I'm not a robot

Conditional statements are used to perform different actions based on different conditions. Conditional Statements Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this. In JavaScript we have the following conditional statements: Use if to specify a block of code to be executed, if a specified condition is true Use else to specify a block of code to be executed, if the same condition is false Use else if to specify a new condition to test, if the first condition is false Use switch to specify many alternative blocks of code to be executed The switch statement is described in the next chapter. The if Statement Use the if statement to specify a block of JavaScript code to be executed if a condition is true. Syntax if (condition) { // block of code to be executed if the condition is true } Note that if is in lowercase letters. Uppercase letters (If or IF) will generate a JavaScript error. Make a "Good day" greeting if the hour is less than 18:00: if (hour < 18) { greeting = "Good day"; } The result of greeting will be: Try it Yourself » Use the else statement to specify a block of code to be executed if the condition is false. if (condition) { // block of code to be executed if the condition is true } else { // block of code to be executed if the condition is false } If the hour is less than 18, create a "Good day" greeting, otherwise "Good evening": if (hour < 18) { greeting = "Good day"; } else { greeting = "Good evening"; } The result of greeting will be: Try it Yourself » The else if Statement Use the else if statement to specify a new condition if the first condition is false. Syntax if (condition1) { // block of code to be executed if condition1 is true } else if (condition2) { // block of code to be executed if the condition1 is false and condition2 is true } else { // block of code to be executed if the condition1 is false and condition2 is false } If time is less than 10:00, create a "Good morning" greeting, if not, but time is less than 20:00, create a "Good day" greeting, otherwise a "Good evening": if (time < 10) { greeting = "Good morning"; } else if (time < 20) { greeting = "Good day"; } else { greeting = "Good evening"; } The result of greeting will be: Try it Yourself » This example will write a link to either W3Schools or to the World Wildlife Foundation (WWF). By using a random number, there is a 50% chance for each of the links. let text; if (Math.random() < 0.5) { text = "Visit W3Schools"; } else { text = "Visit WWF"; } document.getElementById("demo").innerHTML = text; Try it Yourself » 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 paradigms listed in this article. Concurrent programming – have language constructs for concurrency, these may involve multi-threading, support for distributed computing, message passing, shared resources (including shared memory), or futures Actor 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 constraints (or constraint networks), directing allowable solutions (uses 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 – have support for multiple autonomous computers that communicate via computer networks Functional programming – uses evaluation of mathematical functions and avoids state and mutable data Generic programming – uses algorithms written in terms of to-be-specified-later types that are then instantiated as needed for specific types provided as parameters Imperative programming – explicit statements that change a program state Logic programming – uses explicit mathematical logic for programming Metaprogramming – writing programs that write or manipulate other programs (or themselves) as their data, or that do part of the work at compile time that would otherwise be done at runtime Template metaprogramming – metaprogramming methods in which a compiler uses templates to generate temporary source code, which is merged by the compiler with the rest of