Continue



```
If you like the book and the online exercises, give a tip of the hat to these readers, who took the time to tell me about some things that werent working in Version 1.0 and made other contributions as well. This program is now so much better because of their generosity. Joel Kohlbusch for his help with the Chapter 7 exercises. Dorian Maiga, who
caught an error in Chapter 13 of the book. Steven Noe, who spotted an errors in the exercises for Chapter 1, spotted errors in the exercises for Chapter 1.
 Brian Sheets, who spotted an error in the exercises for Chapter 1. Simon Harms at, who told me about a psychological barrier that I need to deal with. GJ Griffiths's friend, who spotted several errors and encouraged me to remove unhelpful remarks from the introduction. Justin Sparks, who suggested a useability improvement. Tom Boyles, who told
 me about a flaw in the timed exercises. And thanks to his son, who provided the test condition. Also an error in a Chapter 54 exercises. Beff Santos, who helped me eliminate an ambiguity in the timed exercises. Rafael Oliveira, whose comments suggested an improvement in my book description. Brad Mellema, for pointing out an error in the exercises
for Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 86, to ambiguous instructions in Chapter 86, to ambiguous instructions in Chapter 13 exercises, found a bug in the Chapter 13 exercises, and spotted errors in Chapter 49, and 72 exercises, and spotted errors in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 13. Callum Makkai, who alerted me to a typo in Chapter 
50, 52, 54, 59, and 62, 63, 64, 65, 79, 82, 85, and 89 exercises. He has been unusually generous and helpful in giving me encouragement and savvy book-marketing advice. Ivan Nikolov, who runs the social site about food, foodolo. Ivan spotted typos in the Chapter 8 exercises, coding errors in Chapters 23, 25, 33, 49, 54, and 61, and 81 of the book,
and errors in Chapter 43 and 65 exercises. He also caught an error in the Chapter 72 exercises. Allie Etcoff, who spotted code mistakes in several locations. Thomas Mechau, who corrected several flawed instructions in Chapter 24 exercises. Kamil Baran, who
made me aware of a shortcoming in my discussion of variables. Vijay Luthra, who suggested I clarify a Chapter 5 exercise. Laine Gebhardt, who corrected some bad code in a Chapter 13 exercise. Laine Gebhardt, who corrected some bad code in a Chapter 13 exercise. Alex Jones, who provided valuable guidance on future books in the series. Gene Kraybill, who spotted a mistake in the table of contents. Chaz Hirales, who
pointed out unnecessary rigidities in Chapter 2 and 3 exercises. Anne Messenger, who spotted errors in the Chapter 4, 13, and 14 exercises. She also made corrections in Chapter 5, 6, 11, 13, 17, and 22 of the book. Matthew Highland, who provided superb guidance on future projects. Alan Forbes, who gave me excellent book-marketing advice.
George Schweitzer, who corrected errors in Chapter 13, 19, and 40 exercises and typos in Chapter 13. Clyde Eugene Makamure, who discovered a weakness that ran through the
entire series of exercises. He also caught errors in Chapter 9 exercises and suggested edits for clarity in the Chapter 9 exercises and helped me clarify an
 instruction in a Chapter 11. He also caught errors in Chapter 12 exercises. And he made a very helpful suggestion for improving exercise instructions. Quint Rahaman, who spotted a coding error in a Chapter 12, 24, 43, 51, 54, and 61 exercises
Nils-Gunnar Nilsson, who told me about a bug in Chapter 3 exercises. Dave Murley. It would almost be easier to say what he didn't help me with. He spotted problems in Chapter 3 exercises. Dave Murley. It would almost be easier to say what he didn't help me with. He spotted problems in Chapter 3 exercises. Dave Murley. It would almost be easier to say what he didn't help me with. He spotted problems in Chapter 3 exercises. Dave Murley. It would almost be easier to say what he didn't help me with. He spotted problems in Chapter 3 exercises. Dave Murley. It would almost be easier to say what he didn't help me with. He spotted problems in Chapter 3 exercises. Dave Murley. It would almost be easier to say what he didn't help me with. He spotted problems in Chapter 3 exercises. Dave Murley. It would almost be easier to say what he didn't help me with. He spotted problems in Chapter 3 exercises. Dave Murley. It would almost be easier to say what he didn't help me with. He spotted problems in Chapter 3 exercises. Dave Murley. It would almost be easier to say what he didn't help me with. He spotted problems in Chapter 3 exercises. Dave Murley. It would almost be easier to say what he didn't help me with. He spotted problems in Chapter 3 exercises. Dave Murley. It would almost be easier to say what he didn't help me with. He spotted problems in Chapter 3 exercises. Dave Murley. It would almost be easier to say what he didn't help me with help me w
exercises. He helped me correct a general problem with some of the live coding exercises and a weakness that ran through the entire collection of exercises. Sean Herrala, who corrected errors in Chapters 3, 4, and
17. Dan Shafer, who corrected errors in Chapter 3 exercises. Tris Nefzger, who suggested an improvement in a Chapter 3 exercise. Brian Eggert, who told me about a bug in a Chapter 3 exercise. Bushir Aziz, who corrected an errors in Chapter 3 exercise. Brian Eggert, who told me about a bug in a Chapter 3 exercise. Mukesh Kumar, who told
me about problems in Chapter 9 and 15 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found ambiguities in Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises. Lene Nytoft Laursen, who found a problem in the Chapter 3 exercises
in Chapter 23 exercises. Al Granberg, who found an error in Chapter 15 and 25 exercises, and tipped me off to something insane in Chapter 31 and 32 exercises, who found an error in Chapter 30 and 48 exercises and suggested
a usability improvement in the exercises that learners are going to love. Leon Robert Walpole, who helped me clear up a problem in a Chapter 21 exercise. Swati Kamtar, who suggested some improvements in Chapter 6 exercises. Brian Miller, who spotted an inaccuracy in a Chapter
6 exercise. Heron, who caught something in a Chapter 8 exercises. Saltosh Walvekar, who caught a typo in a Chapter 25 exercise. Martin Andrews, who
told me about errors in Chapter 6, 53, and 72 exercises. Casey McCann, who found an error in a Chapter 54 exercises. Uniod Nair, who suggested an overall design improvement for the exercises. David White, who spotted an error in Chapter 37, 42, 54, and 57
exercises. Mike Armishaw, who told me about an error in a Chapter 18 exercise. Paul Hume, who pointed out a mistake in Chapter 66. Ryan Smith, who suggested an improvement to a Chapter 66. Ryan Smith, who suggested an improvement to a Chapter 66. Ryan Smith, who suggested an improvement to a Chapter 66. Ryan Smith, who suggested an improvement to a Chapter 66. Ryan Smith, who suggested an improvement to a Chapter 66. Ryan Smith, who suggested an improvement to a Chapter 66. Ryan Smith, who suggested an improvement to a Chapter 66. Ryan Smith, who suggested an improvement to a Chapter 66. Ryan Smith, who suggested an improvement to a Chapter 66. Ryan Smith, who suggested an improvement to a Chapter 66. Ryan Smith, who suggested an improvement to a Chapter 66. Ryan Smith, who suggested an improvement to a Chapter 66. Ryan Smith, who suggested an improvement to a Chapter 66. Ryan Smith, who suggested an improvement to a Chapter 66. Ryan Smith, who suggested an improvement to a Chapter 66. Ryan Smith Sm
between the book and the exercises and a wrong word in Chapter 37. Andy Schwartz, who found bugs in Chapter 71 exercises. Jason Bray, who spotted a problem in a Chapter 7 exercise. Osvaldo Dias dos Santos, who told me about an error in Chapter 21 and 44
exercises. Uriel Cota, who went to a good deal of trouble to document a recurring bug in the exercises. Shang, who alerted me to a typo. Eric Carraway, who told me about several errors in the book. Francesco Badraun, who made a great suggestion that improves clarity in the autocorrect exercises. Unnat Jain, who spotted several typos in the book.
and another in an exercise. Dean Bunnell, who spotted a typo in the book. Chris Heinze, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Matt Jared, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book. Tomas Sandala, who spotted a typo in the book.
problems. I remove the problems, and you start having fun. The first problem is retention. You remember only ten or twenty percent of what you read. That spells failure. To become fluent in a computer language, you have to retain pretty much everything. How can you retain everything? Only by constantly being asked to play everything back. That's
why people use flashcards. But my system does flashcards one better. After reading a short chapter, you go to my website and complete twenty interactive exercises. Algorithms check your work to make sure you know what you think you know. When you stumble, you do the exercise again. You keep trying until you know the chapter cold. The
exercises are free. The second problem is comprehension. Many learners hit a wall when they try to understand advanced concepts like variable scope and prototypes. Unfortunately, they blame themselves. That's why the Dummies books sell so well. But the fault lies with the authors, coding virtuosos who lack teaching talent. I'm the opposite of the
typical software book author. I'll never code fast enough to land a job at Google. But I can teach. Anyway, most comprehension problems are just retention problems in disguise. If you get lost trying to understand variable scope, it's because you don't remember how functions work. Thanks to the interactive exercises on my website, you'll always
understand and remember everything necessary to confidently tackle the next concept. "I've signed up to a few sites like Udemy, Codecademy, FreeCodeCamp, Lynda, YouTube videos, even searched on Coursera but nothing seemed to work for me. This book takes only 10 minutes each chapter and after that, you can exercise what you've just learned
right away!" Amazon reviewer Constanza MoralesBetter than just reading. And more fun. You'll spend two to three times as much time practicing as reading. It's how you wind up satisfied, confident, and proud, instead of confused, discouraged, and defeated. And since many people find doing things more enjoyable than reading things, it can be a
pleasure to learn this way, quite apart from the impressive results you achieve. "Very effective and fun." Amazon reviewer A. BergaminiWritten especially for people who are new to programming. Making no assumptions about what you already know, I walk you through JavaScript slowly,
 patiently. I explain every little thing in sixth-grade English. I avoid unnecessary technical jargon like the plaque.) "The layman syntax he uses...makes it much easier to suddenly realize a concept that seemed abstract and too hard to wrap your head around is suddenly not complicated at all." Amazon reviewer
 IMHOThe exercises keep you focused, give you extra practice where you're shaky, and prepare you for each next step. Every lesson is built on top of a solid foundation that you and I have carefully constructed. Each individual step is small. But, as Amazon reviewer James Toban says, when you get to the end of the book, you've built "a tower of
JavaScript."If you're an accomplished programmer already, my book may be too elementary for you. (Do you really need to be told what a variable is?) But if you're new to programming, more than a thousand five-star reviews are pretty good evidence that my book may be just the one to get you coding JavaScript successfully."Mark Myers' method of
 getting what can be...difficult information into a format that makes it exponentially easier to consume, truly understand, and synthesize into real-world application is beyond anything I've encountered before." Amazon reviewer Jason A. Ruby This article possibly contains original research. Please improve it by verifying the claims made and adding
inline citations. Statements consisting only of original research should be removed. (August 2009) (Learn how and when to remove this message) Programming languages can be grouped by the number and types of paradigms supported. A concise reference for the programming languages can be grouped by the number and types of paradigms supported. A concise reference for the programming languages can be grouped by the number and types of paradigms supported. A concise reference for the programming languages can be grouped by the number and types of paradigms supported. A concise reference for the programming languages can be grouped by the number and types of paradigms supported. A concise reference for the programming languages can be grouped by the number and types of paradigms supported. A concise reference for the programming languages can be grouped by the number and types of paradigms supported.
constructs for concurrency, these may involve multi-threading, support for distributed computing, message passing, shared memory), or futuresActor programming concurrent computation with actors that make local decisions in response to the environment (capable of selfish or competitive behaviour)Constraint
programming relations between variables are expressed as constraint satisfaction or simplex algorithm) Dataflow programming forced recalculation of formulas when data values change (e.g. spreadsheets) Declarative programming describes what computation should perform,
 without specifying detailed state changes cf. imperative programming (functional and logic programming are major subgroups of declarative programming) Distributed programming uses evaluation of mathematical functions and avoids
state and mutable dataGeneric programming uses algorithms written in terms of to-be-specified-later types that are then instantiated as needed for specific types provided as parametersImperative programming west explicit mathematical logic for programming with the specific types provided as parameters.
writing programs that write or manipulate other programs (or themselves) as their data, or that do part of the work at compiler with the rest of
the source code and then compiledReflective programming metaprogramming metaprogramming methods in which a program modifies or extends itselfObject-oriented programming in which inheritance is
achieved by defining classes of objects, versus the objects themselvesPrototype-based object-oriented programming a simple syntax to nest function calls to language originally designed with noneRule-based programming a network of
rules of thumb that comprise a knowledge base and can be used for expert systems and problem deduction & resolutionVisual programming languagesLanguageParadigm
[24]YesYes[25]Yes[26]Library[27]Yes[28]Multiple dispatch, [29] Agents[30]Common Lisp7 (14)Library[31]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[32]Library[3
2]Curry4YesYesYesYesYesYesO (version 2.0)[43][44]7Yes[a 3]YesYesYes[a 2]ECMAScript, E4X, JavaScript, E4X, JavaScript, JScript)4 (5)Partial[a 9][a 10]Library[48][49]YesYesYesYes[a 2]ECMAScript[46][47] (ActionScript, E4X, JavaScript, E4X, JavaScript, JScript)4 (5)Partial[a 9][a 10]Library[48][49]YesYesYes[a 2]ECMAScript[46][47] (ActionScript, E4X, JavaScript, E
 (10) Toolbox [69] Toolbox [70] Yes [71] Toolbox [70] Yes [73] Ye
like based a b c d e f g h i j k l m n o p q r s t u v w x y z aa ab ac ad ae af ag ah ai class-based a b c d e template metaprogramming ab c using Reactive Extensions (Rx) multiple dispatch, method combinations ab c d e actor programming promises, native
extensions^ using Node.js' cluster module or child process.fork method, web workers in the browser, etc.^ a b c d Prototype-based^ using Reactive Extensions (RxJS)^ in Node.js via their events module^ in browsers via their native EventTarget API^ a b c purely functional^ parameterized classes^ immutable^ Uses structs with function
polymorphism and multiple dispatch Akka Archived 2013-01-19 at the Wayback Machine Bragg, S.D.; Driskill, C.G. (2022 September 1994). "Diagrammatic-graphical programming languages and DoD-STD-2167A". Proceedings of AUTOTESTCON '94 (IEEEXplore). Institute of Electrical and Electronics Engineers (IEEE). pp.211220.
doi:10.1109/AUTEST.1994.381508. ISBN978-0-7803-1910-3. S2CID62509261.^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: Tasks and Synchronization Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 9: 
Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Se
2019-09-15. ^OpenMPI^ Boost.MPI^ Boost.MPI^ Boost.MPI^ Boost.MPI^ Boost.MPI^ Boost.MPI^ Boost.MPI^ Boost.MPI^ Boost.MPI^ Boost.MPI - Core.logic - Clojure - Concurrent Programming - Clojure - 
Hierarchies Agents and Asynchronous Actions [2] dataflow extensions [2] dataflow extensions [3] by creating DSLs using the built-in metaprogramming; also see note on functional, constraint and logic paradigms, which are part of declarative [4] MPI, etc via language extensions
template metaprogramming using macros (see C++)^[5][6][7] Prolog implemented as a language extension^ Common Lisp Object System. implemented by the user via a short macro, example of implementation - Visual programming tool based on Common Lisp Object System.
programming extension [9] Archived 2018-04-26 at the Wayback Machine through the Meta Object Protocol D Language Feature Table Phobos std.algorithm D language String Mixins The Little JavaScripter demonstrates fundamental commonality with Scheme, a functional language. Object-Oriented Programming in JavaScript Archived
2019-02-10 at the Wayback Machine gives an overview of object-oriented programming techniques in JavaScript library for building user interfaces". 2019-04-08.^ "Rod-Hooks". GitHub. 2019-04-08.^ "Prolog embedding".
Haskell.org.^ "Functional Reactive Programming". HaskellWiki.^ Cloud Haskell. Template Haskell. Template Haskell (PDF). Oregon State University.^ JSR 331: Constraint
 Programming API Google Cloud Platform Dataflow SDK "JuliaOpt/JuMP.jl". GitHub. JuliaOpt. 11 February 2020. Retrieved 12 February 2020. Retriev
 Kanren implementations in Julia^ "GitHub - abeschneider/PEGParser.jl: PEG Parser for Julia". GitHub. 2018-12-03.^ "GitHub - gitfoxi/Parsimonious.jl: A PEG parser generator for Julia". GitHub. 2017-08-03.^ "Execute loop iterations in parallel". mathworks.com. Retrieved 21 October 2016.^ "Write Constraints". mathworks.com. Retrieved 21
October 2016.^ "Getting Started with SimEvents". mathworks.com. Retrieved 21 October 2016.^ "Execute MATLAB eval". mathworks.com. Re
2016.^ "Class Metadata". mathworks.com. Retrieved 21 October 2016.^ "Simulink". mathworks.com. Retrieved 21 October 2016.
