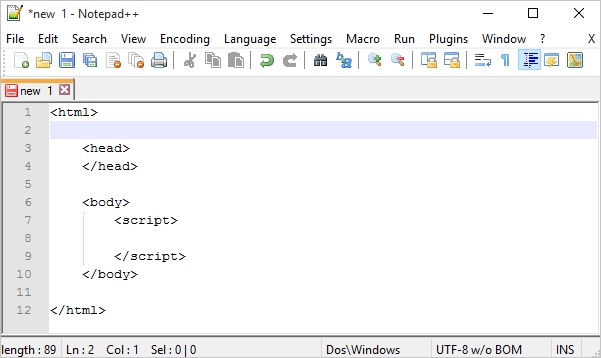
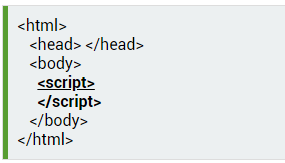
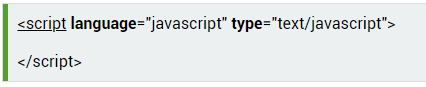
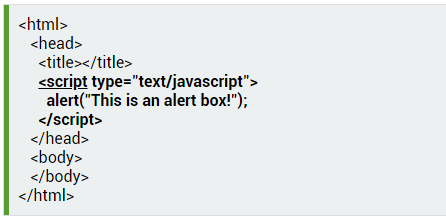
**Introduction**

* JavaScript is used to add interactivity to webpages, process data, as well as create various applications (mobile apps, desktop apps, games, and more).
* JavaScript on the web lives inside the **HTML**document.
* In HTML, JavaScript code must be inserted between **<script>** and **</script>**tags.
* JavaScript can be placed in the HTML page's **<body>** and **<head>**sections.
* 
* It's a good idea to place scripts at the bottom of the <body> element.
* This can improve page load, because HTML display is not blocked by scripts loading.
* 
* The <script> tag can take two attributes, language and type, which specify the script's type:
* 
* Every written "instruction" is called a statement. JavaScript statements are separated by semicolons.
* In the example below, we created an alert box inside the script tag, using the **alert()** function.
* 
* Scripts can also be placed in**external files**. External scripts are useful and practical when the same code is used in a number of different web pages. JavaScript files have the **file extension .js**.
* To use an external script, put the name of the script file in the **src**(source) attribute of the <script>tag.
* 
* External scripts cannot contain <script> tags.
* Placing a JavaScript in an external file has the following advantages:

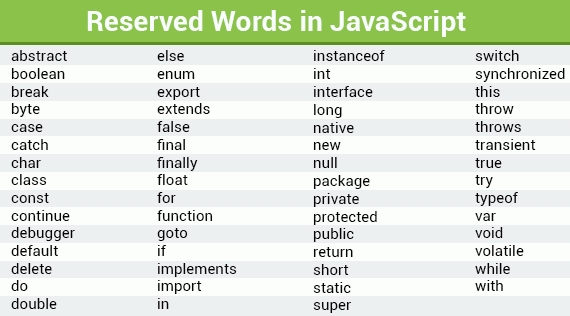
- It separates HTML and code.

- It makes HTML and JavaScript easier to read and maintain.

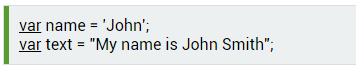
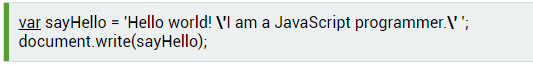
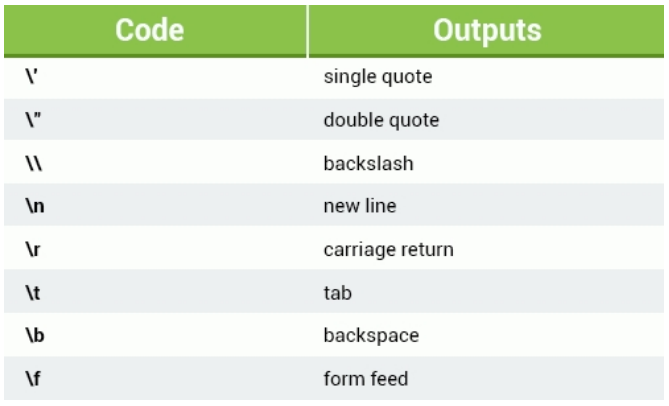
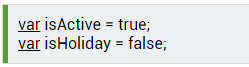
- Cached JavaScript files can speed up page loads

