Intro to JavaScript

ISS Lab 2

<script> tag

In HTML, Javascript code is inserted between <script> and </script> tags.

Usually goes inside the <head> tag.

But can be placed inside <body> tag as well for faster execution.

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript in Body</h2>
<p id="demo"></p>
<script>
    document.getElementById("demo").innerHTML = "My First JavaScript";
</script>
</body>
</html>
```

JavaScript in Body

My First JavaScript

Attaching JS to HTML

- **External:**

```
<script src="myScript.js"></script>
```

- **Internal:**

```
<script>
  function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
```

- **Inline:**

```
<button type="button" onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!"'>Click Me!</button>
```

JS Display

- Writing into an HTML element:

  document.getElementById("demo").innerHTML = 5 + 6;

- Writing into the HTML output

  document.write(5 + 6);

- Writing into an alert box:

  alert(5 + 6);

- Writing into the browser console (will show in the terminal, used for debugging purposes)

  console.log(5 + 6);

JS Variables

let

```
if(i  == 1) {
   let x = 5;
   let y = 6;
   let x = 2;                   // doesn't work
}
document.write(x)        // doesn't work
```

Use let if the variables and the type can be allowed to changed later.

Only has block scope.

Variables must be declared before use.

Cannot be redeclared in the same scope.

var

```
if(car == 1) {
   var carName = "Volvo";
}
document.write(carName);    // works
```

Use let if you want to increase the scope of the variable to global scope.

const

```
const PI = 3.1415926;
PI = 3.14;     // This will give an error
PI = PI + 10;   // This will also give an error
```

Use const if the value should not be changed.

Use const if the type should not be changed (Arrays and Objects)

JS Comments

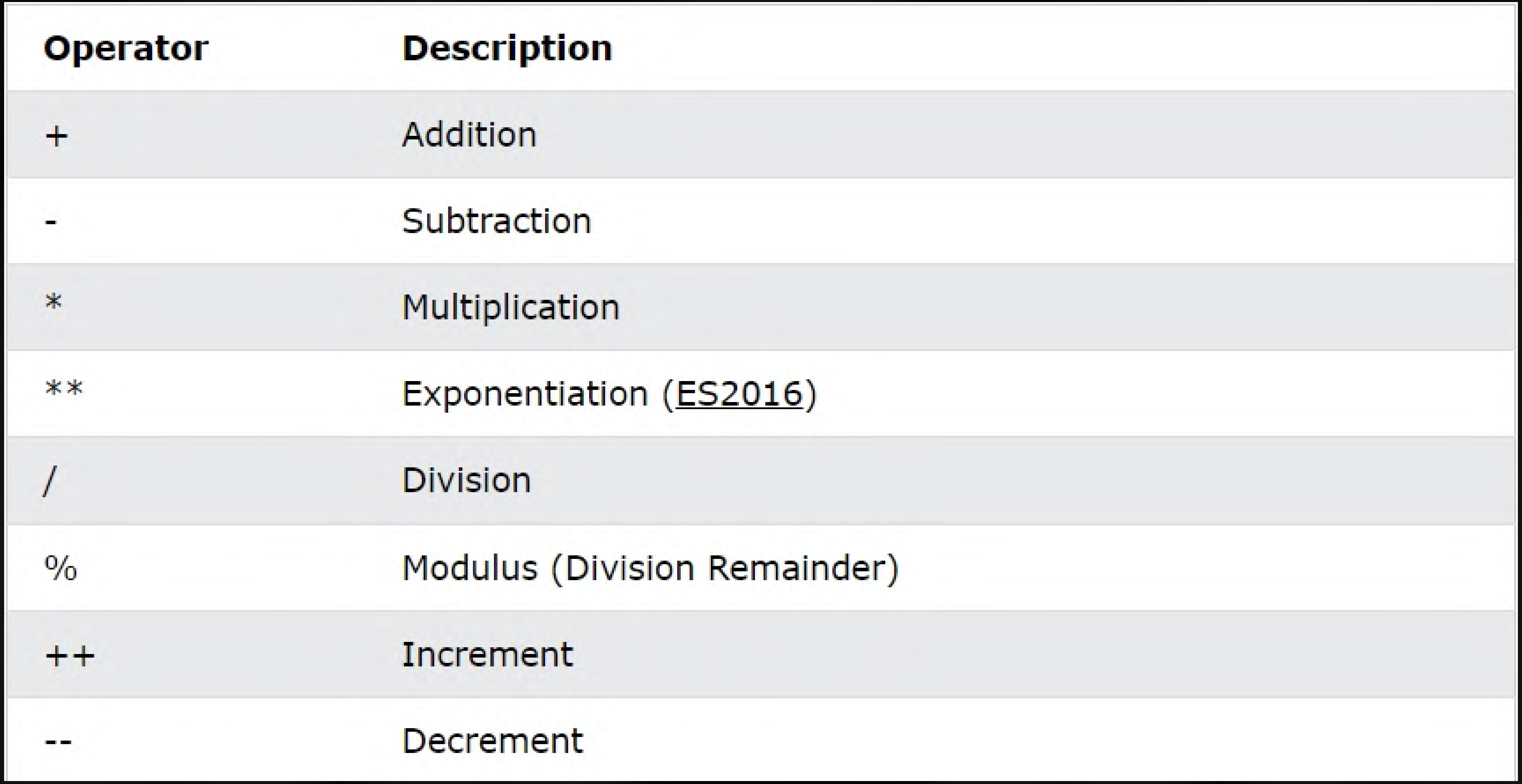
SIngle Line