5)^ PHP Manual, Anonymous functions "Parallel Processing and Multiprocessing in Python.org. Retrieved 21 October 2016.^ "python.org. Retrieved 21 October 2016.^
21 October 2016. Chapter 9. Metaprogramming. chimera.labs.oreilly.com. Archived from the original on 23 October 2016. Retrieved 22 October 2016. Retrieved 2
Retrieved 22 October 2016. "PyDatalog". Retrieved 22 October 2016. "Future batchtools". "Futu
Guide: Distributed Places Lazy Racket Channels and other mechanisms "Problem Solver module". Feed operator Cro module Rust compiler Rust compiler. Rust macros guide". Rust. Retrieved 19 January 2015. "The Rust compiler
plugins guide". Rust. Retrieved 19 January 2015. The Rust Reference 6.1.3.1 An Overview of the Scala Programming/Introduction". en.wikibooks.org. Retrieved 22 October 2016. The Rust Reference 6.1.3.1 An Overview of the Scala Programming/Introduction and Retrieved 22 October 2016. The Rust Reference 6.1.3.1 An Overview of the Scala Programming Language Specification and Retrieved 22 October 2016. The Rust Reference 6.1.3.1 An Overview of the Scala Programming/Introduction and Retrieved 22 October 2016. The Rust Reference 6.1.3.1 An Overview of the Scala Programming Language Specification and Retrieved 22 October 2016. The Rust Reference 6.1.3.1 An Overview of the Scala Programming Language Specification and Retrieved 22 October 2016. The Rust Reference 6.1.3.1 An Overview of the Scala Language Specification and Retrieved 22 October 2016. The Rust Reference 6.1.3.1 An Overview of the Scala Language Specification and Retrieved 22 October 2016. The Rust Reference 6.1.3.1 An Overview of the Scala Language Specification and Retrieved 22 October 2016. The Rust Reference 6.1.3.1 An Overview of the Scala Language Specification and Retrieved 22 October 2016. The Rust Reference 6.1.3.1 An Overview of the Scala Language Specification and Retrieved 22 October 2016. The Rust Reference 6.1.3.1 An Overview of the Scala Language Specification and Retrieved 22 October 2016. The Rust Reference 6.1.3.1 An Overview of the Scala Language Specification and Retrieved 22 October 2016. The Rust Reference 6.1.3.1 An Overview of the Scala Language Specification and Retrieved 22 October 2016. The Rust Reference 6.1.3.1 An Overview of the Scala Language Specification and Retrieved 22 October 2016. The Rust Reference 6.1.3.1 An Overview of the Scala Language Specification and Retrieved 22 October 2016. The Rust Reference 6.1.3.1 An Overview of the Scala Language Specification and Retrieved 22 October 2016. The Rust Retrieved 22 October
 Programming Language Experts, Wolfram Language Documentation. External Programs, Wolfram Language Documentation. External Programs, Wolfram Language Documentation. Sim Coplien, Multiparadigm Design for C++, Addison-Wesley Professional, 1998. Retrieved from Grams, Wolfram Language Documentation.
 friend of Edward, Prince of Wales, for over 20 years, in 1890 he attended a house party at Tranby Croft, where he took part in a game of baccarat at the behest of the prince. During the course of two nights' play he was accused of cheating, which he denied. After news of the affair leaked out, he sued five members of the party for slander; Edward was
 called as a witness. The case was a public spectacle in the UK and abroad, but the verdict went against Gordon-Cumming and he was ostracised from polite society. After the court case he married an American heiress, but their relationship was unhappy. (Fullarticle...)Recently featured: Great Wilbraham (causewayed enclosure)Henry de
 HinuberHurricane Claudette (2003)ArchiveBy emailMore featured articlesAboutPostcard with a Fula woman... that Francis-Edmond Fortier published more than 3,300 postcards of French West Africa (example pictured) between 1901 and 1920?... that a language riot broke out between members of Our Lady of the Rosary in 1917?... that Oleksandi
 Rodin's opera Kateryna was staged despite barricades, bombings, and an air-raid alarm?... that Paul Among the People treats the Pauline epistles as sources comparable to Homer, Aristophanes and Virgil on Greco-Roman attitudes?... that Gyula Kakas competed at two Olympics in gymnastics, set the Hungarian pole-vault record, and played for a
 national-champion football club?... that a lyric in Beautiful Chaos was praised for "spreading queer joy"?... that defending champions Bermuda did not compete in the women's football tournament at the 2015 Island Games?... that Vatican Taekwondo has no registered athletes or coaches?... that Iceland's entry for Eurovision in 2025 brought out a line
of Ash Wednesday costumes? ArchiveStart a new articleNominate an articleMuhammadu Buhari (pictured) dies at the age of 82. Clashes between Druze militias and the Syrian Armed Forces result in hundreds of deaths. The International Criminal Court issues arrest warrants for Taliban leaders
 Hibatullah Akhundzada and Abdul Hakim Haqqani over their alleged persecution of women in Afghanistan. Flooding in Central Texas, United States, leaves at least 140 people dead. Ongoing: Gaza warRussian invasion of Ukrainetimeline Sudanese civil wartimeline Recent deaths: Andrea Gibson Raymond Guiot Felix Baumgartner Fauja Singh Bradley John
 MurdochFrank BarrieNominate an articleJuly 20Forensic experts at the site of the Suru bombing1807 French brothers Claude and Nicphore Nipce received a patent for their Pyrolophore, one of the world's first internal combustion engines. 1951 Abdullah I of Jordan was assassinated while visiting the Al-Aqsa Mosque in Jerusalem. 1976 The Viking 1
lander became the first spacecraft to successfully land on Mars and perform its mission.1997 USSConstitution, one of the United States Navy's original six frigates, sailed for the first time in 116 years after a full restoration.2015 A suicide attack (aftermath pictured) in Suru, Turkey, for which Islamic State of Iraq and the Levant (ISIL) claimed
responsibility, killed 34 people and injured 104 others. Alexander the Great (b.356 BC) Amanda Clement (d.1971) Bruce Lee (d.1973) Gisele Bndchen (b.1980) More anniversaries: July 19 July 20 July 21 Archive By emailList of days of the year About C/2022 E3 (ZTF) is a non-periodic comet from the Oort cloud that was discovered by the Zwicky Transient
Facility (ZTF) in 2022. With a comet nucleus of around 1 kilometer, C/2022 E3 rotates on its axis once every 8.5 to 8.7 hours. Its tails of dust and gas extended for millions of kilometers and, during January 2023, an anti-tail was also visible. The comet reached its most recent perihelion in January 2023, at a distance of 1.11AU
(166millionkm; 103millionmi) from the sun, and the closest approach to Earth was a few weeks later, at a distance of 0.28AU (42millionkm; 26millionmi). The comet reached magnitude 5 and was visible with the naked eye under moonless dark skies. This photograph of C/2022 E3 was taken in January 2023 and released by the Italian National
 Institute for Astrophysics. Photograph credit: Alessandro Bianconi; National Institute for AstrophysicsRecently featured: Passion fruitBasilica of StPaul, RabatClouded ApolloArchiveMore featured picturesCommunity portal The central hub for editors, with resources, links, tasks, and announcements. Village pump Forum for discussions about Wikipedia
itself, including policies and technical issues. Site news Sources of news about Wikipedia and the broader Wikipedia and t
navigate the encyclopedia. Wikipedia is written by volunteer editors and hosted by the Wikimedia Foundation, a non-profit organization that also hosts a range of other volunteer projects: CommonsFree media repository MediaWikiWiki software development Meta-WikiWikimedia project coordination WikibooksFree textbooks and manuals
WikidataFree knowledge base WikinewsFree-content news Wikipedia is written in English. Many other Wikipedias are available; some of the largest are
 listed below. 1,000,000+ articles DeutschEspaolFranaisItalianoNederlandsPolskiPortugusSvenskaTing Vit 250,000+ articles Bahasa IndonesiaBahasa MelayuBn-lm-gCataletinaDanskEestiEsperantoEuskaraMagyarNorsk bokmlRomnSimple EnglishSloveninaSrpskiSrpskohrvatskiSuomiTrkeOzbekcha 50,000+ articles
 AsturianuAzrbaycancaBosanskiFryskGaeilgeGalegoHrvatskiKurdLatvieuLietuviNorsk nynorskShqipSloveninaRetrieved from "2This article by adding citations to reliable sources. Unsourced material may be challenged and removed. Find sources: "1807" news newspapers books
 scholar JSTOR (November 2015) (Learn how and when to remove this message) Calendar yearYearsMillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmillennium2ndmill
Eylau. June 14: Napoleon triumphs over Russia's General Benningsen, at the Battle of Friedland. 1807 (MDCCCVII) was a common year starting on Tuesday of the Julian calendar, the 1807th year of the Common Era (CE) and Anno Domini (AD) designations, the 807th year of the
2ndmillennium, the 7th year of the 19thcentury, and the 8th year of the 1800s decade. As of the start of 1807, the Gregorian calendar was 12 days ahead of the Julian calendar was 12 days ahead of the J
countryAustraliaBrazilCanadaDenmarkFranceGermanyNew ZealandNorwayRussiaSouth AfricaSwedenUnited KingdomUnited StatesOther topicsRail transportScienceSportsLists of leadersLawBirth and death categoriesBirthsDeathsEstablishments and disestablishments
categoriesEstablishmentsDisestablishmentsDisestablishmentsWorks categoryWorksvte1807 in various calendar1256Assyrian calendar1256Assyrian calendar1257Balinese saka calendar1256Assyrian calendar1251Burmese
calendar1169Byzantine calendar73157316Chinese calendar2973Ethiopian calendar15231524Discordian calendar15231524Discordian calendar17291800Hebrew calendar15231524Discordian calendar17291800Hebrew calendar172
 calendar807808Iranian calendar11851186Islamic calendar12211222Japanese calendar1851186Islamic calendar4140Minguo calendar4140Minguo calendar4140Minguo calendar339Thai solar calendar23492350Tibetan calendar4140Minguo calend
Hare)1934 or 1553 or 781Wikimedia Commons has media related to 1807. January 7 The United Kingdom of Great Britain and Ireland issues an Order in Council prohibiting British ships from trading with France or its allies. [1] January 20 The Sierra Leone Company, faced with bankruptcy because of the imminent abolition of the slave trade in British
colonies, petitions the British government for purchase and transfer of its property to the Crown; Parliament approves the transfer on July 29, and it takes effect on January 1, 1808.[2] February 3 Napoleonic Wars and Anglo-Spanish War: Battle of Montevideo The British Army captures Montevideo from the Spanish Empire, as part of the British
 invasions of the Ro de la Plata. February 7 Napoleon leads the forces of the French Empire in an invasion of the Russian Empire, and begins fighting at the Battle of Eylau against the Russians under Bennigsen. February 10 The Survey of
the Coast (renamed the United States Coast Survey in 1836 and the United States Coast Survey in 1878) is established; work begins on August 3, 1816. February 17 Henry Christopher is elected first President of the State of Haiti, ruling the northern part of the country. February 19 Burr conspiracy: In Alabama, former Vice President of
the United States Aaron Burr is tried for conspiracy, but acquitted. February 23 The Slave Trade Act is passed in the House of Commons of the United States Congress passes the Act Prohibiting Importation of Slaves "into any port or place within the jurisdiction of the United States ... from
any foreign kingdom, place, or country" (to take effect January 1, 1808). March 25The United Kingdom Slave Trade Act becomes law abolishing the slave trade in most of the British Empire[5] with effect from 1 May (slavery itself is abolished in British colonies in 1833). The Swansea and Mumbles Railway in South Wales, at this time known as the
Oystermouth Railway, becomes the first passenger-carrying railway in the world. March 29 H. W. Olbers discovers the asteroid Vesta. April 412 Froberg mutiny: The British suppress a mutiny at Fort Ricasoli, Malta, by men of the irregularly-recruited Froberg Regiment. April 14 African Institution holds its first meeting in London; it is intended to
improve social conditions in Sierra Leone. May 22 A grand jury indicts former Vice President of the United States Aaron Burr for treason. [6] May 24 Siege of Danzig ends after 6 weeks with Prussian and Russian defenders capitulating to French forces. May 29 Selim III, Ottoman Emperor since 1789, is deposed in favour of his nephew Mustafa IV. May
31 Primitive Methodism originates in an All Day of Prayer at Mow Cop, in the north midlands of England.[7]June 9 The Duke of Portland is chosen as Prime Minister after the United Kingdom general election.June 10 The Battle of Friedland: Napoleon decisively defeats Bennigsen's Russian army.June 22
Chesapeake Leopard affair: British Royal Navy fourth rate HMSLeopard attacks and boards United States Navy frigate USS Chesapeake off Norfolk, Virginia, seeking deserters. This act of British aggression plays a role in the run-up to the War of 1812. July 5 A disastrous British attack is mounted against Buenos Aires, during the second failed invasion
of the Ro de la Plata. July 79 The Treaties of Tilsit are signed between France, Prussia and Russia. Napoleon and Russian Emperor Alexander I ally together against the British. The Prussians are forced to cede more than half their territory, which is formed into the Duchy of Warsaw in their former Polish lands, and the Kingdom of Westphalia in
 western Germany. The Free City of Danzig is also formed (established September 9 by Napoleon). July 13 With the death of Henry Benedict Stuart, the last Stuart claimant to the throne of the United Kingdom, Jacobitism comes to an effective end. July 20 Nicphore Nipce is awarded a patent by Napoleon Bonaparte for the Pyrolophore, the world's first
internal combustion engine, after it successfully powers a boat upstream on the river Sane in France. August 17 The North River, inaugurating the first commercial steamboat, Robert Fulton's first American steamboat steamboat, Robert Fulton's first American steamboat, Robert Fulton'
Burr is acquitted of treason. He had been accused of plotting to annex parts of Louisiana and Mexico, to become part of an independent republic. September 27 Battle of Copenhagen: The British Royal Navy bombards Copenhagen with fire bombs and phosphorus rockets, to prevent the Dano-Norwegian navy from surrendering to Napoleon; 30% of the
city is destroyed, and 2,000 citizens are killed. September 7 Robert Morrison, the first Protestant missionary to China, arrives in Guangzhou (Canton). [8] September 13 Beethoven's Mass in C major, Op. 86, is premiered, commissioned by Nikolaus I, Prince Esterhzy, and displeasing him. [9] September 27 Napoleon purchases the Borghese art collection,
including the Antinous Mondragone, and brings it to Paris.[10]October 9 Prussian Reform Movement: Serfdom is founded.October 30 El Escorial Conspiracy: Ferdinand, Prince of Asturias is arrested for conspiring against his father Charles IV of Spain.November 24 Battle
of Abrantes, Portugal: The French under Jean-Andoche Junot take the town. November 29 Portuguese Queen Maria I and the Court embark at Lisbon, bound for Brazil. Rio de Janeiro becomes the Portuguese capital. December 511 Napoleonic Wars: Raid on Griessie A British Royal Navy squadron attacks the Dutch port of Griessie on Java in the Dutch
East Indies, eliminating the last Dutch naval force in the Pacific and concluding the Java campaign of 18061807.[11]December 17 Napoleonic Wars: France issues the Milan Decree which confirms the Continental System (i.e. no European country is to trade with the United Kingdom). December 22 The United States Congress passes the Embargo Act
a trade embargo on all foreign nations. Battle of Hingakaka between two factions of Mori people, the largest battle ever fought in New Zealand, and the Battle of Moremonui, first of the Musket Wars. Robert E. Lee January 13 Napoleon Bonaparte Buford, American general, railroad
executive (d. 1883)January 19 Robert E. Lee, American Confederate general (d. 1870)January 28 Robert McClure, Irish-born Arctic explorer (d. 1882)[13]March 1 Wilford Woodruff, American religious leader (d. 1849)February 27 Henry Wadsworth Longfellow, American poet (d. 1882)[13]March 1 Wilford Woodruff, American religious leader (d. 1870)
1898)March 14 Josephine of Leuchtenberg, Queen of Sweden and Norway (d. 1875)April 2 William F. Packer, American politician (d. 1865)April 26 Charles Auguste Frossard, French general (d. 1875)May 28 Louis Agassiz, Swiss-born zoologist and
geologist (d. 1873)June 6 Adrien-Franois Servais, Belgian musician (d. 1886)June 16 John Westcott, American surveyor and politician (d. 1886)August 11 David Rice Atchison, American politician (d. 1886)August 15 Jules Grvy, 4th President of France (d. 1891)August 18 Charles
 Francis Adams Sr., American historical editor, politician and diplomat (d. 1886)September 2 Fredrika Runeberg, Finnish writer (d. 1879)[14]September 7 Henry Sewell, 1st Premier of New Zealand (d. 1879)September 16 John Lenthall, American naval architect and shipbuilder (d. 1882)October 8 Harriet Taylor, English philosophical writer (d. 1858)
[15]October 26 Barbu Catargiu, 1st Prime Minister of Romania (d. 1862)October 29 Aneo Kraljevi, Herzegovinian Catholic bishop (d. 1879)October 30 Christopher Wordsworth, Bishop of Lincoln (d. 1885)November 16 Eduard von Fransecky, Prussian general (d. 1890)December 8 Friedrich Traugott Ktzing, German pharmacist, botanist and
 phycologist (d. 1893)December 17 John Greenleaf Whittier, American Quaker poet and abolitionist (d. 1892)Pasquale Paoli, Corsican patriot, military leader (b. 1725)February 27 Louise du Pierry, French astronomer (b.1746)March 10 Jean Thurel,
 French soldier (b. 1698) April 4 Jrme Lalande, French astronomer (b. 1732) April 10 Duchess Anna Amalia of Brunswick-Wolfenbttel, regent of Weimar and Eisenach (b. 1725) May 13 Eliphalet Dyer, American statesman, judge (b. 1721) May 17 John Gunby,
Maryland soldier in the American Revolutionary War (b. 1745)May 18 John Douglas, Scottish Anglican bishop, man of letters (b. 1721)June 9 Andrew Sterett, American naval officer (b. 1725)July 19 Uriah Tracy, American politiciar
 and congressman from Connecticut, 1793 until 1807 (b. 1755)September 14 George Townshend, 1st Marquess Townshend, British field marshal (b. 1752)November 2 Louis Auguste Le Tonnelier de Breteuil, Prime Minister of King Louis XVI of France (b. 1730)November 5 Angelica
Kauffman, Swiss painter (b. 1741)November 25 Jean-Franois Rewbell, French engineer, hydrographer, politician (b. 1747)November 26 Oliver Ellsworth, American founding father and 3rd acres (b. 1752)November 27 Jean-Franois Rewbell, French engineer, hydrographer, politician (b. 1747)November 26 Oliver Ellsworth, American founding father and 3rd acres (b. 1752)November 27 Jean-Franois Rewbell, French engineer, hydrographer, politician (b. 1747)November 26 Oliver Ellsworth, American founding father and 3rd acres (b. 1748)November 27 Jean-Franois Rewbell, French engineer, hydrographer, politician (b. 1752)November 27 Jean-Franois Rewbell, French engineer, hydrographer, politician (b. 1752)November 28 Jean-Franois Rewbell, French engineer, hydrographer, politician (b. 1752)November 28 Jean-Franois Rewbell, French engineer, hydrographer, politician (b. 1752)November 28 Jean-Franois Rewbell, French engineer, hydrographer, politician (b. 1752)November 28 Jean-Franois Rewbell, French engineer, hydrographer, politician (b. 1752)November 28 Jean-Franois Rewbell, French engineer, hydrographer, politician (b. 1752)November 28 Jean-Franois Rewbell, French engineer, hydrographer, politician (b. 1752)November 28 Jean-Franois Rewbell, French engineer, hydrographer, hydrographer, politician (b. 1752)November 28 Jean-Franois Rewbell, hydrographer, hydrographe
Chief Justice of the United States Supreme Court (b. 1745)December 19 Friedrich Melchior, Baron von Grimm, German writer (b. 1723)December 29 Diogo de Carvalho e Sampayo, Portuguese diplomat, scientist (b. 1750)^ William S. Dudley, ed. The Naval War of 1812: A Documentary
History (Naval Historical Center, 1985) p34^ Stephen Tomkins, The Clapham Sect: How Wilberforce's Circle Transformed Britain (Lion Books, 2012) p200^ William Hodgson, The Life of Napoleon Bonaparte, Once Emperor of the French, who Died in Exile, at St. Helena, After a Captivity of Six Years' Duration (Orlando Hodgson, 1841) p384^
 "William Wilberforce (17591833)". Retrieved January 18, 2021. "Abolition of the Slave Trade 1807". BBC. Archived from the original on October 11, 2007. The Secret of Mow Cop: a new appraisal of the origins of Primitive Methodism. London: Epworth
 Press. "Sketch of the Canton Protestant Mission", by Rev. John Chalmers, in The Chinese Recorder and Missionary Journal, Volume 7 (American Presbyterian Mission Press, 1876) p174. Marston, Nicholas (2006). Beethoven: Mass in C major, Op. 86 (PDF) (CD). Hyperion Records. CDH55263. Retrieved May 15, 2015. Francis Haskell and Nicholas
 Penny, Taste and the Antique: The Lure of Classical Sculpture, 1500-1900 (Yale University Press, 1982) p281^ Clowes, William Laird (1997) [1900]. The Royal Navy: A History from the Earliest Times to 1900. Vol.V. Chatham Publishing. ISBN1-86176-014-0.^ Anderson, Atholl (2022). "War is their principal profession: On the frequency and causes of
Maori warfare and migration, 12501850 CE". In Clark, Geoffrey; Litster, Mirani (eds.). Archaeological Perspectives on Conflict and Warfare in Australia and the Pacific. Canberra: ANU Press. p.51. ISBN 978-1-76046-489-9. Retrieved March 29, 2025 via Google Books. Calhoun, Charles C (2004). Longfellow: A Rediscovered Life. Boston: Beacon Press
p.5. ISBN978-0807070260.^ "Fredrika Runeberg". Svenska Littaratursllskapet i Finland. Archived from the original on May 6, 2014. Retrieved May 14, 2014.^ Robson, Ann P. "Mill [ne Hardy; other married name Taylor], Harriet". Oxford Dictionary of National Biography (onlineed.). Oxford University Press. doi:10.1093/ref:odnb/38051. (Subscription
or UK public library membership required.)Retrieved from " 3Second-largest asteroid of the main asteroid beltThis article is about the asteroid. For the Roman goddess, see Vesta (mythology). For other uses, see Vesta (mythology). For other uses, see Vesta (mythology).
