**Language Map for JavaScript**

|  |  |
| --- | --- |
| **Variable Declaration**  *Is this language strongly typed or dynamically typed? Provide at least three examples (with different data types or keywords) of how variables are declared in this language.* | **JavaScript is Dynamically-typed language** where the interpreter assigns variables a type at runtime based on the variable's value at the time. There are Four Ways to Declare a JavaScript Variable:  * Using var * Using let * Using const * Using nothing   Examples:  const price1 = 5;  let total = 4;  var carName;  you cannot redeclare variable with let.  let x = "John Doe";  let x = 0;  // SyntaxError: 'x' has already been declared  with var you can:  var x = "John Doe";  var x = 0; |
| **Data Types**  *List all of the data types (and ranges) supported by this language.* | JavaScript variables can hold different data types: numbers, strings, objects , Boolean, undefined, and null  For Example:  let x=null; // null  let length = 16;                               // Number let lastName = "Johnson";                      // String  var val6; // undefined  var val3=true; // boolean  let x = {firstName:"John", lastName:"Doe"};    // Object  JavaScript has dynamic types. This means that the same variable can be used to hold different data types:  For Example:  let x;           // Now x is undefined x = 5;           // Now x is a Number x = "John";      // Now x is a String  JavaScript does support Booleans as well usually for conditional statements |
| **Selection Structures**  *Provide examples of all selection structures supported by this language (if, if else, etc.)* ***Don’t just list them, show code samples of how each would look in a real program.*** | 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   **Example of an if statement in JavaScript:**  if (new Date(2020,11,5,12).getHours() < 18) {  document.getElementById("demo").innerHTML = "Good day!";  }// output is Good day!  **Example of an else statement in JavaScript:**  const hour = new Date(2020,11,5,19).getHours();  let greeting;  if (hour < 18) {  greeting = "Good day";  } else {  greeting = "Good evening";  }  document.getElementById("demo").innerHTML = greeting;}// output is Good evening  **Example of an else if statement in JavaScript:**  const time = new Date(2020,11,5,23).getHours();  let greeting;  if (time < 10) {  greeting = "Good morning";  } else if (time < 20) {  greeting = "Good day";  } else {  greeting = "Good evening";  }// output is Good evening  **Example of a switch statement in JavaScript:**  let day;  switch (new Date(2020,11,5,23).getDay()) {  case 0:  day = "Sunday";  break;  case 1:  day = "Monday";  break;  case 2:  day = "Tuesday";  break;  case 3:  day = "Wednesday";  break;  case 4:  day = "Thursday";  break;  case 5:  day = "Friday";  break;  case 6:  day = "Saturday";  }  document.getElementById("demo").innerHTML = "Today is " + day; // output is today is Saturday |
| **Repetition Structures**  *Provide examples of all repetition structures supported by this language (loops, etc.)* ***Don’t just list them, show code samples of how each would look in a real program.*** | Javascript supports the following repetition structures:   * for - loops through a block of code a number of times * for/in - loops through the properties of an object * for/of - loops through the values of an iterable object * while - loops through a block of code while a specified condition is true * do/while - also loops through a block of code while a specified condition is true   **Example of a for loop in JavaScript:**  let text = "";  for (let i = 0; i < 5; i++) {  text += "The number is " + i + "<br>";  }  document.getElementById("demo").innerHTML = text;  /\*output  The number is 0 The number is 1 The number is 2 The number is 3 The number is 4  \*/  **Example of a for/in loop in JavaScript:**  const numbers = [45, 4, 9, 16, 25];  let txt = "";  for (let x in numbers) {  txt += numbers[x] + "<br>";  }  /\*output  45  4  9  16  25  \*/  **Example of the for/of statement in JavaScript:**  const cars = ["Jeep", "Ford", "Mini"];  let text = "";  for (let x of cars) {  text += x + "<br>";  }  /\*output  Jeep  Ford  Mini  \*/  **Example of a while statement in JavaScript:**  let text=0;  let i = 1;  while (i < 10) {  text =text+ "<br>" +i;  i++;  }  /\*output  1  2  3  4  5  6  7  8  9  \*/  **Example of a do/while statement in JavaScript:**  let text = ""  let i = 0;  do {  text += "<br>The number is " + i;  i++;  }  while (i < 10);  /\*output  The number is 0 The number is 1 The number is 2 The number is 3 The number is 4 The number is 5 The number is 6 The number is 7 The number is 8 The number is 9  \*/ |
| **Arrays**  *If this language supports arrays, provide at least two examples of creating an array with a primitive or String data types (e.g. float, int, String, etc.)* | Yes, Javascript supports arrays.  const nums = new Array(1, 2, 3);//output 1,2,3  const cars = new Array("ford", "mustang", "camaro");//output ford,mustang,camaro |
| **Data Structures**  *If this language provides a standard set of data structures, provide a list of the data structures and their Big-Oh complexity.* | | **Name** | **Insert** | **Access** | **Search** | **Delete** | **Comments** | | --- | --- | --- | --- | --- | --- | | [Array](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Array) | [O(n)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Insert-element-on-an-array) | [O(1)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Access-an-element-in-an-array) | [O(n)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Search-an-element-in-an-array) | [O(n)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Deleting-elements-from-an-array) | Insertion to the end is O(1). [Details here.](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Array-operations-time-complexity) | | [HashMap](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#HashMaps) | [O(1)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Insert-element-on-a-HashMap-runtime) | [O(1)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Search-Access-an-element-on-a-HashMap-runtime) | [O(1)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Search-Access-an-element-on-a-HashMap-runtime) | [O(1)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Edit-Delete-element-on-a-HashMap-runtime) | Rehashing might affect insertion time. [Details here.](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#HashMap-operations-time-complexity) | | Map (using Binary Search Tree) | O(log(n)) | - | O(log(n)) | O(log(n)) | Implemented using Binary Search Tree | | [Set (using