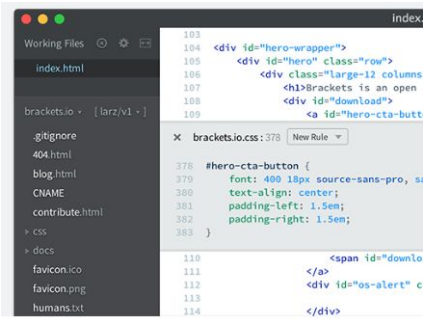
JavaScript Fundamentals

ES6 for Beginners

Developer Tools

http://brackets.io/



https://developers.google.com/web/tools/chrome-devtools/

Keyboard shortcuts for opening DevTools

To open DevTools, press the following keyboard shortcuts while your cursor is focused on the browser viewport:

Action	Mac	Windows / Linux
Open whatever panel you used last	`Command+Option+I`	`F12` or `Control+Shift+I`
Open the **Console** panel	`Command+Option+J`	`Control+Shift+J`
Open the **Elements** panel	`Command+Option+C`	`Control+Shift+C`

Try Console console.log

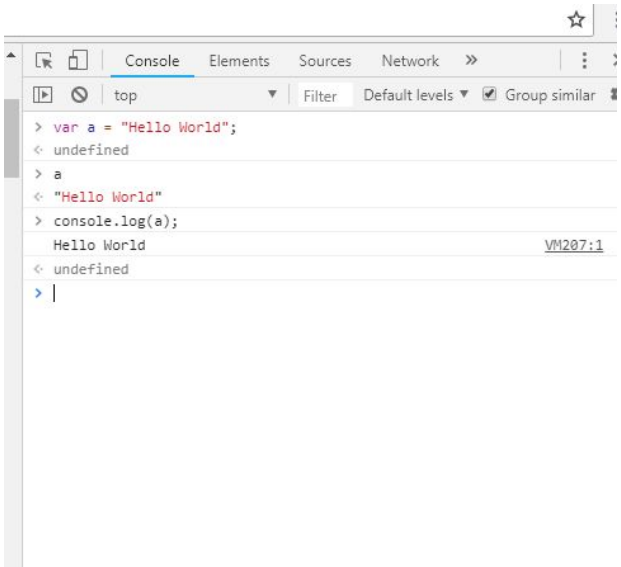
Outputs a message to the Web Console.

Try it: console.log(“Hello”);

Options:

console.log prints the element in an HTML-like tree

console.dir prints the element in a JSON-like tree



Compatibility Table

http://kangax.github.io/compat-table/es6/

http://babeljs.io/

Variables - Let & Const

Variables are ‘a named space in the memory’ that stores values.

Identifier rules

- Cannot be keywords
- Cannot begin with a number
- Cannot contain spaces or special characters other than _ or $

Variables must be declared before using it. Only declare a variable once within its scope otherwise you get a already declared error in the code.

Data Types

The latest ECMAScript standard defines seven data types:

Six data types that are primitives:

- Boolean
- Null
- Undefined
- Number
- String
- Symbol (new in ECMAScript 2015)