 pole.DiscoveryDiscoveredbyHeinrich Wilhelm OlbersDiscoverydate29 March 1807DesignationsMPCdesignation(4) VestaPronunciation/vst/[1]Named afterVestaMinorplanet categoryMain belt (Vesta family)AdjectivesVestanVestian[a]Symbol (historically astronomical, now astrological)Orbital characteristics[6]Epoch 13September
 2023 (JD2453300.5)Aphelion2.57AU (384millionkm)Perihelion2.15AU (322millionkm)Semi-major axis2.36AU (353millionkm)Eccentricity0.0894Orbital period (sidereal)3.63yr (1325.86d)Average orbital speed19.34km/sMean anomaly169.4Inclination7.1422 to ecliptic5.58 to invariable plane [7]Longitude of ascending node 103.71T imeof perihelion2.64m (353m) invariable plane [7]Longitude of ascending node 103.71T imeof perihelion2.64m (353m) invariable plane [7]Longitude of ascending node 103.71T imeof perihelion2.64m (353m) invariable plane [7]Longitude of ascending node 103.71T imeof perihelion2.64m (353m) invariable plane [7]Longitude of ascending node 103.71T imeof perihelion103.71T invariable plane [7]Longitude of ascending node 103.71T invariable plane 103.71T inv
December 2021[8]Argumentof perihelion151.66SatellitesNoneEarthMOID1.14AU (171millionkm)Proper orbital elements[9]Proper semi-major axis2.36151AUProper eccentricity0.098758Proper inclination6.39234Proper mean motion99.1888deg/ yrProper orbital period3.62944 yr(1325.654 d)Precession of perihelion36.8729 (2343 years)arcsec/
yr Precession of the ascending node 39.5979~(2182~years) arcsec/~yr Physical~characteristics Dimensions 572.6 km~557.2 km~446.4 km [10] Mean~diameter 525.40.2 km [10] Flattening 0.2204 Surface~area (8.660.2) 105 km 2 [b] [11] Volume 7.4970 107 km 3 [10] Mass (2.59027 10.000058) 1020 kg [12] Mean~density 3.4560.035 g/cm 3 [10] Equatorial~surface~gravity 0.22 m/s 2 m/s 
 (0.022g0)Equatorial escape velocity 0.36km/sSynodic rotation period 0.2226d (5.342h)[6][13]Equatorial rotation velocity 93.1m/s[c]Axial tilt 29Northpole right ascension 20h 32m[d]Northpole right ascension 2
magnitude(H)3.20[6][15]Angular diameter 0.70 to 0.22Vesta (minor-planet designation: 4 Vesta) is one of the largest objects in the asteroid belt, with a mean diameter of 525 kilometres (326mi).[10] It was discovered by the German astronomer Heinrich Wilhelm Matthias Olbers on 29 March 1807[6] and is named after Vesta, the virgin goddess of
 home and hearth from Roman mythology.[19]Vesta is thought to be the second-largest asteroid, both by mass and by volume, after the dwarf planet Ceres.[20][21][22] Measurements give it a nominal volume only slightly larger than that of Pallas (about 5% greater), but it is 25% to 30% more massive. It constitutes an estimated 9% of the mass of them
 asteroid belt.[23] Vesta is the only known remaining rocky protoplanet of the kind that formed the terrestrial planets.[24] Numerous fragments of Vesta were ejected by collisions one and two billion years ago that left two enormous craters occupying much of Vesta's southern hemisphere.[25][26] Debris from these events has fallen to Earth as
 howardite eucritediogenite (HED) meteorites, which have been a rich source of information about Vesta. [27][28][29]Vesta is the brightest asteroid visible to the naked eye. Its maximum distance from the Sun is slightly greater than the minimum distance of
 Ceres from the Sun,[e] although its orbit lies entirely within that of Ceres.[30]NASA's Dawn spacecraft entered orbit around Vesta on 5 September 2012[31] en route to its final destination, Ceres. Researchers continue to examine data collected by Dawn for additional insights into
the formation and history of Vesta.[32][33]Vesta, Ceres, and the Moon with sizes shown to scaleHeinrich Olbers discovered Pallas in 1802, the year after the discovery of Ceres. He proposed that the two objects were the remnants of a destroyed planet. He sent a letter with his proposed to the British astronomer William Herschel, suggesting that a
not fragments of a larger body. Because the asteroid Juno had been discovered in 1804, this made Vesta the fourth object to be identified in the region that is now known as the asteroid belt. The discovery was announced in a letter addressed to German astronomer Johann H. Schrter dated 31 March.[35] Because Olbers already had credit for
 discovering a planet (Pallas; at the time, the asteroids were considered to be planets), he gave the honor of naming his new discovery to German mathematician Carl Friedrich Gauss, whose orbital calculations had enabled astronomers to confirm the existence of Ceres, the first asteroid, and who had computed the orbit of the new planet in the
remarkably short time of 10 hours.[36][37] Gauss decided on the Roman virgin goddess of home and hearth, Vesta, or national variants thereof, is in international use with two exceptions: Greece and China. In Greek, the name adopted
was the Hellenic equivalent of Vesta, Hestia (4); in English, that name is used for 46 Hestia (Greeks use the name "Hestia" for both, with the minor-planet numbers used for disambiguation). In Chinese, Vesta is called the 'hearth-god(dess) star', Zoshnxng, naming the asteroid for Vesta's role, similar to the Chinese names of Uranus, Neptune, and
Pluto.[f]Upon its discovery, Vesta was, like Ceres, Pallas, and Juno before it, classified as a planet and given a planet and g
[g]The asteroid symbols were gradually retired from astronomical use after 1852, but the symbols for the first four asteroids were resurrected for astrology in the 1970s. The abbreviated modern astrology in the 1970s.
 time the Solar System was thought to have eleven planets.[47] However, in 1845, new asteroids started being discovered at a rapid pace, and by 1851 there were fifteen, each with its own symbol, in addition to the eight major planets (Neptune had been discovered in 1846). It soon became clear that it would be impractical to continue inventing new
  planetary symbols indefinitely, and some of the existing ones proved difficult to draw guickly. That year, the problem was addressed by Benjamin Apthorp Gould, who suggested numbering asteroid. Thus, the fourth asteroid, Vesta, acquired the
 generic symbol . This was soon coupled with the name into an official numbername designation, Vesta, as the number of minor planets increased. By 1858, the circle had been simplified to parentheses, (4) Vesta, was also briefly used, but had more or less completely died
out by 1949.[48]SPHERE image is shown on the left, with a synthetic view derived from Dawn images shown on the right for comparison.[49]Photometric observations of Vesta were made at the Harvard College Observatory in 18801882 and at the Observatory in 18801882 and at the Observatory in 1801882 and at the Observations of Vesta were made at the Harvard College Observatory in 1801882 and at the Observatory in 180
be determined by the 1950s. However, the early estimates of the rotation rate came into question because the light curve included variations in both shape and albedo. [50] Early estimates of the diameter of 51317km (31911mi) in
1879, which is close to the modern value for the mean diameter, but the subsequent estimates ranged from a low of 390km (242mi) up to a high of 602km (374mi) during the next century. The measured estimates were based on photometry. In 1989, speckle interferometry was used to measure a dimension that varied between 498 and 548km (309).
and 341mi) during the rotational period [51] In 1991, an occultation of the star SAO 93228 by Vesta was observed from multiple locations in the eastern United States and Canada, Based on observations from 14 different sites, the best fit to the data was an elliptical profile with dimensions of about 550km 462km (342mi 287mi).[52] Dawn confirmed
this measurement.[i] These measurements will help determine the thermal history, size of the core, role of water in asteroid evolution and what meteorites found on Earth come from these bodies, with the ultimate goal of understanding the conditions and processes present at the solar system's earliest epoch and the role of water content and size in
planetary evolution.[53]Vesta became the first asteroid to have its mass determined. Every 18 years, the asteroid 197 Arete approaches within 0.04AU of Vesta at (1.200.08)1010M (solar masses).[54] More refined estimates
followed, and in 2001 the perturbations of 17 Thetis were used to calculate the mass of Vesta to be (1.310.02)1010M.[55] Dawn determined it to be 1.30291010M.Vesta orbits the Sun between Mars and Jupiter, within the asteroid belt, interior to the Kirkwood gap at 2.50AU. Its
orbit is moderately inclined (i = 7.1, compared to 7 for Mercury and 17 for Pluto) and moderately eccentric (e = 0.09, about the same as for Mars).[6]True orbital resonances between asteroids are considered unlikely. Because of their small masses relative to their large separations, such relationships should be very rare.[56] Nevertheless, Vesta is
able to capture other asteroids into temporary 1:1 resonant orbital relationships (for periods up to 2 million years or more) and about forty such objects have been identified.[57] Decameter-sized objects detected in the vicinity of Vesta by Dawn may be such quasi-satellites rather than proper satellites.[57]Olbers Regio (dark area) defines the prime
meridian in the IAU coordinate system. It is shown here in a Hubble shot of Vesta, because it is not visible in the more detailed Dawn images. Claudia crater (indicated by the arrow at the bottom of the closeup image at right) defines the prime meridian in the Dawn/NASA coordinate system. Vesta's rotation is relatively fast for an asteroid (5.342h) and
prograde, with the north pole pointing in the direction of right ascension 20h32min, declination +48 (in the constellation Cygnus) with an uncertainty of about 10. This gives an axial tilt of 29.[58]Two longitudinal coordinate systems are used for Vesta, with prime meridians separated by 150. The IAU established a coordinate system in 1997 based on
Hubble photos, with the prime meridian running through the center of Olbers Regio, a dark feature 200km across. When Dawn arrived at Vesta, mission scientists found that the location of the pole assumed by the IAU was off by 10, so that the location of the pole assumed by the IAU was off by 10, so that the IA
not discernible from up close, and so was not adequate to define the prime meridian with the precision they needed. They corrected the pole, but also established a new prime meridian 4 from the center of Claudia, a sharply defined crater 700 metres across, which they say results in a more logical set of mapping quadrangles. [59] All NASA
publications, including images and maps of Vesta, use the Claudian meridian, which is unacceptable to the IAU. The IAU Working Group on Cartographic Coordinates and Rotational Elements recommended a coordinate system, correcting the pole but rotating the Claudian longitude by 150 to coincide with Olbers Regio.[60] It was accepted by the
IAU, although it disrupts the maps prepared by the Dawn team, which had been positioned so they would not bisect any major surface features. [59][61] Relative sizes of the four largest asteroids. Vesta is second from left. This graph was using the legacy Graph extension, which is no longer supported. It needs to be converted to the new Chart
extension. The mass of 4 Vesta (blue) compared to other large asteroids: 1 Ceres, 2 Pallas, 10 Hygiea, 704 Interamnia, 15 Eunomia and the remainder of the Main Belt. The unit of mass is 1018 kg. Other objects in the Solar system with well-defined masses within a factor of 2 of Vesta's mass are Varda, Gknhmdm, and Salacia (245, 136, and 4921018).
kg, respectively). No moons are in this range: the closest, Tethys (Saturn II) and Enceladus (Saturn II), are over twice and less than half of Vesta's massive body. [62][23] Vesta is, however, the most massive body that formed in the
asteroid belt, as Ceres is believed to have formed between Jupiter and Saturn. Vesta's density is lower than those of the four terrestrial planets but is higher than those of most asteroids, as well as all of the moons in the Solar System except Io. Vesta's surface area is about the same as the land area of Pakistan, Venezuela, Tanzania, or Nigeria; slightly
under 900,000km2 (350,000sqmi; 90million ha; 220million acres). It has an only partially differentiated interior. [63] Vesta is only slightly larger (525.40.2km [10]) than 2 Pallas (5123km) in mean diameter, [64] but is about 25% more massive. Vesta's shape is close to a gravitationally relaxed oblate spheroid, [58] but the large concavity and protrusion at
the southern pole (see 'Surface features' below) combined with a mass less than 51020kg precluded Vesta from automatically being considered a dwarf planet under International Astronomical Union (IAU) Resolution XXVI 5.[65] A 2012 analysis of Vesta's shape [66] and gravity field using data gathered by the Dawn spacecraft has shown that Vesta is
currently not in hydrostatic equilibrium.[10][67]Temperatures on the surface have been estimated to lie between about 20C (253K) with the Sun overhead, dropping to about 190C (83.1K) at the winter pole. Typical daytime and nighttime temperatures are 60C (213K) and 130C (143K), respectively. This estimate is for 6 May 1996, very close to
perihelion, although details vary somewhat with the seasons.[16]Further information: List of geological features on VestaBefore the arrival of the Dawn spacecraft, some Vestan surface features had already been resolved using the Hubble Space Telescope and ground-based telescopes (e.g., the Keck Observatory).[68] The arrival of Dawn in July 2011eron features on VestaBefore the arrival of the Dawn spacecraft, some Vestan surface features had already been resolved using the Hubble Space Telescope and ground-based telescopes (e.g., the Keck Observatory).[68] The arrival of Dawn in July 2011eron features had already been resolved using the Hubble Space Telescope and ground-based telescopes (e.g., the Keck Observatory).[68] The arrival of Dawn in July 2011eron features had already been resolved using the Hubble Space Telescope and ground-based telescopes (e.g., the Keck Observatory).[68] The arrival of Dawn in July 2011eron features had already been resolved using the Hubble Space Telescope (e.g., the Keck Observatory).[68] The arrival of Dawn in July 2011eron features had already been resolved using the Hubble Space Telescope (e.g., the Keck Observatory).[68] The arrival of Dawn in July 2011eron features had already been resolved using the Hubble Space Telescope (e.g., the Keck Observatory).[68] The arrival of Dawn in July 2011eron features had already been resolved using the Hubble Space Telescope (e.g., the Keck Observatory).[68] The arrival of Dawn in July 2011eron features had already been resolved using the Hubble Space Telescope (e.g., the Keck Observatory).[68] The arrival of Dawn in July 2011eron features had already been resolved using the Hubble Space Telescope (e.g., the Keck Observatory).[68] The arrival of Dawn in July 2011eron features had already been resolved (e.g., the Keck Observatory).[68] The arrival of Dawn in July 2011eron features had already been resolved (e.g., the Control features).