the source code and then compiled Reflective programming – metaprogramming methods in which a program modifies or extends itself Object-oriented programming – uses data structures consisting of data fields and methods together with their interactions (objects) to design programs Class-based – object-oriented programming in which inheritance is achieved by defining classes of objects, versus the objects themselves Prototype-based – object-oriented programming that avoids classes and implements inheritance via cloning of instances Pipeline programming – a simple syntax change to add syntax to nest function calls to language originally designed with none Rule-based programming – a network of rules of thumb that comprise a knowledge base and can be used for expert systems and problem deduction & resolution Visual programming – manipulating program elements graphically rather than by specifying them textually (e.g. Simulink); also termed diagrammatic programming[1] List of multi-paradigm programming languages Language Paradigm count Concurrent Constraints Dataflow Declarative Distributed Functional Metaprogramming Generic Imperative Logic Reflection Object-oriented Pipelines Visual Rule-based Other Ada[2][3][4][5][6] 5 Yes[a 1] — — — Yes — — Yes Yes — — Yes[a 2] — — — — ALF 2 — — — — — Yes — — Yes — — — — — — AmigaE[citation needed] 2 — — — — — — — — Yes — — Yes[a 2] — — — — APL 3 — — — — — Yes — — Yes — — — — — — Array (multi-dimensional) BETA[citation needed] 3 — — — — — Yes — — Yes — — Yes[a 2] — — — — C++ 7 (15) Yes[7][8][9] Library[10] Library[11][12] Library[13][14] Library[15][16] Yes Yes[17] Yes[a 3] Yes Library[18][19] Library[20] Yes[a 2] Library[21] — Library[22] Array (multi-dimensional; using STL) C# 6 (7) Yes — Library[a 4] — — Yes[a 5] — Yes Yes — Yes Yes[a 2] — — — Reactive[a 6] ChucK[citation needed] 3 Yes — — — — — — — Yes — — Yes[a 2] — — — — Claire 2 — — — — — Yes — — — — — Yes[a 2] — — — — Clojure 5 Yes[23][24] — — Yes — Yes[25] Yes[26] — — Library[27] — — Yes[28] — — Multiple dispatch,[29] Agents[30] Common Lisp 7 (14) Library[31] Library[32] Library[33] Yes[34] Library[35] Yes Yes Yes[36] Yes Library[37] Yes Yes[a 7][a 2][38] Library[39] Library[40] Library[41] Multiple dispatch, meta-OOP system,[42] Language is extensible via metaprogramming. Curl 5 — — — — — Yes — Yes[a 3] Yes — Yes Yes[a 2] — — — — Curry 4 Yes Yes — — — Yes — — — Yes — — — — — — D (version 2.0)[43][44] 7 Yes[a 8] — — — — Yes Yes[45][a 3] Yes[a 3] Yes — Yes Yes[a 2] — — — — Delphi 3 — — — — — — — Yes[a 3] Yes — — Yes[a 2] — — — — Dylan[citation needed] 3 — — — — — Yes — — — — Yes Yes[a 2] — — — — E 3 Yes — — — Yes — — — — — — — Yes[a 2] — — — — ECMAScript[46][47] (ActionScript, E4X, JavaScript, JScript) 4 (5) Partial[a 9][a 10] — — Library[48][49] — Yes — — Yes — Yes Yes[a 11] Library[50][51] — — Reactive,[a 12][52] event driven[a 13][a 14] Erlang 3 Yes — — Yes Yes Yes — — — — — — Yes — — — Elixir 4 Yes — — — Yes Yes Yes — — — — — Yes — — — Elm 6 Yes — Yes Yes — Yes (pure)[a 15] — Yes — — — — Yes — — Reactive F# 7 (8) Yes[a 8] — Library[a 4] Yes — Yes — Yes Yes — Yes Yes[a 2] — — — Reactive[a 6] Fortran 4 (5) Yes — — — — Yes[a 15] — Yes[a 16] — — — Yes[a 2] — — — Array (multi-dimensional) Go 4 Yes — — — — — — — Yes — Yes — Yes — — — Haskell 8 (15) Yes Library[53] Library[54] Yes Library[55] Yes (lazy) (pure)[a 15] Yes[56] Yes Yes Library[57] — Partial[a 17] Yes Yes Library[58] Literate, reactive, dependent types (partial) Io 4 Yes[a 8] — — — — Yes — — Yes — — Yes[a 11] — — — — J[citation needed] 3 — — — — — Yes — — Yes — — Yes[a 2] — — — — Java 6 Yes Library[59] Library[60] — — Yes — Yes Yes — Yes Yes[a 2] — — — — Julia 9 (17) Yes Library[61] Library[62][63] Library[64] Yes Yes (eager) Yes Yes Yes Library[65] Yes Partial[a 18] Yes — Library[66][67] Multiple dispatch,Array (multi-dimensional); optionally lazy[68] and reactive (with libraries) Kotlin 8 