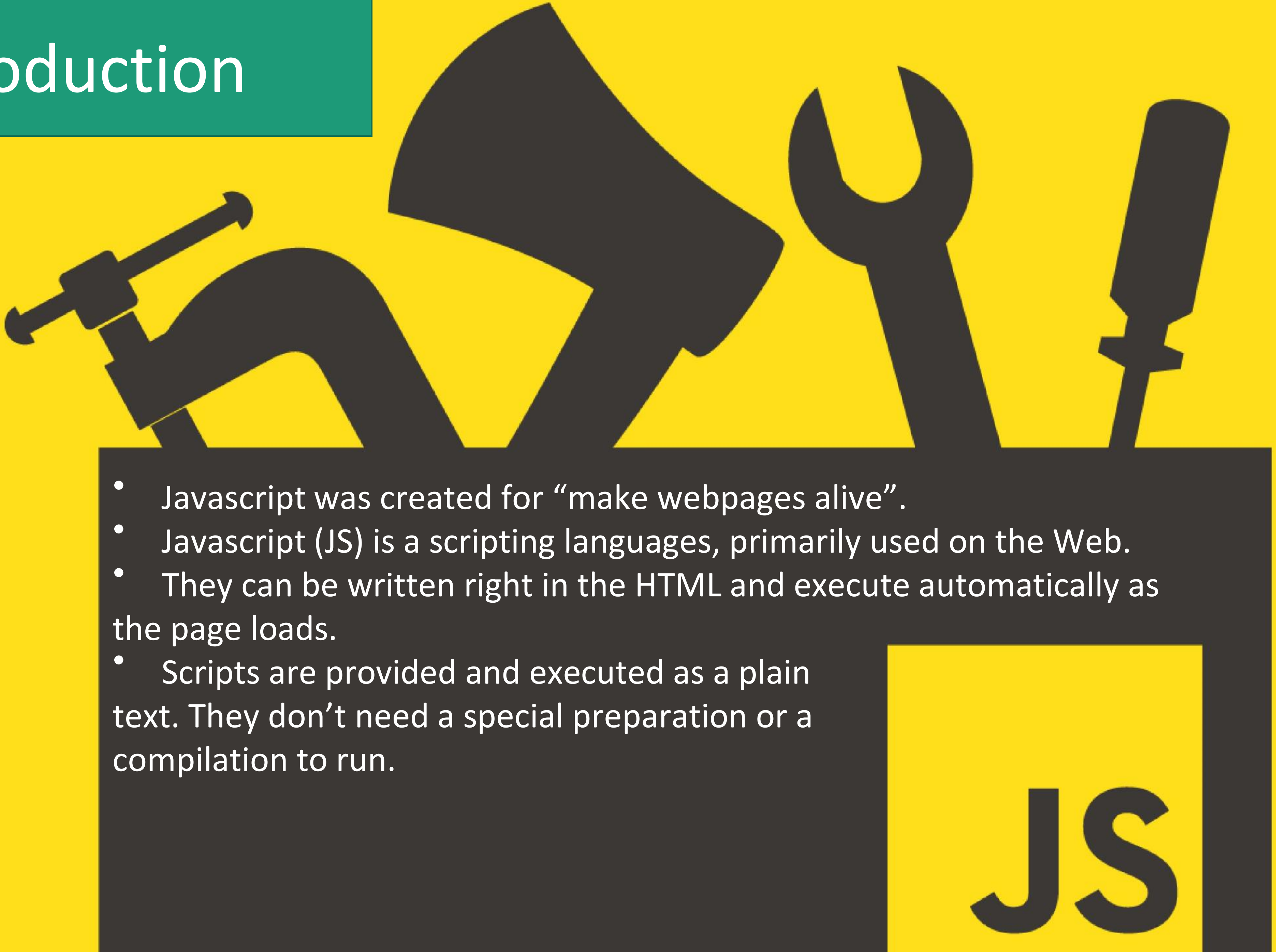
# JavaScript

## Introduction



- Javascript was created for “make webpages alive”.
- Javascript (JS) is a scripting languages, primarily used on the Web.
- They can be written right in the HTML and execute automatically as the page loads.
- Scripts are provided and executed as a plain text. They don’t need a special preparation or a compilation to run.

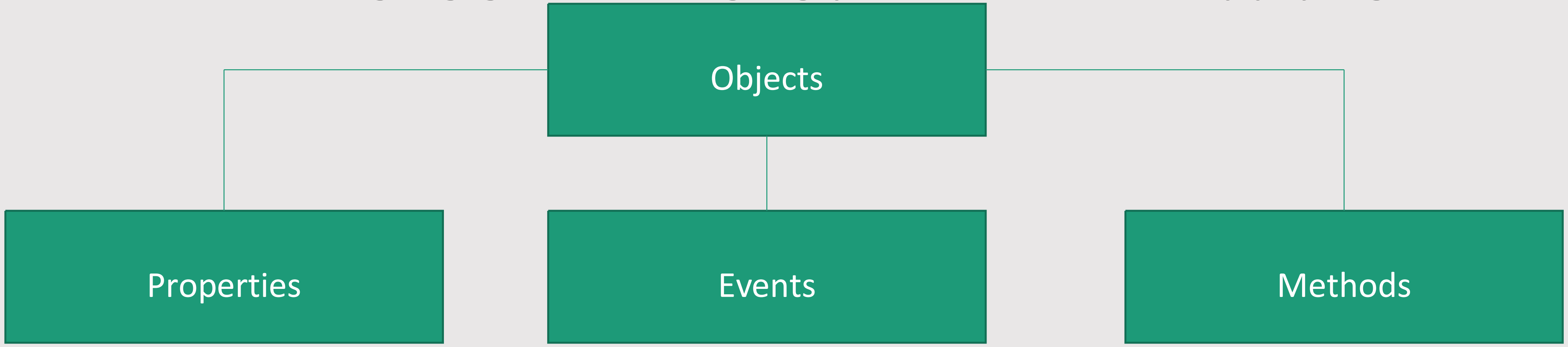
## JavaScript advantages

- Allowing the pages to react to events, exhibit special effects, accept variable text, validate data, create cookies, detect a user’s browser, etc.
- JavaScript renders web pages in an interactive and dynamic fashion.

*Definition source: https://www.techopedia.com/definition/3929/javascript-js*

## Objects

+ Object is the main point of object – oriented technology. In the real life, we have many objects: your dog, your bicycle, etc. Moreover, each object has two characteristic: Dogs have state (name, color, breed, hungry) and behavior (barking, fetching, wagging tail). Bicycles also have state (current gear, current pedal cadence, current speed) and behavior (changing gear, changing pedal cadence, applying brakes).



## Objects – Properties, Events

+ As I said, dog is an object.

+ A dog has **properties** like name, weight and color, and **events** like barking, fetching and wagging tail.

Object Properties Events

dog.name = corgi dog.barking() dog.weight = 5kg dog.fetching() dog.color = yellow dog.wagging()

# Objects - Methods

+ Methods are the represents of the things people need to do with objects. They can retrieve or update the values of an object’s properties.

changeWeight()

## How JS works along with HTML

External

Internal

Inline

It is a separate .js file and we link that file with your HTML document. It is more easier to manage and maintain.
It increases the speed of page load.

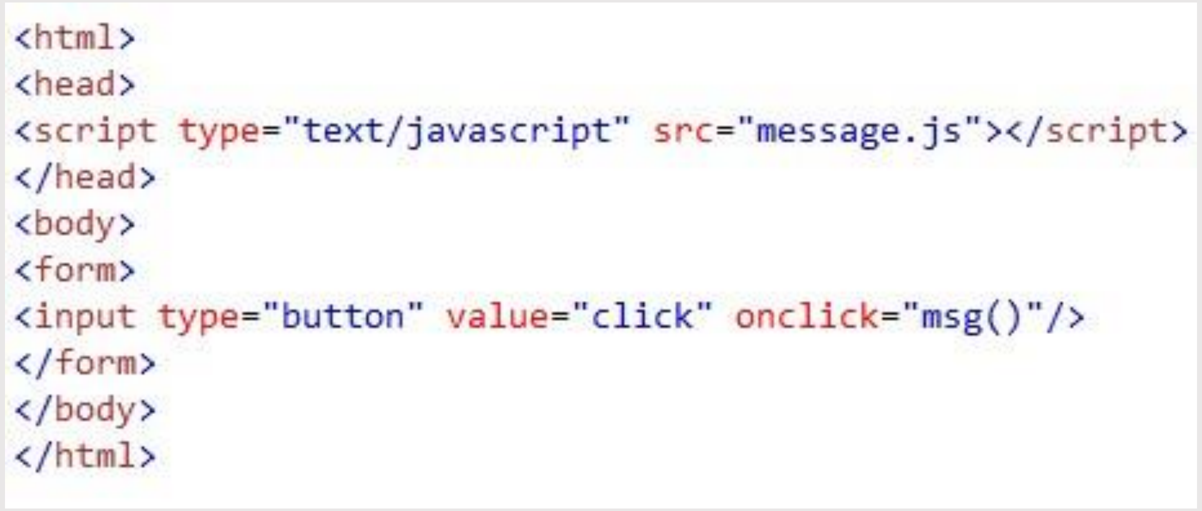
Embedded the JS code internally within your HTML document With the<script> tag

The JS code is applied directly to the current element.

