JavaScript

Dynamic and Interactive Web pages

Web Developer Setup Tools and Resources

Editor to write the code - Visual Studio Code (https://code.visualstudio.com/)

Browser to render and see the code - Chrome

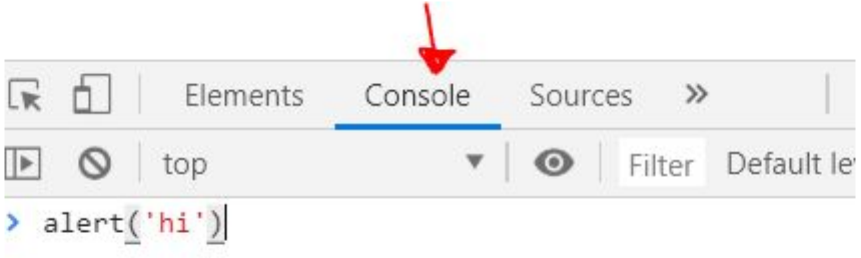
Online editor - (https://codepen.io/)

Developer tools - Chrome developer tools (https://developers.google.com/web/tools/chrome-devtools/)

The browser console

Inspect to open developer tools console

Console tab - try it write some JavaScript in the console.

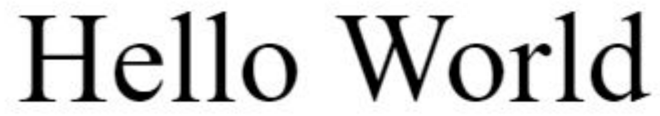


```
> alert('hi')
```

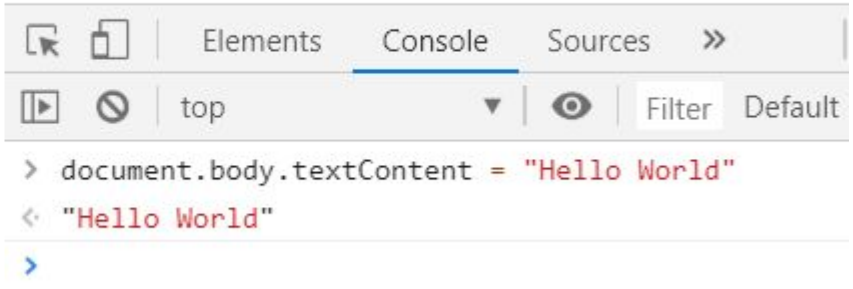
Web Page Document Object Model

Update webpage Content

document.body.textContent = "Hello World"


Hello World

```
document.body.textContent = "Hello World"
```



```
> document.body.textContent = "Hello World"
< "Hello World"
>
```

Laurence Svekis https://basescripts.com/ JavaScript Courses

Add JavaScript to HTML Page

Add script tags to html page.

save the html file

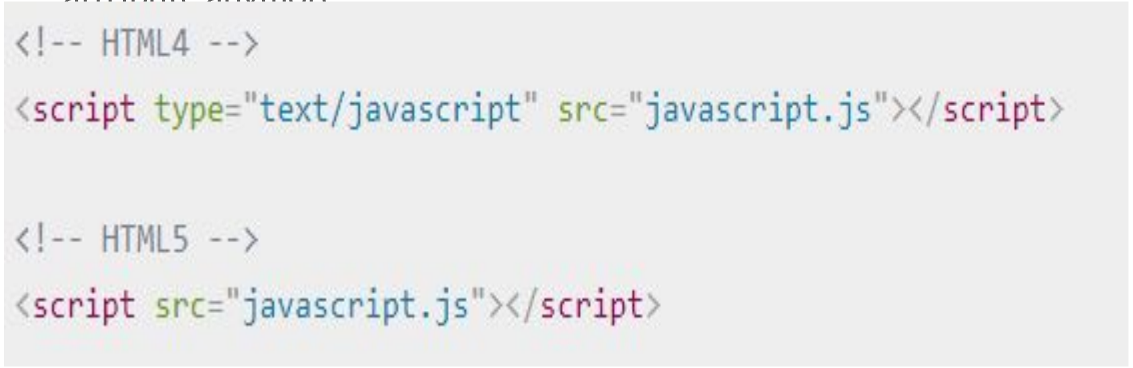
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title></title>
</head>
<body>
    <script>
        alert('hello');
        document.body.textContent = "Hello World";
    </script>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>learn JavaScript</title>
</head>
<body>
    <script src="1js.js"></script>
</body>
</html>
```

```
alert('hello');
document.body.textContent = "Hello World";
```

Add JavaScript to HTML Page

With HTML5 you don't need to include the type
attribute anymore



```
<!-- HTML4 -->
<script type="text/javascript" src="javascript.js"></script>

<!-- HTML5 -->
<script src="javascript.js"></script>
```

Console and commenting - debugging

Use console.log within code to check variable values. Use the console to test.