```
// Change heading:
document.getElementById("myH").innerHTML = "Hello";
```

Multi-line

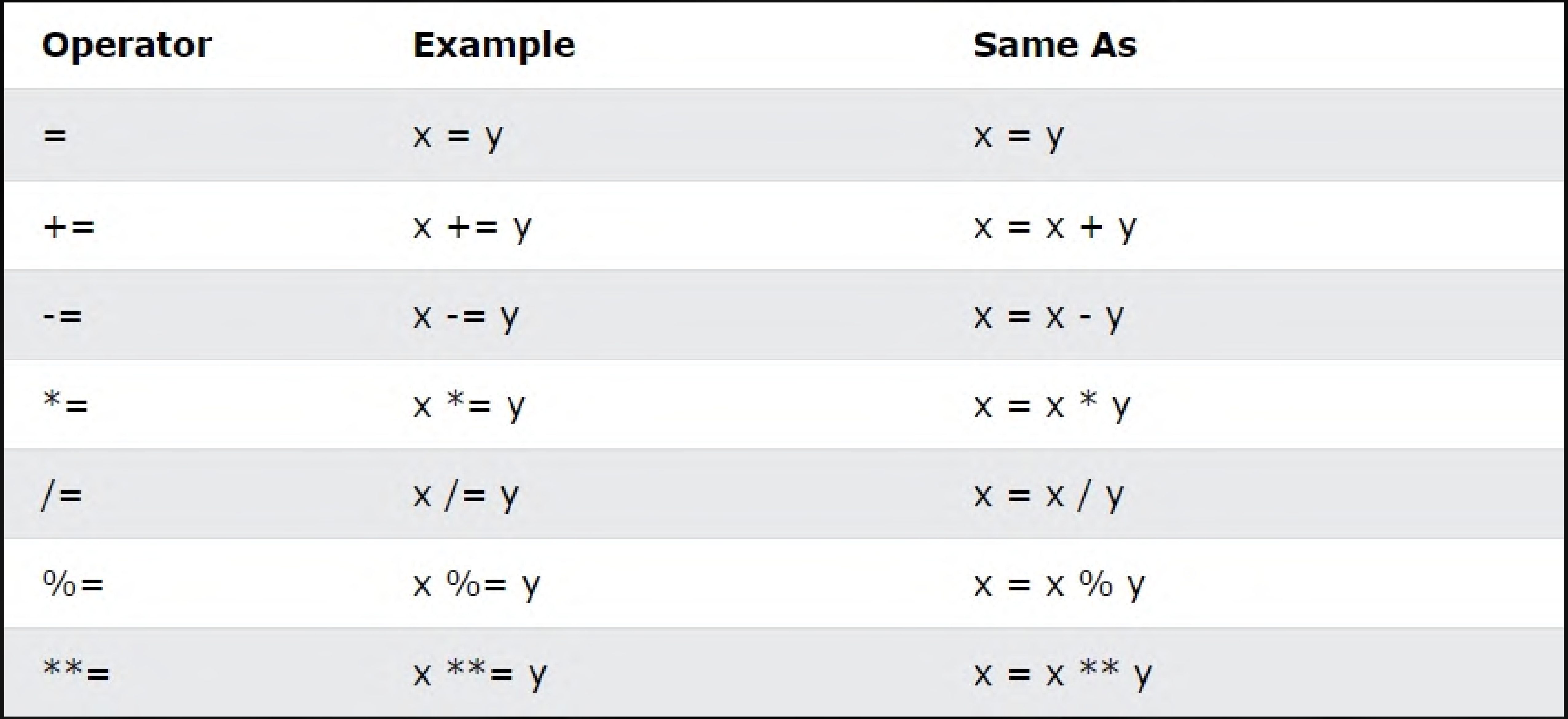
```
/*
The code below will change
my web page:
*/
document.getElementById("myH").innerHTML = "Hello";
```

JS Arithmetic Operators



Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

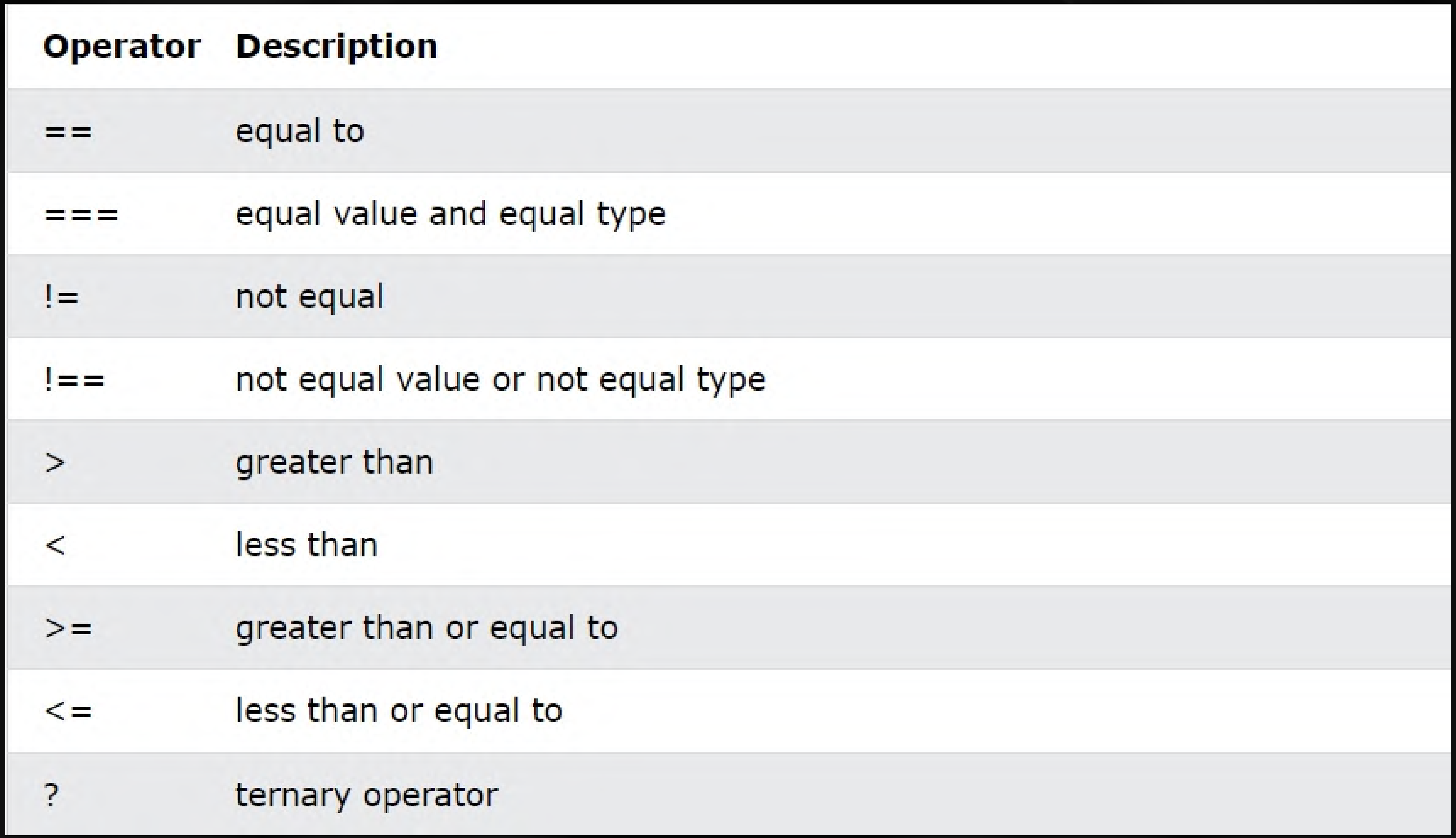
JS Assignment Operators



Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

Try Pitch

JS Comparison Operators



Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

JS Datatypes

