

Intro to JavaScript #2

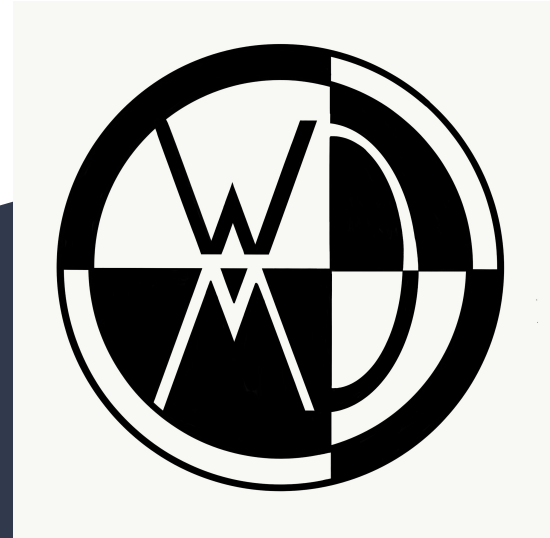
Data: 03/1/2020

Follow us on social media!

Website: webdvt.org

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


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



How to run JS?

1. Create a HTML file
2. 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>
```



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

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."
```

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.

Another example

```
$(document).ready(function () {  
  {  
    var myVar = "bla";  
    console.log(myVar);  
  }  
  console.log(myVar);  
  
  {  
    let myLet = "blub";  
    console.log(myLet);  
  }  
  console.log(myLet);  
  
  {  
    const myConst = "blib";  
    myConst = "blob";  
    console.log(myConst);  
  }  
  console.log(myConst);  
});
```

How should you declare variable in JS?

CONST vs LET vs VAR

ES6 Conventions:

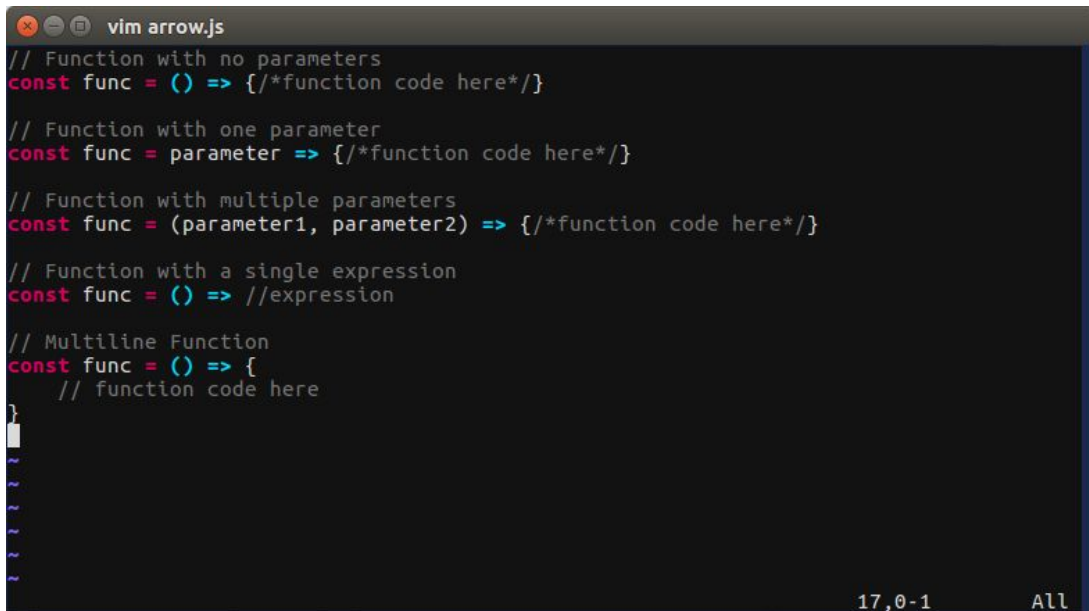
1. Use ``const`` by default.
2. Use ``let`` if you have to rebind a variable.
3. Use ``var`` to signal untouched legacy code

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

JS

How to write functions in JS?

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



```
vim arrow.js  
// Function with no parameters  
const func = () => { /*function code here*/ }  
  
// Function with one parameter  
const func = parameter => { /*function code here*/ }  
  
// Function with multiple parameters  
const func = (parameter1, parameter2) => { /*function code here*/ }  
  
// Function with a single expression  
const func = () => //expression  
  
// Multiline Function  
const func = () => {  
    // function code here  
}
```

17,0-1 All

https://medium.com/@luke_smaki/javascript-es6-arrow-functions-450985f27fdb

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

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
```

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'  
}; */
```

Spread Operator with Array



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

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

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']
```

JavaScript Challenge Solution

<https://github.com/webdvt/js-cheat-sheet/blob/master/challenge.js>

Let's test your
knowledge!

Join at **www.kahoot.it**
or with the **Kahoot!**
App

with Game PIN:

6803661