

JavaScript 101

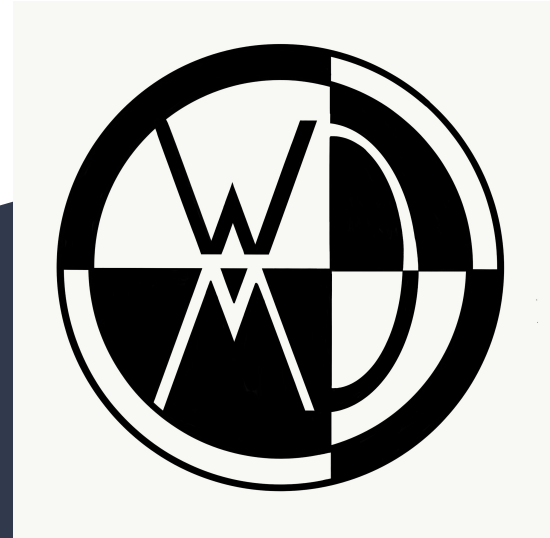
Data: 9/27/2020

Follow us on social media!

Website: webdvt.org

Instagram: [@WebDVT](https://www.instagram.com/WebDVT)

Facebook: [@WebDVT](https://www.facebook.com/WebDVT)



What is JavaScript?

- Interpreted programming language
- Statically typed
- Conforms to **ECMAScript** specification
- Runs on the client/browser as well as on server (Node.js)



Why Learn JavaScript?

- Programming language of the browser
- Can be used for both front-end (React) and back-end (Node.js)→ **full stack** applications
- Used in **mobile** development (React Native, Ionic)
- Used in **desktop** application development (Electron JS)

Directions


1. Create a folder on your computer called **"webdvt_workspace"** (if you missed out previous meetings).
2. Then create another folder (inside **"webdvt_workspace"**) called **"js-cheat-sheet"**.
3. Open up WebStorm (or any text editors of your choosing).
4. Inside Webstorm, open the folder **"js-cheat-sheet"**.
5. Right click on the folder on your left sidebar. Create an HTML file named **"index.html"**.
6. Create another file called **"main.js"**.



How to run JS?

1. Add `<script src="NAME_OF_JS_FILE.js"></script>` tag inside the `<body>` of your HTML document

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>JavaScript Cheat Sheet</title>
</head>
<body>
  <h1>JS Cheat Sheet sd</h1>
  <p>Open developer tools and view the console to see the output of the Javascript code!</p>
  <script src="main.js "></script>
</body>
</html>
```



2. Open the HTML document in browser
3. Right-click on the page and select "inspect"
4. Go to the "console" tab
5. Refresh your web page
6. You should see the output from your JS code

Variables & Data Types

- Var, let, const
 - Var is the old standard for declaring variables
 - Let is the new standard
 - Const - value cannot be changed
- Data types
 - String
 - Number
 - Boolean
 - Null
 - Undefined
 - Object

```
// --- Primitive data types ---  
// string, number, boolean, null, undefined  
const name = 'John'; // string  
const age = 27; // number  
const isMarried = false; // boolean  
const height = 5.6; // number  
const x = null;  
const y = undefined;
```

Var vs. let and const?

var vs. **let** vs. **const**

```
function order(x, y) {  
  if (x > y) {  
    let tmp = x;  
    x = y;  
    y = tmp;  
  }  
  console.log(tmp===x);  
  // ReferenceError: tmp is not defined  
  return [x, y];  
}
```

let works similarly to **var**, but the variable it declares is block-scoped, it only exists within the current block. **var** is function-scoped.

How should you declare variable in JS?

CONST vs LET vs VAR

ES6 Conventions:

1. Use ``const`` by default.
2. Use ``let`` if you have to rebind a variable.
3. Use ``var`` to signal untouched legacy code

Source: <https://twitter.com/raganwald/status/564792624934961152>

JS

JS Template Literal

Template literals are **string literals** allowing embedded expressions.

Template literals are enclosed by the **backtick** (```) character instead of double or single quotes.

```
const name = 'John';
const age = 27;

// String concatenation
console.log("My name is " + name + " and I am " + age + " years old.");
// Template Literal: lets you to inject variables & logic directly into a string
console.log(`My name is ${name} and I am ${age} years old.`);
// Both logs: "My name is John and I am 27 years old."
```

Arrays

- Type of object
- Single variable used to store multiple elements
- Can store elements of different data types
- Elements are accessed by passing index

```
const fruits = ['apples', 'oranges', 'bananas', 'mangoes'];
```

Spread Operator with Array



```
const array1 = [🍏, 🍌, 🥑];  
const array2 = [🥦, 🌽, 🌶️];  
  
const array3 = [...array1, ...array2];  
  
// => [🍏, 🍌, 🥑, 🥦, 🌽, 🌶️]
```

<https://medium.com/openmindonline/js-monday-02-the-formidable-spread-operator-f2d9177350ca>

Object Literals

- A set of key-value pairs that make up properties for an object (similar to dictionary in python)
- Key: value
- JSON (JavaScript Object Notation) is derived from Object Literals