```
// Numbers:
let length = 16;
let weight = 7.5;
// Strings:
let color = "Yellow";
let lastName = 'Johnson';
// Booleans
let x = true;
let y = false;
// Object:
const person = {firstName:"John", lastName:"Doe"};
// Array object:
const cars = ["Saab", "Volvo", "BMW"];
// Date object:
const date = new Date("2022-03-25");
```

JS Functions

A JavaScript function is defined with the function keyword, followed by a **name**, followed by parentheses ().
Function parameters are listed inside the parentheses () in the function definition.
Function arguments are the values received by the function when it is invoked.
Inside the function, the arguments (the parameters) behave as local variables.

```
// Function is called, the return value will end up in x
let x = myFunction(4, 3);

function myFunction(a, b) {
// Function returns the product of a and b
  return a * b;
}
```

JS Objects

Object	Properties	Methods
	car.name = Fiat	car.start()
	car.model = 500	car.drive()
	car.weight = 850kg	car.brake()
	car.color = white	car.stop()

In real life, a car is an object.

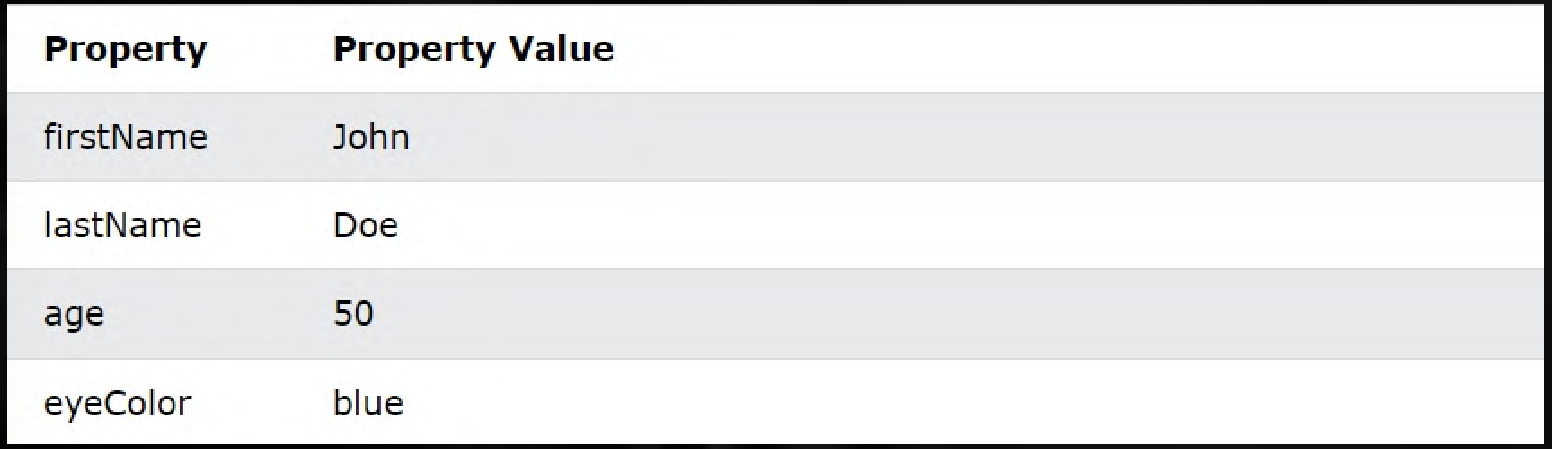
A car has properties like weight and color, and methods like start and stop.

All cars have the same properties, but the property values differ from car to car.

All cars have the same methods, but the methods are performed at different times.

JS Objects

```
const person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue"
};
```



Property	Property Value
firstName	John
lastName	Doe
age	50
eyeColor	blue

You can access properties in two ways:

1) objectName.propertyName

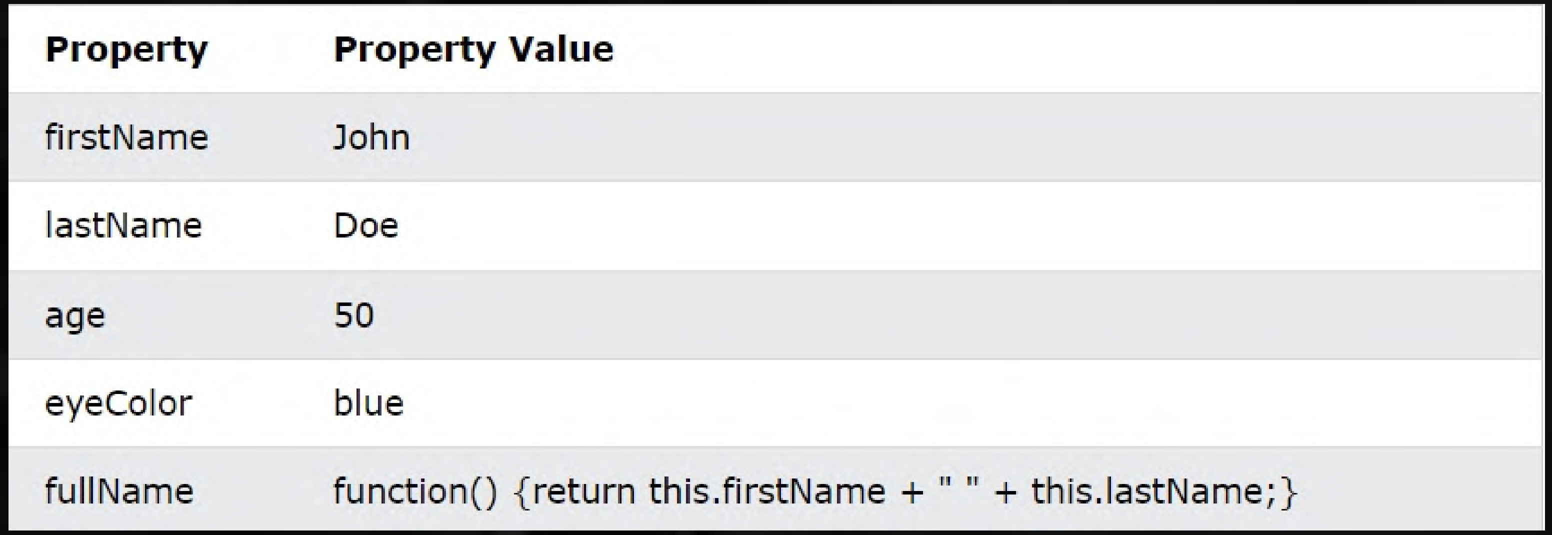
2) objectName["propertyName"]

JS Object Methods

Methods are actions that can be performed on objects.

Methods are stored in properties as function definitions.