Yes — — — — Yes Yes Yes Yes — Yes Yes Yes — — — LabVIEW 4 Yes — Yes — — — — — — — — — Yes — Yes — — Lava 2 — — — — — — — — — — — — Yes[a 2] — Yes — — LispWorks (version 6.0 with support for symmetric multi-processing, rules, logic (Prolog), CORBA) 9 Yes — — — Yes Yes Yes — Yes Yes Yes[a 2] — — Yes — Lua[citation needed] 3 — — — — — Yes — — Yes[a 11] — — — — MATLAB 6 (10) Toolbox[69] Toolbox[70] Yes[71] — Toolbox[72] — Yes[73] Yes[74] — — Yes[75] Yes[76] — Yes[77] — Array (multi-dimensional) Nemerle 7 Yes — — — — Yes Yes Yes Yes — Yes Yes[a 2] — — — — Object Pascal 4 Yes — — — — Yes — — Yes — — Yes[a 2] — — — — OCaml 4 — — — — — Yes — Yes Yes — — Yes[a 2] — — — — Oz 11 Yes Yes Yes Yes Yes Yes — — Yes Yes — Yes[a 2] Yes — Yes — Perl[citation needed] 8 (9) Yes[78] — Yes[79] — — Yes Yes — Yes — Yes[a 2] Yes[a 2] Yes — — — PHP[80][81][82] 4 — — — — — Yes — — Yes — Yes Yes[a 2] — — — — Poplog 3 — — — — — Yes — — Yes Yes — — — — — — Prograph 3 — — Yes — — — — — — — — Yes[a 2] — Yes — — Python 5 (10) Library[83][84] Library[85] — — Library[86] Yes Yes[87][88] Yes[89][90] Yes Library[91] Yes Yes[a 2] — — — Structured R 4 (6) Library[92] — — — Library[93] Yes — — Yes — Yes Yes Yes[94] — — Array (multi-dimensional) Racket 10 Yes[95] Yes[96] Yes[97] — Yes[98] Yes Yes — Yes Yes Yes Yes — — — Lazy[99] Raku 10 Yes[100] Library[101] Yes[102] — Library[103] Yes Yes[104] Yes[105] Yes — Yes[106] Yes[107] Yes — — Multiple dispatch, lazy lists, reactive. ROOP 3 — — — — — — — — — Yes Yes — — — — Yes — Ruby 5 — — — — — Yes Yes — Yes — Yes Yes[a 2] — — — — Rust (version 1.0.0-alpha) 6 Yes[a 8] — — — — Yes Yes[108][109] Yes[110] Yes — — Yes — — — Linear, affline, and ownership types Sather[citation needed] 2 — — — — — Yes — — — — — — Yes[a 2] — — — — Scala[111][112] 9 Yes[a 8] — Yes[a 19] Yes — Yes Yes Yes Yes — Yes Yes[a 2] — — — — Simula[citation needed] 2 — — — — — — — — Yes — — Yes[a 2] — — — — SISAL 3 Yes — Yes — — Yes — — — — — — — — — — Spreadsheets 2 — — — — — Yes — — — — — — — Yes — — Swift 7 Yes — — — — Yes Yes Yes Yes — Yes Yes[a 2] — — — Block-structured Tcl with Snit extension[citation needed] 3 — — — — — Yes[113] — — Yes — — Yes[a 11][114] — — — — Visual Basic .NET 6 (7) Yes — Library[a 4] — — Yes — Yes Yes — Yes Yes[a 2] — — — Reactive[a 6] Windows PowerShell 6 — — — — — Yes — Yes Yes — Yes Yes[a 2] Yes — — — Wolfram Language & Mathematica 13[115] (14) Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes[116] — Yes Knowledge Based Programming paradigm List of programming languages by type Domain-specific language Domain-specific multimodeling ^ rendezvous and monitor-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 ^ a b c using TPL Dataflow ^ only lambda support (lazy functional programming) ^ a b c using Reactive Extensions (Rx) ^ multiple dispatch, method combinations ^ a b 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. (20–22 September 1994). "Diagrammatic-graphical programming languages and DoD-STD-2167A". Proceedings of AUTOTESTCON '94 (IEEEXplore). Institute of Electrical and Electronics Engineers (IEEE). pp. 211–220. doi:10.1109/AUTEST.1994.381508. ISBN 978-0-7803-1910-3. S2CID 62509261. ^ 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 Annex E: Distributed Systems ^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 12: Generic Units ^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, Section 6: Subprograms ^ Ada Reference Manual, ISO/IEC 8652:2005(E) Ed. 3, 3.9 Tagged Types and Type Extensions ^ Thread support ^ Atomics support ^ Memory model ^ Gecode ^ SystemC ^ Boost.Iostreams ^ Boolinq ^ "AraRat" (PDF). Archived from the original (PDF) on 2019-08-19. Retrieved 2019-09-15. ^ OpenMPI ^ Boost.MPI ^ Boost.MPL ^ LC++ ^ Castor Archived 2013-01-25 at the Wayback Machine ^ Reflect Library ^ N3534 ^ Boost.Spirit ^ Clojure - Concurrent Programming ^ Clojure - core.async ^ Clojure - Functional Programming ^ Clojure - Macros ^ Clojure - core.logic ^ Clojure - Threading Macros Guide ^ Multimethods and Hierarchies ^ Agents and Asynchronous Actions ^ "concurrency". CLiki. ^ [1] constraint programming inside CL through extensions ^ [2] dataflow extension ^ [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 see Wikipedia article on CLOS, the Common Lisp Object System. ^ implemented by the user via a short macro, example of implementation ^ - Visual programming tool based on Common Lisp ^ [8] rule-based 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. ^ "React – A JavaScript library for building user interfaces". 2019-04-08. ^ "TNG-Hooks". GitHub. 2019-04-08. ^ "Lodash documentation". 2019-04-08. ^ "mori". 2019-04-08. ^ "TNG-Hooks". GitHub. 2019-04-08. ^ "Prolog embedding". Haskell.org. ^ "Functional Reactive Programming". HaskellWiki. ^ Cloud Haskell ^ "Template Haskell". HaskellWiki. ^ "Logict: A backtracking logic-programming monad". Haskell.org. ^ Kollmansberger, Steve; Erwig, Martin (30 May 2006). "Haskell Rules: Embedding Rule Systems in 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. ^ "GitHub - MikeInnes/DataFlow.jl". GitHub. 2019-01-15. ^ "GitHub - JuliaGizmos/Reactive.jl: Reactive programming primitives for Julia". GitHub. 2018-12-28. ^ Query almost anything in julia ^ A collection of 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. ^ Lazy ^ "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 loop iterations in parallel". mathworks.com. Retrieved 21 October 2016. ^ "Execute MATLAB expression in text - MATLAB eval". mathworks.com. Retrieved 21 October 2016. ^ "Determine class of object". mathworks.com. Retrieved 21 October 2016. ^ "Class Metadata". mathworks.com. Retrieved 21 October 2016. ^ "Object-Oriented Programming". mathworks.com. Retrieved 21 October 2016. ^ "Simulink". mathworks.com. Retrieved 21 October 2016. ^ interpreter based threads ^ Higher Order Perl ^ PHP Manual, Chapter 17. Functions ^ PHP Manual, Chapter 19. Classes and Objects (PHP 5) ^ PHP Manual, Anonymous functions ^ "Parallel Processing and Multiprocessing in Python". Python Wiki. Retrieved 21 October 2016. ^ "threading — Higher-level threading interface". docs.python.org. Retrieved 21 October 2016. ^ "python-constraint". pypi.python.org. Retrieved 21 October 2016. ^ "DistributedProgramming". Python Wiki. Retrieved 21 October 2016. ^ "Chapter 9. Metaprogramming". chimera.labs.oreilly.com. Archived from the original on 23 October 2016. Retrieved 22 October 2016. ^ "Metaprogramming". readthedocs.io. Retrieved 22 October 2016. ^ "PEP 443 – Single-dispatch generic functions". python.org. Retrieved 22 October 2016. ^ "PEP 484 – Type Hints". python.org. Retrieved 22 October 2016. ^ "PyDatalog". Retrieved 22 October 2016. ^ "Futureverse". ^ "future batchtools". ^ "Magrittr: A Forward Pipe Operator for R". cran.r-project.org\access-date=13 July 2017. 17 November 2020. ^ Racket Guide: Concurrency and Synchronization ^ The Rosette Guide ^ FrTime: A Language for Reactive Programs ^ Racket Guide: Distributed Places ^ Lazy Racket ^ Channels and other mechanisms ^ "Problem Solver module". ^ Feed operator ^ Cro module ^ "Meta-programming: What, why and how". 2011-12-14. ^ Parametrized Roles ^ "Meta-object protocol (MOP)". ^ Classes and Roles ^ "The 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 Language ^ Scala Language Specification ^ "Tcl Programming/Introduction". en.wikibooks.org. Retrieved 22 October 2016. ^ "TCLLIB - Tcl Standard Library: snitfaq". sourceforge.net. Retrieved 22 October 2016. ^ Notes for Programming Language Experts, Wolfram Language Documentation. ^ External Programs, Wolfram Language Documentation. Jim Coplien, Multiparadigm Design for C++, Addison-Wesley Professional, 1998. Retrieved from " I need to pare two hex values that are ing from a xml tag attribute field, I'm trying this:var fiI need to pare two hex values that are ing from