```
let person = {  
  firstName: 'John',  
  lastName: 'Doe',  
  age: 70,  
  hobbies: ['hiking', 'drinking', 'science', 'inventing'],  
  address: {  
    street: '123 main st',  
    city: 'Blacksburg',  
    state: 'Virginia'  
  }  
};
```

JavaScript Object Literal

Object Literal Notation

```
// create a person object
var person = {};
person.firstName = "Joe";
person.lastName = "Jones";
person.address = {};
person.address.street = "123
    main";
person.address.zip = "12345";
person.address.state = "MO";
```

```
// same thing in object literal notation
var person = {
    firstName: "Joe",
    lastName: "Jones",
    address: {
        street: "123 main",
        zip: "12345",
        state: "MO"
    }
};
```

<https://www.slideshare.net/MetaThis/javascript-literacy>

JS Object Destructuring

```
const developer = {  
  name: "Mitch",  
  age: 24,  
  languages: {  
    favorite: "Haskell",  
    mostUsed: "JavaScript"  
  }  
};  
  
const { name, age, languages: { favorite, mostUsed } } = developer;  
  
const bio = `${name} is a ${age} years old developer.\n`  
  + `He codes in ${mostUsed} but prefers ${favorite}`;  
  
console.log(bio);  
  
// => "Mitch is a 24 old developer.  
//      He codes in JavaScript but prefers Haskell"
```

https://miro.medium.com/max/2720/1*mUcxSZsz3xwfKPrWR1yYEw.png

How to copy object properties?

→ Use Spread Operator!

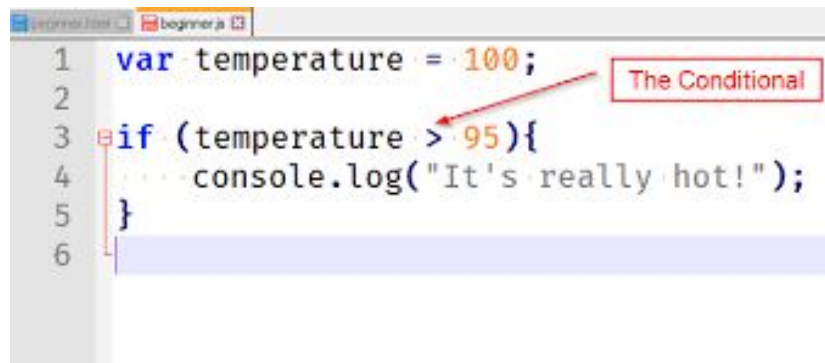
```
// How to merge arrays or object literal?  
const person1 = {  
  firstName: 'John',  
  lastName: 'Doe',  
  age: 27  
};  
  
const job = {jobTitle: 'developer', company: 'companyX'};  
  
const person1Merged = {...person1, ...job};  
/* person1Merged = {  
  firstName: 'John',  
  lastName: 'Doe',  
  age: 27,  
  jobTitle: 'developer',  
  company: 'companyX'  
}; */
```

Object Destructuring & assigning new name

```
let john = {  
  name: 'John',  
  age: 40  
}  
  
const employee = john;  
  
let { name: n, age: a } = employee;  
// n = employee.name  
// a = employee.age
```


Conditionals

- If-else if-else
- Used for decision branching
- Equals in comparison context
 - Always use triple equals (===)
 - Compares data types of the left hand and right hand value



The screenshot shows a code editor with two tabs: 'beginner.html' and 'beginner.js'. The 'beginner.js' tab is active, displaying the following JavaScript code:

```
1 var temperature = 100;  
2  
3 if (temperature > 95){  
4   console.log("It's really hot!");  
5 }  
6
```

A red arrow points from a box labeled "The Conditional" to the condition `temperature > 95` in line 3. The code block is highlighted with a light blue background.

Loops

- Loops are common structures among programming languages
- Main types
 - For
 - For each
 - While
 - Do while
- Allows for repetitive actions and array traversal

```
// Array traversal
for (let i = 0; i < todos.length; i++) {
  console.log(todos[i].text);
}

for (let todo of todos) {
  console.log(todo.id);
}

// forEach
todos.forEach(function(todo : {...}) {
  console.log(todo.text);
});
```

Array map

```
const users = [
  { firstName: 'John', lastName: 'Doe'},
  { firstName: 'Sarah', lastName: 'Smith'},
  { firstName: 'Sam', lastName: 'Williams'},
];

// Array map --> creates a new array by calling a function on each element in the input array.
const firstNames = users.map(function(user : {firstName: string, lastName: string} | ... ){ return user.firstName});

console.log({firstNames});
// firstNames = ['John', 'Sarah', 'Sam']
```

Functions

- Block of code that is essential for completing certain task
- Helps keep code clean and readable
- Makes code reusable

```
// 'function' syntax
function isEven(num) {
    return num % 2 === 0;
}

// modern ES6 arrow syntax
const isEven2 = (num) => {
    return num % 2 === 0;
};
```