## External JAVASCRIPT

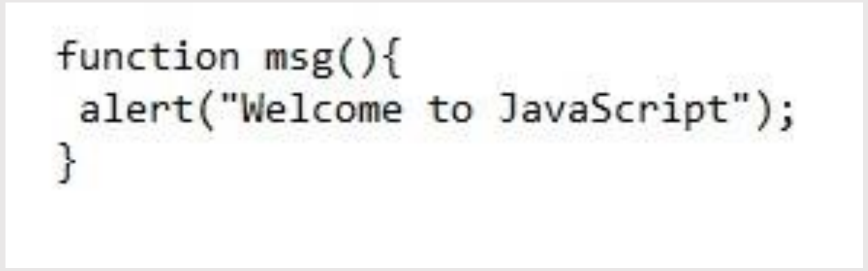
## Putting the name of the script file in src attribute of <script> tag.

example.html



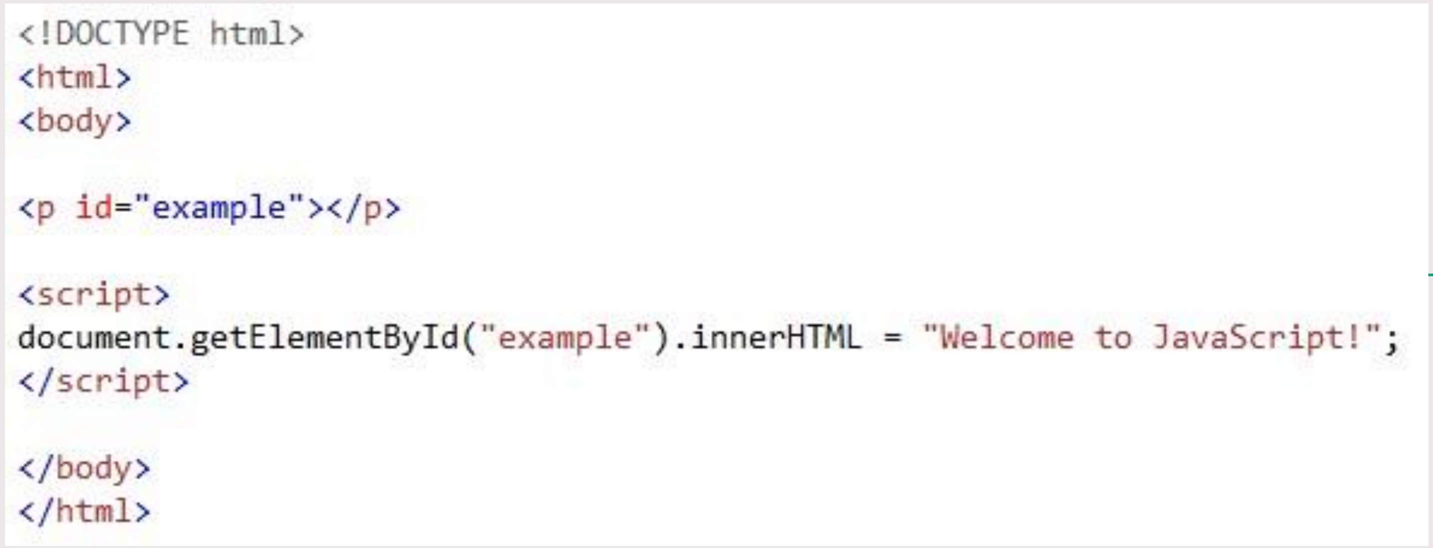
```html
<html>
<head>
<script type="text/javascript" src="message.js"></script>
</head>
<body>
<form>
<input type="button" value="click" onclick="msg()"/>
</form>
</body>
</html>
```

example.js



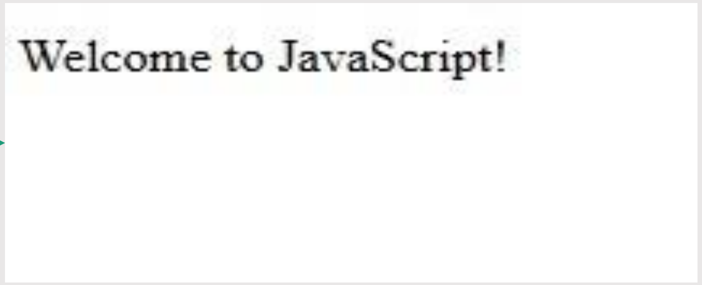
```javascript
function msg(){
 alert("Welcome to JavaScript");
}
```

## Internal JAVASCRIPT



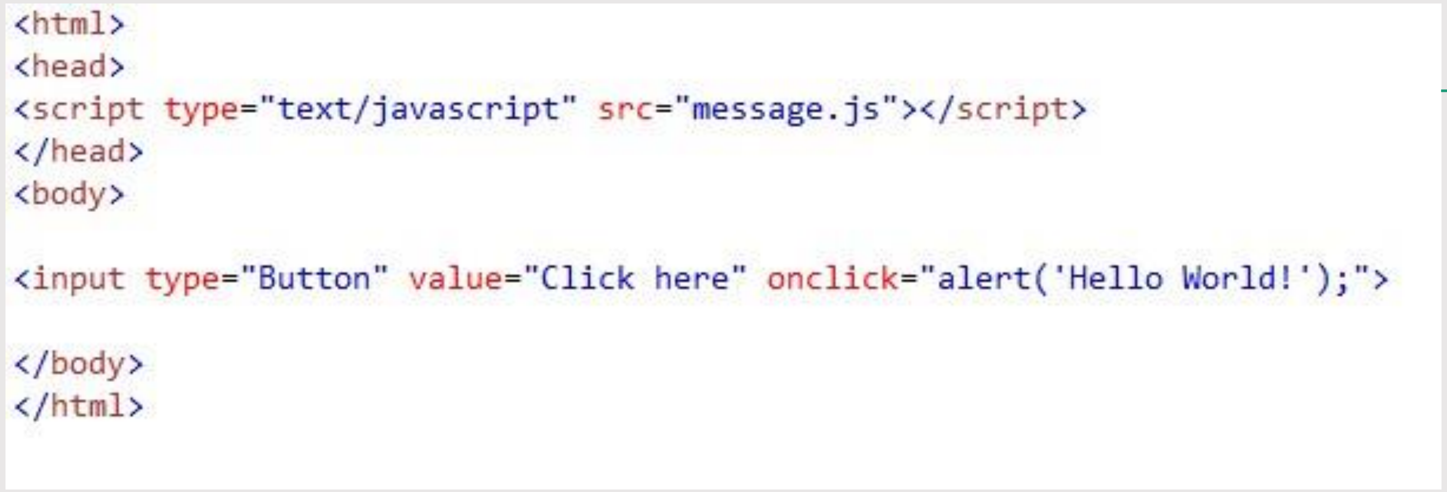
```
<!DOCTYPE html>
<html>
<body>

<p id="example"></p>

<script>
document.getElementById("example").innerHTML = "Welcome to JavaScript!";
</script>

</body>
</html>
```

Output



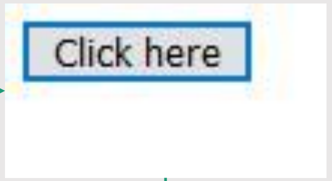
```
Welcome to JavaScript!
```

## INLINE

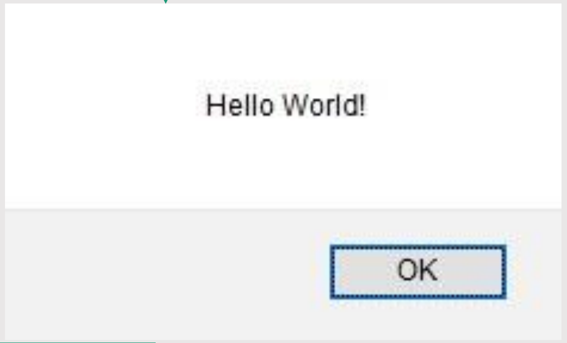


```
<html>
<head>
<script type="text/javascript" src="message.js"></script>
</head>
<body>

<input type="Button" value="Click here" onclick="alert('Hello World!');">

</body>
</html>
```

Output



Output



Hello World!

## The use of three-method

- External: easy to maintain, to analyze and debug more efficiently. external JS files are cached by browsers and can load pages more faster.

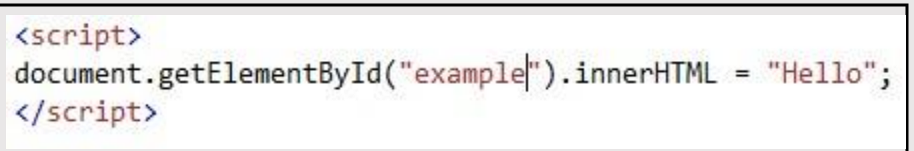
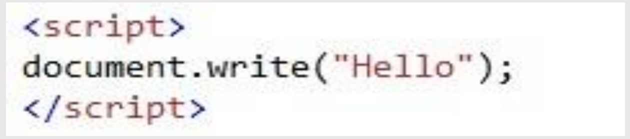
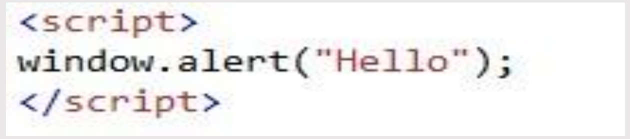
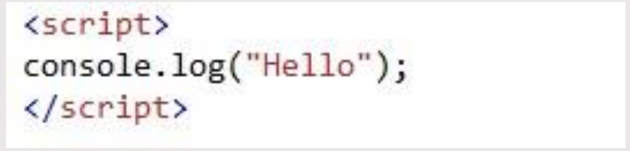
- Internal: when you have few lines of code which is specific to a webpage, it is better to keep the JS code internally within your HTML document.

- Inline: not recommend with a big website because it cannot be minified, cached and hard to debug.

## JavaScript Output()

JS can display the data in following ways:

| JS OUTPUT | FEATURES | EXAMPLES |
|---|---|---|

| | | |
|---|---|---|
| **innerHTML** | Writing into an HTML element |  `<script>`<br>`document.getElementById("example").innerHTML = "Hello";`<br>`</script>` |
| **document.write()** | Writing into the HTML output |  `<script>`<br>`document.write("Hello");`<br>`</script>` |
| **window.alert()** | Writing into an alert box |  `<script>`<br>`window.alert("Hello");`<br>`</script>` |
| **console.log()** | Writing into the browser console (Press F12-Tools Console in your browser for seeing the result(Console tab)- for debug purpose |  `<script>`<br>`console.log("Hello");`<br>`</script>` |

## Comment Syntax

Two different types of JS comment syntax :

Single-line

```
// This is a comment
```

Multi-line

```
/* This is
a comment */
```

Everything between the opening and closing tag in the code block above will be ignored.
Both single-line and multi-line comments are written above or next to the code they are designated to explain.

## Datatype/variables

+ Javascript variable can hold many kind of datatypes: number, string, boolean, object.

```
<script type="text/javascript">

var name = "Ali";
var money;
money = 2000.50;
var open = true;
var x = {firstName:“Ali", lastName:“Bli"};

</script>
```

+ Unlike many other languages, you don't have to tell JavaScript during variable declaration what type of value the variable will hold. The value type of a variable can change during the execution of a program and JavaScript takes care of it automatically.

## Datatype/variables

+ Naming rules:

1) Can’t start with letter “_”, “$” or a number.

2) Can’t use “–” and “.”.

3) No reserved words or JS keywords.

4) All variables are case sensitive so “name” is not equal to “Name”.

5) Use related names.

6) Camel case if more than one word (firstName).

## Datatype/variables

+ Undefined: a variable that has been declared, but no value has been assigned to it yet.

+ Null: a variable with no value - it may have had one at some point, but no longer has a value. Unfortunately, in JavaScript, the data type of null is an object.

+ Difference Between Undefined and Null:

```
typeof undefined          // undefined
typeof null               // object

null === undefined        // false null ==
undefined         // true
```

## JavaScript Function

A function is a subprogram to demonstrate a particular task.

Functions are executed when they are called. It is invoked function. They are always return values; otherwise, it will return undefined.