```
//console for debugging
console.log('hello');
```

```
///commenting
alert('hello'); // this is a test
document.body.textContent = "Hello World"; // this is to output content
/*
Block comment
alert('hello');
document.body.textContent = "Hello World";
*/
```

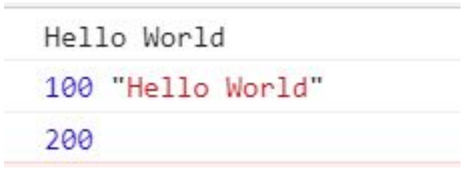
Variables and Constants

Variables are one of the most fundamental notions.

Give us a way to store values and then use that value.

Stores the value in memory and can be accessed later in the code using the variable.

let allows us to change the value



```
Hello World
100 "Hello World"
200
```

```
let message = "Hello World";
console.log(message);
let points = 100;
console.log(points,message);
points = 200;
console.log(points);
```

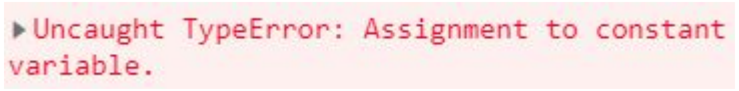
var let const

const - does not allow you to change the value. Avoid overwriting values.

Both **let** and **const** are modern ways of setting variables.

Use **let** if you plan to change the value and **const** if no changes are going to be done.

var keyword is older - allows us to change the value.



```
▶Uncaught TypeError: Assignment to constant variable.
```

```
let message = "Hello World";
console.log(message);
let points = 100;
console.log(points,message);
points = 200;
console.log(points);

const val = 100;
//val = 200; // will not work
var year = 2019;
year = 2020;
console.log(year,val,points,message);
```

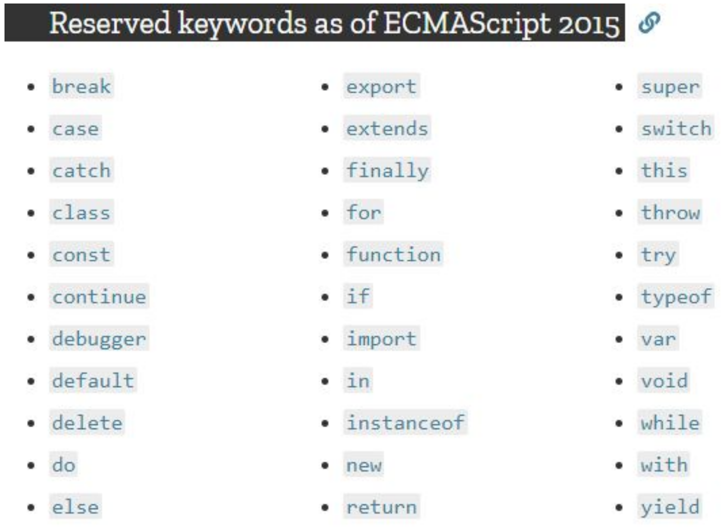
TIPS - Keep in Mind

JavaScript is Case Sensitive

camelCase when writing JavaScript variables

Can’t use reserved words (const const = 100) <- won’t work

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Lexical_grammar#Reserved_keywords_as_of_ECMAScript_2015



Reserved keywords as of ECMAScript 2015

- `break`
- `case`
- `catch`
- `class`
- `const`
- `continue`
- `debugger`
- `default`
- `delete`
- `do`
- `else`
- `export`
- `extends`
- `finally`
- `for`
- `function`
- `if`
- `import`
- `in`
- `instanceof`
- `new`
- `return`
- `super`
- `switch`
- `this`
- `throw`
- `try`
- `typeof`
- `var`
- `void`
- `while`
- `with`
- `yield`

TIPS - Keep in Mind

Comments Comments are used to add hints, notes, suggestions, or warnings to JavaScript code. This can make it easier to read and understand. Add comments and info to document what is happening.

Whitespace makes it readable - White space characters improve the readability of source text and separate tokens from each other.

Line terminator Statements - end each line statement with a semicolon; Line terminator characters are used to improve the readability of the source text

Data Types

strings = need quotes can contain digits and characters. Can be single or double quotes. If you want to include the “ or ‘ then you need to break out of the string.

