Javascript Basics

… a useless introduction

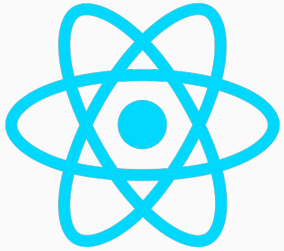
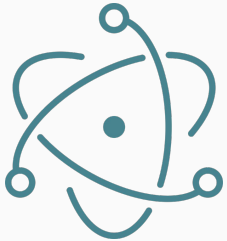
Pre-requisites

What do you need to get started with Javascript

- Any form of basic programming knowledge
- Some basic HTML, CSS Knowledge
- Basic understanding of procedural programming and object-oriented programming

What is Javascript used for?

Basically Everything

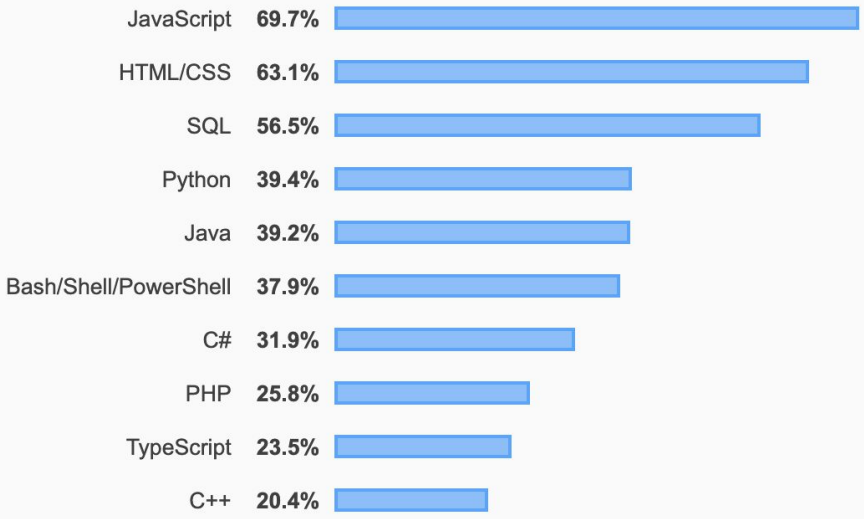



Introduction

- Compiled vs Interpreted Programming Language*
  - Has a two phase compilation and execution phase
  - Doesn't need a build step, unlike traditional compilers like **gcc**
  - Runs and executes code until an error occurs
- Multi-paradigm language
  - Allows procedural, object-oriented, functional and more paradigms of programming
- Loosely / Dynamically Typed
  - Does not need explicit type definition
  - Uses type inference and implicit type setting
  - A type assigned to a variable can be overridden by another

But why though?

- The de facto scripting language for Web based applications
- Easy to learn and get started with
- Build applications for a large range of platforms
- Becoming the most popular programming language



Stackoverflow Developer Survey 2019

Getting Started

How to write and run Javascript

- Internal Script

```
<script>/* Write some JS code */</script>
```

- External Script

```
<script src="./scripts/home.js"></script>
```

- Browser Console

Open the console tab in browser developer tools and write commands there.

Ctrl + Shift + J OR Cmd + Alt + J

The easy stuff...

Variables

```
var myNum = 10;
var myStr = 'Hello World';
var myBool = true;

console.log(myNum, typeof myNum);
console.log(myStr, typeof myStr);
console.log(myBool, typeof myBool);

myNum = '10';
console.log(myNum, typeof myNum);
```

Data Types in JavaScript

Primitive Data Types

- Number
- Boolean
- String
- Null
- Undefined

Non Primitive Data Types

- Object
- Array

String Operations

```
var myStr = 'Hello World';

console.log(myStr.length);
console.log(myStr.indexOf('ell'), myStr.indexOf('www'));
console.log(myStr[0], myStr[myStr.length - 1]);
console.log(myStr.concat('hello'), myStr);
console.log(myStr.split('ll'));
console.log(myStr.toUpperCase());
console.log(myStr.substr(1, 5));
```

Null and Undefined

```
var a;
console.log(a);

var b = null;
console.log(b);
```

null, undefined, '', 0 and false are `false` values.
Everything else is true.

