JavaScript

by
Conor Twomey

History

- Written in 1995 by Brendan Eich
- Took only 2 weeks to write!
- Originally called LiveScript
- Renamed to JavaScript to take advantage of the popular release of Java
- JavaScript is now standardised as ECMAScript

History

JavaScript is NOT Java

JavaScript

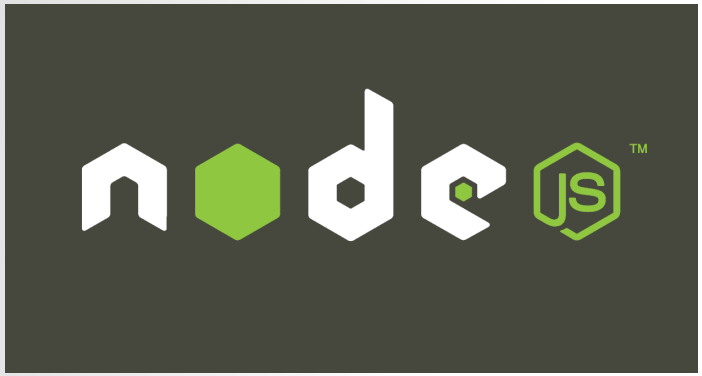
- Scripting Language
- Scripts in Browsers
- Weakly Typed
- Prototype

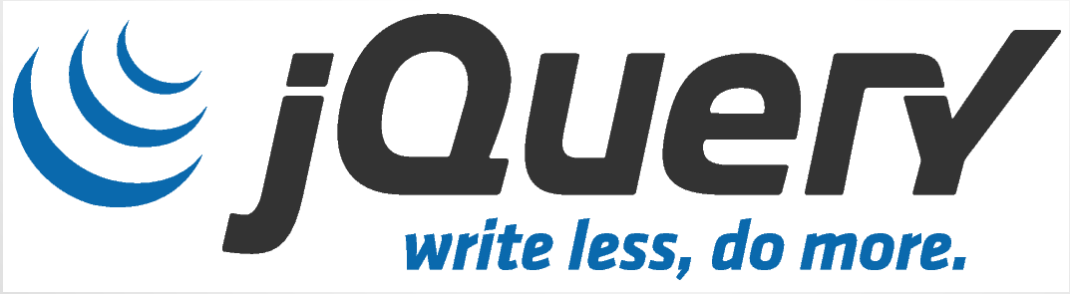
Java

- Compiled Language
- Applets in Browsers
- Strongly Typed

Do NOT Confuse them!

Current / Future





Let's Code!

HTML Template

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
</head>
<body>
</body>
</html>
```

Hello World!

```
<!DOCTYPE html>
<html>
<head>
    <title>Hello World</title>
</head>
<body>
<script type="text/javascript">
  function helloWorld() {
    var message = "Hello World!";
    alert(message);
  }
  helloWorld();
</script>
</body>
</html>
```

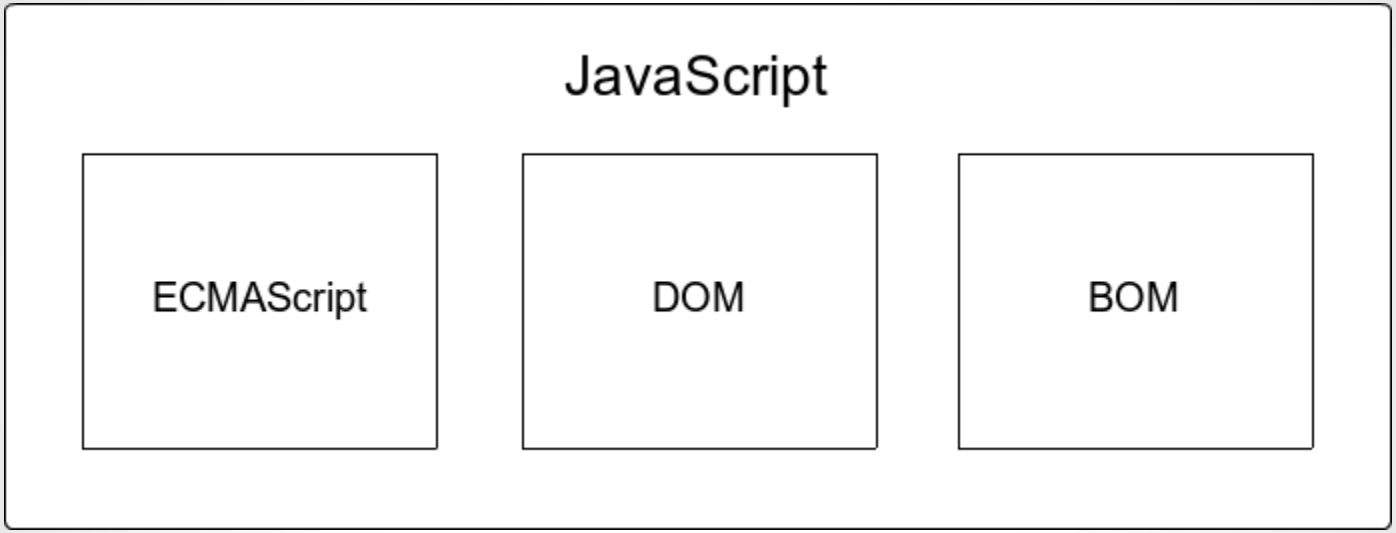
<script> Tag

- The **<script>** and **</script>** Tags are used to mark where the JavaScript starts and ends
- JavaScript can be local or in an external file

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