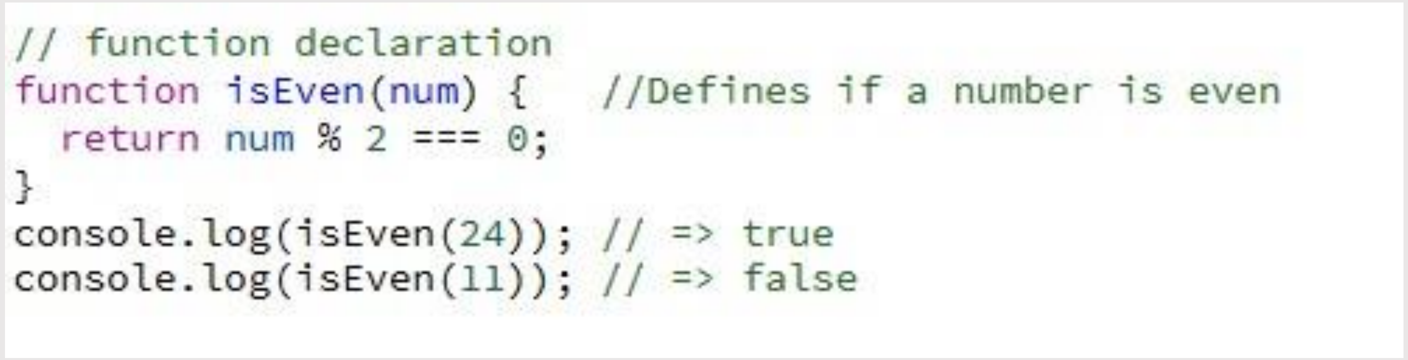
*How to define a function?*

- Function Declaration
- Function Expression
- Many more …

## Function Declaration

- Defines a function name and using function keyword.

- To use the function declaration, the function declaration is hoisted which allows the function to be used before it is defined.



```
// function declaration
function isEven(num) {   //Defines if a number is even
  return num % 2 === 0;
}
console.log(isEven(24)); // => true
console.log(isEven(11)); // => false
```

Explaining

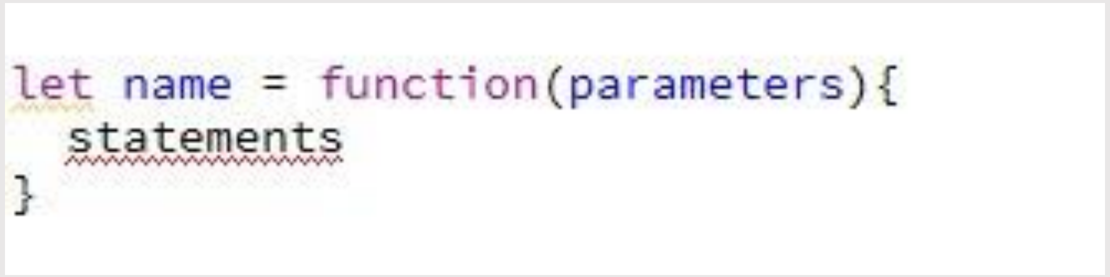
The function declaration creates a variable in the current scope with the identifier equal to function name *( isEven and num).*

The func. variable is hoisted to the top of the scope means the function can be invoked before declaration.

## Function Expression

- Defines a named or anonymous function(no name func.)

- Function expressions are not hoisted which means you can't use function expressions before you define them.



```
let name = function(parameters){
  statements
}
```

In the example above, we are setting the anonymous function object equal to a variable.

## Arrays

+ An array doesn't just store one value, it can stores a list of values.

```
var cities = ['Helsinki', 'Espoo', 'Vantaa’];       // Print out: Helsinki
console.log(cities[0]);
Index 0 >> Helsinki
Index 1 >> Espoo
Index 2 >> Vantaa
```

+

```
var cities = ['Helsinki', 'Espoo', 'Vantaa’];
 cities[0] = 'Tampere’;
 console.log(cities[0]);
                                                     // Print out: Tampere
```

## Array length:

```
var cities = ['Helsinki', 'Espoo', 'Vantaa’];
console.log(cities.length);
                                                     //Print out: 3
```

## Change value:

+

## Global Objects or

**Value properties**

These global properties return a simple value; they have no properties or methods.

- Infinity
- NaN
- undefined
- null literal **Function properties**

**Numbers and dates**

These are the base objects representing numbers, dates, and mathematical calculations.

- Number
- Math
- Date

**Text processing**

These objects represent strings and support

These global functions—functions which are called globally rather than on an object—directly return their results to the caller.

- isNaN() • eval()
- parseInt() ...

manipulating them.

- String
- RegExp

## Standard built-in objects

Read more: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects

# String

11 12 13 14 15 16 17 181920 21 var

str = “Welcome to JavaScript!";

0 1 2 3 4 5 6 7 8 9 10

| Property/Method | Description | Example | Output |
|---|---|---|---|
| **String Properties** | | | |
| length | Returns the length of a string(number of characters) | str.length; | 22 |
| **String Methods** | | | |
| charAt() | returns the character at the specified index | str.charAt(0) | ‘W’ |
| endsWith() | returns true if the string ends with the characters of a specified string, and false if not | str.endsWith(“JavaScript."); | False<br>(because string ends with JavaScript!) |

| | | | |
|---|---|---|---|
| indexOf() | returns the position of the first occurrence of a specified value in a string. | str.indexOf(“to"); | 8 |
| slice() | extracts parts of a string and returns the extracted parts in a new string | str.slice(1, 10); | ‘elcome to’ |
| toUpperCase() | converts a string to uppercase letters. | str.toUpperCase(); | ‘WELCOME TO JAVASCRIPT!’ |
| … | … | … | … |

## Math

| Property/Method | Description | Example | Output |
|---|---|---|---|
| **Math Properties** | | | |
| PI | Returns PI value | Math.PI; | 3.141592653589793 |
| SQRT2 | returns the square root of 2 | Math.SQRT2; | 1.4142135623730951 |
| **Math Methods** | | | |

|  |  |  |  |
|---|---|---|---|
| abs() | returns the absolute value of a number. | Math.abs(-5.50) | 5.50 |
| acos() | returns the arccosine of a number | Math.acos(-1); | 3.141592653589793 |
| asin(x) | returns the arcsine of a number, in radians | Math.asin(1); | 1.5707963267948966 |
| ceil() | rounds a number UPWARDS to the nearest integer, and returns the result. | Math.ceil(1.2); | 2 |
| toUpperCase() | rounds a number DOWNWARDS to the nearest integer, and returns the result. | Math.floor(1.9); | 1 |
| … | … | … | … |

| Method | Description | Example |
|---|---|---|
| getDate() | returns the day of the month (from 1 to 31) for the specified date. | d.getDate(); |
| getDay() | returns the day of the week (from 0 to 6) for the specified date | d.getDay(); |
| getFullYear() | returns the year (four digits for dates between year 1000 and 9999) of the specified date. | d.getFullYear(); |
| getHours() | returns the hour (from 0 to 23) of the specified date and time. | d.getHours(); |
| getMinutes() | returns the minutes (from 0 to 59) of the specified date and time. | d.getMinutes(); |
| getMonth() | returns the month (from 0 to 11) for the specified date, according to local time. | d.getMonth(); |
| getSeconds() | returns the seconds (from 0 to 59) of the specified date and time | d.getSeconds(); |

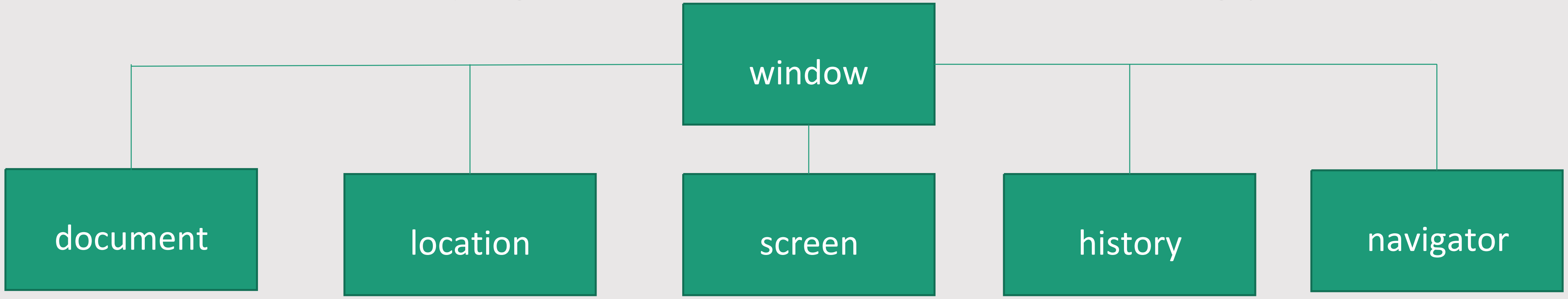
|  |  |  |
|---|---|---|
| d.getTime(); | returns the number of milliseconds between midnight of January 1, 1970 and the specified date. | d.getTime(); |
| … | … | … |

## Date

var d= new Date()

## BOM

- The **BOM** (Browser Object Model) consists of the objects navigator , history , screen , location and document which are children of window . In the document node is the DOM (Document Object Model), the document object model, which represents the contents of the page. You can manipulate it using javascript.



Using Browser Object Model

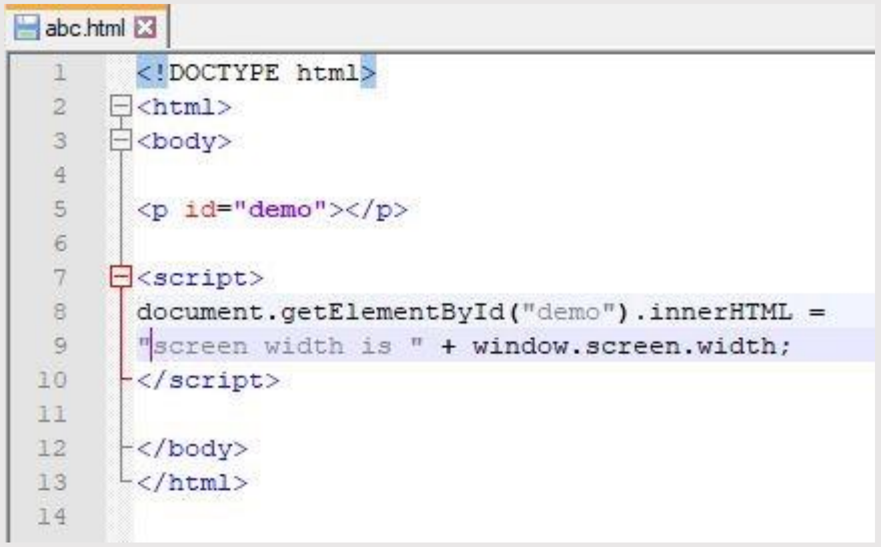
## Properties

- window.innerHeight
- window.innerWidth
- window.pageXOffset
- window.pageYOfset
- window.screenX
- window.screenY
- window.location
- window.document
- window.history
- window.history.length
- window.screen
- window.screen.width
- window.screen.height