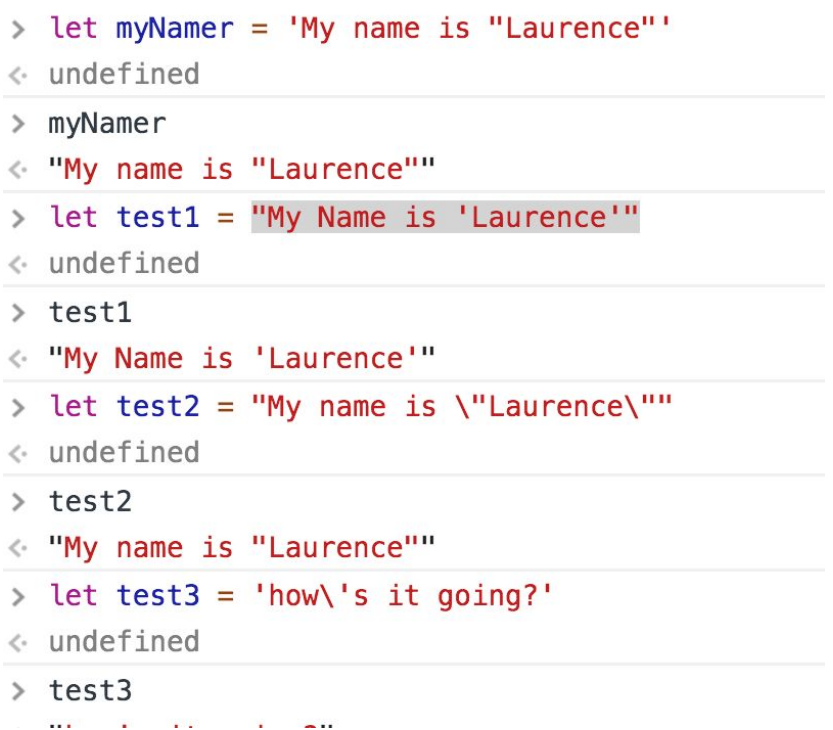
and Object

Data Types Examples

```
var a = true; // true of false = Boolean

var b = 100; // can be written with or without decimal point
= Number

var c = ‘Hello World’; // can be inside single or double
quotes = String

var d = null; // It is explicitly nothing. = Null

var e; // has no value but is declared = Undefined

var f = Symbol("value"); // represents a unique identifier.
= Symbol
```

Break out of quotes

Use backslash to break out of quotes



Variables - let

new keyword to declare variables: let

‘let’ is similar to var but has scope. Only accessible within the block of code that it is defined. let restricts access to the variable to the nearest enclosing block.

Variables - const

new keyword to declare variables: const

‘const’ is similar to var but has scope. Only accessible within the block of code that it is defined. Also for values that are not going to be changed and are fixed with a read-only reference to a value.

- const cannot be reassigned a value
- const variable value is immutable
- const cannot be redeclared
- const requires an initializer

```
let e = 100;
e++;
const d = 100;
d++;
```

```
Only available in this block                    lesson1.html:17
▸Uncaught TypeError: Assignment to constant variable.   lesson1.html:24
    at lesson1.html:24
```

```
if(a){
    console.log(a);
    const c = 'Only available in this block';
    console.log(c);
}
console.log(c);

</script>
```

```
undefined                                       lesson1.html:3
Hello world                                     lesson1.html:5
Hello world                                     lesson1.html:8
Only available in this block                    lesson1.html:10
Hello world                                     lesson1.html:15
Only available in this block                    lesson1.html:17
▸Uncaught ReferenceError: c is not defined      lesson1.html:19
    at lesson1.html:19
```

Have combining data types works

Try adding and multiplying different data type together

```
"" + 5
"5"
let a = "" + 6 + 7;
undefined
a
"67"
let b = "Hello"+"World"
undefined
b
"HelloWorld"
let c = true + false
undefined
c
1
```

```
typeof c
"number"
"4"*"5"
20
"4"+"5"
"45"
null + 5
5
```

Objects

An object is a collection of related data and/or functionality (which usually consists of several variables and functions — which are called properties and methods when they are inside objects.) Objects are not a primitive data Type.

```
var person = {age:25};
```

person.age - Dot notation

person['age'] - Bracket notation

Arrays

Arrays are generally described as "list-like objects"; they are basically single objects that contain multiple values stored in a list. Array objects can be stored in variables and dealt with in much the same way as any other type of value, the difference being that we can access each value inside the list individually, and do super useful and efficient things with the list, like loop through it and do the same thing to every value.

```
var shopping = ['bread', 'milk', 'cheese', 'hummus',
'noodles'];

console.log(shopping);
```

Variables from Objects

The destructuring assignment syntax is a JavaScript expression that makes it possible to unpack values from arrays, or properties from objects, into distinct variables.

```
var x = [1, 2, 3, 4, 5];
var [y, z] = x;
console.log(y); // 1
console.log(z); // 2
```