revealed the complex surface of Vesta in detail.[69]Geologic map of Vesta (Mollweide projection).[70] The most ancient and heavily cratered regions are brown; areas modified by the Veneneia and Rheasilvia impacts are purple (the Saturnalia Fossae Formation, in the north)[71] and light cyan (the Divalia Fossae Formation, equatorial),[70]
respectively; the Rheasilvia impact basin interior (in the south) is dark blue, and neighboring areas of Rheasilvia ejecta (including an area within Veneneia) are light purple-blue; [72][73] areas modified by more recent impacts or mass wasting are yellow/orange or green, respectively. Main articles: Rheasilvia and VeneneiaNorthern (left) and southern
(right) hemispheres. The "Snowman" craters are at the top of the left image; Rheasilvia and Veneneia (green and blue) dominate the right. Parallel troughs are seen in both. Colors of the two hemispheres are not to scale, [i] and the equatorial region is not shown. South pole of Vesta, showing the extent of Rheasilvia crater. The most prominent of these
surface features are two enormous impact basins, the 500-kilometre-wide (311mi) Rheasilvia, centered near the south pole; and the 400-kilometre-wide (249mi) Veneneia. The Rheasilvia impact basins, the 500-kilometre-wide (311mi) Rheasilvia, after the mother of Romulus
and Remus and a mythical vestal virgin.[75] Its width is 95% of the mean diameter of Vesta. The crater floor and the highest measured part of the crater rim is 31km (19mi) above the crater floor low point. It is estimated that the impact
responsible excavated about 1% of the volume of Vesta, and it is likely that the Vesta family and V-type asteroids are the products of this collision. If this is the case, then the fact that 10km (6mi) fragments have survived bombardment until the present indicates that the crater is at most only about 1 billion years old. [76] It would also be the site of
origin of the HED meteorites. All the known V-type asteroids taken together account for only about 6% of the ejected volume, with the rest presumably either in small fragments, ejected by approaching the 3:1Kirkwood gap, or perturbed away by the Yarkovsky effect or radiation pressure. Spectroscopic analyses of the Hubble images have shown that
this crater has penetrated deep through several distinct layers of the crust, and possibly into the mantle, as indicated by spectral signatures of olivine. [58] Subsequent analysis of data from the Dawn mission provided much greater detail on Rheasilvia's structure and composition, confirming it as one of the largest impact structures known relative to
its parent body size.[74] The impact clearly modified the pre-existing very large, Veneneia structure, indicating Rheasilvia's younger age.[74] Rheasilvia's size makes Vesta's southern topography unique, creating a flattened southern hemisphere and contributing significantly to the asteroid's overall oblate shape.[69] Rheasilvia's ~22km (14mi) central
peak stands as one of the tallest mountains identified in the Solar System.[74] Its base width of roughly 180km (110mi) and complex morphology distinguishes it from the simpler central peaks seen in smaller craters.[77] Numerical modeling indicates that such a large central structure within a ~505km (314mi) diameter basin requires formation on a
 differentiated body with significant gravity. Scaling laws for craters on smaller asteroids fail to predict such a feature; instead, impact dynamics involving transient crater collapse and rebound of the underlying material (potentially upper mantle) are needed to explain its formation. [77] Hydrocode simulations suggest the impactor responsible was
likely 6070km (3743mi) across, impacting at roughly 5.4 km/s.[78] Models of impact angle (around 30-45 degrees from vertical) better match the detailed morphology of the basin and its prominent peak.[77] Crater density measurements on Rheasilvia's relatively unmodified floor materials and surrounding ejecta deposits, calibrated using standard
lunar chronology functions adapted for Vesta's location, place the impact event at approximately 1 billion years ago. [79][70] This age makes Rheasilvia a relatively young feature on a protoplanetary body formed early in Solar System history. The estimated excavation of ~1% of Vesta's volume [74] provides a direct link to the Vesta family of asteroids
(Vestoids) and the HED meteorites. Since Vesta's spectral signature matches that of the Vestoids and HEDs, this strongly indicates they are fragments ejected from Vesta most likely during the Rheasilvia impact. [27][79] The Dawn mission's VIR mapping
revealed spectral variations across the basin consistent with the mixing of different crustal layers expected in the HED meteorites. Signatures matching eucrites (shallow crustal basalts) and diogenites (deeper crustal orthopyroxenites) were identified, which usually correlate with specific morphological features like crater walls or slump blocks.[80]
[27] The confirmed signature of olivine-rich material, which were first hinted at by Hubble observations is strongest on the flanks of the central peak and in specific patches along the basin rim and walls, suggesting it is not uniformly distributed but rather exposed in distinct outcrops. [81] [80] As the dominant mineral expected in Vesta's mantle
beneath the HED-like crust, [10] the presence of olivine indicates the Rheasilvia impact penetrated Vesta's entire crust (~2040km (1225mi) thick in the region) and excavated material from the upper mantle. [81] Furthermore, the global stresses resulting from this massive impact are considered the likely trigger for the formation of the large trough
systems, like Divalia Fossa, that encircle Vesta's equatorial regions.[82][69]The crater AeliaFeralia Planitia, an old, degraded impact basin or impact basin 
although none are quite so large. They include Feralia Planitia, shown at right, which is 270km (168mi) across.[83] More-recent, sharper craters range up to 158km (98mi) Varronilla and 196km (122mi) Postumia.[84]Dust fills up some craters range up to 158km (98mi) Varronilla and 196km (122mi) Postumia.
without a significant atmosphere. These are smooth deposits of dust accumulated in depressions on the surface of Vesta, we have identified both type 1 (formed from impact melt) and type 2 (electrostatically made) dust ponds within 030N/S, that is,
Equatorial region. 10craters have been identified with such formations. [86] The "snowman craters" are a group of three adjacent craters in Vesta's northern hemisphere. Their official names, from largest to smallest (west to east), are Marcia, Calpurnia, and Minucia is the youngest and cross-cuts Calpurnia. Minucia is the oldest.
[70]"Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" cratersThe majority of the equatorial region of Vesta is sculpted by a series of parallel troughs designated Divalia Fossae; its longest trough in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed image of the "Snowman" craters by Dawn from 5,200km (3,200mi) in 2011Detailed 
 seventh the size of the Moon, Divalia Fossae dwarfs the Grand Canyon. A second series, inclined to the equator, is found further north. This northern trough system is named Saturnalia Fossae, with its largest trough to be large-scale graben resulting from the
impacts that created Rheasilvia and Veneneia craters, respectively. They are some of the longest chasms in the Solar System, nearly as long as Ithaca Chasma on Tethys. The troughs may be graben that formed after another asteroid collided with Vesta, a process that can happen only in a body that is differentiated, [82] which Vesta may not fully be.
Alternatively, it is proposed that the troughs may be radial sculptures created by secondary cratering from Rheasilvia. [87] A section of Divalia Fossae Compositional information from the visible and infrared spectrometer (VIR), gamma-ray and neutron
detector (GRaND), and framing camera (FC), all indicate that the majority of the surface composition of Vesta is consistent with the composition of the howardite, eucrite, and diogenite meteorites. [88][89][90] The Rheasilvia region is richest in diogenite meteorites.
```

presence of olivine within the Rheasilvia region would also be consistent with excavation of mantle material. However, olivine has only been detected in localized regions of the northern hemisphere, not within Rheasilvia. [32] The origin of this olivine is currently unclear. Though olivine was expected by astronomers to have originated from Vesta's mantle prior to the arrival of the Dawn orbiter, the lack of olivine within the Rheasilvia and Veneneia impact basins excavated Vestian material down to 60100km, far deeper than the expected thickness of ~3040km for Vesta's crust. Vesta's crust may be far thicker than expected or the violent impact events

```
that created Rheasilvia and Veneneia may have mixed material enough to obscure olivine from observations. Alternatively, Dawn observations of olivine could instead be due to delivery by olivine-rich impactors, unrelated to Vesta's internal structure. [91] Pitted terrain has been observed in four craters on Vesta: Marcia, Cornelia, Numisia and Licinia
[92] The formation of the pitted terrain is proposed to be degassing of impact-heated volatile-bearing material. Along with the pitted terrain, curvilinear gullies are found in Marcia and Cornelia craters. The curvilinear gullies end in lobate deposits, which are sometimes covered by pitted terrain, and are proposed to form by the transient flow of liquic
water after buried deposits of ice were melted by the heat of the impacts.[71] Hydrated materials have also been detected, many of which are associated with areas of dark material is thought to be largely composed of carbonaceous chondrite, which was deposited on the surface by impacts. Carbonaceous chondrites
are comparatively rich in mineralogically bound OH.[90]Cut-away schematic of Vestan core, mantle, and crustEucrite meteorites (Vestan achondrites), giving insight into Vesta's geologic history and structure. NASA Infrared Telescope
Facility (NASA IRTF) studies of asteroid (237442) 1999 TA10 suggest that it originated from deeper within Vesta than the HED meteorites.[94]Vesta is thought to consist of a metallic ironnickel core, variously estimated to be 90km (56mi)[63] to 220km (140mi)[10] in diameter, an overlying rocky olivine mantle, with a surface crust of similar
composition to HED meteorites. From the first appearance of calciumaluminium-rich inclusions (the first solid matter in the Solar System, forming about 4.567billion years ago), a likely time line is as follows: [95][96][97][98][99]Timeline of the evolution of Vesta23million years Accretion completed 45million years Complete melting
due to radioactive decay of 26Al, leading to separation of the metal core67million yearsProgressive crystallization of a convection stopped when about 80% of the material had crystallization of the remaining molten matter. Convection stopped when about 80% of the material had crystallization of a convection stopped when about 80% of the material had crystallization of the remaining molten material to form the crust, either as basaltic lavas in progressive eruptions, or possibly forming a
short-lived magma ocean. The deeper layers of the crust crystallize to form plutonic rocks, whereas older basalts are metamorphosed due to the pressure of newer surfaced in this manner. Because of this, some scientists refer to Vesta as a protoplanet.
[100]Composition of the Vestan crust (by depth)[101]A lithified regolith, the source of howardites and brecciated eucrites. Plutonic rocks consisting of pyroxene with large grain sizes, the source of
 diogenites.On the basis of the sizes of V-type asteroids (thought to be pieces of Vesta's crust ejected during large impacts), and the depth of Rheasilvia crater (see below), the crust is thought to be roughly 10 kilometres (6mi) thick.[102]Findings from the Dawn spacecraft have found evidence that the troughs that wrap around Vesta could be graben
formed by impact-induced faulting (see Troughs section above), meaning that Vesta has more complex geology than other asteroids. The impacts that created the Rheasilvia and Veneneia craters occurred when Vesta was no longer warm and plastic enough to return to an equilibrium shape, distorting its once rounded shape and prohibiting it from
being classified as a dwarf planet today.[citation needed]Vesta's surface is covered by regolith distinct from that found on the Moon or asteroids such as Itokawa. This is because space weathering acts differently. Vesta's surface shows no significant trace of nanophase iron because the impact speeds on Vesta are too low to make rock melting and
vaporization an appreciable process. Instead, regolith evolution is dominated by brecciation and subsequent mixing of bright and dark component is the original Vesta basaltic soil.[104]Some small Solar System bodies are suspected to be
fragments of Vesta caused by impacts. The Vestian asteroids and HED meteorites are examples. The V-type asteroid 1929 Kollaa has been determined to have a composition akin to cumulate eucrite meteorites, indicating its origin deep within Vesta's crust. [28] Vesta is currently one of only eight identified Solar System bodies of which we have
physical samples, coming from a number of meteorites suspected to be Vestan fragments. It is estimated that 1 out of 16 meteorites from Mars, meteorites from the Moon, and samples returned from the Moon, the comet Wild 2, and the asteroids
25143 Itokawa, 162173 Ryugu, and 101955 Bennu. [29][k] Animation of Dawn's trajectory from 27 September 2007 to 5 October 2018 Dawn Earth Mars 4 Vesta 1 CeresFirst image of asteroids (Ceres and Vesta) taken from Mars. The image was made by the Curiosity rover on 20 April 2014. Animation of Dawn's trajectory around 4 Vesta from 15 July
2011 to 10 September 2012 Dawn 4 VestaIn 1981, a proposal for an asteroid mission was submitted to the European Space Agency (ESA). Named the Asteroidal Gravity Optical and Radar Analysis (AGORA), this spacecraft was to launch some time in 19901994 and perform two flybys of large asteroids. The preferred target for this mission was Vesta
AGORA would reach the asteroid belt either by a gravitational slingshot trajectory past Mars or by means of a small ion engine. However, the proposal was refused by the ESA. A joint NASAESA asteroid mission was then drawn up for a Multiple Asteroid Orbiter with Solar Electric Propulsion (MAOSEP), with one of the mission profiles including an
orbit of Vesta. NASA indicated they were not interested in an asteroid belt were proposed in the 1980s by France, Germany, Italy and the United States, but none were approved.[106] Exploration of Vesta by fly-by and impacting
penetrator was the second main target of the first plan of the multi-aimed Soviet Vesta mission, developed in cooperation with European countries for realisation in 19911994 but canceled due to the dissolution of the Soviet Union. Artist's conception of Dawn orbiting VestaIn the early 1990s, NASA initiated the Discovery Program, which was intended
to be a series of low-cost scientific missions. In 1996, the program's study team recommended a mission to explore the asteroid belt using a spacecraft with an ion engine as a high priority. Funding for this program remained problematic for several years, but by 2004 the Dawn vehicle had passed its critical design review[107] and construction
proceeded.[citation needed]It launched on 27 September 2007 as the first space mission to Vesta. On 3 May 2011, NASA confirmed that it received telemetry from Dawn indicating that the spacecraft successfully entered Vesta's orbit.[109]
It was scheduled to orbit Vesta for one year, until July 2012.[110] Dawn's arrival coincided with late summer in the southern hemisphere of Vesta, with the large crater at Vesta's south pole (Rheasilvia) in sunlight. Because a season on Vesta lasts eleven months, the northern hemisphere, including anticipated compression fractures opposite the crater,
would become visible to Dawn's cameras before it left orbit.[111] Dawn left orbit around Vesta on 4 September 2012 11:26 p.m. PDT to travel to Ceres.[112]NASA/DLR released imagery and summary information from a survey orbit, two high-altitude orbits (6070m/pixel) and a low-altitude mapping orbit (20m/pixel), including digital terrain models,
videos and atlases.[113][114][115][116][117][118] Scientists also used Dawn to calculate Vesta's precise mass and gravity field. The subsequent determination of the J2 component yielded a core diameter estimate of about 220km (140mi) assuming a crustal density similar to that of the HED.[113]Dawn data can be accessed by the public at the UCLA
website.[119]Albedo and spectral maps of 4 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 4 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 4 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 4 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 4 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 4 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 4 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 4 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 4 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 4 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 5 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 6 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 8 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 8 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 8 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 8 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 8 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 8 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 8 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 8 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 8 Vesta, as determined from Hubble Space Telescope images of May 1996Elevation map of 8 Vesta (May 1996Elevation map of 8 Vesta).
Rheasilvia crater at the south pole and Feralia Planitia near the equatorVesta seen by the Hubble Space Telescope in May 2007The 2006 IAU draft proposal on the definition of a planet listed Vesta as a candidate.[120] Vesta is shown fourth from the left along the bottom row. Vesta comes into view as the Dawn spacecraft approaches and enters
orbit:Vesta from 100,000km(1 July 2011) From 5,200km(23 July 2011) In orbit from 5,200km(24 July 2011) In orbit from 5,200km(24 July 2011) In orbit from 10,500km(18 July 2011) In orbit from 5,200km(24 July 2011) In orbit from 10,500km(18 July 2011) In orbit from 5,200km(24 July 2011) In orbit from 10,500km(18 July 2011) In orbit from 5,200km(24 July 20
hills and ridges(6 August 2011)Densely cratered terrain near terminator(6 August 2011)Vestan craters in various states of degradation, with troughs at bottom(6 August 2011)Densely cratered terrain near terminator(6 August 2011)Vestan craters in various states of degradation, with troughs at bottom(6 August 2011)Densely cratered terrain near terminator(6 August 2011)Densely cratered terrain nea
orbits are available on the Dawn Mission website of JPL/NASA.[121]Annotated image from Earth's surface make Vesta the bright s
+5.4, the brightest since 1989.[122] At that time, opposition, reaching a magnitude of +5.3.[124]Less favorable opposition, reaching a magnitude of from +6.5 to +7.3.[125] Even when in
conjunction with the Sun, Vesta will have a magnitude around +8.5; thus from a pollution-free sky it can be observed with binoculars even at elongations much smaller than near opposition. [125] a brightness that makes it visible in
binocular range but generally not for the naked eye. Under perfect dark sky conditions where all light pollution is absent it might be visible to an experienced observer without the use of a telescope or binoculars. Vesta came to opposition again on 5 August 2011, in the constellation of Capricornus at about magnitude 5.6.[126][127]Vesta was at
opposition again on 9 December 2012.[128] According to Sky and Telescope magazine, this year Vesta came within about 6 degrees of 1 Ceres during the winter of 2012 and spring 2013.[129] Vesta orbits the Sun in 3.63 years and Ceres in 4.6 years, so every 17.4 years Vesta overtakes Ceres (the previous overtaking was in April 1996).[129] On 1
December 2012, Vesta had a magnitude of 6.6, but it had decreased to 8.4 by 1 May 2013.[129]Conjunction of Ceres and Vesta came within one degree of each other in the night sky in July 2014.[129]3103 Eger 3551 Verenia 3908 Nyx 4055 Magellan Asteroids in
fictionDiogeniteEucriteList of former planetsHowarditeVesta family (vestoids)List of tallest mountains in the Solar System^ Marc Rayman of the JPL Dawn team used "Vestian" (analogous to the Greek cognate Hestian) a few times in 2010 and early 2011 in his Dawn Journal, and the Planetary Society continued to use that form for a few more years.
