Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

JavaScript

- It is lightweigh (just in time compiled) programming language.

Just in time- Compilation of code during executing of program i.e. at runtime rather than before execution.

→ How to link JS code in HTML?

(a) Internal JS -

We can add JS directly to our HTML file by writing code inside the <script> tag.

<script> tag can either be placed inside the head or body tag according to the requirement.

(b) External JS:

We can write JS code in another file having an extension .js & then link this file inside head tag of HTML file in which we want to add this code

→ JS is a (weakly typed) language.

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

=> Why JS in known as lightweight programming language?

It is considered as lightweight due to the fact that it has low CPU usage, easy to implement and has minimalist syntax.

Minimalist Syntax means, has no data types.

- Everything here is treated as an object.

=> JavaScript is Compiled or Interpreted language?

Answer is Both, previously it was ~~compiled~~ Intepreter language but after the V8, the JIT compiler was also incorporated to optimize the execution & display the result more quickly.

→ Template Literals : When we declare strings using backticks (`), this are called template literals, it has special properties, we can embed ${} variables or even expression in them.

```
const greeting = `Hello ${name}`;
```

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

→ Useful Number Methods -

(a) toFixed() - Round no. to a fixed no. of decimal places.

E.g.:

```
const numValue = 1.7654321
~~numValue~~
const updatedValue = numValue.toFixed(2)
console.log(updateValue);
```

O/P : 1.76

(b) Converting to Number data types -

Number()

E.g.

```
const textValue = "53";
const noValue = Number(textValue)
console.log(typeof noValue);

// O/P: Number
```

→ String() : It converts anything into string.

String()

E.g.

```
const value = 53
const strValue = String(value);
console.log(typeof strValue);
```

O/P : String

Date :.....

MON TUE WED THU FRI SAT SUN

→ Useful string methods -

(a) length of the string: length property.

e.g: const value = "abc"

console.log(value.length);

// O/P: 3

(b) Retreiving specific string characters: We assume string as an array, so we can retrieve character from string using [] square bracket.

e.g.: const value = "Hello";

console.log(value[0]);

// O.P: H

(c) Check if strings contain substring: Includes() method.

e.g. const value = "Hello world";

```
if (value.includes("World")) {
    console.log("Found World");
}
else {
    console.log("Not Found");
}
```

Includes() returns true if string contains substring, else false otherwise.

Date :.....

MON TUE WED THU FRI SAT SUN

(d) startsWith() & endsWith();

If you want to know if a string starts or ends with particular substring. Returns true or false.

E.g.
```
const value = "mozilla";
if (value.startsWith("moz")) {
    console.log("Moz");
}
else {
    console.log("No Moz");
}
```

(e) Find position of substring in a string: indexOf()

It takes two parameter - (1) substring that we want to search
(2) Optional parameter that specifies starting point of search.

It returns index of the first occurence of the substring. If string doesn't contain the substring, indexOf() returns -1.

E.g.
```
const tagline = "MDN - Resources for developers, by developers";
console.log(tagline.indexOf("developers"))
```

O/P: 20

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

(f) Extracting substring from string: slice() method.
It takes two parameters:
(i) Index at which start extracting.
(ii) Index at which to stop extracting, this index is exclusive.

```
const type = "mozilla";
console.log(type.slice(2, 7));
```

O/P: "zilla"

We can also do, if we want to extract all the remaining characters after a certain index, then don't include second parameter.

```
const type = "mozilla";
console.log(type.slice(2));
```

O/P: "zilla"

(g) Changing Case:
(i) toLowerCase()
(ii) toUpperCase()

```
const data = "Mozilla";

console.log(data.toLowerCase()); // "mozilla"
console.log(data.toUpperCase()); // "MOZILLA"
```

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

(b) Updating parts of a string:

We can replace one substring inside a string with another substring using the replace().

It takes two parameters, (i) string we want to replace
(ii) string we want to replace it with.

E.g. const browerType = "mozilla";
const updated = browerType.replace("moz", "van");
console.log(updated); // ~~vanilla~~ vanilla

- replace() in this form only changes the first occurences of the substring. If we want to change all occurences, we can use replaceAll().