```
let num = 1000; /// Number
let str = 'Hello World'; //string single quotes
let str2 = "Hello World"; // string double quotes
let boo = true; //Boolean
let nu = null; // Null set a variable to no value
let undef = undefined; // used for variables that have not been defined
// Objects like arrays and objects are complex data types
// Symbol are used with objects
```

Fun with Strings

```
let first = 'Laurence';
let last = 'Svekis';
console.log(first,last);
let fullName = first + ' ' + last;
console.log(fullName);
console.log(first.length);
console.log(first[0]);
console.log(last[0]);
let initials = first[0] + last[0];  //store the values that are returned
console.log(initials);

//built in functions - will take the value and do something with it
console.log(first.toUpperCase());
console.log(first.toLowerCase());
console.log(first); // value hasn't changed.
```

```
//Finding a String in a String
let str = 'I love JavaScript because its JavaScript';
let val = str.indexOf('JavaScript');
console.log(str,val);
let lastVal = str.lastIndexOf('JavaScript');
console.log(lastVal);

let valSearch = str.search('JavaScript');
console.log(valSearch);

let val2 = str.indexOf('JavaScript',20); // Second parameter sets start
position of search
console.log(val2);
```

Parts of Strings

Get the part of a string.

- slice(start, end) - extracts a part of a string and returns the extracted part in a new string.
- substring(start, end)
- substr(start, length)

```
let first = 'Laurence';
let last = 'Svekis';
let needle = 'JavaScript';
let str = 'I love ' + needle + ', because its ' + needle + ', and ' + needle +
' is the best.';
let start = str.indexOf(needle);
let len = needle.length;
let end = start + len;
console.log(start, len, end);
let firstSlice = str.slice(7); // parameter is starting index.
console.log(firstSlice);
let mySlice = str.slice(7, 17); // second parameter is ending index.
console.log(mySlice);
let betterSlice = str.slice(start, end);
console.log(betterSlice);
let subby = str.substring(start, end);
console.log(subby);
let subby2 = str.substr(start, len); //uses length of the extracted part as
second parameter.
console.log(subby2);
```

Changing Strings

You can use **replace** within the string to replace all instances of searched term.

```
let replacer = str.replace('JavaScript', 'HTML');
console.log(replacer);
let str2 = str.replace('HTML', needle);
console.log(str2);
```

Numbers

No quotes needed.

Can be with or without decimals

Positive or negative

Assign a value with = sign

Do arithmetic with the numbers.

Math operators +,-,*,/,%

```
let a = 10;
let b = 25;
console.log(a, b);
// Math operators +,-,*,/,%
let c = a * b;
console.log(c);
console.log(a + b); //35
console.log(a - b); // -15
console.log(a * b); // 250
console.log(a / b); // 0.4
console.log(b % a); // 5 remainder
```

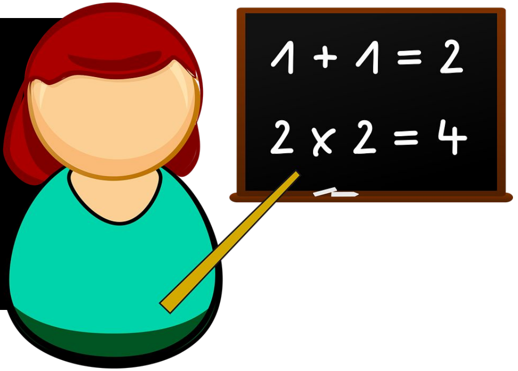
More Numbers

Uses the order of operations you might have learned in school.

```
/*
Parentheses means brackets()
Exponents (and Roots) means power
Multiplication & Division
Addition & Subtraction
*/
console.log( 5 + (a * b) - 10); // 245
```

You can also use shorthand methods

```
a = a + 1; //11
a++; // 12
a = a + 10; //22
a += 10; //32
a -= 22; //10
console.log(a);
```



Number Methods

```
let a = 10;
let b = '10';
console.log(a + b); //1010 - javaScript converts to string
let c = '20 hello';
console.log(a + c); //10hello
b++;
console.log(b); //11
//c++;
//console.log(c); // NaN
let d = Number(b); //convert JavaScript variables to numbers
console.log(a+d); // 21
console.log(Number(c)); //NaN
console.log(parseInt(c)); //20 parses string returns the first number if available.
```

Template literals