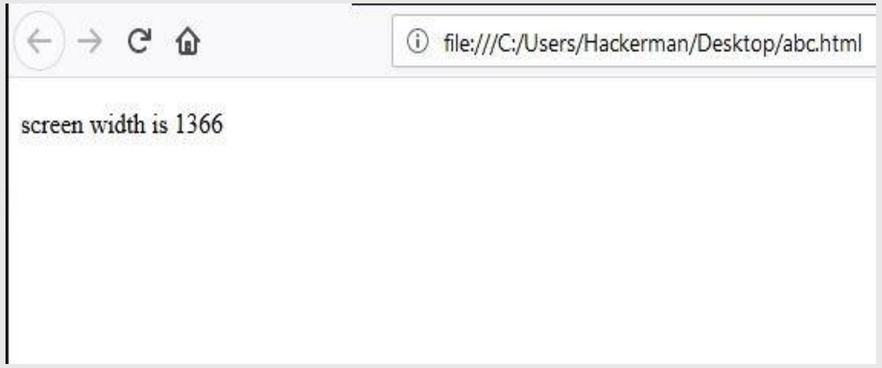
## Methods

- window.alert()
- window.open()
- window.print()

## Example



```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML =
"screen width is " + window.screen.width;
</script>

</body>
</html>
```



```
screen width is 1366
```

# Operators

## Assignment Operators

- city = ‘Kokkola’;

## Arithmetic Operators

- area = 5 * 6;

## String Operators

- message = ‘Welcome ’ + name;

Comparison Operators

- open = 4 > 5;

Logical Operators

- open = (3 < 5) && (14 < 16);

## Operators

| Name | Operator | Example | Result |
|---|---|---|---|
| Addition | + | 7+2 | 9 |
| Subtraction | - | 14-6 | 8 |
| Division | / | 6/2 | 3 |
| Multiplication | * | 2*3 | 6 |

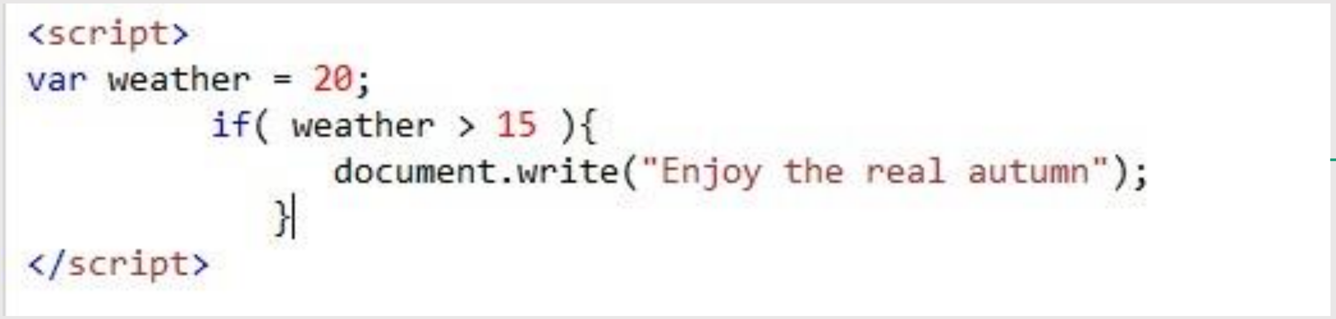
| | | | |
|---|---|---|---|
| Increment | ++ | i=5;<br>i++; (postfix-increment)<br>++i; (prefix-increment) | 6 |
| Decrement | -- | i=5;<br>i--; (postfix-decrement)<br>--i; (prefix-decrement) | 4 |
| Modulus | % | 5%3 | 2 |

## if statement

if statement gets a condition, evaluates it and, if the result is true, executes the code.

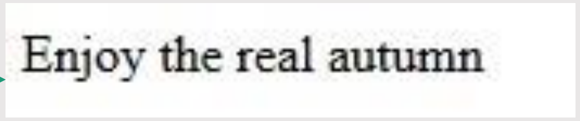
**Syntax**

```
if (expression){
        Statement(s) to be executed if expression is true }
```



```
<script>
var weather = 20;
         if( weather > 15 ){
               document.write("Enjoy the real autumn");
            }
</script>
```

Output



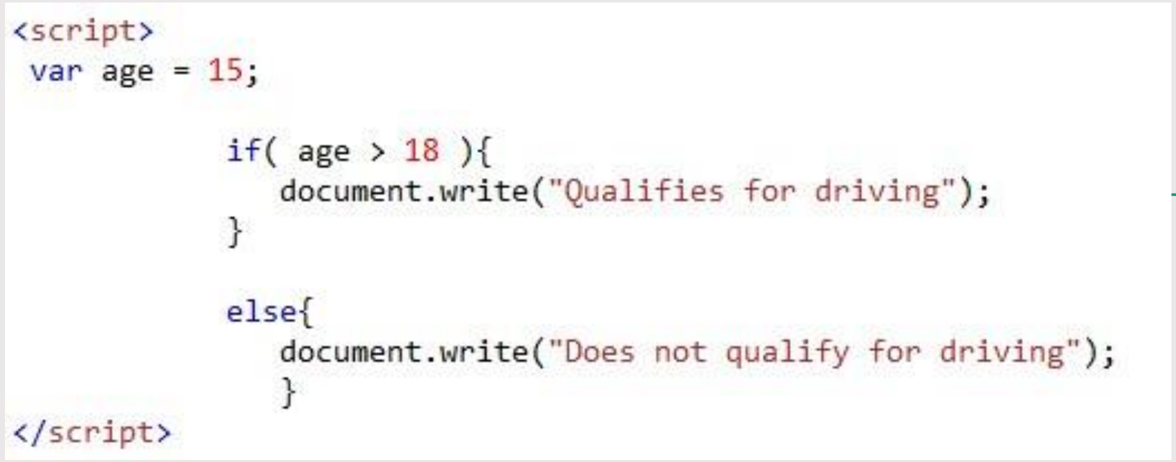
```
Enjoy the real autumn
```

## if...else statement

**Syntax**

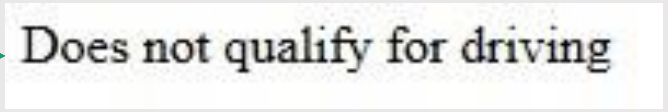
```
if (expression){
Statement(s) to be executed if expression is true

} else{
Statement(s) to be executed if expression is false
}
```

# if...else if... statement



```
<script>
 var age = 15;

            if( age > 18 ){
               document.write("Qualifies for driving");
            }

            else{
               document.write("Does not qualify for driving");
               }
</script>
```

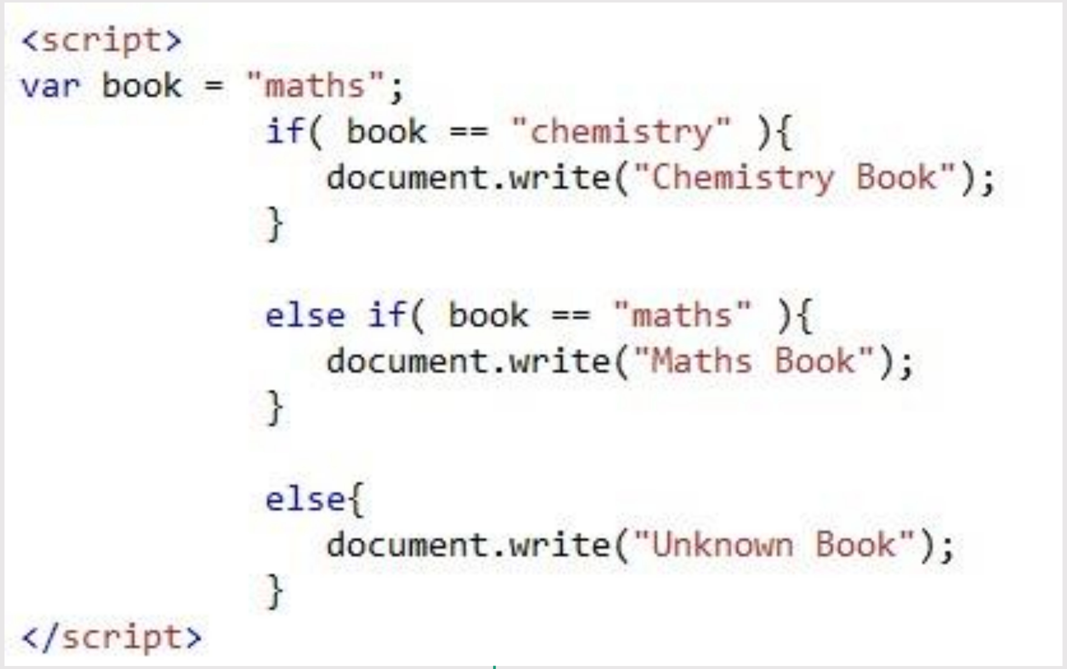
Output



```
Does not qualify for driving
```

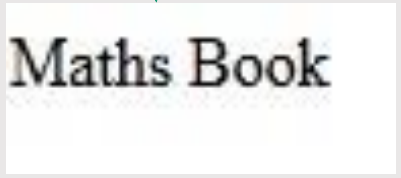
**Syntax**

```
if (expression 1){
Statement(s) to be executed if expression 1 is true
} else if (expression 2){
Statement(s) to be executed if expression 2 is true
} else{
Statement(s) to be executed if no expression is true
}
```



```
<script>
var book = "maths";
            if( book == "chemistry" ){
               document.write("Chemistry Book");
            }

            else if( book == "maths" ){
               document.write("Maths Book");
            }

            else{
               document.write("Unknown Book");
            }
</script>
```

Output



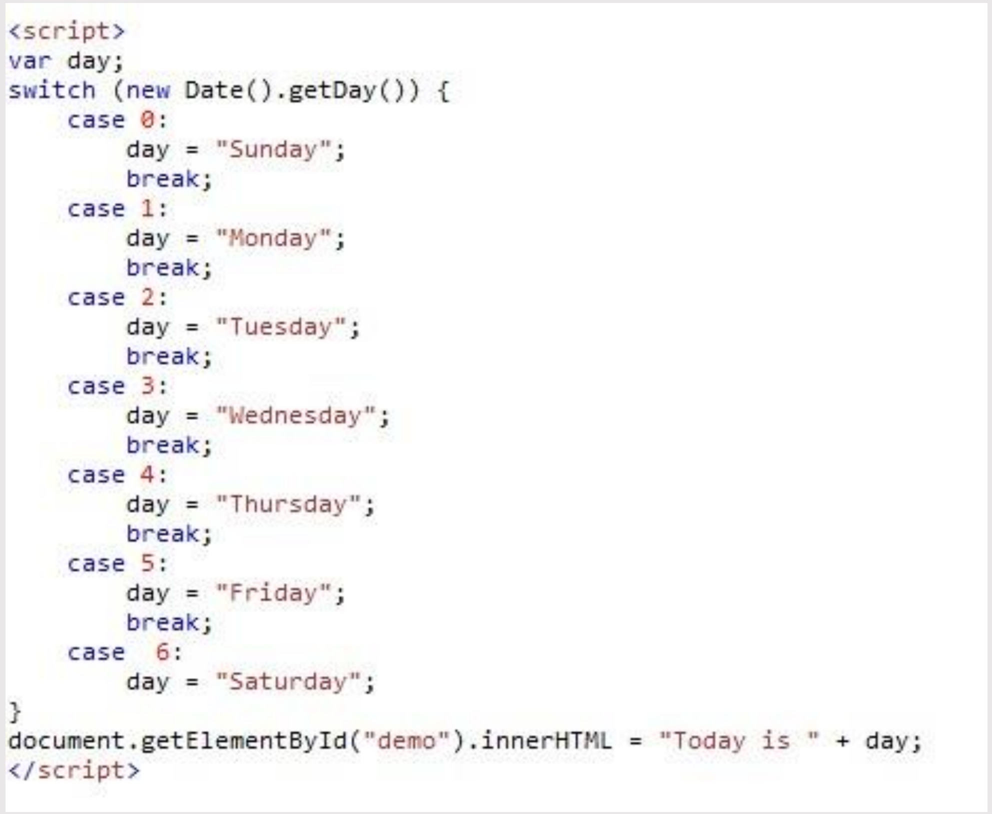
```
Maths Book
```

## switch statements

**Syntax**

- The expression is evaluated.
- The 1st case is checked if it matches the expression value, the code executes.
- the break keyword ends the switch block.