[2] The word had been used elsewhere, e.g. in Tsiolkovsky (1960) The call of the cosmos. However, otherwise the shorter form "Vestan" has been used by JPL.[3] Most modern print sources also use "Vestan virgins, not to Vesta herself.^
Calculated using the known dimensions assuming an ellipsoid. Academic Calculated using (1) the known rotation period (5.342h)[6] and (2) the equatorial radius Req (285km)[10] of the best-fit biaxial ellipsoid to Asteroid 4 Vesta. A b topocentric coordinates computed for the selected location: Greenwich, United Kingdom[14] On 10 February 2009, during
Ceres perihelion, Ceres was closer to the Sun than Vesta, because Vesta has an aphelion distance greater than Ceres's perihelion distance. (10 February 2009: Vesta 2.56AU; Ceres 2.54AU)^ wist is the closest Chinese approximation of the Latin pronunciation westa. Some sources contemporaneous to Gauss invented more elaborate forms, such as
and .[43][44] A simplification of the latter from c.1930, [45] never caught on. This symbol can be seen in the top of the most elaborate of the earlier forms, . It dates from 1973, at the beginning of astrological interest in asteroids.
elemental abundances, topographic profiles, gravity fields, and mapping of remnant magnetism, if any.[53]^ that is, blue in the north does not mean the same thing as blue in the south.^ Note that 6 Hebe may be the parent body for H chondrites, one of the most common meteorite types.^ "Vesta". Dictionary.com Unabridged (Online). n.d.^ "Search
Results". Planetary Society. Archived from the original on 5 March 2016. Meteoritics & planetary science, Volume 42, Issues 68, 2007; Origin and evolution of Earth, National Research Council et al., 2008 E.g in Meteoritics & planetary science
(volume 42, issues 68, 2007) and Origin and evolution of Earth (National Research Council et al., 2008). a b c d e f g h "JPL Small-Body Database Browser: 4 Vesta". Archived from the original on 26 September 2021. Retrieved 1 June 2008. Souami, D.; Souchay, J. (July 2012). "The solar system's invariable plane". Astronomy & Astrophysics. 543: 11
 Bibcode:2012A&A...543A.133S. doi:10.1051/0004-6361/201219011. A133.^ "Horizons Batch for 4 Vesta on 2021-Dec-26" (Perihelion occurs when rdot flips from negative to positive). JPL Horizons. Retrieved 26 September 2021. (Epoch 2021-Jul-01/Soln.date: 2021-Apr-13)^ "AstDyS-2 Vesta Synthetic Proper Orbital Elements". Department of
Mathematics, University of Pisa, Italy. Retrieved 1 October 2011.^ a b c d e f g h i j Russell, C. T.; etal. (2012). "Dawn at Vesta: Testing the Protoplanetary Paradigm" (PDF). Science. 336 (6082): 684686. Bibcode: 2012Sci...336..684R. doi:10.1126/science.1219381. PMID22582253. S2CID206540168.^ "surface ellipsoid 286.3x278.6x223.2". Wolfram
Alpha: Computational Knowledge Engine. Konopliv, A. S.; et al. (2014). "The Vesta gravity field, spin pole and rotation period, landmark positions, and ephemeris from the Dawn tracking and optical data". Icarus. 240: 118132. Bibcode: 2014Icar. 240..103K. doi:10.1016/j.icarus.2013.09.005. PDF copy Harris, A. W. (2006). Warner, B. D.; Pravec, P.
(eds.). "Asteroid Lightcurve Derived Data. EAR-A-5-DDR-DERIVED-LIGHTCURVE-V8.0". NASA Planetary Data System. Archived from the original on 9 April 2009. Retrieved 26 December 2013. "Asteroid 4 Vesta". The SkyLive. Retrieved 13 December 2022. a b Tedesco, E. F.; Noah, P. V.; Noah, M.; Price, S. D. (2004). "Infra-Red Astronomy Satellite
(IRAS) Minor Planet Survey. IRAS-A-FPA-3-RDR-IMPS-V6.0". NASA Planetary Data System. Archived from the original on 9 April 2009. Retrieved 15 March 2007. Nasa Planetary Data System. Archived (PDF) from the original on 11 September 2005. Neese
C.; Ed. (2005). "Asteroid Taxonomy EAR-A-5-DDR-TAXONOMY-V5.0". NASA Planetary Data System. Archived from the original on 10 March 2007. Retrieved 25 December 2013. a b Menzel, Donald H. & Pasachoff, Jay M. (1983). A Field Guide to the Stars and Planets (2nded.). Boston, MA: Houghton Mifflin. p.391. ISBN 978-0-395-34835-2. "In
Depth - 4 Vesta". NASA Solar System Exploration. 10 November 2017. Retrieved 13 December 2022. "Dawn Mission Overview". NASA. Archived from the original on 16 October 2011. Retrieved 13 December 2017. Retrieved 13 December 2011. "Lang, Kenneth (2011). The Cambridge Guide to the Solar System. Cambridge University Press. pp.372, 442. ISBN 9780521198578."
Russell, C. T.; etal. (2011). "Exploring the smallest terrestrial planet: Dawn at Vesta" (PDF). EPSC Abstracts. 2011 EPSC-DPS Joint Meeting. Vol.6. EPSC-DP
(PDF). Solar System Research. 39 (3): 176186. Bibcode: 2005SoSyR...39..176P. doi:10.1007/s11208-005-0033-2. S2CID120467483. Archived from the original (PDF) on 31 October 2008. Amos, Jonathan (11 May 2012). "Asteroid Vesta is 'last of a kind' rock". BBC News. Jutzi, M.; E. Asphaug; P. Gillet; J.-A. Barrat; W. Benz (14 February 2013). "The
structure of the asteroid 4 Vesta as revealed by models of planet-scale". Nature. 494 (7436): 207210. Bibcode:2013Natur.494..207J. doi:10.1038/nature11892. PMID23407535. S2CID4410838.^ Cook, Jia-Rui. "Dawn Reality-Checks Telescope Studies of Asteroids". Archived from the original on 2 May 2014. Retrieved 30 April 2014.^ a b c McSween, H.
Y.; R. P. Binzel; M. C. De Sanctis; etal. (27 November 2013). "Dawn; the Vesta-HED connection; and the geologic context for eucrite, diogenites, and howardites". Meteoritics & Planetary Science. 48 (11): 2090214. Bibcode: 2013M&PS...48.2090M. doi:10.1111/maps.12108. S2CID131100157.^ a b Kelley, M. S.; etal. (2003). "Quantified mineralogical
evidence for a common origin of 1929 Kollaa with 4 Vesta and the HED meteorites". Icarus. 165 (1): 215218. Bibcode: 2003Icar..165..215K. doi:10.1016/S0019-1035(03)00149-0.^ a b "Vesta". NASA/JPL. 12 July 2011. Archived from the original on 29 June 2011. Retrieved 30 July 2011.^ "Ceres, Pallas, Vesta, and Hygiea". Gravity Simulator. Archived
from the original on 17 June 2008. Retrieved 31 May 2008. Retrieved 6 September 2012. a b Ammannito, E.; M. C. De Sanctis; E. Palomba; etal. (2013). "Olivine in an unexpected location on Vesta's surface". Nature. 504 (7478): 122125.
Bibcode: 2013 Natur. 504..122A. doi:10.1038/nature12665. PMID24196707. S2CID4464889.^ Cook, Jia-Rui. "It's Complicated: Dawn Spurs Rewrite of Vesta's Story". Archived from the original on 2 May 2014. Retrieved 30 April 2014.^ Littmann, Mark (2004). Planets Beyond: Discovering the Outer Solar System. Dover Books on Astronomy. Courier
Dover Publications. p.21. ISBN 978-0-486-43602-9. Lynn, W. T. (February 1907). "The discovery of Vesta". The Observatory. 30: 103105. Bibcode:1907Obs....30..103L. Dunnington, Guy Waldo; Gray, Jeremy; Dohse, Fritz-Egbert (2004). Carl Friedrich Gauss: Titan of Science. The Mathematical Association of America. p.76. ISBN 978-0-88385-547-8.
Rao, K. S.; Berghe, G. V. (2003). "Gauss, Ramanujan and Hypergeometric Series Revisited". Historia Scientiarum. 13 (2): 123133. Schmadel, Lutz D. (2003). Dictionary of Minor Planet Names: Prepared on Behalf of Commission 20 Under the Auspices of the International Astronomical Union. Springer. p.15. ISBN 978-3-540-00238-3. von Zach, Franz
Xaver (1807). Monatliche correspondenz zur befrderung der erd- und himmels-kunde. Vol.15. p.507.^ Carlini, Francesco (1808). Effemeridi astronomiche di Milano per l'anno 1809.^ a b Bala, Gavin Jared; Miller, Kirk (18 September 2023). Unicode request for historical asteroid symbols (PDF) (Report). Unicode Consortium. Retrieved 26 September 2023).
2023 via unicode.org. "Proposed new characters". unicode.org. The Pipeline. Unicode Consortium. Retrieved 6 November 2023. Bureau des longitudes (1807). Annuaire pour l'an 1808. p.5. Canovai, Stanislao; del-Ricco, Gaetano (1810). Elementi di fisica matematica. p.149. Koch, Rudolf (1955) [1930]. The Book of Signs (reprinted.). Dover.
Bach, Eleanor (1973). Ephemerides of the asteroids: Ceres, Pallas, Juno, Vesta, 19002000. Celestial Communications. Bibcode:1973eacp.book.....B. Wells, David A. (1851). Bliss, George Jr. (ed.). "The Planet Hygiea". Annual of Scientific Discovery for the year 1850, quoted by spaceweather.com archives, 2006-09-13. Retrieved 1 June 2008. Hilton.
James L. When Did the Asteroids Become Minor Planets? (Report). Archived from the original on 24 March 2008 via U.S. Naval Observatory website, particularly the discussion of Gould, B. A. (1852). "On the Symbolic Notation of the Asteroids". Astronomical Journal. 2: 80. Bibcode:1852AJ.....2...80G. doi:10.1086/100212. and immediate subsequent
 history. Also, the discussion of C. J. Cunningham (1988) Introduction to Asteroids explains the parenthetical part. "New SPHERE view of Vesta". www.eso.org. Retrieved 25 June 2018. McFadden, L. A.; Emerson, G.; Warner, E. M.; Onukwubiti, U.; Li, J.-Y. "Photometry of 4 Vesta from its 2007 Apparition". Proceedings, 39th Lunar and Planetary
Science Conference. League City, Texas. Bibcode: 2008LPI....39.2546M. 1014 March 2008^ Hughes, D. W. (September 1994). "The Historical Unravelling of the Diameters of the First Four Asteroids". Quarterly Journal of the Royal Astronomical Society. 35 (3): 331. Bibcode: 1994QJRAS...35...331H. Povenmire, H. (September 2001). "The January 4.
1991 Occultation of SAO 93228 by Asteroid (4) Vesta". Meteoritics & Planetary Science. 36 (Supplement): A165. Bibcode: 2001M&PSA..36Q.165P. doi:10.1111/j.1945-5100.2001.tb01534.x.^ a b "Dawn-NASA-NSSDCA-Spacecraft-Details". Archived from the original on 23 May 2022. Retrieved 16 December 2022.^ Hertz, Hans G. (19 April 1968). "Massacraft-Details".
of Vesta". Science. 160 (3825): 299300. Bibcode:1968Sci...160...299H. doi:10.1126/science.160..3825.299. PMID17788233. S2CID2555834.^ Kovaevi, A. (January 2005). "Determination of the mass of (4) Vesta based on new close approaches". Astronomy and Astrophysics. 430 (1): 319325. Bibcode:2005A&A...430...319K. doi:10.1051/0004-
6361:20035872.^ Christou, A. A. (2000). "Co-orbital objects in the main asteroid belt". Astronomy and Astrophysics. 356: L71 L74. Bibcode:2000A&A...356L..71C.^ a b Christou, A. A.; Wiegert, P. (January 2012). "A population of Main Belt Asteroids co-orbiting with Ceres and Vesta". Icarus. 217 (1): 2742. arXiv:1110.4810.
Bibcode:2012Icar..217...27C. doi:10.1016/j.icarus.2011.10.016. ISSN0019-1035. S2CID59474402.^ a b c Thomas, P. C.; etal. (1997). "Vesta: Spin Pole, Size, and Shape from HST Images". Icarus. 128 (1): 8894. Bibcode:1997Icar..128...88T. doi:10.1006/icar.1997.5736.^ a b Hand, Eric (2012). "Space missions trigger map wars". Nature. 488 (7412)
442443. Bibcode:2012Natur.488..442H. doi:10.1038/488442a. PMID22914145.^ "IAU WGCCRE Coordinate System for Vesta | USGS Astrogeology Science Center". Astrogeology Science Center". Astrogeology.usgs.gov. 15 November 2013. Retrieved 25 June 2014.^ Li, Jian-Yang; Mafi, Joseph N. "Body-Fixed Coordinate Systems for Asteroid (4) Vesta" (PDF). Planetary Data System
Archived (PDF) from the original on 6 November 2015. Astronomy. 100 (1): 2742. Bibcode: 2008CeMDA.100...27B. doi:10.1007/s10569-007-9103-8. Astronomy. 100 (1): 2742. Bibcode: 2008CeMDA.100...27B. doi:10.1007/s10569-007-9103-8. Astronomy. 100 (1): 2742. Bibcode: 2008CeMDA.100...27B. doi:10.1007/s10569-007-9103-8.
Vaughan, A. T.; Rambaux, N.; Bills, B. G.; Castillo-Rogez, J. C.; Fu, R. R.; Jacobson, S. A.; Stewart, S. T.; Toplis, M. J. (23 April 2025). "A small core in Vesta inferred from Dawn's observations". Nature Astronomy: 111. doi:10.1038/s41550-025-02533-7. ISSN2397-3366. Carry, B.; et al. (2009). "Physical properties of (2) Pallas". Icarus. 205 (2): 460472
arXiv:0912.3626v1. Bibcode:2010Icar..205..460C. doi:10.1016/j.icarus.2009.08.007. S2CID119194526. "The IAU draft definition of "planet" and "plutons"". IAU. August 2006. Archived from the original on 9 January 2010. Retrieved 16 December 2009. (XXVI) Fu, R. R.; Hager, B. H.; Ermakov, A.I.; Zuber, M.T. (2013). "Early Viscous Relaxation of "planet" and "plutons".
Asteroid Vesta and Implications for Late Impact-Driven Despinning" (PDF). 44th Lunar and Planetary Science Conference (1719): 2115. Bibcode: 2013LPI....44.2115F. Archived (PDF) from the original on 26 July 2014. Asmar, S. W.; Konopliv, A. S.; et al. (2012). "The Gravity Field of Vesta and Implications for Interior Structure" (PDF).
43rd Lunar and Planetary Science Conference (1659): 2600. Bibcode: 2012LPI....43.2600A. Archived (PDF) from the original on 20 October 2013. Sellner, N. E. B.; Gibbard, S.; de Pater, I.; etal. (2005). "Near-IR imaging of Asteroid 4 Vesta" (PDF). Icarus. 177 (1): 190195. Bibcode: 2005Icar...177...190Z. doi:10.1016/j.icarus.2005.03.024. Archived from
the original (PDF) on 23 November 2008.^ a b c Jaumann, R.; etal. (2012). "Vesta's Shape and Morphology". Science. 336 (6082): 687690. Bibcode: 2012Sci...336..687J. doi:10.1126/science.1219122. PMID22582254. S2CID206540010.^ a b c d Williams, D. A.; Yingst, R. A.; Garry, W. B. (December 2014). "Introduction: The geologic mapping of Vesta"
Icarus. 244: 112. Bibcode:2014Icar...244....1W. doi:10.1016/j.icarus.2014.03.001. hdl:2286/R.I.28071.Williams, D. A.; etal. (December 2014). "The geology of the Marcia quadrangle of asteroid Vesta: Assessing the effects of large, young craters". Icarus. 244: 7488. Bibcode:2014Icar...244....74W. doi:10.1016/j.icarus.2014.01.033. hdl:2286/R.I.28061
July 2018. Scully, J. E. C.; etal. (2014). "Sub-curvilinear gullies interpreted as evidence for transient water flow on Vesta Geologic mapping of
quadrangles Gegania and Lucaria" (PDF). Icarus. 244: 6073. Bibcode: 2014Icar...244...60S. doi:10.1016/j.icarus.2014.06.026. hdl:2286/R.I.28060. Archived from the original (PDF) on 28 April 2019. Retrieved 24 September 2019. Netrieved 24 September 2019. Retrieved 24 September 2
Rheasilvia craters Geology of guadrangle Tuccia, Vesta". Icarus. 244: 133157. Bibcode: 2014Icar.. 244.. 133K. doi:10.1016/j.icarus.2014.02.012. hdl: 2286/R.I. 28058. a b c d e Schenk, P.; etal. (2012). "The Geologically Recent Giant Impact Basins at Vesta's South Pole". Science. 336 (6082): 694697. Bibcode: 2012Sci... 336... 694S.