```
<script type="text/javascript" src="helloWorld.js">
</script>
```

Browsers, DOM, BOM

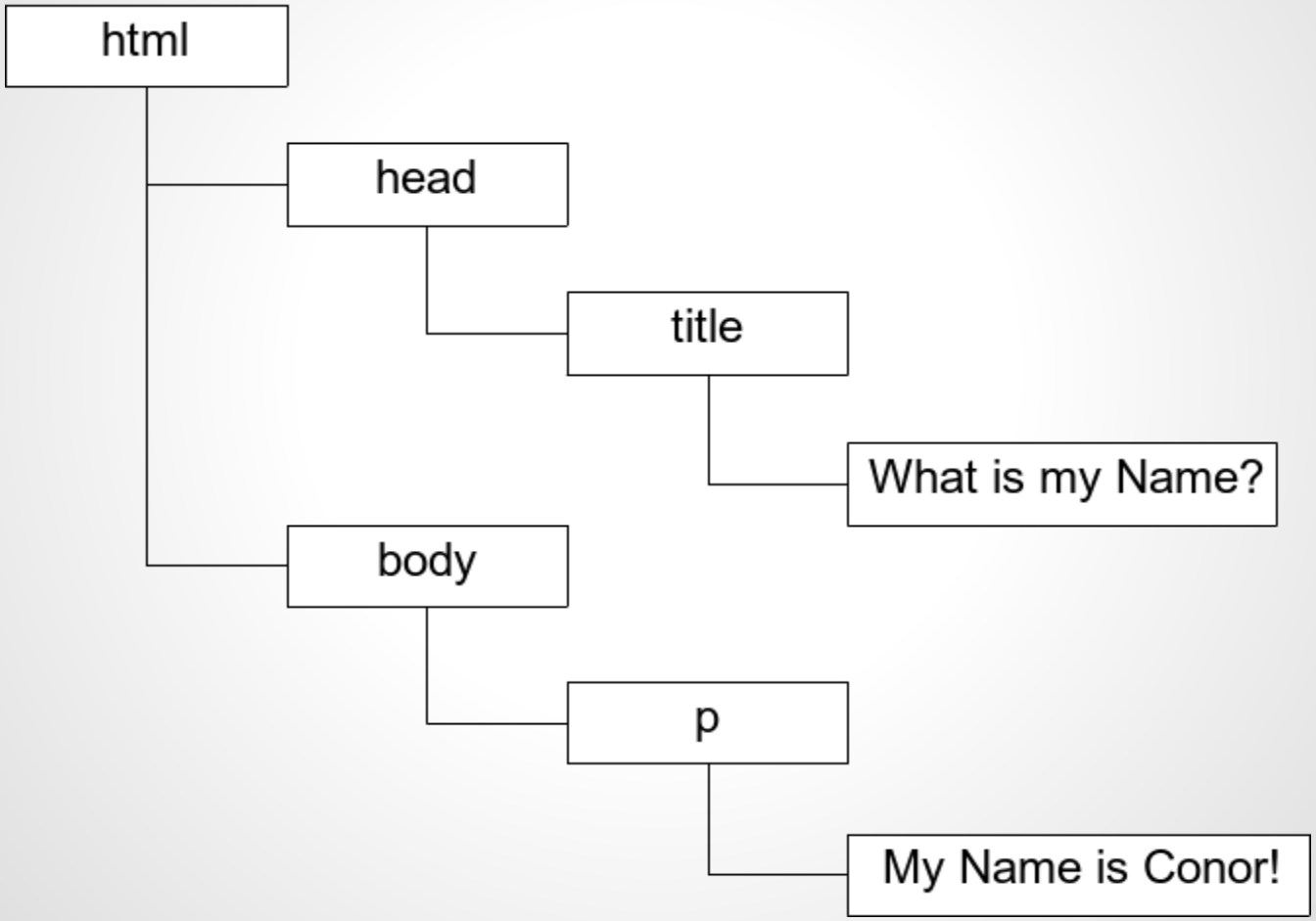


- Document Object Model (DOM)
- Browser Object Model (BOM)

What is my Name?