E.g.: let quote = "To be or not to be";
quote = quote.replaceAll("be", "code");
console.log(quote); // To code or not to code

MON TUE WED THU FRI SAT SUN

Arrays:

Arrays are "list-like objects", they are basically single object that contain multiple values stored in a list.

→ Find length of array:

```
const shopping = ["bread", "milk", "cheese"];
console.log(shopping.length);
```

→ Finding the index of items in an array:

```
const birds = ["Parrot", "Falcon", "Owl"];
console.log(birds.indexOf("Falcon")); // 1
console.log(birds.indexOf("Owl")); // 2
console.log(birds.indexOf("Rabbit")); // -1
```

→ Adding Items:

```
const cities = ["Manchester"];
cities.push("Cardiff");
console.log(cities); // ["Manchester", "Cardiff"];
cities.push("Bradford", "Brighon");
console.log(cities); // ["Manchester", "Cardiff", "Bradf
                     "Brighon"].
```

- If we want to add item at start, use unshift()

```
cities.unshift("Cardiff");
```

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

- push() returns new length of an array.

```
console.log(cities.push("Armsterdam")); // 5
```

→ Removing Items:

- To removing the last item from the array, use pop()

```
const cities = ["Manchester", "Cardiff"];
console.log("Removing items is", cities.pop());

// Removing item is Cardiff
```

pop() return the item that was removed.

- To remove the first item from an array, use shift()

```
const cities = ["Manchester", "Cardiff"];
console.log("Removing item is", cities.shift());

// Removing item is Manchester
```

- To remove an item of specific index, use splice()

```
const cities = ["Manchester", "Cardiff", "Liverpool"];
cities.splice(cities.indexOf("Cardiff"), 1);

console.log(cities);
// ["Manchester", "Liverpool"]
```

splice() takes two arguments, first says where to start removing items. & second says, how many items should we removed.

Date :.....

MON TUE WED THU FRI SAT SUN

☐ ☐ ☐ ☐ ☐ ☐ ☐

Accessing every item

(1) for ... of :

```
for (const city of cities) {
    console.log (city);
}
```

(2) map (): If you want to some operation of array element.

```
const num = [2, 3, 5, 7];
const double = num.map (value => {
    return value * 2;
});
console.log (double); // [4, 6, 10, 14]
```

Converting between strings & array:

- Use the split() method to split the string into an array. It takes single parameter, the character you want to seperate the string at, & returns the substrings between the seperator as items in an array.

```
const cities = "Manchester, London, Liverpool";
const arr = cities.split (", ");
console.log (arr);
```

Date :.....

MON TUE WED THU FRI SAT SUN

- Now, if you want to convert array into string, use join()

```
const arr = ["Manchester", "London", "Liverpool"];
const cities = arr.join(", ");
console.log(cities);
```

- Another way of converting array to string is toString(). its not take any parameter, & it's limiting than join() because it always use comma.

```
const arr = ["Manchester", "London", "Liverpool"];
const cities = arr.toString();
console.log(cities); // Manchester, London, Liverpool
```

Date :.....

MON TUE WED THU FRI SAT SUN

☐ ☐ ☐ ☐ ☐ ☐ ☐

⇒ ~~Event Propegation~~

⇒ Event Bubbling :

It is a concept in the DOM. It happens when an element receives an event, & that event bubbles up (or you can ~~stay~~ say is transmitted or propogated) to its parent & ancestor elements in the DOM tree until it gets to root element.

This is default behaviour of events on elements unless you stop the propogation.

- How to stop event bubbling?

event.stopPropogation()

Propogation is an act of spreading something. The stopPropogation method is used to prevent the spreading of events when an event is triggred on an element.

* event.preventDefault() :

This method prevents default action that browsers make when an event is triggered.

Date :.....

MON TUE WED THU FRI SAT SUN

→ target & currentTarget property in Event:

There are two different properties of event object to access the element that was clicked.

event.target : target refers to the element on which the event was initially fired, ~~while~~

event.currentTarget : refers to the element to which this event handler has been attached

- target remains same while event bubbles, currentTarget will be different for event handlers that are attached to different elements in hierarchy.