Icarus. 128 (1): 95103. Bibcode:1997Icar..128...95B. doi:10.1006/icar.1997.5734.^ a b c Ivanov, B. A.; Melosh, H. J. (2013). "Rheasilvia impact basin on Vesta: Constraints on formation models from the central peak topography". Journal of Geophysical Research: Planets. 118 (7): 15451555. Bibcode:2013JGRE..118.1545I. doi:10.1002/jgre.20108.
 Bowling, T. J.; Richard, G.; Melosh, H. J. (2013). "Numerical simulations of the Rheasilvia impact basin on Vesta". Journal of Geophysical Research: Planets. 118 (8): 16221639. Bibcode: 2013JGRE..118.1622B. doi:10.1002/jgre.20113. a b Marchi, S.; McSween, H. Y.; O'Brien, D. P. (2012). "The Violent Collisional History of Vesta". Science. 336 (6082)
Vol.14. p.5711. Bibcode: 2012EGUGA..14.5711G. EGU2012-5711-1. Archived (PDF) from the original on 8 April 2014. "Nomenclature Search Results / Target: VESTA / Feature Type: Crater, craters". Gazetteer of Planetary Nomenclature Search Results / Target: VESTA / Feature Type: Crater, craters". Sky and Telescope. Retrieved 18 October 2023
via skyandtelescope.org. To geologists' surprise, the asteroid Eros has more than 250'ponds' thought to contain compacted deposits of finely ground dust. Parekh, R.; Otto, K.A.; Matz, K.D.; Jaumann, R.; Krohn, K.; Roatsch, T.; etal. (28 February 2022) [1 November 2021]. "Formation of ejecta and dust pond deposits on asteroid Vesta". Journal of
Geophysical Research: Planets. 126 (11): e2021JE006873. doi:10.1029/2021JE006873. ^ Hirata, N. (2023). "Secondary Cratering From Rheasilvia as the Possible Origin of Vesta's Equatorial Troughs". Journal of Geophysical Research: Planets. 128 (3). arXiv:2303.14955. Bibcode:2023JGRE..12807473H. doi:10.1029/2022JE007473.
hdl:20.500.14094/0100482053. Retrieved 4 March 2024.^ De Sanctis, M. C.; etal. (2012). "Spectroscopic Characterization of Mineralogy and Its Diversity Across Vesta". Science. 336 (6082): 697700. Bibcode:2012Sci...336..697D. doi:10.1126/science.1219270. PMID22582257. S2CID11645621.^ Prettyman, T. H.; etal. (2012). "Elemental Mapping by
Science. 338 (6104): 246249. Bibcode:2012Sci...338..246D. CiteSeerX10.1.1.656.1476. doi:10.1126/science.1225374. PMID22997131. S2CID22892716.^ De Sanctis, M. C.; etal. (2012). "Detection of Widespread Hydrated Materials on Vesta by the vir Imaging Spectrometer on Board Thedawnmission". The Astrophysical Journal Letters. 758 (2): L36
doi:10.1006/icar.1998.5956. Righter, K.; Drake, M. J. (1997). "A magma ocean on Vesta: Core formation and petrogenesis of eucrites and diogenites". Meteoritics & Planetary Science. 32 (6): 929944. Bibcode:1997M&PS...32..929R. doi:10.1111/j.1945-5100.1997.tb01582.x. S2CID128684062. Drake, M. J. (2001). "The eucrite/Vesta story"
Meteoritics & Planetary Science. 36 (4): 501513. Bibcode:2001M&PS...36..501D. doi:10.1111/j.1945-5100.2001.tb01892.x. S2CID128394153. Sahijpal, S.; Soni, P.; Gagan, G. (2007). "Numerical simulations of the differentiation of accreting planetesimals with 26Al and 60Fe as the heat sources". Meteoritics & Planetary Science. 42 (9): 15291548
2011). "When Is an Asteroid Not an Asteroid?". NASA/JPL. Archived from the original on 29 June 2011. Retrieved 30 July 2011. Archived from the original on 29 June 2011. Retrieved 30 July 2011. Takeda, H. (1997). "Mineralogical records of early planetary processes on the HED parent body with reference to Vesta". Meteoritics & Planetary Science. 32 (6): 841853. Bibcode:1997M&PS...32..841T. doi:10.1111/j.1945
5100.1997.tb01574.x.^ Yamaguchi, A.; Taylor, G. J.; Keil, K. (1995). "Metamorphic History of the Eucritic Crust of 4 Vesta". Meteoritical Society. 30 (5): 603. Bibcode:1995Metic...30..603Y.^ Pieters, C. M.; Ammannito, E.; Blewett, D. T.; et al. (2012). "Distinctive space weathering on Vesta from regolith mixing processes". Nature. 491 (7422): 7982.
Marc (8 April 2015). Now Appearing At a Dwarf Planet Near You: NASA's Dawn Mission to the Asteroid Belt (Speech). Silicon Valley Astronomy Lectures. Foothill College, Los Altos, CA. Archived from the original on 14 November 2021. Retrieved 7 July 2018. Ulivi, Paolo; Harland, David (2008). Robotic Exploration of the Solar System: Hiatus and
Renewal, 19831996. Springer Praxis Books in Space Exploration. Springer. pp.117125. ISBN 978-0-387-78904-0. Russell, C. T.; Capaccioni, F.; Coradini, A.; et al. (October 2007 EM&P..101...65R. doi:10.1007/s11038-007-9151-9. S2CID 46423305
Archived (PDF) from the original on 27 September 2011. Retrieved 13 June 2011. Retrieved 13 June 2011. Retrieved 14 May 2011. Retrieved 14 May 2011. Retrieved 14 May 2011. Retrieved 14 May 2011. Retrieved 15 June 2011. "NASA'S Dawn Captures First Image of Nearing Asteroid". NASA/JPL. Archived from the original on 29 January 2012. Retrieved 14 May 2011. "NASA'S Dawn Captures First Image of Nearing Asteroid". NASA/JPL. Archived from the original on 29 January 2012. Retrieved 14 May 2011. "NASA'S Dawn Captures First Image of Nearing Asteroid". NASA/JPL. Archived from the original on 29 January 2012. Retrieved 13 June 2011. "NASA'S Dawn Captures First Image of Nearing Asteroid". NASA/JPL. Archived from the original on 29 January 2012. Retrieved 14 May 2011. "NASA'S Dawn Captures First Image of Nearing Asteroid". NASA/JPL. Archived from the original on 29 January 2012. Retrieved 14 May 2011. "NASA'S Dawn Captures First Image of Nearing Asteroid". NASA/JPL. Archived from the original on 29 January 2012. Retrieved 14 May 2011. "NASA/JPL. Archived from the original on 29 January 2012. Retrieved 15 June 2011. "NASA/JPL. Archived from the original on 29 January 2012. Retrieved 16 June 2011. "NASA/JPL. Archived from the original on 29 January 2012. Retrieved 18 June 2011. "NASA/JPL. Archived from the original on 29 January 2012. Retrieved 18 June 2011. "NASA/JPL. Archived from the original on 29 January 2012. Retrieved 18 June 2011. "NASA/JPL. Archived from the original on 29 January 2012. Retrieved 19 June 2011. "NASA/JPL. Archived from the original on 29 January 2012. Retrieved 19 June 2011. "NASA/JPL. Archived from the original on 29 June 2011. "NASA/JPL. Archived from the original on 29 June 2011. "NASA/JPL. Archived from the original on 29 June 2011. "NASA/JPL. Archived from the original on 29 June 2011. "NASA/JPL. Archived from the original on 29 June 2011. "NASA/JPL. Archived from the original on 29 June 2011. "NASA/JPL. Archived from the original on 20 June 2
Spacecraft Enters Orbit Around Asteroid Vesta". NASA. Archived from the original on 19 June 2012. Retrieved 17 July 2011. Dawn mission timeline Archived 19 October 2010). "Dawn Mission: Mission: Mission: Mission: Mission: Mission: Mission timeline Archived 19 October 2010)."
2011.^ "Dawn has Departed the Giant Asteroid Vesta". NASA JPL. NASA. 5 September 2012. Retrieved 5 September 2012. a b Russell, C. T.; etal. (2013). "Dawn completes its mission at 4 Vesta". Meteoritics & Planetary Science. 48 (11): 20762089. Bibcode: 2013M&PS...48.2076R. doi:10.1111/maps.12091.^ Roatsch, Thomas; etal. (2012). "High
resolution Vesta High Altitude Mapping Orbit (HAMO) Atlas derived from Dawn framing camera images". Planetary and Space Science. 73 (1): 283286. Bibcode:2012P&SS...73...283R. doi:10.1016/j.pss.2012.08.021.^ Roatsch, Thomas; et al. (2013). "High-resolution Vesta Low Altitude Mapping Orbit Atlas derived from Dawn Framing Camera images"
Planetary and Space Science. 85: 293298. Bibcode: 2013P&SS...85...293R. doi:10.1016/j.pss.2013.06.024.^ "NASA's Journey Above Vesta". DLR Institute of Planetary Research video with NASA JPL imagery. NASA. 16 September 2011. Archived from the original on 22 April 2021. Retrieved 18 September 2011.^ "DLR public Dawn products site".
Archived from the original on 16 October 2015. "NASA Dawn". "Dawn Public Data". Dawn [website]. Univ. California, Los Angeles. Retrieved 6 March 2015. "Gingerich, Owen (2006). "The Path to Defining Planets" (PDF). Dissertatio cum Nuncio Sidereo Ill. Tertia. Vol.VIII, no.16. pp.45. Archived (PDF) from the original on 15 March 2015.
Retrieved 13 March 2007. "Dawn Mission website of JPL/NASA". 22 October 2018. Bryant, Greg (2007). "Sky & Telescope: See Vesta at Its Brightest!". Retrieved 7 May 2007. Harrington, Philip S. (21 October 2010). Cosmic Challenge: The
Ultimate Observing List for Amateurs. Cambridge University Press. p.75. ISBN 9781139493680. a b James, Andrew (2008). "Vesta". Southern Astronomical Delights. Retrieved 6 November 2008. a b Yeomans, Donald K.; Chamberlin, Alan B. "Horizons Ephemeris". JPL Solar System Dynamics. Retrieved 9 January 2010. "Elements and Ephemeris".
for (4) Vesta". Minor Planet Center. Archived from the original on 4 March 2016. "2012 Astronomy Special". Nightskyonline.info. Archived from the original on 20 April 2013 Sky & Telescope. The Dawn Mission to Minor Planets 4 Vesta and 1 Ceres.
Christopher T. Russell and Carol A. Raymond (Editors), Springer (2011), ISBN 978-1-4614-4903-4Keil, K.; Geological History of Asteroid 4 Vesta: The Smallest Terrestrial Planet in Asteroids III, William Bottke, Alberto Cellino, Paolo Paolicchi, and Richard P. Binzel (editors), University of Arizona Press (2002), ISBN 0-8165-2281-2Wikimedia Common
has media related to Vesta (asteroid). This video explores Vesta's landscape, history and planet-like characteristics. Interactive 3D gravity simulation of the Dawn spacecraft in orbit around Vesta Archived 11 June 2020 at the Wayback Machine Vesta Trek An integrated map browser of datasets and maps for 4 VestaJPL Ephemeris Views of the Solar
System: VestaHubbleSite: Hubble Maps the Asteroid VestaEncyclopdia Britannica, Vesta full articleHubbleSite: short movie composed from Hubble Space Telescope images at ESA/Hubble Archived 22 January 2009 at the Wayback MachineDawn at Vesta from System: VestaHubbleSite: Short movie composed from Hubble Space Telescope images from November 1994. Adaptive optics views of Vesta from System:
(NASA press kit on Dawn's operations at Vesta)NASA video Archived 22 April 2021 at the Wayback MachineVesta at AstDyS-2, AsteroidsDynamic SiteEphemeris Observation prediction Orbital info Proper elements Observation Orbital info Proper 
parameters Physical parametersPortals: Stars Spaceflight Outer space ScienceRetrieved from " 4The following pages link to 4 Vesta External tools(link counttransclusion countsorted list) See help page for transcluding these entriesShowing 50 items. View (previous 50 | next 50) (20 | 50 | 100 | 250 | 500) Asteroid (links | edit) Apparent magnitude (links | edit) Apparent magnitud
 | edit)Comet (links | edit)Classical Kuiper belt object (links | edit)Deep Space 1 (links | edit)Earth (links | edit)Earth (links | edit)Galileo project (links | edit)Galileo project (links | edit)Human spaceflight (links | edit)Human spaceflight (links | edit)Moon (links | edit)Mercury
 (planet) (links | edit)Meteorite (links | edit)Plate tectonics (li
edit)Space exploration (links | edit)Space colonization (links | edit)Trans-Neptunian object (links | edit)Jupiter trojan (links | edit)Jupiter trojan (links | edit)Tau Ceti (links | edit)Trans-Neptunian object (links | edit)Tau Ceti (links | edit)Tau 
(links | edit)Callisto (moon) (links | edit)Europa (moon) (links | edit)Europa (moon) (links | edit)Europa (moon) (links | edit)Uiew (previous 50 | next 50) (20 | 50 | 100 | 250 | 500)Retrieved from "WhatLinksHere/4 Vesta"Enjoy sharper detail, more
accurate color, lifelike lighting, believable backgrounds, and more with our new model update. Your generated images will be more polished thanever. See What's NewExplore how consumers want to see climate stories told today, and what that means for yourvisuals. Download Our Latest VisualGPS ReportData-backed trends. Generative AI demos
Answers to your usage rights questions. Our original video podcast covers it allnow ondemand. Watch NowEnjoy sharper detail, more accurate color, lifelike lighting, believable backgrounds, and more with our new model update. Your generated images will be more polished than ever. See What's NewExplore how consumers want to see climate stories
told today, and what that means for yourvisuals. Download Our Latest VisualGPS ReportData-backed trends. Generative AI demos. Answers to your usage rights questions. Our original video podcast covers it allnow ondemand. Watch NowEnjoy sharper detail, more accurate color, lifelike lighting, believable backgrounds, and more with our new model
update. Your generated images will be more polished thanever. See What's NewExplore how consumers want to see climate stories told today, and what that means for yourvisuals. Download Our Latest VisualGPS ReportData-backed trends. Generative AI demos. Answers to your usage rights questions. Our original video podcast covers it allnow
ondemand. Watch Now Share copy and redistribute the material in any medium or format for any purpose, even commercially. The licensor cannot revoke these freedoms as long as you follow the license terms. Attribution You must give appropriate credit
provide a link to the license, and indicate if changes were made . You may do so in any reasonable manner, but not in any way that suggests the licenser endorses you or your use. ShareAlike If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. No additional restrictions You
may not apply legal terms or technological measures that legally restrict others from doing anything the license permits. You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation. No warranties are given. The license may not give you all of the
permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material. Computer science conceptThis article is about type systems in computer programming. For the formal study of type systems, see Type theory. This article includes a list of general references, but it
lacks sufficient corresponding inline citations. Please help to improve this article by introducing more precise citations. (October 2010) (Learn how and when to remove this message)This article is written like a personal feelings or presents an original
argument about a topic. Please help improve it by rewriting it in an encyclopedic style. (July 2016) (Learn how and when to remove this message) Type systems General concepts Type safety Strong vs. weak typing Major categories Static vs. dynamic Manifest vs. inferred Nominal vs. structural Duck typing Minor categories Abstract Dependent Flow-
sensitiveGradualIntersectionLatentRefinementSubstructuralUniqueSessionvteIn computer programming, a type system is a logical system comprising a set of rules that assigns a property called a type (for example, integer, floating point, string) to every term (a word, phrase, or other set of symbols). Usually the terms are various language constructs
of a computer program, such as variables, expressions, functions, or modules.[1] A type system determines the allowed values of that term. Type system determines the operations that can be performed on a term. For variables, the type system determines the programmer uses for algebraic data types, data
structures, or other data types, such as "string", "array of float", "function returning boolean". Type systems are often specified as part of programming languages and built into interpreters and compilers, although the type system of a language sand built into interpreters and compilers, although the type system of a language sand built into interpreters and compilers, although the type system of a language sand built into interpreters and compilers, although the type system of a language sand built into interpreters and compilers, although the type system of a language sand built into interpreters and compilers, although the type system of a language sand built into interpreters and compilers, although the type system of a language sand built into interpreters and compilers, although the type system of a language sand built into interpreters and compilers, although the type system of a language sand built into interpreters and compilers, although the type system of a language sand built into interpreters and compilers, although the type system of a language sys
grammar. The main purpose of a type system in a programming language is to reduce possibilities for bugs in computer programs due to type errors. [2] The given type system in question determines what constitutes a type error, but in general, the aim is to prevent operations expecting a certain kind of value from being used with values of which that
operation does not make sense (validity errors). Type systems allow defining interfaces between different parts of a computer program, and then checking that the parts have been connected in a consistent way. This checking that the parts have other
purposes as well, such as expressing business rules, enabling certain compiler optimizations, allowing for multiple dispatch, and providing a form of documentation. One function is invoked by another function. The interface of a
function states the name of the function and a list of parameters that are passed to the function's code. The code of an invoking function states the name of the invoked, along with the names of variables that hold values to pass to it. During a computer program's execution, the values are placed into temporary storage, then execution jumps to the
code of the invoked function. The invoked function are written with the assumption of receiving an integer value, but the calling code passed a floating-point value, then the wrong result will be computed by the invoked function. The C compiler checks the types
of the arguments passed to a function when it is called against the types of the parameters declared in the function's definition. If the types do not match, the compiler throws a compiler thr
many C compilers the float data type, for example, is represented in 32 bits, in accord with the IEEE specification for single-precision floating-point addition, multiplication, etc.). The depth of type constraints and the manner of their evaluation
affect the typing of the language. A programming language may further associate an operation with various resolutions for each type, in the case of type polymorphism. Type theory is the study of type systems. The concrete types of some programming languages, such as integers and strings, depend on practical issues of computer architecture,
compiler implementation, and language design. Formally, type theory studies type systems. A programming language must have the opportunity to type check using the type system whether at compile time or runtime, manually annotated or automatically inferred. As Mark Manasse concisely put it:[3]The fundamental problem addressed by a type
theory is to ensure that programs have meaning. The fundamental problem caused by a type theory is that meaningful programs may not have meaning to a sequence of bits such as a value in memory or some object such
as a variable. The hardware of a general purpose computer is unable to discriminate between for example a memory address and an instruction between any of the possible values that a sequence of bits might mean. [note 1] Associating a
sequence of bits with a type conveys that meaning to the programmable hardware to form a symbolic system composed of that hardware and some program. A program associated with many subtypes. Other entities, such as objects, modules, communication
channels, and dependencies can become associated with a type. Even a type of a value), class (a type of a type of a value), class (a type of a type, or metatype). These are the abstractions that typing can go through, on a
hierarchy of levels contained in a system. When a programming language evolves a more elaborate type system, it gains a more finely grained rule set than basic type checking, but this comes at a price when the type inferences (and other properties) become undecidable, and when more attention must be paid by the programmer to annotate code or
to consider computer-related operations and functioning. It is challenging to find a sufficiently expressive type system that satisfies all programming practices in a type safe manner. A programming practices in a type safe manner and functioning. It is challenging to find a sufficiently expressive type system that satisfies all programming practices in a type safe manner. A programming practices i
checker. Beyond simple value-type pairs, a virtual "region" of code is associated with an "effect" component describing what is being done with what, and enabling for example to "throw" an error report. Thus the symbolic system may be a type and effect system, which endows it with more safety checking alone. Whether automated
by the compiler or specified by a programmer, a type system renders program behavior illegal if it falls outside the type-system rules. Advantages provided by programmer-specified type system sinclude: Abstraction (or modularity) Types enable programmers to think at a higher level than the bit or byte, not bothering with low-level implementation.