- If the case does not match the next case will be checked.
- If the next one matches , the code executes and exit.
- If no cases match, the default code block is executed.



```
<script>
var day;
switch (new Date().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;
</script>
```

switch (expression) { case condition 1: statement(s) break;
case condition 2: statement(s) break;

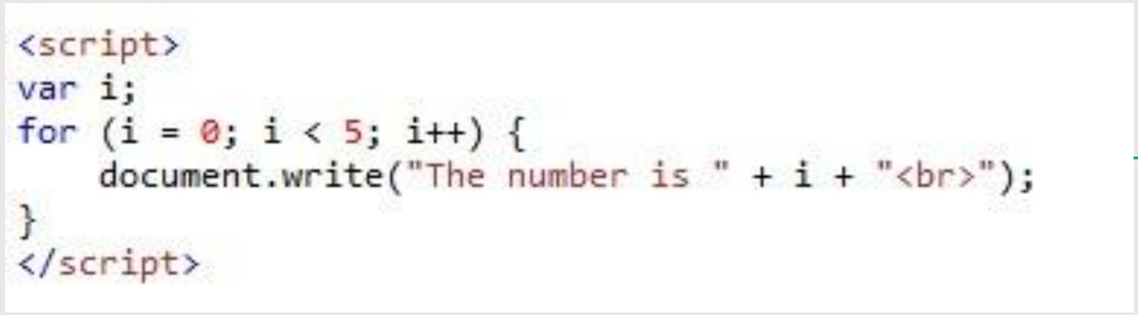
...
case condition n: statement(s) break;
default: statement(s)

```
}
```

Run your code to see the result. It can be different based on the default time zone.

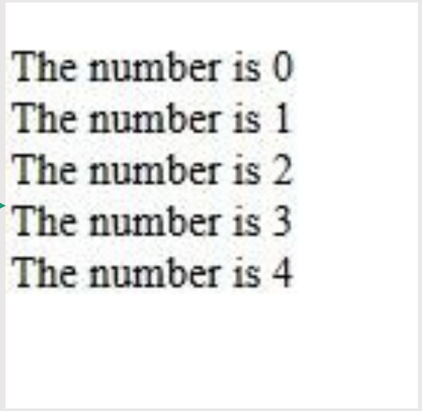
## for loops

A for loop is useful if you want to run same code over and over again.



```
<script>
var i;
for (i = 0; i < 5; i++) {
    document.write("The number is " + i + "<br>");
}
</script>
```

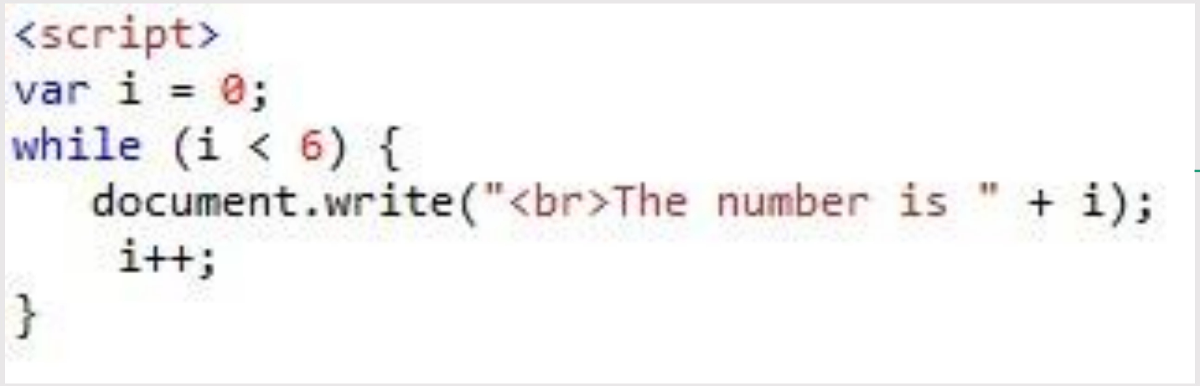
Output



```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
```

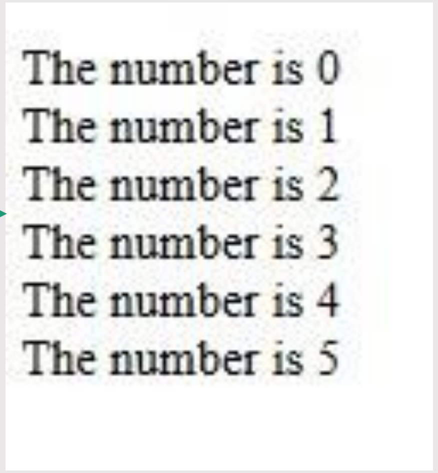
## while loops

A while loop is to execute a statement repeatedly as long as an expression is true. Once the expression becomes false, the loop terminates.



```
<script>
var i = 0;
while (i < 6) {
   document.write("<br>The number is " + i);
    i++;
}
```

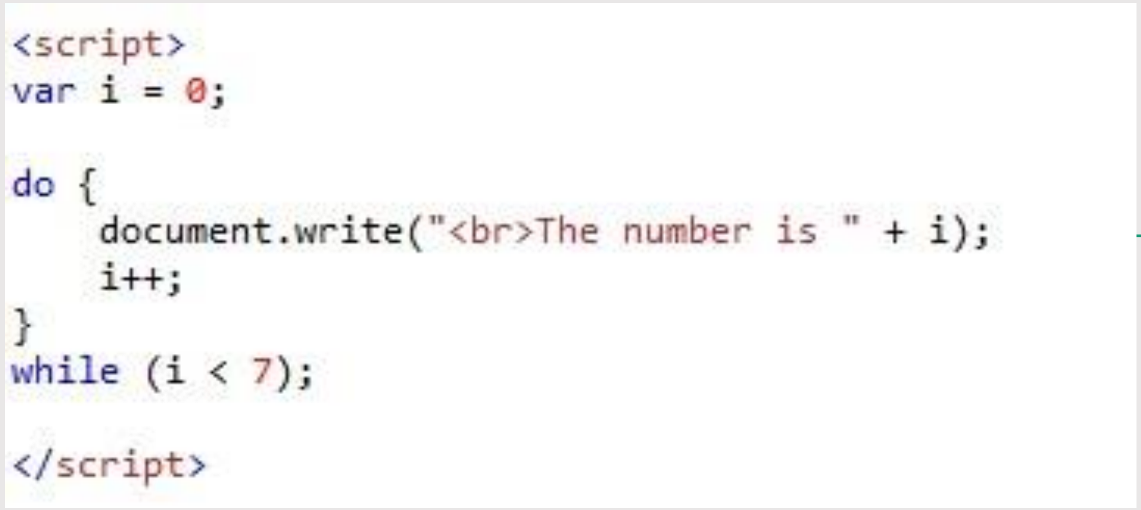
Output



```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

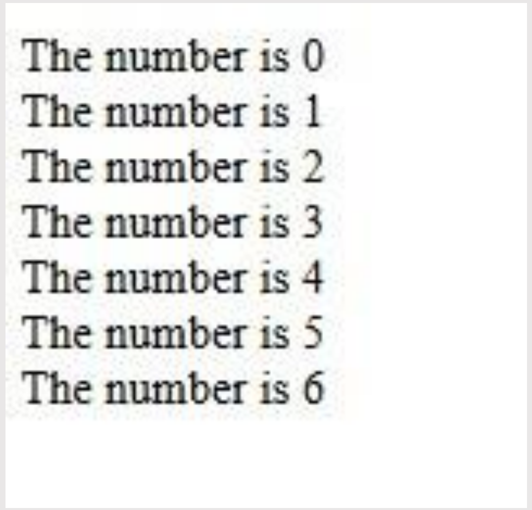
## Do/While Loop

The do/while loop is a variant of the while loop. This loop will execute before checking if the condition is true, then it will repeat the loop as long as the condition is true.



```
<script>
var i = 0;

do {
    document.write("<br>The number is " + i);
    i++;
}
while (i < 7);

</script>
```

Output

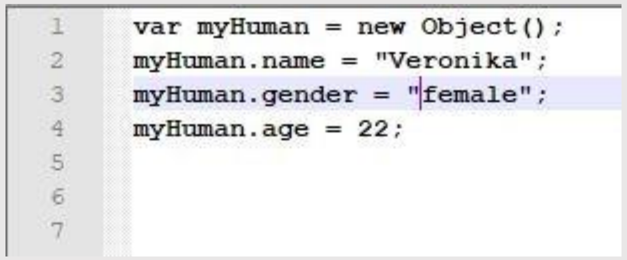


```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
```

## JavaScript classes/objects

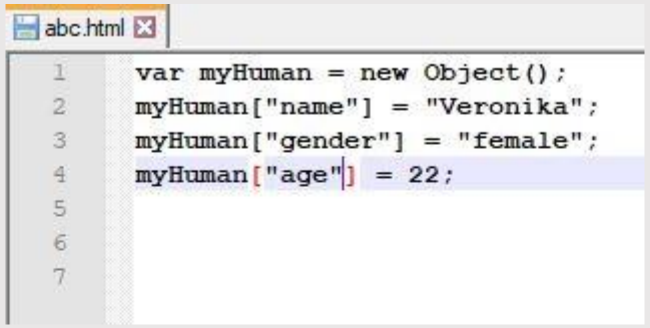
+ An object is a collection of properties, and a property is an association between a name and a value.

Method 1