```
const person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```



Property	Property Value
firstName	John
lastName	Doe
age	50
eyeColor	blue
fullName	function() {return this.firstName + " " + this.lastName;}

What is 'this' keyword above?

this refers to the object.

Here, this.firstName means the firstName proerpty of person.

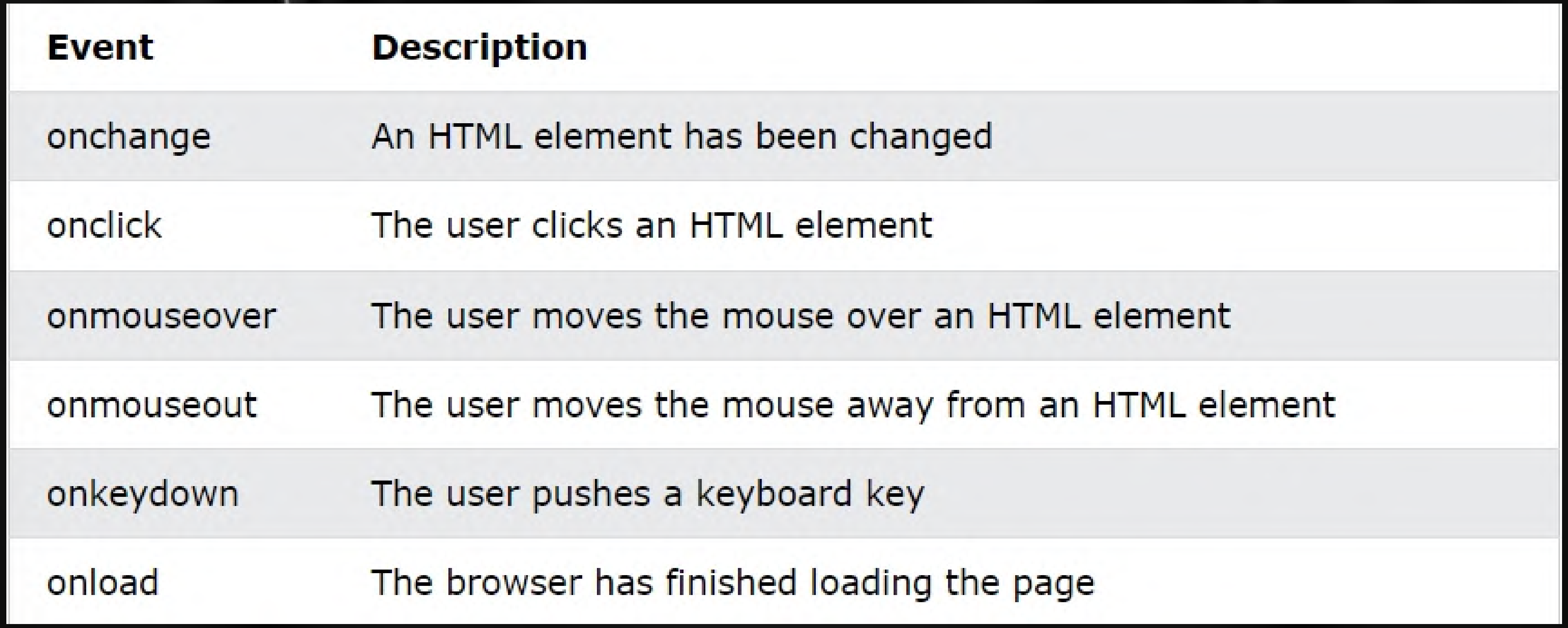
this.lastName means the lastName property of person.