* Code after a double slash //, or between /\* and \*/, is treated as a **comment**.
* **Single line** comments use double slashes.
* Everything you write between /\*and \*/ will be considered as a multi-line comment.

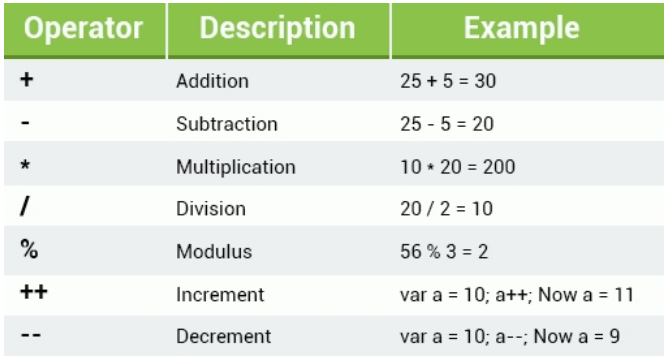
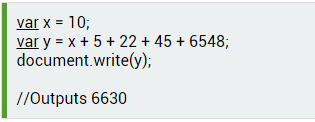
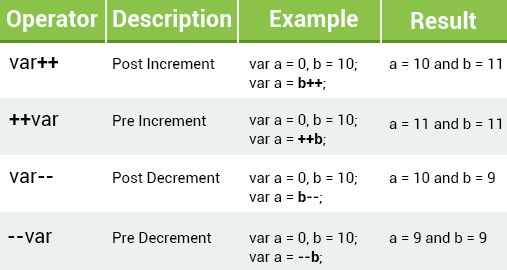
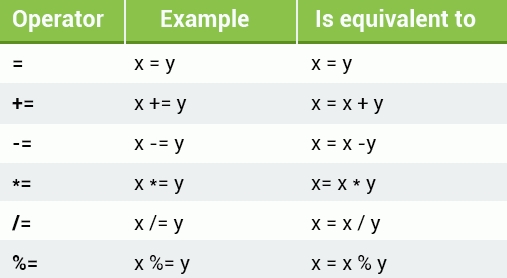
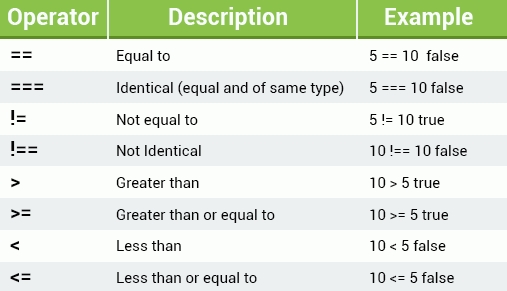
**Basic Concepts**

* Variables are containers for storing data values.
* Use the **var**keyword to declare a variable.
* JavaScript is case sensitive. For example, the variables lastName and lastname, are two different variables.
* In JavaScript, the equal sign (=) is called the "**assignment**" operator, rather than an "equal to" operator.  
  For example, **x = y** will assign the value of **y** to **x**.
* A variable can be declared without a value. A variable declared without a value will have the value undefined.
* Variable Naming rules:  
  - The first character **must be** a letter, an underscore (\_), or a dollar sign ($). Subsequent characters may be letters, digits, underscores, or dollar signs.  
  - Numbers are **not allowed** as the first character.  
  - Variable names**cannot**include a **mathematical or logical operator**in the name. For instance, *2\*something* or *this+that*;  
  - JavaScript names **must not contain spaces**.
* Hyphens are not allowed in JavaScript. It is reserved for subtractions.
* You **must not** use any **special symbols**, like *my#num,* *num%*, etc.
* Be sure that you do not use any of the following JavaScript reserved words.
* 