```
var myHuman = new Object();
myHuman.name = "Veronika";
myHuman.gender = "female";
myHuman.age = 22;
```

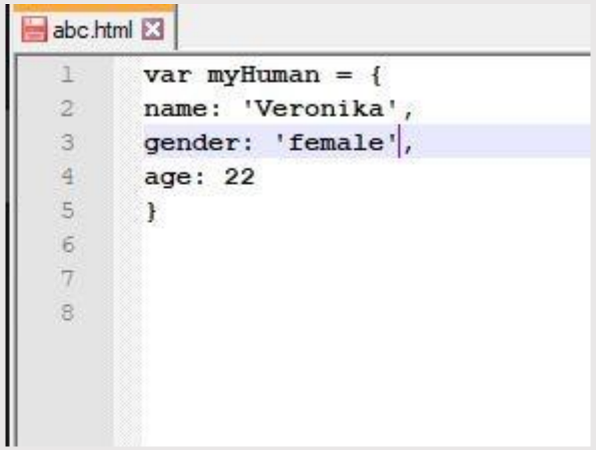
Method 2



abc.html

```
var myHuman = new Object();
myHuman["name"] = "Veronika";
myHuman["gender"] = "female";
myHuman["age"] = 22;
```

Method 3



abc.html

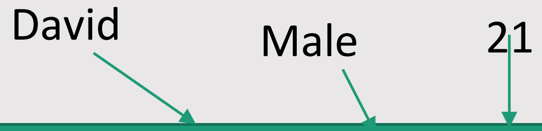
```
var myHuman = {
name: 'Veronika',
gender: 'female',
age: 22
}
```

# Objects – Constructor Notation

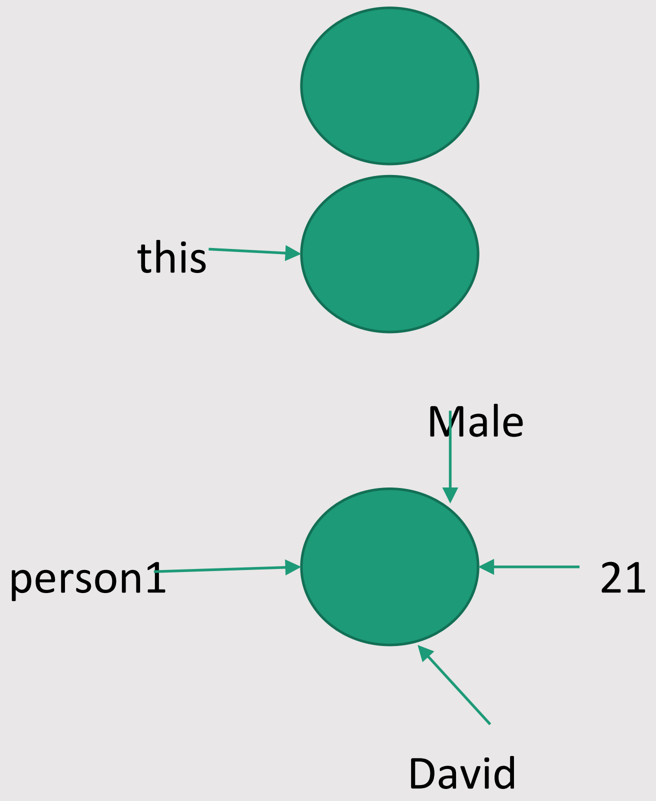
```
var person1 = new Human("David" , "Male" , 21);
```

The first thing new does is create a new, empty object

Next, new sets **this** to point to the new object



```
function Human(name, gender, age) {
this.name = name;
this.gender = gender;
this.age = age;
}
```



## Events

+ The Event interface represents any event of the DOM. It contains common properties and methods to any event. Events trigger a function or script.

UI Events

Keyboard Events

Mouse Events

Form Events

Mutation Events

## Events – UI Events

| Event | Description |
|---|---|
| load | Web page has finished loading |
| unload | Web page is unloading |
| error | Browser confront a JavaScript error, or a file is missing |
| resize | Browser window has been resized |
| scroll | User has scrolled up or down the page |

## Events – Keyboard Events

| Event | Description |
|---|---|
| keydown | User presses a key |
| keyup | User releases a key |
| keypress | Character is being inserted |

## Events – Mouse Events

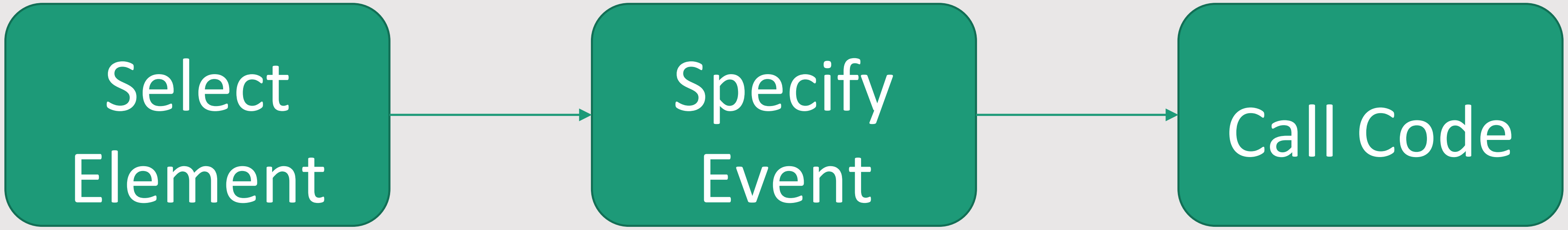
| Event | Description |
|---|---|
| click | User presses and releases a button over an element |
| dblclick | User presses and releases a button twice over an element |
| mousedown | User presses a mouse button over an element |
| mouseup | User releases a mouse button over an element |
| mousemove | User moves the mouse |
| mouseover | User moves the mouse over an element |
| mouseout | User moves the mouse off an element |

## Events – Mutation Events

| Event | Description |
|---|---|

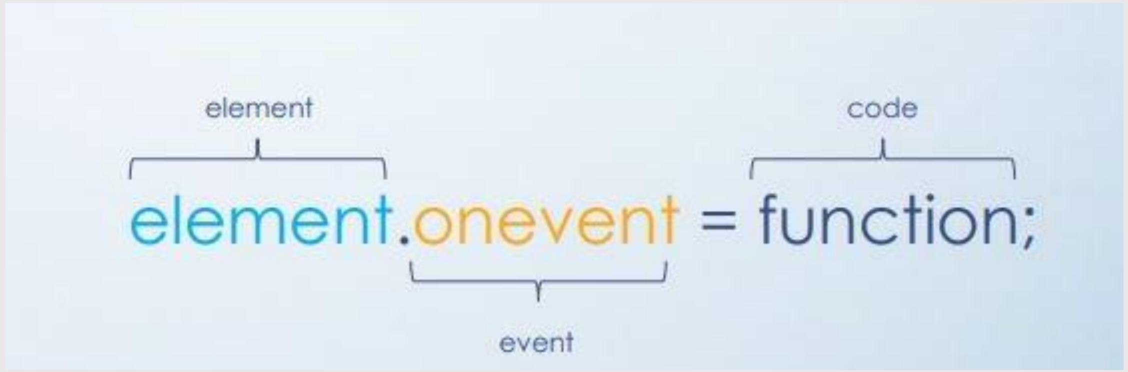
| | |
|---|---|
| DOMSubtreeModified | Change has been made to document |
| DOMNodeInserted | Node has been inserted as a direct child of another node |
| DOMNodeRemoved | Node has been removed from another node |
| OOMNodeInsertedIntoDocument | Node has been inserted as a descendant of another node |
| DOMNodeRemovedFromOocument | Node has been removed as a descendant of another node |

## How to use Event Handler



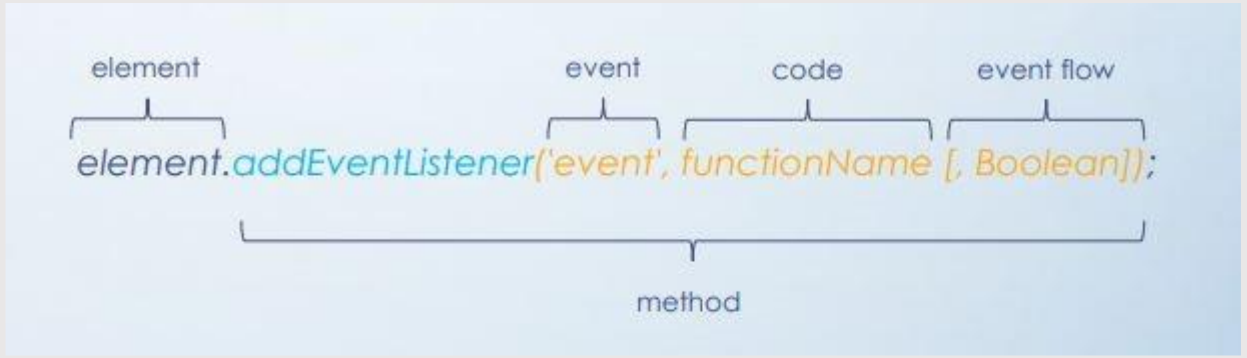
```
<script>
document.getElementById("demo").onclick = showDate;
</script>;
```

## DOM Event Handlers



```
function showData()
{ // code to show Data }
var el = document.getElementById('myButton');
el.onclick = showData;
```

## Event Listeners


```
element.addEventListener('event', functionName [, Boolean]);
```

```
function showData() { // code to show Data }
var el =
document.getElementById('myButton');
el.addEventListener('click', showData, false);
```

## Events – Form Events

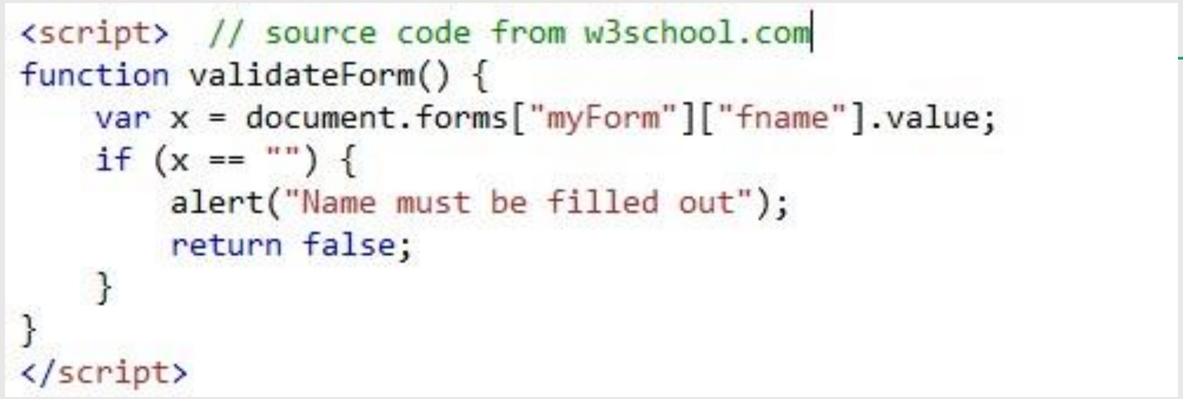
| Event | Description |
|---|---|
| input | Value has been changed in any or |
| change | Value in select box, radio button or check box changes |
| submit | User submits a form |
| reset | User clicks on a reset button inside a form |
| cut | User cuts content from a form field |
| copy | User copies content from a form field |

|  |  |
|---|---|
| paste | User pastes content into a form field |
| select | User selects content of a form element |

## JS Form Validation

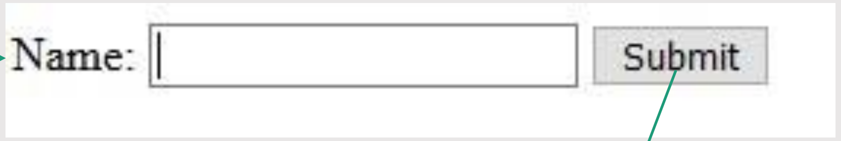
We have found out about form in HTML Part and it also can be worked out by JS.

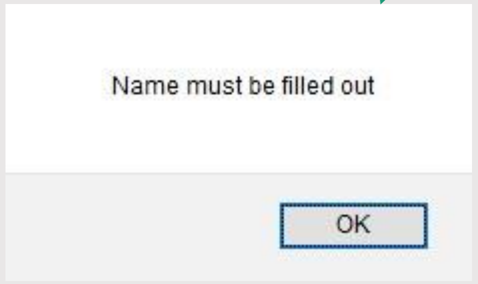
It is used to validate a form field, numeric output,…