```
let greeting = 'Hello everyone';
let fav = 'JavaScript';
let mes1 = 'I\'d rather be coding ' + fav;
let mes2 = "Did you say \"JavaScript\"";
let mes3 = `This is a better way to ${greeting} and add variables ${fav}!`;
let message = mes3;
console.log(message);
let html = `
    <h1>${fav}</h1>
    <p>${mes1}</p>
    <p>${mes2}</p>
`;
console.log(html);
```

Arrays

Store multiple values in one variable

```
let friends = ['Mike', 'John', 'Lisa', 'Jane'];
console.log(friends);
console.log(friends[2]); //zero based starts with zero
console.log(friends.length);
friends[2] = 'Jack'; // Lisa becomes Jack
console.log(friends);
let mixer = ['Hello', 50, true, 100, 600, null, [1, 2, 3]];
console.log(mixer);
console.log(mixer.toString());
let joiner = friends.join(' -*- ');
console.log(joiner);
```

```
let itemLast = friends.pop(); // changes array removes last item
console.log(itemLast); //'Jane'
console.log(friends); //'Mike','John','Jack'
friends.push('Linda'); //adds to array at end 'Linda'
console.log(friends); //'Mike','John','Jack','Linda'
let itemFirst = friends.shift(); //removes first item 'Mike'
console.log(friends); //'John','Jack','Linda'
friends.unshift('Bob'); //adds to beginning of array
console.log(friends); //'Bob',John','Jack','Linda'
let middle = friends.splice(2, 2, 'Larry', 'Joe'); //removes middle starting at 2 for 2 items
console.log(middle); //'Jack','Linda'
console.log(friends); //'Bob',John','Larry','Joe'
let first = friends.splice(0, 1); //removes first item 'Bob'
console.log(friends); //John','Larry','Joe'
let myArray = friends.concat(mixer); // merges 2 arrays together
console.log(myArray);
```

Arrays Sort

Can be sorted or reverse sorted order. Includes can be used to check if a value is in the array

```
let friends = ['Mike', 'John', 'Lisa', 'Jane'];
friends.sort();//"Jane", "John", "Lisa", "Mike"
console.log(friends);
friends.reverse(); //"Mike", "Lisa", "John", "Jane"
console.log(friends);

console.log(friends.includes('Mike'));
```

Booleans

True or False

```
let okay = true;
let notOkay = false;
let message = 'Hello JavaScript';
let word = 'JavaScript';
let checker = message.includes('JavaScript');
console.log(checker); //true
console.log(message.includes('HTML')); //false
console.log(okay);
```

```
console.log(word == 'JavaScript'); //truthy true
console.log(word == 'HTML'); //falsey false
console.log(5 == 5); //true;
console.log(5 != 5); //false;
console.log(10 > 5); //true;
console.log(10 < 5); //false;
console.log(5 < 5); //false;
console.log(5 <= 5); //true;
console.log(5 >= 5); //true;
console.log('5' == 5); //true not checking data type
console.log('5' === 5); //false absolute check
console.log('5' !== 5); //true
console.log('5' != 5); //false
console.log('6' !== 5); //true
console.log('6' != 5); //true
console.log(typeof('5')); //string
console.log(typeof(5)); //number
```

Conditions

```
let okay = true;
let notOkay = false;
let message = 'Hello JavaScript';
let word = 'JavaScript';
let checker = message.includes('JavaScript');
if (true) {
    console.log('was true');
    //do something
}
if (false) {
    console.log('was true');
    //do something
}
if (okay) {
    console.log('yes that worked'); //outputs
} else {
    console.log('did not work');
}
```

```
if (notOkay) {
    console.log('yes that worked');
}
else {
    console.log('did not work'); //outputs
}
if (word === 'HTML') {
    console.log('it was HTML');
}
else if (word === 'JavaScript') {
    console.log('it was JavaScript');
}
else {
    console.log('no matches');
}
```

Ternary operator

condition ? val1 : val2

var status = (age >= 18) ? 'adult' : 'minor';

```
let okay = true;
if (okay) {
    console.log('yes that worked'); //outputs
}
else {
    console.log('did not work');
}
let message = okay == true ? 'yes that worked' : 'did not work';
console.log(message);
```

For loops while and do