For example, programmers can begin to think of a string as a set of character values instead of as an array of bytes. Higher still, types enable programmers to think about and express interfaces between two of any-sized subsystems. This enables more levels of localization so that the definitions required for interoperability of the subsystems remain
consistent when those two subsystems communicate. Documentation In more expressive type systems, types can serve as a form of documentation clarifying the intent of the programmer. For example, if a programmer declares a function as returning a timestamp type, this documents the function when the timestamp type can be explicitly declared
deeper in the code to be an integer type. Advantages provided by compiler-specified type systems include: Optimization Static type-checking may provide useful compiler may be able to use more efficient machine
instructions. Safety A type system enables the compiler to detect meaningless or invalid code. For example, we can identify an expression 3 / "Hello, World" as invalid, when the rules do not specify how to divide an integer by a string. Strong typing offers more safety, but cannot guarantee compilet type safety. A type error occurs when an operation
receives a different type of data than it expected.[4] For example, a type error would happen if a line of code divides two integers, and is passed a string of letters instead of an integer.[4] It is an unintended condition[note 2] which might manifest in multiple stages of a program's development. Thus a facility for detection of the error is needed in the
type system. In some languages, such as Haskell, for which type inference is automated, lint might be available to its compiler to aid in the detection of error. Type safety contributes to program correctness, but might only guarantee correctness, but might only guarantee correctness, but might be available to its compiler to aid in the detection of error. Type safety contributes to program correctness, but might only guarantee correctness, but might only guarantee correctness.
defects, other kinds of formal methods, collectively known as program analyses, are in common use. Alternatively, a sufficiently expressive type system, such as in dependently typed languages, can prevent these kinds of errors (for example, expressive type of non-zero numbers). In addition, software testing is an empirical method for finding
errors that such a type checker would not detect. The process of verifying and enforcing the constraints of typestype checkingmay occur at compile time (a static check) or at run-time (a dynamic check). If a language specification requires its typing rules strongly, more or less allowing only those automatic type conversions that do not lose information
one can refer to the process as strongly typed; if not, as weakly typed. The terms are not usually used in a strict sense. See also: Category: Statically typed program based on analysis of a program based on analysis of a program passes a static type checker,
then the program is guaranteed to satisfy some set of type safety properties for all possible inputs. Static type checking can be considered an optimization. If a compiler can prove that a program is well-typed, then it does not need to emit
 dynamic safety checks, allowing the resulting compiled binary to run faster and to be smaller. Static type checking for Turing-complete languages is innerently conservative. That is, if a type system is both sound (meaning that it rejects all incorrect programs) and decidable (meaning that it is possible to write an algorithm that determines whether a
program is well-typed), then it must be incomplete (meaning there are correct programs, which are also rejected, even though they do not encounter runtime errors).[7] For example, consider a program containing the code:if then else Even if the expression always evaluates to true at run-time, most type checkers will reject the program as ill-typed,
because it is difficult (if not impossible) for a static analyzer to determine that the else branch will not be taken.[8] Consequently, a static type checking, even code coverage may be unable to find such type errors. The tests may fail to detect
such type errors, because the combination of all places where a certain value is used must be taken into account. A number of useful and common programming language features cannot be checked statically, such as downcasting. Thus, many languages will have both static and dynamic type checking; the static
type checker verifies what it can, and dynamic checks verify the rest. Many languages with static type checking provide a way to bypass the type checker. Some languages allow programmers to choose between static and dynamic type safety. For example, historically C# declares variables statically, [9]:77, Section 3.2 but C# 4.0 introduces the
dynamic keyword, which is used to declare variables to be checked dynamically at runtime.[9]:117, Section 4.1 Other languages allow writing code that is not type-safe; for example, in C, programmers can freely cast a value between any two types that have the same size, effectively subverting the type concept. See also: Dynamic programming
language, Interpreted language, and Category: Dynamically typed programming languages generally associate each runtime object with a type tag (i.e., a reference to a type) containing its type
information. This runtime type information (RTTI) can also be used to implement dynamic dispatch, late binding, downcasting, reflective programming (reflection), and similar features. Most type-safe languages include some form of dynamic type checking, even if they also have a static type checker. [10] The reason for this is that many useful features.
or properties are difficult or impossible to verify statically. For example, suppose that a program defines two types, A and B, where B is a subtype of A. If the program tries to convert a value of type B. Thus, a dynamic
check is needed to verify that the operation is safe. This requirement is one of the criticisms of downcasting. By definition, dynamic type checking may cause a program to fail at runtime. In some programming languages, it is possible to anticipate and recover from these failures. In others, type-checking errors are considered fatal. Programming
languages that include dynamic type checking but not static type checking are often called "dynamically typed programming languages". "Type hinting of typefaces, see font hinting. Certain languages support
downcasting types to their subtypes, querying an object to discover its dynamic type and other type operations that depend on runtime type information. Another example is C++ RTTI. More generally, most programming languages include mechanisms for dispatching over different 'kinds' of data, such as disjoint unions, runtime polymorphism, and
variant types. Even when not interacting with type annotations or type checking, such mechanisms are materially similar to dynamic typing. Objects in object-oriented languages are usually accessed by a reference whose static
target type (or manifest type) is equal to either the object's run-time type (its latent type) or a supertype thereof. This is conformant with the Liskov substitution principle, which states that all operations performed on an instance of a subtype. This concept is also known as subsumption or subtype
polymorphism. In some languages subtypes may also possess covariant or contravariant return types and argument types checked by default, but allow programs to opt into static type checking by providing optional annotations. One reason to use such
hints would be to optimize the performance of critical sections of a program. This is formalized by gradual typing. The programming environment based on Lisp, and a precursor of the language Racket is also soft-typed. [11] Conversely, as of version 4.0, the C# language provides a way to indicate that a variable
should not be statically type checked. A variable whose type is dynamic will not be subject to static type checking. Instead, the program relies on runtime type information to determine how the variable may be used.[12][9]:113119In Rust, the dynamic typing of 'static types.[13]The choice between static and dynamic
typing requires certain trade-offs. Static typing can find type errors reliably at compile time, which increases the reliability of the delivered program. However, programmers disagree over how commonly type errors occur, resulting in further disagreements over the proportion of those bugs that are coded that would be caught by appropriately
representing the designed types in code.[14][15] Static typing advocates[who?] believe programs are more reliable when they have been well type-checked, whereas dynamic-typing advocates[who?] point to distributed code that has proven reliable and to small bug databases.[citation needed] The value of static typing increases as the strength of the
type system is increased. Advocates of dependent typing, [who?] implemented in languages such as Dependent ML and Epigram, have suggested that almost all bugs can be considered by the compiler. [16] Static typing usually results in compiled
code that executes faster. When the compiler knows the exact data types that are in use (which is necessary for static verification, either through declaration or inference) it can produce optimized machine code. Some dynamically typed languages such as Common Lisp allow optional type declarations for optimization for this reason. By contrast,
dynamic typing may allow compilers to run faster and interpreters to dynamically load new code, because changes to source code in dynamically typed languages that lack type inference
(such as C and Java prior to version 10) require that programmers declare the types that a method or function must use. This can serve as added program documentation, that is active and dynamic, instead of static. This allows a compiler to prevent it from drifting out of synchrony, and from being ignored by programmers. However, a language can
be statically typed without requiring type declarations (examples include Haskell, Scala, OCaml, F#, Swift, and to a lesser extent C# and C++), so explicit type declaration is not a necessary requirement for static typing in all languages. Dynamic typing allows constructs that some (simple) static type declaration is not a necessary requirement for static typing in all languages. Dynamic typing allows constructs that some (simple) static type declaration is not a necessary requirement for static typing in all languages.
functions, which execute arbitrary data as code, become possible. An eval function is possible with static typing, but requires advanced uses of algebraic data types. Further, dynamic typing better accommodates transitional code and prototyping, such as allowing a placeholder data structure (mock object) to be transparently used in place of a full
data structure (usually for the purposes of experimentation and testing). Dynamic typing typically allows duck typing or other mechanisms like generic programming that also enable easier code reuse. Dynamic typing typically makes
metaprogramming easier to use. For example, C++ templates are typically more cumbersome to write than the equivalent Ruby or Python code since C++ has stronger rules regarding type definitions (for both functions and variables). This forces a developer to write more boilerplate code for a template than a Python developer would need to. More
advanced run-time constructs such as metaclasses and introspection are often harder to use in statically typed languages, such features may also be used e.g. to generate new types and behaviors on the fly, based on run-time data. Such advanced constructs are often provided by dynamic programming languages; many of these are
dynamically typed, although dynamic typing need not be related to dynamic programming languages. Main article: Strong and weak typingLanguages are often colloquially referred to as strongly typed or weakly typed. In fact, there is no universally accepted definition of what these terms mean. In general, there are more precise terms to represent
the differences between type systems that lead people to call them "strong" or "weak". Main articles: Type safety and Memory safety of typed operations. Computer scientists use the term type-safe language to describe languages that do not allow
operations or conversions that violate the rules of the type system. Computer scientists use the term memory-safe language (or just safe language that do not allow programs to access memory that has not been assigned for their use. For example, a memory-safe language will check array bounds, or else statically guarantee (i.e.,
at compile time before execution) that array accesses out of the array boundaries will cause compile-time and perhaps runtime errors. Consider the following program of a language that is both type-safe and memory-safe:[17]var x:= 5; var y:= "37"; var z:= x + y; In this example, the variable z will have the value 42. Although this may not be what the
programmer anticipated, it is a well-defined result. If y were a different string, one that could not be converted to a number (e.g. "Hello World"), the result would be well-defined as well. Note that a program can be type-safe or memory-safe and still crash on an invalid operation. This is for languages where the type system is not sufficiently advanced
to precisely specify the validity of operations on all possible operands. But if a program encounters an operation that is not type-safe, terminating the program is often the only option. Now consider a similar example in C:int x = 5; char y[] = "37"; char* z = x + y; printf("%c", *z); In this example z will point to a memory address five characters beyond y,
equivalent to three characters after the terminating zero character of the string pointed to by y. This is memory that the program may do anything; with a simple compiler it might actually print whatever byte is stored after the string "37". As this example shows,
C is not memory-safe. As arbitrary data was assumed to be a character, it is also not a type-safe language. In general, type-safety and memory-safety go hand in hand. For example, a language that supports pointer arithmetic and number-to-pointer conversions (like C) is neither memory-safe nor type-safe, because it allows arbitrary memory to be
accessed as if it were valid memory of any type. Some languages allow different regions of code. Examples include: The use strict directive in JavaScript[18][19][20] and Perl applies stronger checking. The declare(strict types=1) in PHP[21] on a per-file basis allows only a variable of exact type of the type
declaration will be accepted, or a TypeError will be thrown. The Option Strict On in VB.NET allows the compiler to require a conversion between objects. Additional tools such as lint and IBM Rational Purify can also be used to achieve a higher level of strictness. It has been proposed, chiefly by Gilad Bracha, that the choice of type system be made
independent of choice of language; that a type system should be a module that can be plugged into a language as needed. He believes this is advantageous, because what he calls mandatory type system does not affect the semantics of the language is
difficult to fulfill. Optional typing is related to, but distinct from, gradual typing. While both typing disciplines can be used to perform static analysis of code (static typing), optional typing is related to, but distinct from, gradual typing is related to, but distinct from, gradual typing disciplines can be used to perform static analysis of code (static typing).
code (especially, functions or classes) to act on values of multiple types, or to the ability of different instances of the same data structure to contain elements of different types. Type systems that allow polymorphism, programmers need only implement a
data structure such as a list or an associative array once, rather than once for each type of element with which they plan to use it. For this reason computer scientists sometimes call the use of certain forms of polymorphism are closely related to those of abstraction, modularity
and (in some cases) subtyping. Many type systems have been created that are specialized for use in certain environments with certain types of data, or for out-of-band static program analysis. Frequently, these are based on ideas from formal type theory and are only available as part of prototype research systems. The following table gives an overview
over type theoretic concepts that are used in specialized type systems. The names M, N, O range over types. The following notation will be used: M: {\displaystyle \sigma }; M (N) {\displ
of M {\displaystyle M} on N {\displaystyle \tau [x:=N]} ) describes the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which results from replacing all occurrences of the type which replaces all occurrences of the type which replaces all occurrences of the type which replaces all occurrences all occurrences all occurrences all occurrences all occurrences all occurrences all o
\sigma \to \tau } If M: {\displaystyle M:\sigma \to \tau } and N: {\displaystyle N:\sigma } and O: {\displaystyle O:\tau } .Sum +
 \{\text{sigma} + \text{sigma} + \text{injection s.t. N} : \{\text{sigma} + \text{injection s.t
\cap \tau } , then M : {\displaystyle M:\sigma \ cup \tau } . Hen M : {\displaystyle M:\sigma \ cup \tau } . Hen M : {\displaystyle M:\sigma \ cup \tau } . Then M : {\displaystyle M:\sigma \ cup \tau \ rangle } . Then M is a member x : {\displaystyle M:\sigma \ cup \tau \ rangle \ 
{\displaystyle x:\tau } .Polymorphic . {\displaystyle M:\tau } .f M : . . .f M : .f M : .f M : . .f M : .f
 some type .Recursive . {\displaystyle \mu \alpha .\tau } If M : . {\displaystyle M:\tau } If M : . {\displaystyle M:\tau } If M : . {\displaystyle M:\tau } If M : [ x : ) {\displaystyle M:\tau } If M : . {\displaystyle M:\tau } .then M (N) : [ x := N] {\displaystyle \mu \alpha .\tau } If M : [ x : ) {\displaystyle M:\tau } ...