**Data Types**

* The term **data type** refers to the types of values with which a program can work.
* Unlike many other programming languages, JavaScript does not define different types of numbers, like integers, short, long, floating-point, etc.
* 
* JavaScript numbers are always stored as double precision floating point numbers.
* A string can be any text that appears within **quotes**. You can use single or double quotes.
* 
* As strings must be written within quotes, quotes inside the string must be handled. The **backslash (\) escape character** turns special characters into string characters.
* 
* The escape character (\) can also be used to insert other special characters into a string.  
  These special characters can be added to a text string using the backslash sign.
* 
* In JavaScript Boolean, you can have one of two values, either **true**or **false**.   
  These are useful when you need a data type that can only have one of two values, such as Yes/No, On/Off, True/False.
* The Boolean value of 0 (zero), null, undefined, empty string is false. Everything with a "real" value is true.
* 

**Arithmetic Operators**

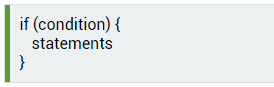
* Arithmetic operators perform arithmetic functions on numbers.
* 
* The addition operator is used to determine the sum of two numbers.
* 
* **Increment ++ / Decrement --**  
  The increment operator increments the numeric value of its operand by one. If placed before the operand, it returns the incremented value. If placed after the operand, it returns the original value and then increments the operand.
* 
* Assignment operators assign values to JavaScript variables.
* 
* The table below explains the comparison operators.
* 

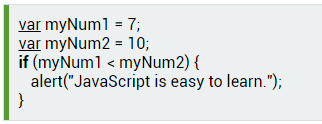
**Conditional (Ternary) Operator**

* Another JavaScript conditional operator assigns a value to a variable, based on some condition.
* Syntax: variable = (condition) ? value1: value2
* 

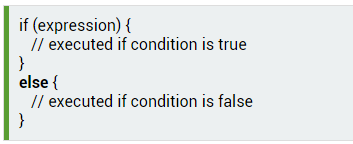
**Conditionals and Loops**

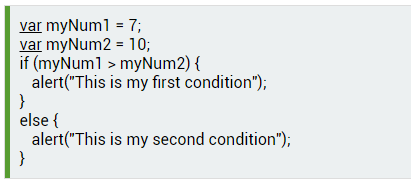
* Very often when you write code, you want to perform different actions based on different conditions.
* If statement 🡪



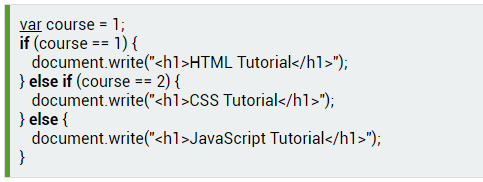


* The else Statement 🡪 Use the **else**statement to specify a block of code that will execute if the condition is **false**.

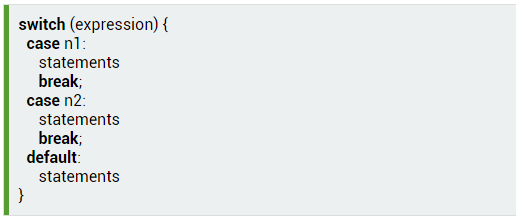




* else if 🡪 You can use the **else if statement** to specify a new condition if the first condition is false.

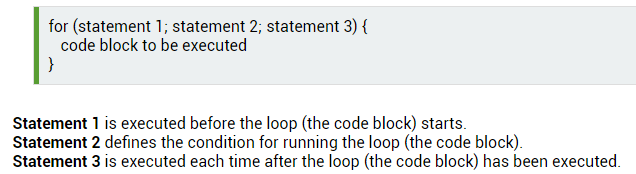


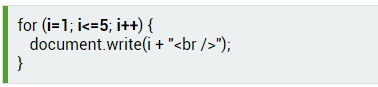
* Switch 🡪 In cases when you need to test for multiple conditions, writing**if else** statements for each condition might not be the best solution. The **switch statement** is used to perform different actions based on different conditions.