Object

```
var obj = {
    num: 10,
    key: 'value',
    isTrue: true,
    arr: [1, 2, 3, 4, 5]
};

console.log(obj.num, obj['key'], obj.val);

obj.isTrue = 1;
obj.val = { newKey: 'newVal' };
console.log(obj.val.newKey);
```

Array

```
var array = [1, 2, 3, 'four', true, [1, 2, 3]];

array.push(10);
array.pop();

[1, 2, 3].push(20);

console.log(array[2]);
console.log(array.length);
console.log(array.indexOf(0));
console.log(array.slice(1, 2), array);
console.log(array.splice(2, 3), array);

[].concat([1, 2, 3], ['a', 'b'], 5);
```

Comparison and operators

```
1 == 1;
1 == 2;
1 == '1';
1 === '1';
1 <= 2;
1 >= 0;
'a' < 'b';
'c' > 'b'
'Hello' < 'Hello World'
```

```
true && true
true || true
!true
!!true

!!0
!!1
!!-1
!![]
!!{}
!!''
!!'Hello'
!!null
!!undefined
```

Conditionals

```
if (condition) {
    // do something
}

if (condition1) {
    // do something
} else if (condition2) {
    // do something
} else {
    // do something
}
```

```
switch (variable) {
    case value1:
        // do something
        break;
    case value2: {
        // do something
        break;
    }
    default:
        // do something
}
```

```
var value = condition
    ? value1
    : value2;
```

Loops

```
var a = [1, 2, 3, 4];

for (var i = 0; i < a.length; i++) {
    console.log(i, a[i]);
}

for (var i = 0; i < a.length; i++) {
    console.log(i, a[i]);
    if (a[i] === 3) {
        break;
    }
}
```

```
var a = [1, 2, 3, 4];

var i = 0;
while (i < a.length) {
    console.log(i, a[i]);
    i++;
}

var j = 0;
do {
    console.log(j, a[j]);
    j++;
} while (j < a.length);
```

Iterating over objects

```
var obj = {
    id: 1,
    name: 'John Doe'
};

var keys = Object.keys(obj);

for (var i = 0; i < keys.length; i++) {
    var key = keys[i];
    var value = obj[key];

    console.log(key, value);
}
```

Try the same with `Object.values(obj)`

Functions

```
function getSum(a, b) {
    return a + b;
}

getSum(1, 3);

var getSum = function(a, b) {
    return a + b;
}

getSum(1, 3);
```

```
function getSum(a, b) {
    function add() {
        return a + b;
    }

    return add();
}

getSum(1, 3);
```

The medium stuff...

Scopes

```
var globalScope = 10;

if (true) {
  var insideIf = 20;
}

console.log(insideIf);

for (var i = 0; i < 10; i++) {
  var insideFor = 30;
}

console.log(insideFor);
```

```
function myFunc() {
  var insideFunction = 40;
}
myFunc();

console.log(insideFunction);
```

Hoisting

```
console.log(myVar);

var myVar = 10;

console.log(myVar);

console.log(myFunc);

function myFunc() {
  return 'Hello';
}
```

```
console.log(myFunc2);

var myFunc2 = function() {
  return 'Hello Again';
}

console.log(myFunc2);
```

Function as a response

```
function init() {
    // do stuff;
    return function(a, b) {
        return a + b;
    }
}

var getSum = init();
var sum = getSum(1, 2);
var anotherSum = init()(1, 2);
```

Function as a parameter - Callback

```
function init(callbackFn) {
    // do stuff;
    callbackFn();
}

function getValue() {
    // do more stuff
}

var getSum = init(getValue);

init(function() {
    // do even more stuff
});
```

```
// Pseudo Code. Don't try this at home.
function getAPIValue(success, err) {
    var value = callAPI();

    if (value.status === 'success') {
        success(value);
     } else {
         err(value);
     }
}

getAPIValue(function(data) {
    console.log('SUCCESS', data);
}, function(err) {
    console.log('ERROR', err);
});
```

Iterating over Arrays - forEach and map

```
var array = [1, 2, 3, 4, 5];

var result = array.forEach(function(val, index) {
    console.log(index, val);
    return val + 1;
});

console.log('ForEach', result, array);

var result = array.map(function(val, index) {
    console.log(index, val);
    return val + 1;
});

console.log('Map', result, array);
```

Iterating over Arrays - filter

```
var array = [1, 2, 3, 4, 5];

var result = array.filter(function(val, index) {
    return val % 2 === 0; // return true or false
});

console.log(result);
```

Transforming Arrays - reduce

```
var result = array.reduce(function(acc, val, index, src) {
    return newAccValue;
}, defaultValue);

acc = accumulator, val = currentValue, index = currentIndex, src = initialSource

var array = [1, 2, 3, 4, 5];

var sum = array.reduce(function(acc, val) {
    return acc + val;
}, 0);
```

The hard stuff...