```
for (let x = 0; x < 5; x++) {
    console.log(x);
}
console.log('loop completed');
let y = 0;
while (y < 5) {
    console.log(y);
    y++;
}
console.log('while completed');
let z = 0;
do {
    console.log(z); //will run at least once if you change z to 10
    z++;
}
while (z < 5);
console.log('dowhile completed');
```

```
let friends = ['Mike', 'John', 'Lisa', 'Jane'];
for (let i = 0; i < friends.length; i++) {
    console.log(friends[i]);
    let html = `<div>${i+1}. ${friends[i]}</div>`;
    log(html);
}
let j = 0;
while (j < friends.length) {
    console.log(friends[j]);
    let html = `<div>${j+1}. ${friends[j]}</div>`;
    log(html);
    j++;
}
```

Functions

Function declarations load before any code is executed while Function expressions load only when the interpreter reaches that line of code.
message1(); //won't work before the line of code that has the expression.
message2(); // will work even before the line of code with the declaration.

```
const message1 = function(){
    console.log('Hello World function expression');
}

message1();
message1();
message1();
```

```
// function declaration
function message2(){
    console.log('Hello World function declaration');

}
message2();
message2();
message2();
```

Functions arguments and parameters

```
const message1 = function (mes) {
    console.log(mes);
}
message1('hello');
message1('welcome');
message1('bye bye');

function message2(mes) {
    console.log(mes);
}
message2('hello');
message2('welcome');
message2('bye bye');
```

```
const message3 = function(mes = 'Hello Everyone',greet ='Nice to see
you'){
    let output = `
    ${mes}
    ${greet}
`;
    console.log(output);
}
message3('hello');
message3('hello','welcome');
message3();


function tester(para1,para2){
    return (para1,para2);
}
console.log(tester('argument1','argument2'));
```

Return and arrow functions

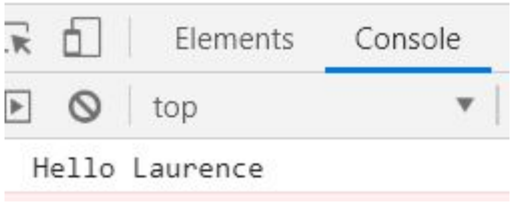
```
const addme = function (a = 0, b = 0) {
    console.log(a,b);
    return (a + b);
}
console.log(addme()); //0
console.log(addme(5)); //5
console.log(addme(5, 7)); //12
```

```
const adder = (a = 0, b = 0) => {
    console.log(a,b);
    return (a + b);
}

console.log(adder()); //0
console.log(adder(5)); //5
console.log(adder(5, 7)); //12


const adder2 = (a = 0, b = 0) => a + b; //return in one line
console.log(adder2()); //0
console.log(adder2(5)); //5
console.log(adder2(5, 7)); //12
```

Callback function

A callback function is a function passed into another function as an argument, which is then invoked inside the outer function to complete some kind of routine or action

```
function greeting(name) {
    console.log('Hello ' + name);
}

function test(callback) {
    let name = 'Laurence';
    callback(name);
}
test(greeting);
```

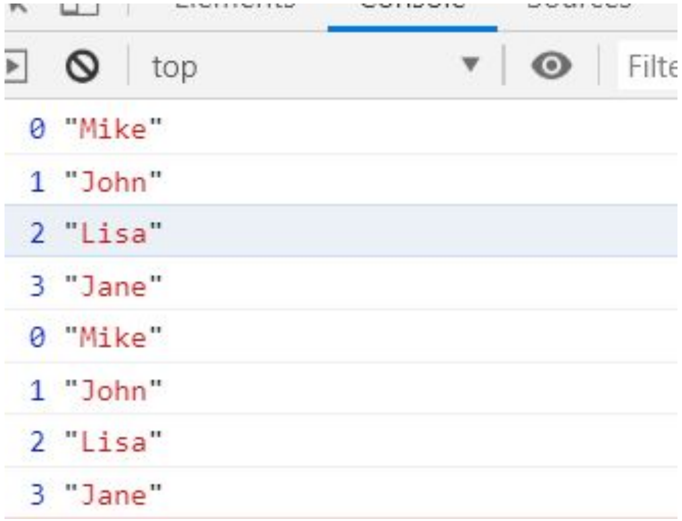


```
Hello Laurence
```

forEach

Iterate data within an array

```
let friends = ['Mike', 'John', 'Lisa', 'Jane'];
friends.forEach(function (friend, index) {
    console.log(index, friend);
})
friends.forEach((friend, index) => {
    console.log(index, friend);
})
```