HashMap)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Sets) | [O(1)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Set-Implementation) | - | [O(1)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Set-Implementation) | [O(1)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Set-Implementation) | Set using a HashMap implementation. [Details here.](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Set-Operations-runtime) | | Set (using list) | [O(n)](https://www.ecma-international.org/ecma-262/6.0/#sec-set.prototype.add) | - | [O(n)](https://www.ecma-international.org/ecma-262/6.0/#sec-set.prototype.has) | [O(n)](https://www.ecma-international.org/ecma-262/6.0/#sec-set.prototype.delete) | Implemented using Binary Search Tree | | Set (using Binary Search Tree) | O(log(n)) | - | O(log(n)) | O(log(n)) | Implemented using Binary Search Tree | | [Linked List (singly)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Singly-Linked-Lists) | [O(n)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#SinglyLinkedList.addLast) | - | [O(n)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#LinkedList.contains) | [O(n)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#LinkedList.remove) | Adding/Removing to the start of the list is O(1). [Details here](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Singly-Linked-Lists-time-complexity). | | [Linked List (doubly)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Doubly-Linked-Lists) | [O(n)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#DoublyLinkedList.add) | - | [O(n)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#LinkedList.contains) | [O(n)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#LinkedList.remove) | Adding/Deleting from the beginning/end is O(1). But, deleting/adding from the middle is O(n). [Details here](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Doubly-Linked-Lists-time-complexity) | | [Stack (array implementation)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Stacks) | [O(1)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Stacks) | - | - | [O(1)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#Stacks) | Insert/delete is last-in, first-out (LIFO) | | [Queue (naïve array impl.)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#QueueNaiveImpl) | [O(1)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#QueueNaiveImpl) | - | - | [O(n)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#QueueNaiveImpl) | Remove (Array.shift) is *O(n)* | | [Queue (array implementation)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#QueueArrayImpl) | [O(1)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#QueueArrayImpl) | - | - | [O(1)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#QueueArrayImpl) | Worst time insert is O(n). However amortized is O(1) | | [Queue (list implementation)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#QueueListImpl) | [O(1)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#QueueListImpl) | - | - | [O(1)](https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/#QueueListImpl) | Using Doubly Linked List with reference to the last eleme | |
| **Objects**  *If this language support object-orientation, provide an example of how you would write a simple object with a default constructor and then how you would instantiate it.* | Yes, Javascript supports creating objects  Almost everything except primitives are considered objects in JS.  // Constructor function for the person object  function Person(first, last, age, eye) {  this.firstName = first;  this.lastName = last;  this.age = age;  this.eyeColor = eye;  }  // create a person object  Create a Person object  const myFather = new Person("John", "Doe", 50, "blue"); |
| **Runtime Environment**  *What runtime environment does this language compile to? For example, Java compiles to the Java Virtual Machine.*  *Do other languages also compile to this runtime?* | When you visit a website you do so within a web browser, like Chrome, Firefox, Edge, or Safari. Each browser has a JS Runtime Environment. In the environment are Web API’s that a developer can access to build a program.  Also in the runtime environment is a Javascript Engine that parses the code. Each browser has its own version of a JS engine. Chrome uses what it calls its V8 JS Engine and that is what we will analyze now. |
| **Libraries/Frameworks**  *What are the popular libraries or frameworks used by programmers for this language? List at least three (3) and describe what they are used for..* | 3 most common frameworks used is Angular, React, and Vue  Angular is One of the most powerful, efficient, and open-source JavaScript frameworks is Angular. Google operates this framework and is implemented to use for [developing a Single Page Application (SPA)](https://hackr.io/blog/why-should-you-learn-angular#angular-supports-single-page-applications). It extends the HTML into the application and interprets the attributes to perform data binding.  React was created by Facebook, the React framework has earned popularity within a short period. It is used to develop and operate the dynamic User Interface of the web pages with high incoming traffic. It makes the use of a virtual DOM, and hence, the integration of the same with any application is more straightforward.  Although Vue was developed in the year 2016, this JavaScript framework has already made its way into the market and has proven its worth by offering various features. Its dual integration mode is one of the most attractive features for creating high-end SPA or Single Page Application. It is a much reliable platform for developing cross-platform. |
| **Domains**  *What industries or domains use this programming language? Provide specific examples of companies that use this language and what they use it for****. E.g. Company X uses C# for its line of business applications.*** | Recently, **Microsoft** has really embraced NodeJS. They thoroughly support Node on the Azure cloud platform. Its one of Azure’s major features, and they’ve integrated Visual Studio support for Node. Microsoft has also developed a version of Node for (IoT) applications. NodeJS is great of IoT because it’s light weight and efficient.  **Pay**Pal has been using JavaScript on the front end of their website for a long time, but that’s only the beginning. The online payment giant was one of the earliest adopters of NodeJS. During an overhaul of their account overview page, they decided to try building the page in Node at the same time as their usual Java development. The NodeJS version worked out so well, that they chose to use it in production and build all client-facing applications in Node going forward. That means that most of what you see in your account is running on Node. |