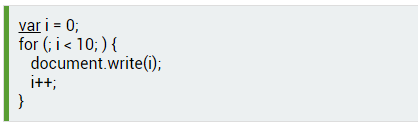
The switch expression is evaluated once. The value of the expression is compared with the values of each **case**. If there is a match, the associated block of code is executed.

* When JavaScript code reaches a **break**keyword, it breaks out of the switch block. This will stop the execution of more code and case testing inside the block. Usually, a break should be put in each case statement.
* **Loops**
* Loops can execute a block of code a number of times. They are handy in cases in which you want to run the same code repeatedly, adding a different value each time.
* The **for**loop is commonly used when creating a loop.

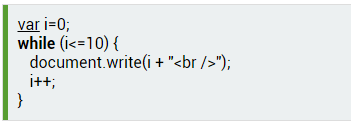




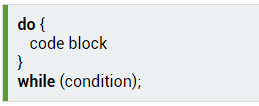
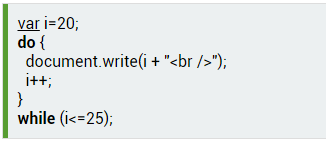
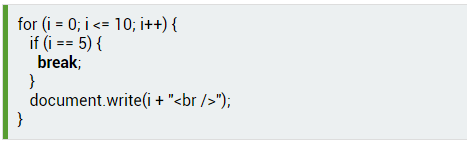
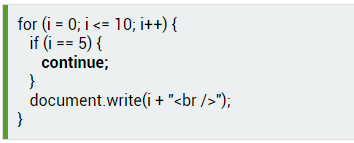
* If you omit statement 2, you must provide a break inside the loop. Otherwise, the loop will never end.
* **Statement 3** is used to change the initial variable. Statement 3 is also optional, and it can be omitted if you increment your values inside the loop.



* The **while**loop repeats through a block of code, as long as a specified condition is **true**.  
  

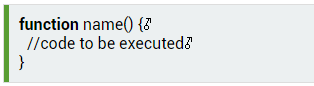


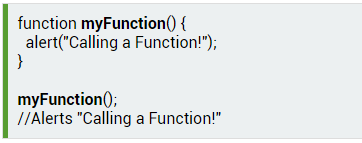
If you forget to increase the variable used in the condition, the loop will never end.

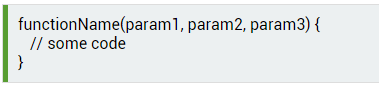
* The **do...while** loop is a variant of the while loop. This loop will execute the code block once, **before**checking if the condition is true, and then it will repeat the loop as long as the condition is true.
* The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested.
* 
* Note the semicolon used at the end of the do...while loop.
* 
* The **break**statement "jumps out" of a loop and continues executing the code after the loop.
* 
* The **continue**statement breaks only one iteration in the loop, and continues with the next iteration.
* 

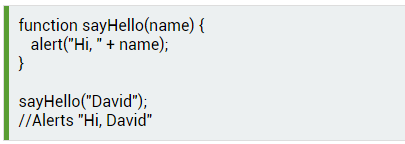
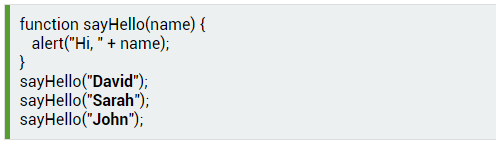
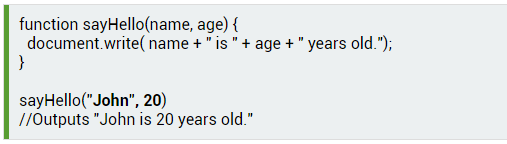
**Functions**