Alerts

Simple interaction with user

```
alert("Hello");
let result = prompt('What is your name', 'None');
console.log(result);
let result2 = prompt('Do you love JavaScript?');
if(result2) {
    console.log(result2);
}
```

Operators

Operators can be used to assign values and perform calculations.

```
console.log(50%6);  //
a++;
b--;
console.log(a);
console.log(b);

let tester = 500;
console.log(tester++);
console.log(++tester);
```

Functions

Functions blocks of reusable code. Functions are one of the fundamental building blocks in JavaScript. A function is a JavaScript procedure—a set of statements that performs a task or calculates a value. To use a function, you must define it somewhere in the scope from which you wish to call

Function declarations

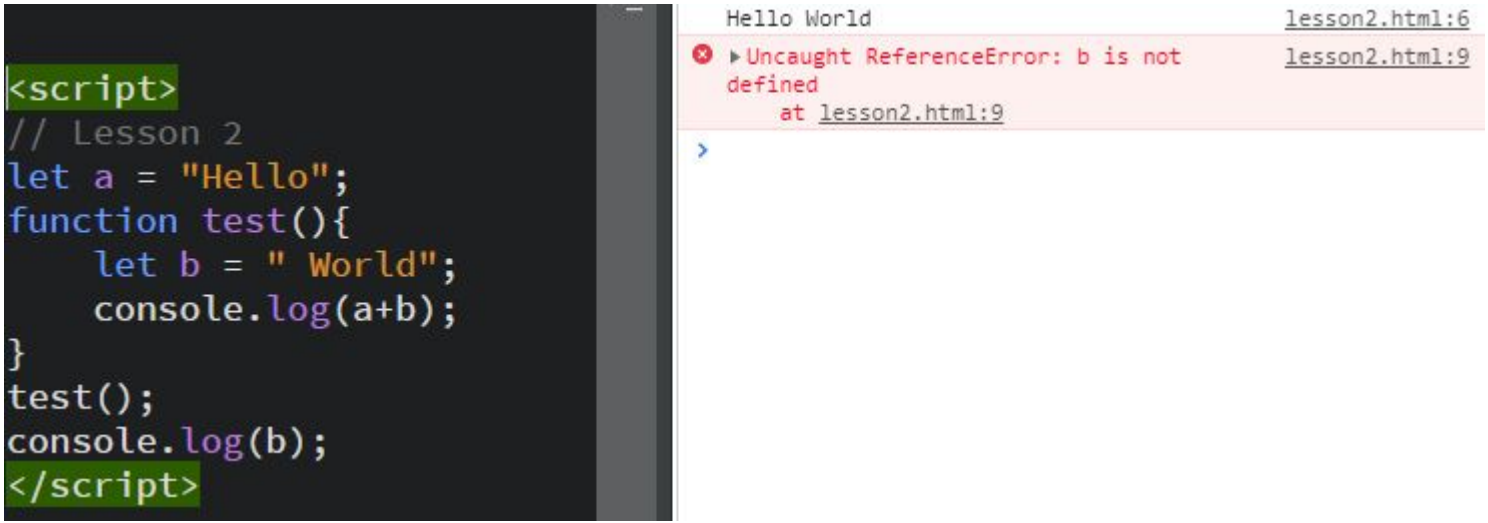
A **function definition** (also called a **function declaration**, or **function statement**) consists of the `function` keyword, followed by:

ctions

- The name of the function.
- A list of parameters to the function, enclosed in parentheses and separated by commas.
- The JavaScript statements that define the function, enclosed in curly brackets, `{ }`.

Functions and Scope

Global vs. Local Variables



```
<script>
// Lesson 2
let a = "Hello";
function test(){
    let b = " World";
    console.log(a+b);
}
test();
console.log(b);
</script>
```

```
Hello World                                          lesson2.html:6
▸Uncaught ReferenceError: b is not defined           lesson2.html:9
    at lesson2.html:9
>
```

Arrow functions

An arrow function expression (previously, and now incorrectly known as fat arrow function) has a shorter syntax compared to function expressions and lexically binds the this value. Arrow functions are always anonymous.

```
var test10 = function (x) {
    return x * 5;
}
const test11 = (x) => x * 5;
console.log(test10(5));
console.log(test11(5));
```

Arrow functions

1. Declare variable - name of function to invoke it
2. Define body within {}

Default parameters

```
const test12 = (x=15) =>{
    return x * 10;

}
console.log(test12()); //150
console.log(test12(5)); //50
```

```
const test13 = (num, x = 15) => {
    return x * 10;
}
console.log(test13(1, 1)); //10
console.log(test13(5)); //150
```

Conditional operators and Comparisons

Evaluation of content whether it’s true or false.