```
<!DOCTYPE html>
<html>
<head>
  <title>What is my Name?</title>
</head>
<body>
  <p id="name">My Name is Conor!</p>
</body>
</html>
```

Document Object Model (DOM)



What is my Name?

```
<!DOCTYPE html>
<html>
<head>
  <title>What is my Name?</title>
</head>
<body>
  <p id="name">My Name is Conor!</p>
  <button type="button" onclick="changeName()">Change Name</button>
<script type="text/javascript">
  function changeName() {
    document.getElementById("name").innerHTML="My Name is Darragh!";
  }
</script>
</body>
</html>
```

Variables

- Loosely Typed
- Use **`var`** for local variables

```
function test() {
  var message1 = "hi";    // local variable
  message2 = "bye";       // global variable
  message3;               // Undefined
}
test();
alert(message1);          // error
alert(message2);          // "bye"
```

Data Types

- 5 x Primative Data Types - **`Undefined, Null, Boolean, Number`** and **`String`**
- 1 x Complex Data Type - **`Object`**

```
var myString = "hi";   // String
var myNumber = 123;    // Number
var finished = true;   // Boolean
var dontKnow;          // Undefined
var empty = null;      // Null
```

Data Types

- Use the **`typeof`** Operator to identify which one a variable is. Returns (**`"undefined"`**, **`"boolean"`**, **`"string"`**, **`"number"`**, **`"object"`**, **`"null"`**, **`"function"`**)

```
var message = "My Name";
var dontKnow;
alert(typeof message);     // "string"
alert(typeof 2013);        // "number"
alert(typeof dontKnow);    // "undefined"
```

Boolean Operators

NOT	!
AND	&&
OR	||

```
var found = true;
var a = false;
var b = true;
alert(!found);              // false
alert(a || b);              // true
alert(a && b);              // false
```

Multiplicative Operators

MULTIPLY	*
DIVIDE	/
MODULUS	%

```
var result1 = 10 * 10;
alert(result1);                 // 100
var result2 = 25 / 5;
alert(result2);                 // 5
var result3 = 26 % 5;
alert(result3);                 // 1
```

Additive Operators

ADD / CONCAT	+
SUBTRACT	-

```
var result1 = 5 + 6;
alert(result1);                    // 11
var result2 = 12 - 2;
alert(result2);                    // 10

var x = 5, y = 2;
alert("x and y = " + x + y); // ???
```

Relational Operators

LESS THAN	<
GREATER THAN	>
LESS THAN OR EQUAL	<=
GREATER THAN OR EQUAL	>=

```
var result1 = 5 > 3;          // true
var result2 = 5 < 3;          // false
var result3 = "a" <= "b";     // true
var result4 = 3 <= 3;         // true
var result5 = "23" < "3";     // ???
```

Equality Operators

EQUAL	==
NOT EQUAL	!=
IDENTICALLY EQUAL	===
IDENTICALLY NOT EQUAL	!==