a xml tag attribute field, I'm trying this: var fill = $(this).attr("fill"); // console.log(fill.toString(16)); if (fill === "#FF00FF") But is not working any ideas? I need to pare two hex values that are ing from a xml tag attribute field, I'm trying this: var fill = $(this).attr("fill"); // console.log(fill.toString(16)); if (fill === "#FF00FF") But is not working any ideas? Share Improve this question asked May 12, 2012 at 7:44 Ricardo SanchezRicardo Sanchez 5,1971212 gold badges6161 silver badges9595 bronze badges 7 1 Isn't it redundant to do a toString() when jQuery's attr already returns a string? – Joseph Commented May 12, 2012 at 7:46 I'm doing toString just to see what I get in the console, I'm not using the value still the if statement fails – Ricardo Sanchez Commented May 12, 2012 at 7:47 Your code should work assuming the "fill" attribute looks like "#FF00FF", hash included. – DanRedux Commented May 12, 2012 at 7:49 Anyways, did you try to check what fill is? like typeof fill? did you try to console.log(fill)? what does it result to? – Joseph Commented May 12, 2012 at 7:50 it results in plain #FF00FF and other hex values – Ricardo Sanchez Commented May 12, 2012 at 7:51 | Show 2 more ments 3 Answers 3 Sorted by: Reset to default 1 attr returns a string, there's no need to call toString on it (and the argument will be ignored, because String's toString doesn't take an argument). Your code is assuming a couple of things: That the attribute es back in #hex form (if it's a color value, this is not reliably true cross-browser). That it will be in all upper case. Not knowing what you see when you log the value, I'll just address the second part: var fill = $(this).attr("fill"); if (fill.toUpperCase() === "#FF00FF") I think you have to use 2 equal signs there, try this... var fill = $(this).attr("fill"); if (fill == "#FF00FF") If that doesn't work, then you probably not identifying $(this) If fill is a color, then it might be returned in RGB-format. And when you log it you write toString(). Either pare it with a RGB-value or pare it with a string as fill.toString(16) 发布者：admin，转转请注明出处： You signed in with another tab or window. Reload to refresh your session. You signed out in another tab or window. Reload to refresh your session. You switched accounts on another tab or window. Reload to refresh your session. Dismiss alert Instantly share code, notes, and snippets. Clone this repository at <script src=" quot;></script> Save tiff/887032 to your computer and use it in GitHub Desktop. Clone this repository at <script src=" quot;></script> Save tiff/887032 to your computer and use it in GitHub Desktop. Compare two color values in different formats (rgb, hex, color keyname, ...) You can't perform that action at this time. Is there an existing function to compare a number (0-255) with a hex value that is written in chars, such as char value_in_chars[] = {'A', '2'}; byte number = 0xA2; // or number =162; ? Or is the only solution ASCII arithmetic? To make matters worse, I can not be certain if the value_in_char only contains upper case or maybe also lowercase letters. Background: The value_in_chars is the checksum of a NMEA0183 sentence which is appended to the sentence, hence it's of the type char. To verify the sentence, I calculate my own checksum and get the result as a number. Comparison and Logical operators are used to test for true or false. Comparison Operators Comparison operators are used in logical statements to determine equality or difference between variables or values. Given that x = 5, the table below explains the comparison operators: Comparison operators can be used in conditional statements to compare values and take action depending on the result: if (age < 18) text = "Too young to buy alcohol"; You will learn more about the use of conditional statements in the next chapter of this tutorial. Logical Operators Logical operators are used to determine the logic between variables or values. Given that x = 6 and y = 3, the table below explains the logical operators: Operator Description Example Try it && and (x < 10 && y > 1) is true Try it » || or (x == 5 || y == 5) is false Try it » ! not !