```
<script>  // source code from w3school.com
function validateForm() {
    var x = document.forms["myForm"]["fname"].value;
    if (x == "") {
        alert("Name must be filled out");
        return false;
    }
}
</script>
```

Output





Name must be filled out

For example: we can easily create a form for submitting a name. With JS Validation, it helps to regcognize the user's act when they leave the form empty and click submit, thus it shows an alert box with a reminding message

## Error Handing

try

- Lets you test a block of code for errors.

catch

- Lets you handle the error.

finally

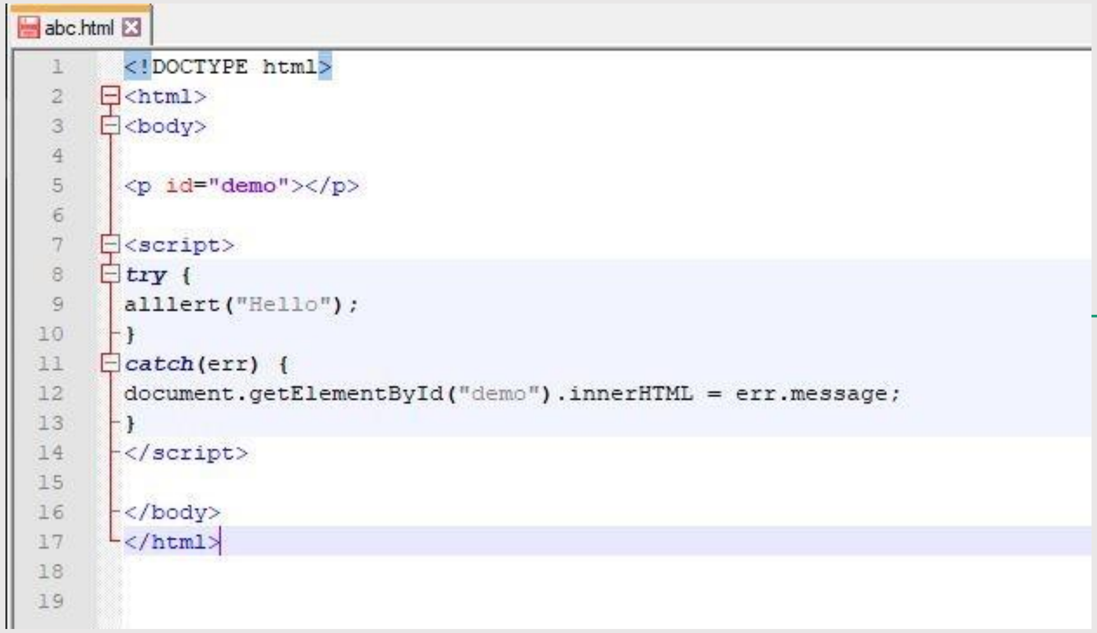
- Lets you create custom errors.

throw

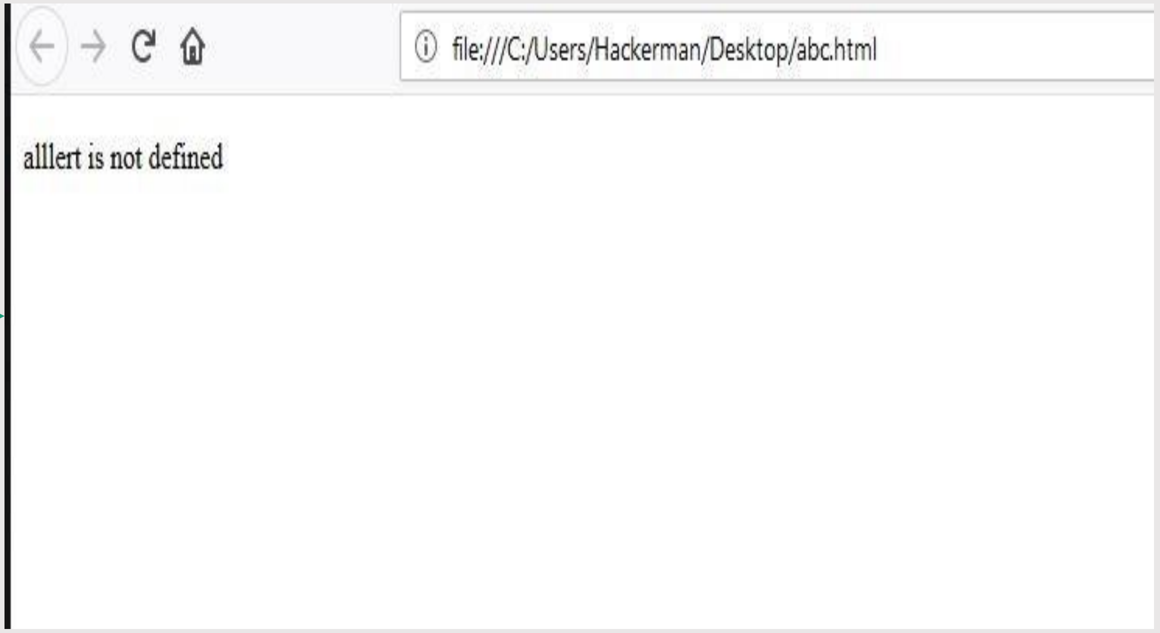
- Lets you execute code, after try and catch, regardless of the result.

# Error Handing

Example



```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>
try {
allllert("Hello");
}
catch(err) {
document.getElementById("demo").innerHTML = err.message;
}
</script>

</body>
</html>
```



```
alllert is not defined
```

## A glance of jQuery

jQuery is a cross-platform JavaScript library designed to simplify the clientside scripting of HTML. It is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers.

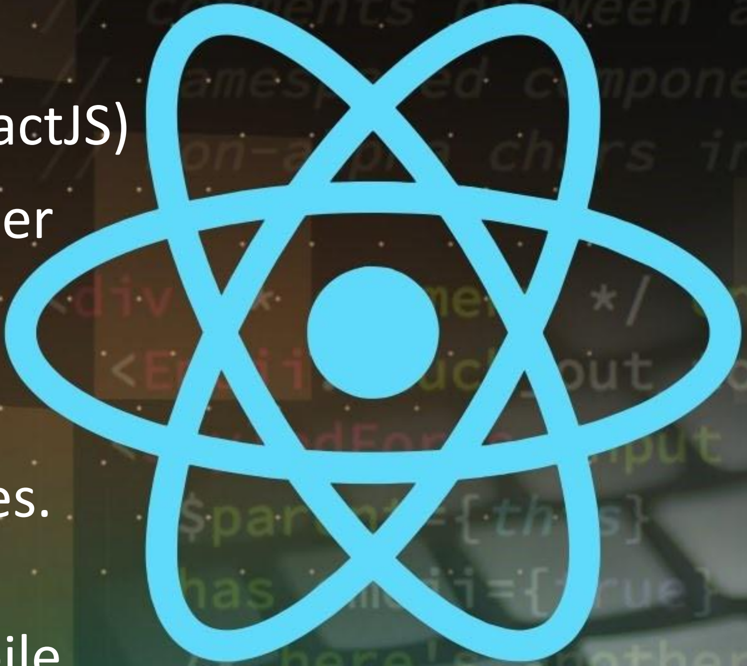
jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications. jQuery also provides capabilities for developers to create plug-ins on top of the JavaScript library. The modular approach to the jQuery library allows the creation of powerful dynamic web pages and Web

application. *Sourse: [https://en.wikipedia.org/wiki/JQuery](https://en.wikipedia.org/wiki/JQuery)*

Find out more about jQuery: https://jquery.com/

# A glance of React JS

React (also known as React.js or ReactJS) is a JavaScript library for building user interfaces.It is maintained by Facebook and a community of individual developers and companies. Reactcan be used as a base in the development of single-page or mobile applications.



React can be used as a base in the development of single-page or mobile applications. Complex React applications usually require the use of additional libraries for state management, routing, and interaction with an API.

*Source: https://en.wikipedia.org/wiki/React_(JavaScript_library)*

## A glance of Bootstrap

Bootstrap includes HTML and CSS based design templates for common user interface components like Typography, Forms, Buttons, Tables, Navigation, Dropdowns, Alerts, Modals, Tabs, Accordion, Carousel and many other as well as optional JavaScript extensions.

It helps building responsive websites-where devices come in all shapes and sizes. It is said to be the most popular HTML, CSS and JavaScript framework for developing a responsive and mobile friendly website. Bootstrap gives you the ability to produce design with less efforts.

*Source: https://medium.com/@samdias9792/introduction-to-bootstrap-4-57616767a81c*

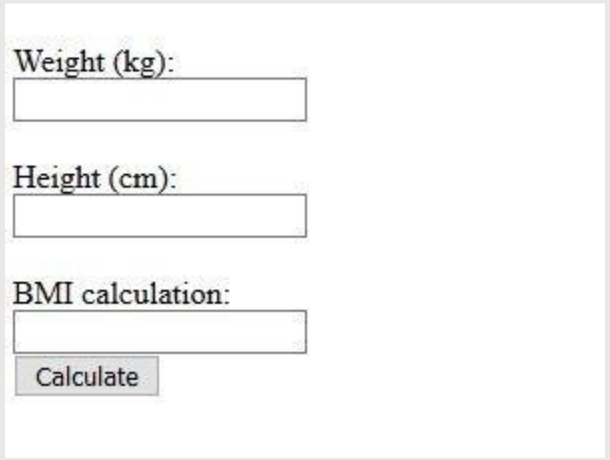
## Example

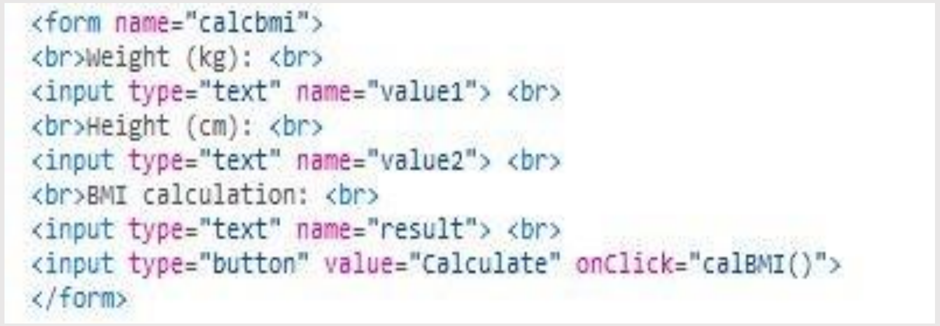
Let’s think about how can we combine all of things in a real website.

We often use html form to insert values, and how can we use those values in Javacript code?

User Interface

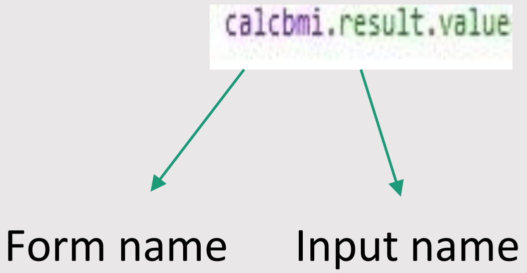
Code