JS Events

Examples of HTML events:

- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked

```
<button onclick="document.getElementById('demo').innerHTML = Date()">The time is?</button>
```



Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

JS Strings

let text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

- length: let length = text.length;
- charAt(position): text.charAt(0);
- Property access [] like in arrays: let letter = name[2];
- slice(start, end): let part = text.slice(-12, -6)
- substring(start, end): text.substring(7,13) // start and end values less than 0 are treated as 0 here
- substr(start, length): text.substr(7,6)
- to upper case: let text2 = text1.toUpperCase();
- to lower case: let text2 = text1.toLowerCase();
- concat():

```
let text1 = "Hello";
let text2 = "World";
let text3 = text1.concat(" ", text2);
```

JS Arrays

It is common practice to declare arrays with the const keyword.

Declaring arrays:

- const cars = ["Saab", "Volvo", "BMW"];
- const cars = [];
  cars[0]= "Saab";
  cars[1]= "Volvo";
  cars[2]= "BMW";
- const cars = new Array("Saab", "Volvo", "BMW");

Accessing array elements:

- let fruit = fruits[0];

Adding array elements:

- const fruits = ["Banana", "Orange", "Apple"];
  fruits.push("Lemon");

JS Switch

```
switch (new Date().getDay()) {          // date object
  case 6:
    text = "Today is Saturday";
    break;
  case 0:
    text = "Today is Sunday";
    break;
  default:
    text = "Looking forward to the Weekend";
}
```

JS Iterables

```
const letters = ["a","b","c"];

for (const x of letters) {
  // code block to be executed
}
```

Iterables are iterable objects like Arrays.

Can be accessed with simple and efficient code.

Can be iterated over with for..of loops.

JS Sets

A set is a collection of unique values.

Each value can only occur once in a set.

```
// Create a Set
const letters = new Set();

// Add Values to the Set
letters.add("a");
letters.add("b");
letters.add("c");
letters.add("a");          // only the first "a" will be saved
```

JS Maps

A map holds key-value pairs where the keys can be any datatype.

A map remembers the original insertion order of the keys.

```
// Create a Map
const fruits = new Map([
  ["apples", 500],
  ["bananas", 300],
  ["oranges", 200]
]);

// Set Map Values
fruits.set("apples", 200);   // can change existing key value or make a new key

fruits.get("apples");   // Returns 200
fruits.delete("apples");
```

JS try-catch-finally

Syntax:

```
try {
    // Block of code to try
}
catch(err) {
    // Block of code to handle errors
}
finally {
    // Block of code to be executed regardless of the try / catch result
}
```

JS Classes

```
// creating car class and constructor
class Car {
  constructor(name, year) {
    this.name = name;
    this.year = year;
  }
  age(x) {
    return x - this.year;
  }
}

// creating car objects
const myCar1 = new Car("Ford", 2014);
const myCar2 = new Car("Audi", 2019);
```

Lab Assignment

- Lab Assignment 2 is uploaded on Moodle.
- Deadline is Saturday at 5:30pm.
- There is a bonus section and will make up for any marks you lose in this lab assignment only.
- Anything not mentioned in the assignment document can be assumed.
- Submission format:
  - Join GitHub classroom.
  - https://classroom.github.com/a/zBKeFQYU
  - Select your Roll Number to join the classroom.
  - You will automatically be added into your repo.
  - Push your code here before the deadline.
- The tutorial session this week will be a doubt clearing session where you can work on your lab assignment and ask the TAs if you have any doubts.
- Hint for the lab assignment: Use Date() object and its get methods.
- DO NOT use ChatGPT.

Pitch

Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It's free)