(x == y) is true Try it » Conditional (Ternary) Operator JavaScript also contains a conditional operator that assigns a value to a variable based on some condition. Syntax variablename = (condition) ? value1:value2 Example let voteable = (age < 18) ? "Too young":"Old enough"; Try it Yourself » If the variable age is a value below 18, the value of the variable voteable will be "Too young", otherwise the value of voteable will be "Old enough". Comparing Different Types Comparing data of different types may give unexpected results. When comparing a string with a number, JavaScript will convert the string to a number when doing the comparison. An empty string converts to 0. A non-numeric string converts to NaN which is always false. When comparing two strings, "2" will be greater than "12", because (alphabetically) 1 is less than 2. To secure a proper result, variables should be converted to the proper type before comparison: age = Number(age); if (isNaN(age)) { voteable = "Input is not a number"; } else { voteable = (age < 18) ? "Too young" : "Old enough"; } Try it Yourself » The Nullish Coalescing Operator (??) The ?? operator returns the first argument if it is not nullish (null or undefined). Otherwise it returns the second argument. The nullish operator is supported in all browsers since March 2020: Chrome 80 Edge 80 Firefox 72 Safari 13.1 Opera 67 Feb 2020 Feb 2020 Jan 2020 Mar 2020 Mar 2020 The Optional Chaining Operator (?.) The ?. operator returns undefined if an object is undefined or null (instead of throwing an error). // Create an object: const car = {type:"Fiat", model:"500", color:"white"}; // Ask for car name: document.getElementById("demo").innerHTML = car?.name; Try it Yourself » The optional chaining operator is supported in all browsers since March 2020: Chrome 80 Edge 80 Firefox 72 Safari 13.1 Opera 67 Feb 2020 Feb 2020 Jan 2020 Mar 2020 Mar 2020 Choose the correct comparison operator to alert true, when x is greater than y. x = 10; y = 5; alert(x @(1) y); x = 10; y = 5; alert(x > y); Correct! Next ❯ What i want to do is to check the elements of array which are bytes are greater or less than some hex value? so i'm trying write like this -> bytes[] chars = new bytes[] {some hex values}; foreach(byte b in chars){ if(b> hex value && b< hex halue) ---> some code } Input your age and click the button: How can financial brands set themselves apart through visual storytelling? Our experts explain how.Learn MoreThe Motorsport Images Collections captures events from 1895 to today's most recent coverage.Discover The CollectionCurated, compelling, and worth your time. Explore our latest gallery of Editors' Picks.Browse Editors' FavoritesHow can financial brands set themselves apart through visual storytelling? Our experts explain how.Learn MoreThe Motorsport Images Collections captures events from 1895 to today's most recent coverage.Discover The CollectionCurated, compelling, and worth your time. Explore our latest gallery of Editors' Picks.Browse Editors' FavoritesHow can financial brands set themselves apart through visual storytelling? Our experts explain how.Learn MoreThe Motorsport Images Collections captures events from 1895 to today's most recent coverage.Discover The CollectionCurated, compelling, and worth your time. Explore our latest gallery of Editors' Picks.Browse Editors' Favorites Share — copy and redistribute the material in any medium or format for any purpose, even commercially. Adapt — remix, transform, and build upon the material 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 licensor 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.