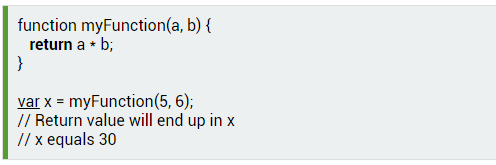
* A JavaScript **function**is a block of code designed to perform a particular task.  
  The main advantages of using functions:  
  Code **reuse**: Define the code once, and use it many times.  
  Use the same code many times with different **arguments**, to produce different results.
* To define a JavaScript function, use the **function**keyword, followed by a **name**, followed by a set of **parentheses ()**. The code to be executed by the function is placed inside curly brackets {}.

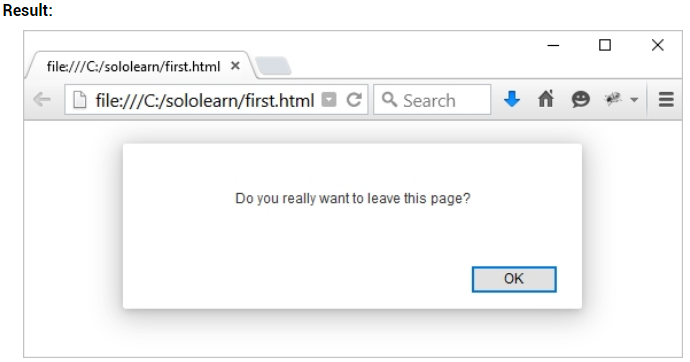
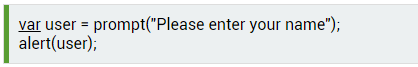
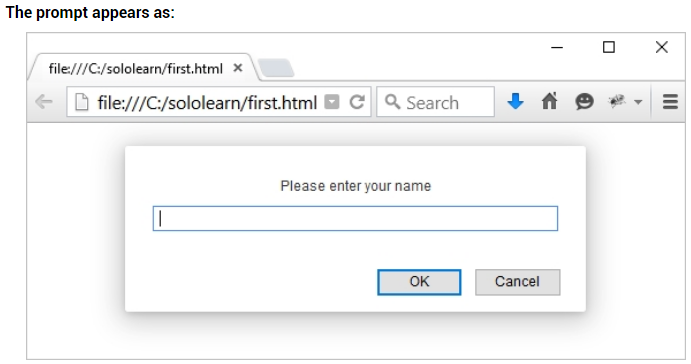
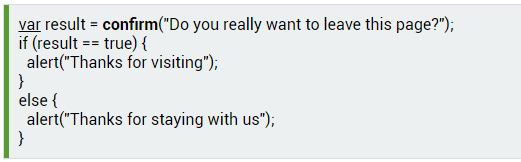
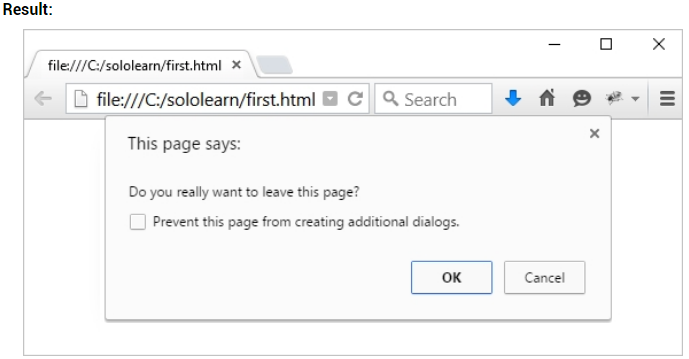


* Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
* To execute the function, you need to call it. To call a function, start with the name of the function, then follow it with the arguments in parentheses.
* 
* Always remember to end the statement with a semicolon after calling the function.
* Functions can take **parameters**. Function **parameters**are the names listed in the function's definition.