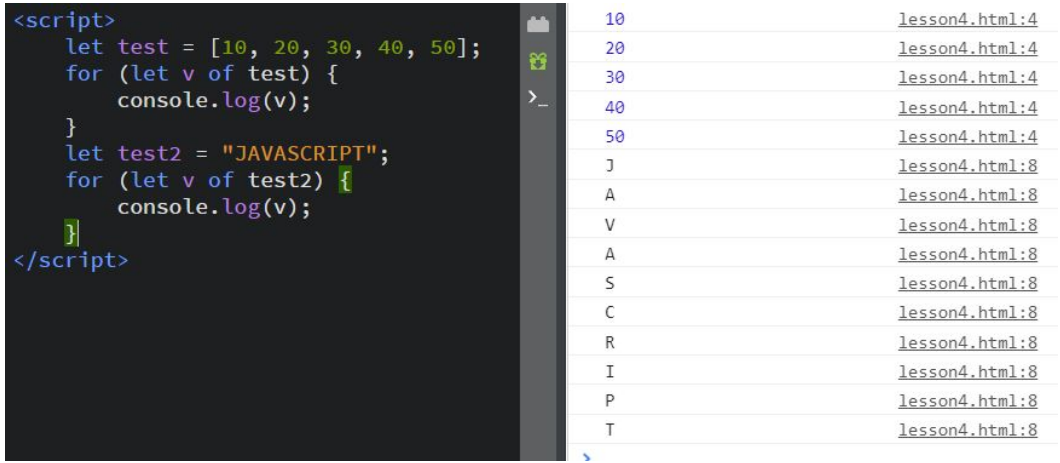
```
let x = 10;
let y = 0;
if (x) {
    console.log('X has a value');
}
if (y) {
    console.log('Y has a value');
}
```

Loops

for (variable of iterable) { statement }

Variable : On each iteration a value of a different property is assigned to variable.

Iterable : Object whose iterable properties are iterated.



MAP()

The Map object holds key-value pairs. Any value (both objects and primitive values) may be used as either a key or a value. Creates a new array with the results of calling a provided function on every element in the calling array.

- map() method creates a new array returning a function execution for every array element.
- map() method iterates through each element in an array, in order.

Map set and get

```
var myMap = new Map();
```

- Indexes are unique in maps.

```
var test1 = new Map();
test1.set('name', 'Laurence');
test1.set('name', 'New Name');
test1.set('last', 'Svekis');
console.log(test1.get('name'));
for (let key of test1.keys()) {
    console.log(key);
}
for (let val of test1.values()) {
    console.log(val);
}
for (let element of test1) {
    console.log(element);
}
for (let [key, val] of test1) {
    console.log(key + ' : ' + val);
}
console.log(test1.size);
console.log(test1.keys());
console.log(test1.values());
```

Sets

```
var mySet = new Set();
```

- Unique values are displayed

```
var test1 = new Set();
test1.add('name');
test1.add('name');
test1.add('last');
console.log(test1);

for (let element of test1) {
    console.log(element);
}
```

Fetch

The Fetch API provides a JavaScript interface for accessing and manipulating parts of the HTTP pipeline, such as requests and responses. It also provides a global fetch() method that provides an easy, logical way to fetch resources asynchronously across the network.

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

```
fetch('http://example.com/movies.json')
  .then(function(response) {
    return response.json();
  })
  .then(function(myJson) {
    console.log(myJson);
  });
```

Thank you for taking the Course

Don’t forget to practice and challenge yourself with JavaScript.