Asynchronous actions

```
var counter = 0;

var interval = setInterval(function() {
    console.log(counter);
    counter++;
}, 1000);

console.log(counter);

// clearInterval(interval);
```

```
var counter = 0;

var timeout = setTimeout(function() {
    console.log(counter);
    counter++;
}, 1000);

console.log(counter);

// clearTimeout(timeout);
```

Consider this code

```
for (var i = 0; i < 10; i++) {
    setTimeout(function() {
        console.log(i);
    }, 1000);
}
```

What is the output?

Now this

```
for (var i = 0; i < 10; i++) {
    setTimeout((function(val) {
        return function() {
            console.log(val);
        }
    })(i), 1000);
}
```

What is the output now?

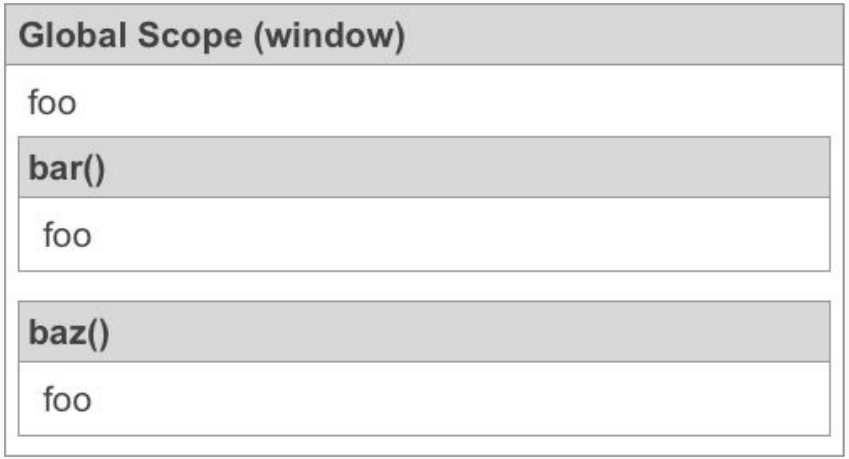
Keywords: **IIFE**, **Closures**

IIFE

```
(function (param) {
    // do something
})(value);
```

Compilation and Advanced Scopes

```
1  var foo = 'Global';
2
3  function bar() {
4      var foo = 'Inside bar';
5  }
6
7  function baz(foo) {
8      foo = 'Inside baz';
9      bam = 'Hello World';
10 }
```



Execution and Advanced Scopes

```
1  var foo = 'Global';
2
3  function bar() {
4      var foo = 'Inside bar';
5  }
6
7  function baz(foo) {
8      foo = 'Inside baz';
9      bam = 'Hello World';
10 }
```

Closures

```
var xyz = 0;

function A (xyz) {
  function C () {
    console.log(xyz);
  }
  function B () {
    console.log(xyz);
  }

  return {
    C: C,
    B: B
  }
}
```

```
var result = A(xyz);

console.log(result)
console.log(result.C())
console.log(result.B())
```

Advanced Concepts

Concepts that are part of the core javascript functionality

- Types and Coercion
  https://www.freecodecamp.org/news/js-type-coercion-explained-27ba3d9a2839/
- Lexical Scoping and Closures
  https://medium.com/@nickbalestra/javascripts-lexical-scope-hoisting-and-closures-without-mystery-c2324681d4be
- Prototypes and Prototype Chain
  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Inheritance_and_the_prototype_chain
- Event Loop
  https://flaviocopes.com/javascript-event-loop/
  https://medium.com/front-end-weekly/javascript-event-loop-explained-4cd26af121d4
  https://www.youtube.com/watch?v=8aGhZQkoFbQ
  https://www.youtube.com/watch?v=cCOL7MC4Pl0

Learning links

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide

https://github.com/getify/You-Dont-Know-JS

https://www.tutorialspoint.com/javascript

https://www.javascript.com/try

https://javascript.info/

https://www.learn-js.org/

And that's it.