* As with variables, parameters should be given names, which are separated by commas within the parentheses.
* After defining the parameters, you can use them inside the function.
* 
* You can define a single function, and pass different parameter values (arguments) to it.
* 
* You can define multiple parameters for a function by **comma-separating** them.
* 
* If a function is called with missing arguments (fewer than declared), the missing values are set to undefined, which indicates that a variable has not been assigned a value.
* A function can have an optional return statement. It is used to return a value from the function.
* This statement is useful when making calculations that require a result. When JavaScript reaches a return statement, the function stops executing.

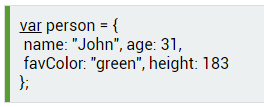


* If you do not return anything from a function, it will return undefined.
* JavaScript offers three types of popup boxes, the **Alert**, **Prompt**, and **Confirm**boxes.
* An**alert box** is used when you want to ensure that information gets through to the user.  
  When an alert box pops up, the user must click OK to proceed. The **alert**function takes a single parameter, which is the text displayed in the popup box.
* Be careful when using alert boxes, as the user can continue using the page only after clicking OK.
* 
* 
* A **prompt box** is often used to have the user input a value before entering a page.  
  When a prompt box pops up, the user will have to click either OK or Cancel to proceed after entering the input value. If the user clicks OK, the box**returns the input value**. If the user clicks Cancel, the box returns **null**.
* The **prompt()** method takes **two parameters**.   
  - The first is the label, which you want to display in the text box.  
  - The second is a default string to display in the text box (optional).
* 
* 
* A **confirm box** is often used to have the user verify or accept something. When a confirm box pops up, the user must click either OK or Cancel to proceed. If the user clicks OK, the box returns **true**. If the user clicks Cancel, the box returns **false**.
* 
* 

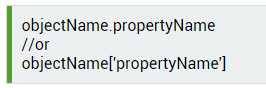
**Objects**

JavaScript variables are containers for data values. **Objects**are variables too, but they can contain many values.

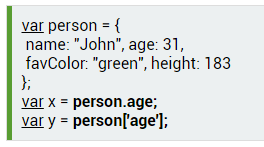
Think of an object as a list of values that are written as **name:value** pairs, with the names and the values separated by colons.



You can access object properties in two ways.



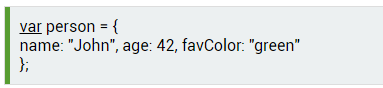
This example demonstrates how to access the age of our person object.



**Object Methods**

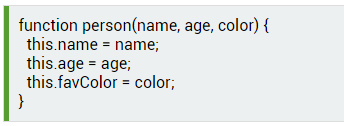
An object **method**is a property that contains a **function definition**. Use the following syntax to access an object method.

**Creating an object using the object literal (or initializer) syntax.**



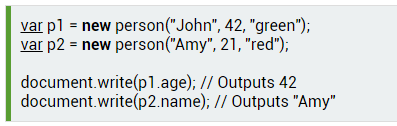
**Object Constructor**

Sometimes, we need to set an "**object type**" that can be used to create a number of objects of a single type. The standard way to create an "object type" is to use an object **constructor function**.



The this keyword refers to the current object. Note that this is not a variable. It is a keyword, and its value cannot be changed.

Once you have an object constructor, you can use the **new**keyword to create new objects of the same type.



p1 and p2 are now objects of the person type. Their properties are assigned to the corresponding values.

**Methods**

**Methods**are functions that are stored as object properties.

Use the following syntax to create an object method:

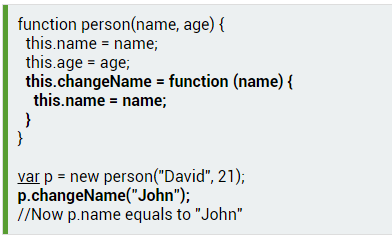


Access an object method using the following syntax:



A method is a function, belonging to an object. It can be referenced using the **this**keyword.  
The **this**keyword is used as a reference to the current object.

Defining methods is done inside the constructor function.