```
<form name="calcbmi">
<br>Weight (kg): <br>
<input type="text" name="value1"> <br>
<br>Height (cm): <br>
<input type="text" name="value2"> <br>
<br>BMI calculation: <br>
<input type="text" name="result"> <br>
<input type="button" value="Calculate" onClick="calBMI()">
</form>
```

# Try to take values to Javascript

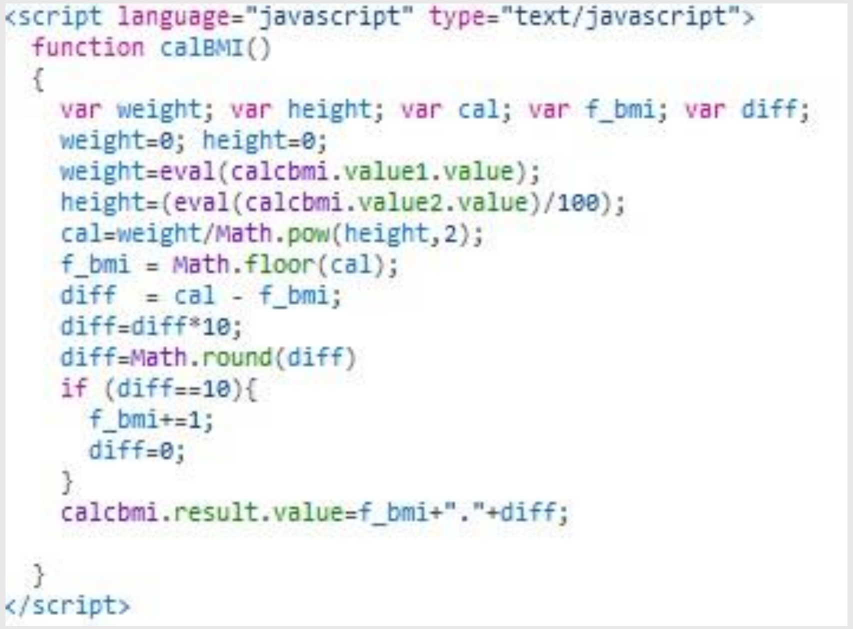


```
calcbmi.result.value
```

Code to call the value from the form html

# Javacript code

- Let’s add js code to your source:

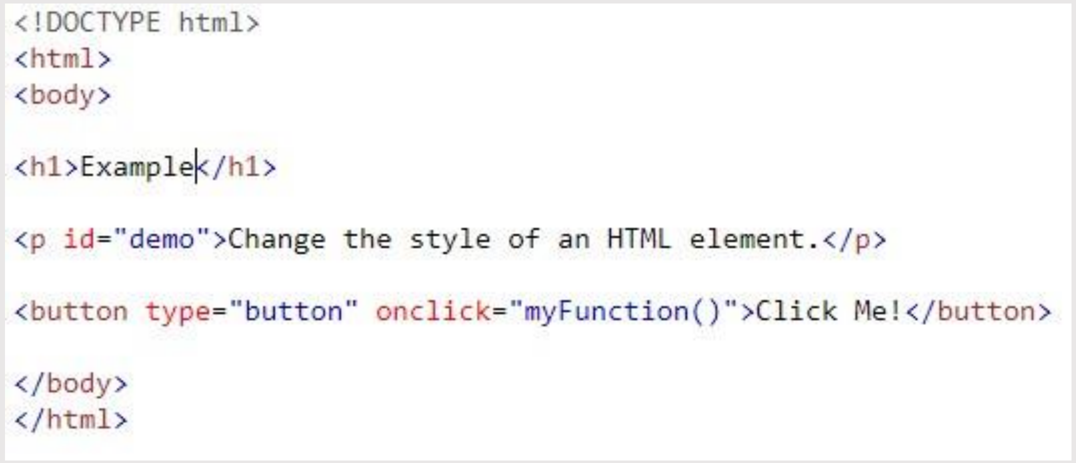


```
<script language="javascript" type="text/javascript">
 function calBMI()
 {
   var weight; var height; var cal; var f_bmi; var diff;
   weight=0; height=0;
   weight=eval(calcbmi.value1.value);
   height=(eval(calcbmi.value2.value)/100);
   cal=weight/Math.pow(height,2);
   f_bmi = Math.floor(cal);
   diff  = cal - f_bmi;
   diff=diff*10;
   diff=Math.round(diff)
   if (diff==10){
     f_bmi+=1;
     diff=0;
   }
   calcbmi.result.value=f_bmi+"."+diff;

 }
</script>
```

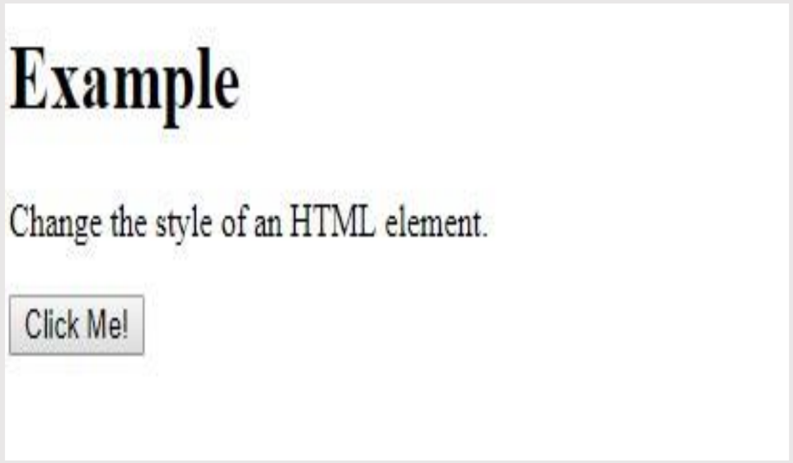
- This code to calculate BMI.

## More Example

- Now, we will try to use javacript to change the CSS style.
- Code    User Interface



```
<!DOCTYPE html>
<html>
<body>

<h1>Example</h1>

<p id="demo">Change the style of an HTML element.</p>

<button type="button" onclick="myFunction()">Click Me!</button>

</body>
</html>
```

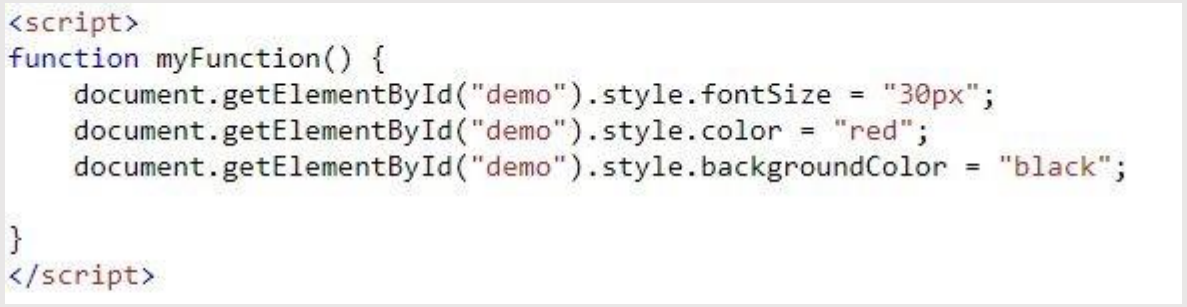


Example

Change the style of an HTML element.

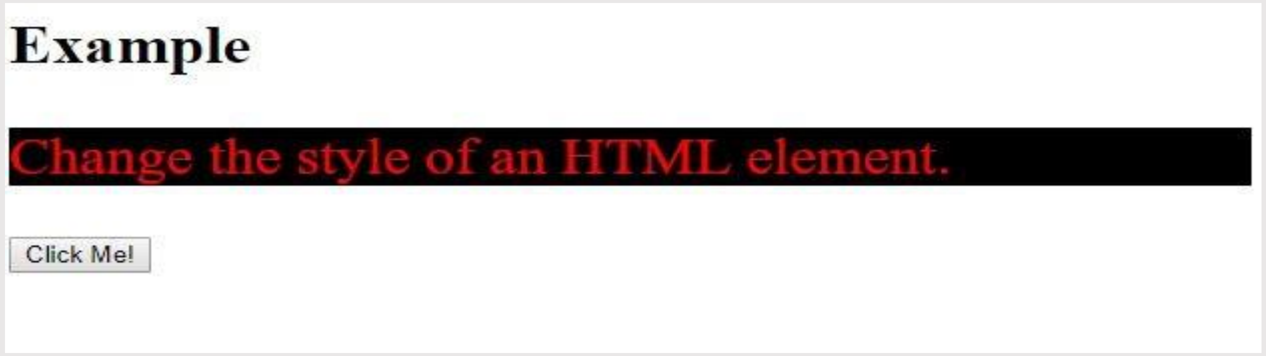
# More Example

- Now add the js code below to your source:



```
<script>
function myFunction() {
    document.getElementById("demo").style.fontSize = "30px";
    document.getElementById("demo").style.color = "red";
    document.getElementById("demo").style.backgroundColor = "black";

}
</script>
```

- The



Example

Change the style of an HTML element.

# Giving feedback and contribution

IF THERE ARE SOME MISTAKES, YOU CAN FREELY REPORT IT.

GIVING FEEDBACK.

ANY SUGGESTION ?

THANK YOU FOR YOUR TIME!