```
var result1 = ("55" == 55);
alert(result1);                    // true

var result2 = ("55" === 55);
alert(result2);                    // false
```

Statements

- Statements control flow in JavaScript
- Common in most Languages
- Statements cover

Condition	**`if`** Statement
Post-Test Loop	**`do-while`** Statement
Pre-Test Loop	**`while`** Statement
Iterative	**`for`** Statement
Break and Continue	**`break, continue`**
Switch	**`switch / case`**

HTML Template for Statements

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>
<form name="myForm" onSubmit="return test()">
  <input name="inputText" type="text"/>
  <input name="Test" type="submit"/>
</form>
<script type="text/javascript">
</script>
</body>
</html>
```

`if` Statement (Condition)

```
if (condition) {
   statement1
}
else if (condition) {
   statement2
}
else {
   statement3
}
```

`if` Statement

```
function test () {
  var i = document.myForm.inputText.value;

  if (i > 25) {
    alert("Greater than 25.");
  }
  else {
    alert("Less than or equal to 25.");
  }
}
```

`do-while` Statement

```
do {
   statement
}
while (expression)
```

- Statement is run at least once

`do-while` Statement

```
function test () {
  var i = document.myForm.inputText.value;

  do {
    alert("Number of Alerts Left: " + i);
    i = i - 1;
  }
  while (i > 0)
}
```

`while` Statement

```
while (expression) {
   statement
}
```

- Expression is tested before statement is run

`while` Statement

```
function test () {
  var i = document.myForm.inputText.value;

  while (i > 0) {
    alert("Number of Alerts Left: " + i);
    i = i - 1;
  }
}
```

`for` Statement

```
for (initialisation;
      expression;
      post-loop expression) {
   statement
}
```

- Variable initialisation
- Pre-Loop expression
- Post-Loop code execution

`for` Statement

```
function test () {
  var i = document.myForm.inputText.value;

  for (var x = 0; x < i; x++) {
    alert("Alert Number: " + x);
  }
}
```

`switch` Statement

```
switch (expression) {
   case value: statement;
     break;
   case value: statement;
     break;
   default: statement;
}
```

- Cleaner than lots of `if` / `else if` statements

`switch` Statement

```
function test () {
  var i = document.myForm.inputText.value;
  var number = parseInt(i);
  switch (number) {
    case 25: alert("25");
      break;
    case 30: alert("30");
      break;
    default: alert("Other number");
  }
}
```

HTML Template for Functions

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>
<script type="text/javascript">
</script>
</body>
</html>
```

Functions

- Core of any Language
- Allow relevant statements to be grouped together

```
function functionName(arg0, arg1, argN) {
  statements
}
```

- **`arg0...argN`** are called Arguments
- Function Arguments are optional

Functions

```
function sayHi(name, message) {
  alert("Hello " + name + ", " + message);
}

sayHi("Conor", "How are you?");
sayHi("Darragh", "How are you?");
```

- Reuseable / Easier to test
- Well written Functions can be shared between projects

Functions

- Functions can return values

```
function sum(x, y) {
   return x + y;
}

var xy = sum(5, 10);
alert("Answer: " + xy);
```

Functions

- Doesn't care how many Arguments passed
- Can use **`arguments`** Object

```
function sum() {
  if (arguments.length == 2) {
    return arguments[0] + arguments[1];
  }
  return "Incorrect number of Arguments";
}
var xy = sum(5, 10);
alert("Answer: " + xy);
```

Functions

- Let's make a more usable **`sum()`** Function

```
function sum() {
  if (arguments.length == 1) {
    return arguments[0];
  }
  else if (arguments.length > 1) {
    var total = arguments[0];
    for (var i = 1; i < arguments.length; i++) {
      total = total + arguments[i];
    }
    return total;
  }
  return "Incorrect number of Arguments";
}
```

Functions

- Let's test the **`sum()`** Function

```
var xy = sum(5, 10);
alert ("Answer: " + xy);
xy = sum(10);
alert ("Answer: " + xy);
xy = sum(10, 5, 12, 46);
alert ("Answer: " + xy);
xy = sum();
alert ("Answer: " + xy);
```