```
0 "Mike"
1 "John"
2 "Lisa"
3 "Jane"
0 "Mike"
1 "John"
2 "Lisa"
3 "Jane"
```

Objects

Objects like arrays hold lots of information

```
let car = {
    make: 'ford'
    , model: 'mustang'
    , year: 2000
    , price: 50000
    , color: 'red'
    , tires: true
};
console.log(car);
car.color = 'blue';
console.log(car);
car['color'] = 'green';
console.log(car);
```

```
let prop = 'color';
car[prop] = 'black';
console.log(car);
car.inside = ['radio', 'seats', 'steering wheel', true];
console.log(car);
car.outside = {
    tires: 4
    , windshield: 'glass'
    , bumper: true
};
console.log(car);
console.log(car.outside.tires);
```

Object Methods

```
let car = {
    make: 'ford'
    , model: 'mustang'
    , year: 2000
    , price: 50000
    , color: 'red'
    , tires: true
    , drive() {
        console.log('its driving');
    }
    , instructions: ['turn key', 'put in gear', 'press gas pedal', 'turn wheel as needed']
    , help() {
        console.log(this);
    }
    , info() {
        console.log(`Make ${this.make} Model ${this.model}`);
    }
    , howto() {
        this.instructions.forEach(step => {
            console.log(step);
        })
    }
};
console.log(car);
car.drive();
car.howto();
```

Object values

```
let friend1 = {
    firstName: 'John'
    , lastName: 'Smith'
    , age: 40
};
console.log(Object.keys(friend1));
for (prop in friend1) {
    console.log(prop);
}
let friend2 = {
    firstName: 'Jane'
    , lastName: 'Doe'
    , age: 25
};
console.log(Object.values(friend2));
console.log(friend1);
console.log(friend2);
```

```
console.log(Object.values(friend2));
for (prop in friend2) {
    console.log(friend2[prop]);
}
console.log(friend1);
console.log(friend2);
let friends = {
    list: [friend1, friend2]
};
console.log(friends);
console.log(friends.list);

friends.list.forEach(function (friend) {
    console.log(`${friend.firstName} ${friend.lastName} `);
})
```

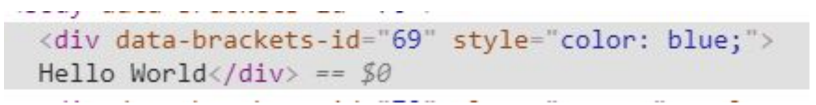
Document Object Model

Use DOM method to select page elements into useable objects in JavaScript.

```
<div>Test 1</div>
<div class="output">Test 2</div>
<div id="main">Test 3</div>
<script>
    console.dir(document);
    const el1 = document.querySelector('div');
    console.log(el1);
    const el2 = document.getElementById('main');
    console.log(el2);
    const el3 = document.querySelector('#main');
    console.log(el3);
    const el4 = document.querySelector('.output');
    console.log(el4);
</script>
```

Use JavaScript to manipulate and get values from elements.

```
        console.log(el1.textContent);
        el1.textContent = "Hello World";
        console.log(el1.textContent);
        el2.innerHTML = "<h1>Welcome to the DOM</h1>";
        console.log(el3.textContent); // el3 and el2 are referencing same
object.
        el4.style.backgroundColor = "red"; //updated style attribute of
element
        el1.style.color = "blue";
```



```
<div data-brackets-id="69" style="color: blue;">
Hello World</div> == $0
```

Select Multiple Elements

Loop all matching elements from selection.

```
<div>Test 1</div>
<div class="output">Test 2</div>
<div id="main">Test 3</div>
<script>
    const els = document.querySelectorAll('div');
    console.log(els); // node list of all elements
    console.log(els.length); //length like an array
    for(let x=0;x<els.length;x++){
        console.log(els[x]);
        els[x].textContent = `${x} ${els[x].textContent}`;
    }
</script>
```

Can use forEach as well.

```
<div>Test 1</div>
<div class="output">Test 2</div>
<div id="main">Test 3</div>
<script>
    const els = document.querySelectorAll('div');
    els.forEach(function(el){
        console.log(el);
        el.style.color = "red";
    })
</script>
```

Congratulations on completing the course!

Thank you

This ebook uses https://developer.mozilla.org/en-US/docs/Web/JavaScript as a source for examples. Check out more about JavaScript at MDN.

Course instructor : Laurence Svekis - providing online training to over 1,000,000 students across hundreds of courses and many platforms.

Laurence Svekis https://basescripts.com/ JavaScript Courses