Date :.....

MON TUE WED THU FRI SAT SUN

☐ ☐ ☐ ☐ ☐ ☐ ☐

⇒ Object prototypes : (Inheritance in JS)

Every object in JavaScript has built-in property, which is called its prototype.

The prototype is itself an object, making what's called a prototype chain.

The chain ends when we reach a prototype that has null for its own prototype.

The standard way to access's an object prototype is the Object.getPrototypeOf() method.

→ When you try to access a property of an object & if the property can't be found in the object itself, the prototype is searched for that prototype. If the prototype still can't be found, then the prototype's prototype is searched & so on until either the property is found or the end of chain is reach, in which case undefined is returned.

E.g.
```
const obj = {
    value: "abc",
    value1 : "xyz"
};
console.log(Object.getPrototypeOf(obj));
// O/P: Object {}
```

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

Prototype of Date Object:

```
const date = new Date();
let object = date;

do {
    object = Object.getPrototypeOf(date);
    console.log(object);
} while (object);
```

O/P: Date.prototype
Object { }
null

=> Setting a prototype:

Two ways to ~~describe~~ set object prototype are Object.create() & constructors.

(a) Object.create():

It helps to creates a new object & allows you to specify an object that will be used as the new object's prototype.

```
E.g. const personPrototype = {
        greet() {
            console.log("Hello");
        }
    };

    const carl = Object.create(personProto);
```

Date :.....

MON TUE WED THU FRI SAT SUN

(b) Using constructor :

In JS, all functions have a property named "prototype". When you call function as a constructor, this property is set as the prototype of newly constructed object.

E.g.

```
const personProto = {
    greet() {
        console.log("Hello");
    }
}

function Person(name) {
    this.name = name;
}

Object.assign(Person.prototype, personProto);
// or
Person.prototype.greet = personProto.greet;

const p = new Person();
p.greet();  // Hello.
```

Date :.....

MON TUE WED THU FRI SAT SUN

=> HasOwn :

- Properties that are defined directly in the object, like name in Person are called own properties.
- We can check that property is ownproperty using Object.hasOwn() method.

```
console.log (Object.hasOwn (p, "name")) ; //true
```

Date :.....

MON TUE WED THU FRI SAT SUN

Classes & Constructors:

We can create class using "class" keyword.

E.g.

```
class Person {
    name;

    constructor (name) {
        this.name = name;
    }

    introduce () {
        console.log (`Hi, My name is ${this.nam
    }
}

const mike = new Person ("Mike");
mike.introduce();
// Hi, My name is Mike.
```

- We can class without constructor as well.

Inheritance: In class, we can achieve inheritance using 'extends' keyword.

```
class Student extends Person {
    year;

    constructor (name, year) {
        this. super (name);
        this.year = year;
    }
}
```

Date :.....

MON TUE WED THU FRI SAT SUN

☐ ☐ ☐ ☐ ☐ ☐ ☐

Encapsulation:

If we want to make any class property 'private' then we can use '#' as prefix at a variable declaration time.

e.g: class Student {

```
    #year ;          // private variable

    Introduce (){
        console.log (`I'm in ${this.#year} year`);
    }
}

const std = new Student ();
std.#year ;   // Syntax Error.
```

- In class method, we can access the private variable like this.#year
- # we need to use every time.

Private Methods: Like class property, if we need to define class method as private.

```
# introduce (){
    //....
}
```

Date :.....

MON TUE WED THU FRI SAT SUN

☐ ☐ ☐ ☐ ☐ ☐ ☐

Working with JSON :

JSON is a text based data format following JS object syntax.

JSON exists as string - useful when you transmit data from across a network.

Methods :

(1) To convert object / Promise into JSON :

.json()

E.g. response.json()

(2) To convert JSON string into JSON object :

```
const vari = "{ ... }"; // JSON in string
const jsonObj = JSON.parse(vari);
```

(3) To convert JSON obj into JSON string :

```
const vari = { ... }
const jsonStr = JSON.stringify(vari);
```