Javascript ES6:

ECMAScript 2015 was the second major revision to JavaScript.

## JavaScript let:

The let keyword allows you to declare a variable with block scope.

<!DOCTYPE html>

<html>

<body>

<h2>Redeclaring a Variable Using let</h2>

<p id=*"demo"*></p>

<script>

let x = 10;

// Here x is 10

{

let x = 2;

// Here x is 2

}

// Here x is 10

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

## JavaScript const

The const keyword allows you to declare a constant (a JavaScript variable with a constant value).

Constants are similar to let variables, except that the value cannot be changed.

<!DOCTYPE html>

<html>

<body>

<h2>Declaring a Variable Using const</h2>

<p id=*"demo"*></p>

<script>

**var** x = 10;

// Here x is 10

{

**const** x = 2;

// Here x is 2

}

// Here x is 10

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

## Arrow Functions

Arrow functions allows a short syntax for writing function expressions.

You don't need the function keyword, the return keyword, and the **curly brackets**.

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Arrow Functions</h2>

<p>With arrow functions, you don't have to type the function keyword, the return keyword, and the curly brackets.</p>

<p>Arrow functions are not supported in IE11 or earlier.</p>

<p id=*"demo"*></p>

<script>

**const** x = (x, y) => x \* y;

document.getElementById("demo").innerHTML = x(5, 5);

</script>

</body>

</html>

Arrow functions do not have their own this. They are not well suited for defining **object methods**.

Arrow functions are not hoisted. They must be defined **before** they are used.

Using const is safer than using var, because a function expression is always a constant value.

You can only omit the return keyword and the curly brackets if the function is a single statement. Because of this, it might be a good habit to always keep them:

## The For/Of Loop

The JavaScript for/of statement loops through the values of an iterable objects.

for/of lets you loop over data structures that are iterable such as Arrays, Strings, Maps, NodeLists, and more.

The for/of loop has the following syntax:

for (*variable* of *iterable*) {  
  // *code block to be executed*  
}

variable - For every iteration the value of the next property is assigned to the variable. Variable can be declared with const, let, or var.

iterable - An object that has iterable properties.

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript For Of Loop</h2>

<p>The for of statement loops through the values of any iterable object:</p>

<p id=*"demo"*></p>

<script>

**const** cars = ["BMW", "Volvo", "Mini"];

let text = "";

**for** (let x of cars) {

text += x + "<br>";

}

document.getElementById("demo").innerHTML = text;

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript For Of Loop</h2>

<p>The for of statement loops through the values of an iterable object.</p>

<p id=*"demo"*></p>

<script>

let language = "JavaScript";

let text = "";

**for** (let x of language) {

text += x + "<br>";

}

document.getElementById("demo").innerHTML = text;

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Map Objects</h2>

<p>Objects as keys in a Map:</p>

<p id=*"demo"*></p>

<script>

// Create Objects

**const** apples = {name: 'Apples'};

**const** bananas = {name: 'Bananas'};

**const** oranges = {name: 'Oranges'};

// Create a new Map

**const** fruits = **new** Map();

// Add the Objects to the Map

fruits.set(apples, 500);

fruits.set(bananas, 300);

fruits.set(oranges, 200);

// Display Object Type

document.getElementById("demo").innerHTML = fruits;

</script>

</body>

</html>

## JavaScript Set Objects:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Sets</h2>

<p>Add values to a Set:</p>

<p id=*"demo"*></p>

<script>

// Create a Set

**const** letters = **new** Set();

// Add Values to the Set

letters.add("a");

letters.add("b");

letters.add("c");

// Display set.size

document.getElementById("demo").innerHTML = letters.size;

</script>

</body>

</html>

## JavaScript Classes

JavaScript Classes are templates for JavaScript Objects.

Use the keyword class to create a class.

Always add a method named constructor():

### Syntax

class ClassName {  
  constructor() { ... }  
}

### Example

class Car {  
  constructor(name, year) {  
    this.name = name;  
    this.year = year;  
  }  
}

The example above creates a class named "Car".

The class has two initial properties: "name" and "year".

A JavaScript class is **not** an object.

It is a **template** for JavaScript objects.

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Class</h2>

<p>How to use a JavaScript Class.</p>

<p id=*"demo"*></p>

<script>

**class** Car {

constructor(name, year) {

**this**.name = name;

**this**.year = year;

}

}

**const** myCar = **new** Car("Ford", 2014);

document.getElementById("demo").innerHTML =

myCar.name + " " + myCar.year;

</script>

</body>

</html>