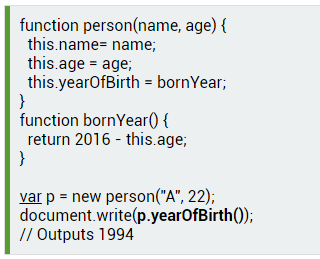
In the example above, we have defined a method named changeName for our person, which is a function, that takes a parameter name and assigns it to the name property of the object.

this.name refers to the name property of the object.

The changeName method changes the object's name property to its argument.

**Calling the method**

Call the method by the property name you specified in the constructor function, rather than the function name.

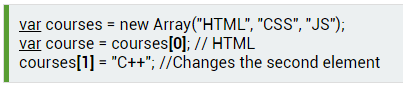


**Core Objects : Array**

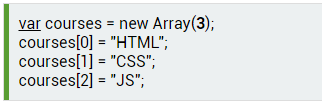
Arrays store multiple values in a single variable.



You refer to an array element by referring to the index number written in square brackets. [0] is the first element in an array. [1] is the second. Array indexes start with 0.



You can also declare an array, tell it the number of elements it will store, and add the elements later.



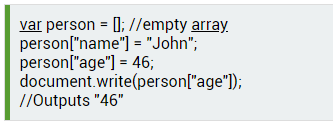
An array is a special type of object. An array uses numbers to access its elements, and an object uses names to access its members.

For greater simplicity, readability, and execution speed, you can also declare arrays using the array literal syntax.



# Associative Arrays

While many programming languages support arrays with named indexes (text instead of numbers), called **associative arrays**,JavaScript **does not**. However, you still can use the named array syntax, which will produce an object.



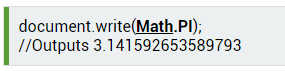
Now, person is treated as an object, instead of being an array. The named indexes "name" and "age" become properties of the person object.

As the person array is treated as an object, the standard array methods and properties will produce incorrect results. For example, person.length will return 0.

It is better to use an object when you want the index to be a string (text). Use an array when you want the index to be a number.

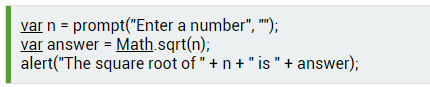
# The Math Object

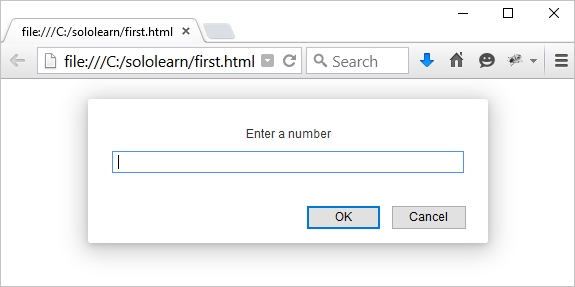
The Math object allows you to perform mathematical tasks, and includes several properties.



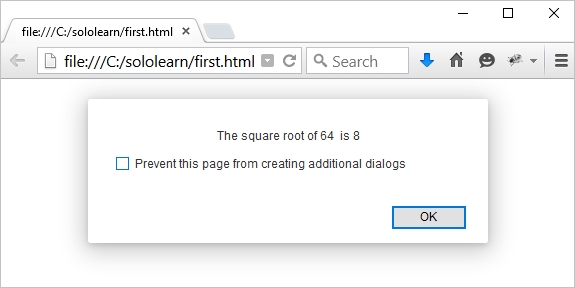
Math has no constructor. There's no need to create a Math object first.

Let's create a program that will ask the user to input a number and alert its square root.



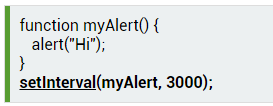


Enter a number, such as 64.



# setInterval

The **setInterval()** method calls a function or evaluates an expression at specified intervals (in milliseconds). It will continue calling the function until **clearInterval()** is called or the window is closed.



This will call the myAlert function every 3 seconds (1000 ms = 1 second). Write the name of the function without parentheses when passing it into the setInterval method.