M(N): tau [x:=N] . Dependent pair[b] ( x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( <math>x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( (x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( (x: ) {\displaystyle M:(x:sigma ) times tau } If M: ( (x: ) {\displaystyle M:(x: ) times tau } If M: ( (x: ) {\displaystyle M:(x: ) times tau } If M: ( (x: ) {\displaystyle M:(x: ) times tau } If M: ( (x: ) {\displaystyle M:(x: ) times tau } If M: ( (x: ) {\displaystyle M:(x: ) times tau } If M: ( (x: ) {\displaystyle M:(x: ) times tau } If M: ( (x: ) {\displaystyle M:(x: ) times tau } If M: ( (x: ) {\displaystyle M:(x: ) times tau } If M: ( (x: ) {\displaystyle M:(x: ) times tau } If M: ( (x: ) {\displaystyle M:(x: ) times tau } If M: ( (x
x \in M : x \in
```

union[24] x : {\displaystyle \bigcup {x:\sigma }\tau } If M : x : {\textstyle M:\sigma }\tau } If M : x : {\textstyle M:\sigma }\tau } .^ Also referred to as dependent product type, since (x:) = x : {\textstyle (x:\sigma)\tau } | x : {\textstyle M:\sigma }\tau } .^ Also referred to as dependent product type, since (x:) = x : {\textstyle M:\sigma }\tau } | x : {\textstyle M:\sigma }\tau } .^ Also referred to as dependent product type, since (x:) = x : {\textstyle M:\sigma }\tau } | x : {\textstyle M:\sigma }\tau }\tau } | x : {\textstyle M:\sigma }\tau } | x : {\textstyle M:\sigma }\tau }\tau }\tau } | x : {\textstyle M:\sigma }\tau }\ sum type, since (x:) = x: {\textstyle (x:\sigma)\times \tau =\sum _{x:\sigma }\tau } . Main article: Dependent types are based on the idea of using scalars or values to more precisely describe the type of a 3 3 {\displaystyle \mathrm {matrix} (3,3)} might be the type of a 3 3 {\displaystyle \mathrm {matrix} (3,3)} 3\times 3\} matrix. We can then define typing rules such as the following rule for matrix { (k,n) times \mathrm {matrix} (k,n) m a t r i x (k, n) m a t r i x (k,n) m a t r i x (k,n) m a t r i x (k,n) where k, m, n are arbitrary positive integer values. A variant of ML called Dependent ML has been created based on this type system, but because type checked without some kind of limits. Dependent ML limits the sort of equality it can decide to Presburger arithmetic. Other languages such as Epigram make the value of all expressions in the language decidable so that type checking can be decidable. However, in general proof of decidable, so many programs require hand-written annotations that may be very non-trivial. As this impedes the development process, many language implementations provide an easy way out in the form of an option to disable this condition. This, however, comes at the cost of making the type-check, causing the type-checker run in an infinite loop when fed programs that do not type-check, causing the type-check, causing the type-checker run in an infinite loop when fed programs that do not type-check, causing the type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-check run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run in an infinite loop when fed programs that do not type-checker run i assigned to values having the property that they have one and only one reference to them at all times. These are valuable for describing large immutable values such as files, strings, and so on, because any operation that simultaneously destroys a linear object (such as str = str + "a") can be optimized "under the hood" into an in-place mutation. Normally this is not possible, as such mutations could cause side effects on parts of the program holding other references to the object, violating referential transparency. They are also used in the prototype operating system Singularity for interprocess communication, statically ensuring that processes cannot share objects in shared memory in order to prevent race conditions. The Clean language (a Haskell-like language) uses this type system in order to gain a lot of speed (compared to performing a deep copy) while remaining safe. Main article: Intersection typeIntersection typeInte value sets. For example, in most implementations of C the signed char has range 0 to 127. Such an intersection type could be safely passed into functions expecting either signed or unsigned chars, because it is compatible with both types. Intersection types are useful for describing overloaded function types: for example, if "int int" is the type of functions taking an integer argument and returning an integer, and "float float, then the intersection of these two types can be used to describe functions that do one or the other, based on what type of input they are given. Such a function could be passed into another function afely; it simply would not use the "float float" function afely; it simply would not use the "float float" function afely; it simply would not use the "float float" function afely; it simply would not use the "float float" function afely; it simply would not use the "float float" function afely; it simply would not use the "float float" function afely; it simply would not use the "float float" function afely; it simply would not use the "float float" function afely; it simply would not use the "float float" function afely; it simply would not use the "float float" function afely; it simply would not use the "float float" function afely; it simply would not use the "float float" function afely; it simply would not use the "float float" function afely; it simply would not use the "float float" function afely; it simply would not use the "float float" function afely; it simply would not use the "float float" function afely float f is empty. The Forsythe language includes a general implementation of intersection types. A restricted form is refinement types. How types are types describing values that belong to either of two types. For example, in C, the signed char has a -128 to 127 range, and the unsigned char has a 0 to 255 range, so the union of these two types would have an overall "virtual" range of -128 to 255 that may be used partially depending on which union member is accessed. Any function handling this union type are operations that are valid on both types being unioned. C's "union" concept is similar to union types, but is not typesafe, as it permits operations that are valid on either type, rather than both. Union types are important in program analysis, where they are used to represent symbolic values whose exact nature (e.g., value or type) is not known. In a subclassing hierarchy, the union of a type and an ancestor type (such as its parent) is the ancestor type. The union of sibling types is a subtype of their common ancestor (that is, all operations in common). Main article: Existential quantifier Existential types are frequently used in connection with record types to represent modules and abstract data types, due to their ability to separate implementation from interface that has a data member named a of type X and a function named f that takes a parameter of the same type X and returns an integer. This could be implemented in different ways; for example:intT = { a: int; f: (int int); }floatT = { a: float; f: (float int); }floatT = { a: int; f: (int int); }floatT = { a: float; f: (float int); }floatT = { a: int; f: (int int); }floatT = { a: int; f: (in regardless of what the abstract type X is. This gives flexibility for choosing types suited to a particular implementation, while clients that use only values of the interface typethe existential type a given module belongs to. In the above example intT { a: int; f: (int int); } could also have the type X { a: X; f: (int int); } as X { a: X; f: (X int); } Although abstract data types and modules had been implemented in programming languages for quite some time, it wasn't until 1988 that John C. Mitchell and Gordon Plotkin established the formal theory under the slogan: "Abstract [data] types have existential type".[25] The theory is a second-order typed lambda calculus similar to System F, but with existential instead of universal quantification. Main article: Gradual typingIn a type system with Gradual typing, variables may be assigned a type either at compile-time (which is static typing), or at run-time (which is dynamic typing). [26] This allows software developers to choose either type paradigm as appropriate, from within a single language. [26] Gradual typing uses a special type named dynamic to represent statically unknown types; gradual typing replaces the notion of type equality with a new relation called consistency that relates the dynamic type to every other type. The consistency relation is symmetric but not transitive. [27] Further information: Type inferenceMany static type systems, such as those of C and Java, require type declarations: the programmer must explicitly associate each variable with a specific type. Others, such as Haskell's, perform type inference: the compiler draws conclusions about the types of variables. For example, given a function f(x, y) that adds x and y together, the compiler can infer that x and y must be numbers included for numbers. Thus, any call to f elsewhere in the program that specifies a non-numeric type (such as a string or list) as an argument would signal an error. Numerical and string constants and expressions in code can and often do imply type in a particular context. For example, an expression 3.14 might imply a type of floating-point, while [1, 2, 3] might imply a list of integerstypically an array. Type inference is in general possible, if it is computable in the type system in question. Moreover, even if inference is often possible for a large subset of real-world programs. Haskell's type system, a version of HindleyMilner, is a restriction of System F to so-called rank-1 polymorphic types, in which type inference is computable. (Type checking is decidable, however, and rank-1 programs still have type inference; higher rank polymorphic programs are rejected unless given explicit type annotations.) Main article: Type theory Decision problems A type environments using typing rules is naturally associated with the decision problems of type environment {\displaystyle e}, and a type {\displaystyle a} and a type {\displaystyle a}. \tau } , decide whether the term e {\displaystyle e} can be assigned the type {\displaystyle \tau } in the type environment {\displaystyle \tau } in the type environment. Given a term e {\displaystyle \tau } in the type environment. In the type environment a term e {\displaystyle \tau } in the type environment. In the type environment. In the type environment. In the type environment environment. In the type environment. In the type environment. In the type environment environment. In the type environment. In the type environment environment. In the type environment. In the type environment environment. In the type environment. In the type environment environment. In the type environment. In the type environment. In the type environment environment. In the type environment. In the type environment environment environment environment. In the type environment environment environment. In the type environment environment environment environment. In the type environment environme type environment {\displaystyle \Gamma } and a type {\displaystyle \tau } in the type environment. Some languages like C# or Scala have a unified type system. [29] This means that all C# types including primitive types inherit from a single root object. Every type in C# inherits from the Object class. Some languages, like Java and Raku, have a root type but also have primitive types that exist together with the primitive types so developers can use either the wrapper object types or the simpler non-object primitive types. Raku automatically converts primitive types to objects when their methods are accessed. [31] A type checker for a statically typed language must verify that the type of any expression is consistent with the type expected by the context in which that expression appears. For example, in an assignment statement of the form x:= e, the inferred type of the expression e must be consistent with the declared or inferred type of the variable x. This notion of consistency, called compatibility, is specific to each programming language. If the type of e and the type of x are the same, and assignment is allowed for that type, then this is a valid expression. Thus, in the simplest type systems, the question of whether two types are compatible reduces to that of whether they are equal (or equivalent). Different languages, however, have different criteria for when two types are compatible reduces to that of whether they are equal (or equivalent). type systems, in which any two types that describe values with the same structure are equivalent, and nominative type systems, in which no two syntactically distinct type expressions denote the same "name" in order to be equal). In languages with subtyping, the compatibility relation is more complex: If B is a subtype of A, then a value of type B can be used in a context where one of type A is expected (covariant), even if the reverse is not true. Like equivalence, the subtype relation is defined differently for each programming language may also have implications for type compatibility. Computer programming portal Comparison of type systems Covariance and contravariance (computer science) Polymorphism in object-oriented programming Type signature Type theory The Burroughs ALGOL computer line determined a memory location's contents by its flag bits. Flag bits specify the contents of a memory location. Instruction, data type, and functions are specified by a 3 bit code in addition to its 48 bit contents. Only the MCP (Master Control Program) could write to the flag code bits. ^ For example, a leaky abstraction might surface during development, which may show that more type development is needed. "The evaluation of a well-typed program always terminates".B. Nordstrm, K. Petersson, and J. M. Smith[5] A systematic change in variables to avoid capture of a free variable can introduce error, in a functional programming language where functions are first class citizens.[6] From the lambda calculus article. Pierce 2002, p.1: "A type system is a tractable syntactic method for proving the absence of certain program behaviors by classifying phrases according to the kinds of values they compute." Cardelli 2004, p.1: "The fundamental purpose of a type system is to prevent the occurrence of execution errors during the running of a program." Pierce 2002, p.208. a b Sethi, R. (1996). Programming languages: Concepts and constructs (2nded.). Addison-Wesley. p.142. ISBN 978-0-201-59065-4. OCLC604732680.^ Nordstrm, B.; Petersson, K.; Smith, J.M. (2001). "Martin-Lf's Type Theory". Algebraic and Logical Structures. Handbook of Logic in Computer Science. Vol.5. Oxford University Press. p.2. ISBN 978-0-19-154627-3.^ Turner, D.A. (12 June 2012). "Some History of Functional Programming Languages" (PDF). invited lecture at TFP12, at St Andrews University. See the section on Algol 60.^ "... anysound, decidable type system must be incomplete" D. Remy, Didier. "Type systems for programming languages" (PDF). Archived from the original (PDF) on 14 November 2017. Retrieved 26 May 2013. Pierce 2002. a b c Skeet, Jon (2019). C# in Depth (4ed.). Manning. ISBN 978-1617294532. Miglani, Gaurav (2018). "Dynamic Method Dispatch or Runtime Polymorphism in Java". Archived from the original on 2020-12-07. Retrieved 2021-03-28. Wright, Andrew K. (1995). Practical Soft Typing (PhD). Rice University. hdl:1911/16900. "dynamic (C# Reference)". MSDN Library. Microsoft. Retrieved 14 January 2014. "std::any Rust". doc.rust-lang.org. Retrieved 2021-07-07. Meijer, Erik; Drayton, Peter. "Static Typing Where Possible, Dynamic Typing When Needed: The End of the Cold War Between Programming Languages" (PDF). Microsoft Corporation. Laucher, Amanda; Snively, Paul (2012). "Types vs Tests". InfoQ.^ Xi, Hongwei (1998). Dependent Types in Practical Programming (PhD). Dependent Types in Practical Programming. Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. ACM. pp.214227. CiteSeerX10.1.1.69.2042. doi:10.1145/292540.292560. ISBN1581130953. S2CID245490.^ Visual Basic is an example of a language that is both type-safe and memory-safe.^ "4.2.2 The Strict Variant of ECMAScript". ECMAScript 2020 Language Specification (11thed.). ECMA June 2020. ECMA-262.^ "Strict mode JavaScript". MDN. Developer.mozilla.org. 2013-07-17.^ "Strict typing". PHP Manual: Language Reference: Functions.^ a b Bracha, G. "Pluggable Types" (PDF).^ "Sure. It's called "gradual typing", and I would qualify it as trendy. ..." Is there a language that allows both static and dynamic typing?. stackoverflow. 2012. a b c Kopylov, Alexei (2003). "Dependent intersection: A new way of defining records in type theory". 18th IEEE Symposium on Logic in Computer Science. LICS 2003. IEEE Computer Society. pp.8695. CiteSeerX10.1.1.89.4223. doi:10.1109/LICS.2003.1210048. Mitchell, John C.; Plotkin, Gordon D. (July 1988). "Abstract Types Have Existential Type" (PDF). ACM Trans. Program. Lang. Syst. 10 (3): 470502. doi:10.1145/44501.45065. S2CID1222153. a b Siek, Jeremy (24 March 2014). "What is gradual typing?". Siek, Jeremy; Taha, Walid (September 2006). Gradual Typing for Functional Languages (PDF). Scheme and Functional Programming 2006. University of Chicago. pp.8192. Barendregt, Henk; Dekkers, Wil; Statman, Richard (20 June 2013). Lambda Calculus with Types. Cambridge University Press. p.66. ISBN 978-0-521-76614-2. Barendregt, Henk; Dekkers, Wil; Statman, Richard (20 June 2013). Lambda Calculus with Types. Cambridge University Press. p.66. ISBN 978-0-521-76614-2. December 2017. ECMA-334. "Native Types". Perl 6 Documentation. "Numerics, Auto-boxing". Perl 6 Documentation. Auto-boxing". Perl 6 Documentation. Auto-boxing Types, Data Abstraction, and Polymorphism (PDF). ACM Computing Surveys. 17 (4): 471523. CiteSeerX10.1.1.117.695. doi:10.1145/6041.6042. S2CID2921816.Pierce, Benjamin C. (2002). Types and Programming Languages. MIT Press. ISBN978-0-262-16209-8. Cardelli, Luca (2004). "Type systems" (PDF). In Allen B. Tucker (ed.). CRC Press. ISBN978-1584883609. Tratt, Laurence (July 2009). "5. Dynamically Typed Languages" Advances in Computers. Vol.77. Elsevier. pp.149184. doi:10.1016/S0065-2458(09)01205-4. ISBN 978-0-12-374812-6. The Wikibook Ada Programming has a page on the topic of: TypesThe Wikibook Debating Type Systems".Retrieved from " JavaScript is hell because of two problems, and you start having fun. The first problem is retention. You remember only ten or twenty percent of what you read. That spells failure. To become fluent in a computer language, you have to retain pretty much everything. How can you retain everything? Only by constantly being asked to play everything back. That's why people use flashcards one better. After reading a short chapter, you go to my website and complete twenty interactive exercises. Algorithms check your work to make sure you know what you think you know. When you stumble, you do the exercise again. You keep trying until you know the chapter cold. The exercises are free. The second prototypes. Unfortunately, they blame themselves. That's why the Dummies books sell so well. But the fault lies with the authors, coding virtuosos who lack teaching talent. I'm the opposite of the typical software book author. I'll never code fast enough to land a job at Google. But I can teach. Anyway, most comprehension problems are just retention problems in disguise. If you get lost trying to understand variable scope, it's because you don't remember how functions work. Thanks to the interactive exercises on my website, you'll always understand and remember everything necessary to confidently tackle the next concept. "I've signed up to a few sites like Udemy, Codecademy, FreeCodeCamp, Lynda, YouTube videos, even searched on Coursera but nothing seemed to work for me. This book takes only 10 minutes each chapter and after that, you can exercise what you've just learned right away!" -Amazon reviewer Constanza MoralesBetter than just reading. It's how you wind up satisfied, confident, and proud, instead of confused, discouraged, and defeated. And since many people find doing things more enjoyable than reading things, it can be a pleasure to learn this way, quite apart from the impressive results you achieve. "Very effective and fun." -Amazon reviewer A. BergaminiWritten especially for beginners. I wrote the book and exercises especially for people who are new to programming. Making no assumptions about what you already know, I walk you through JavaScript slowly, patiently. I explain every little thing in sixth-grade English. I avoid unnecessary technical jargon like the plague. (Face it, fellow authors, it is the plague.) "The layman syntax he uses...makes it much easier to suddenly realize a concept that seemed abstract and too hard to wrap your head around is suddenly not complicated at all." - Amazon reviewer IMHOThe exercises keep you focused, give you extra practice where you're shaky, and prepare you for each next step. Every lesson is built on top of a solid foundation that you and I have carefully constructed. Each individual step is small. But, as Amazon reviewer James Toban says when you get to the end of the book, you've built "a tower of JavaScript." If you're an accomplished programmer already, my book may be just the one to get you coding JavaScript successfully. "Mark Myers' method of getting what can be...difficult information into a format that makes it exponentially easier to consume, truly understand, and synthesize into real-world application is beyond anything I've encountered before." -Amazon reviewer Jason A. Ruby

Smarter way to learn javascript exercises. A smarter way to learn javascript pdf github. A smarter way to learn javascript latest edition. A smarter way to learn javascript review. A smarter way to learn javascript reddit. A smarter way to learn javascript quiz. A smarter way to learn javascript epub. Smarter way to learn javascript book.

- https://hpfwater.com/admin/upload/files/35581352867.pdf
- http://yidaig.com/uploadfile/file/20250712214054.pdfcokoje
- https://tailormade-sales-marketing.com/userfiles/file/1194315338.pdf
 niyayaya