(Udemy)

# DOM Javascript

## way to insert javascript

1) inline javascript → inside this
<body onload = "alert ('Hello');" >
(not a good practice to use)

2) <script type = "text/javascript"> (internal javascript)
alert ("Hello");
</script>

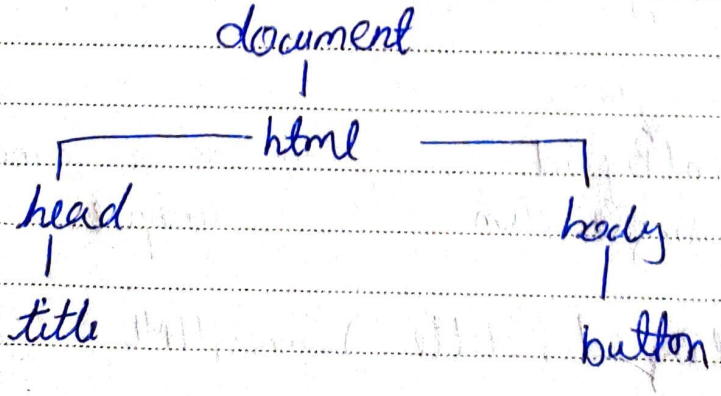
3) <script src = "index.js" > </script> (external javascript)

गुरुवार 14

Note

put javascript at end (before </body>)
as

- like you made change in h1 & h1 comes below javascript tag so it will throw error.
- also run faster



- img → src
- all others → href

## Get Elements

1) `document.getElementsByTagName("li");`
→ return an array of all li

to change color of 2nd li (2nd element)
`document.getElementsByTagName("li")[1].style.color = "red";`

to get length of array
`document.getElementsByTagName("li").length;`

2) `document.getElementsByClassName`
→ returns array

16 शनिवार

Note
can't do
`document.getElementsByClassName("wow").style.color = "red";`
↑ as returns array so error
↓ do
`document.getElementsByClassName("wow")[0].style.color = "red";`

3) `document.getElementById` (no s) → return single item → no s since all Id's are unique

`document.getElementById("title").innerHTML = "Good Bye";`

4) document.querySelector('h1');

or

document.querySelector("#title");

or

document.querySelector(".wow");

↳ returns array so [ ] use index

→ Manipulating things & properties

) font-style changes to fontStyle (no -) → camel case

so document.getElementById("title").style.fontStyle = "Italic"

attribute change

2) document.querySelector("a").setAttribute("href", "https://www.bing.com");

↑

सोमवार

New part

Adding Event

document.getElementByClassName("drum")[1].addEventListener("click", getClicked);

function getClicked() {

alert("I got clicked");

}

* to add a emoji → ⌘ to gogle emoji

---

selecting all drum classes

```
var numberOfDrumButtons = document.querySelectorAll(".drum").length;

for (var i=0; i<numberOfDrumButtons; i++) {
    document.querySelectorAll(".drum")[i].addEventListener("click", function() {
        alert("I got clicked!");
    });
}
```

anonymous function

## combining functions

```
function add(a,b) { return a+b; }
function mul(a,b) { return a*b; }

function calculator(a, b, operator) {
    return operator(a, b);
}
```

to call `calculator(8, 3, add);`

## this

```
console.log(this);
```
→ returns which one is selected

to change selected one

```
this.style.color = "white";
```

---

20 बुधवार

## Constructor function

used in place of object → `var bellBoy2 = { name : "Tom" , age : 19 , lang : ["french", "English"] }` → object so

Constructor fn

```
function BellBoy (name, age, lang) {
    this.name = name;
    this.age = age;
    this.lang = lang; → fn so;
}
```

to initialise new object

```
var bellboy1 = new BellBoy("Tom", 19, ["french", "English"]);
```

शुक्रवार 22

## Switch

```
var selected = this.innerHTML;
switch (selected) {
    case "w":
        var audio = new Audio ("tom.mp3");
        audio.play();
        break;
    case "a":

    default:
}
```

## Dot Notation

bellBoy1.moveSuitcase();

to make fn. inside constructor

```
function Audio(fileLocation) {
    this.play = function () {
        ---
        - code
    }
}
```

## Keypress event

24 रविवार

to add Event to a whole document

to

```
document.addEventListener("keypress", function(event) {
    console.log(event);
});
```

to get which key is selected

```
document.addEventListener("keypress", function(event) {
    makeSound(event.key);
});
```

↑ made a function

# Higher Order FUNCTION

functions which are able to take functions as input,

eg

```
document.addEventListener("keypress", respond(event));
```

↑ addEventListener

↑ respond(event) — & this function is a call back function

```
function respond(event) {
    console.log(event);
}
```

मंगलवार 20

# FUNCTIONAL Programming

free code camp

⊙ The functional programming software development approach breaks a program into small, testable part.

Input → Process → output

## Types of functions :-

1) functions that can be assigned to a variable, passed into another function, or returned from another function just like any other normal value, are called First class functions. In Javascript, all functions are called first class functions.

शनिवार

2) Functions that take a function as an argument, or return a function as a return value are called higher order functions.

3) When the functions are passed into another function or returned from another function, then those functions which gets passed in or returned can be called a lambda.

## ⊙ Avoid mutation And Side Effects

→ One of the core principles of functional programming is to not change things. changes lead to bugs

→ function should not change arguments, global variables etc.

→ In functional programming, changing or altering things is called mutation, and the outcome is called side-effect. A function should ideally a pure function, meaning it does not have any side effect.

eg
```
var fixedValue = 4;
function incrementer () {
    return fixedValue + 1;
}
```
✓

4 सोमवार

not
```
function incrementer () {
    fixedValue += 1;
    return fixedValue;
}
```
✗

eg2
```
function add (x, bookname) {
    x.push(bookname);
    return x;
}
```
x = list of books

✗ x is changed

do
```
function add (x, bookName) {
    var temp = [...x];
    temp.push(bookName);
    return temp;
}
```
✓

; ✗ unchange

to copy array

# map function

map method iterates over each item in an array and returns a new array containing result of calling the callback function on each element. It does this without mutating original array.

```
const users = [
    { name : 'John', age : 34 },
    { name : 'Amy', age : 20 },
    { name : 'Cat', age : 10 },
];
const names = users.map(user => user.name);
```

callback function

names = ['John', 'Amy', 'cat']

बुधवार

both name & age (more than one)

```
const info = users.map( item => ({
    Username : item['name'],
    Userage : item['age'] }));
```

info = [ {Username : John, Userage : 34}, { }, { } ]

## ⊙ implementing map function

map is a pure function, and it's output depends solely on its input. Plus it takes another function as its argument.

§ given s = [23, 65, 5]
want new_s = [46, 130, 10]

Sol

```
var s = [23, 65, 5]

Array.prototype.myMap = function (callback) {
  var newArray = [];
  this.forEach (a => newArray.push (callback (a)))
  return newArray;
};

var new_s = s.myMap( function (item ) {
  return item * 2;
});
```

→ call back function

8 शुक्रवार

or in place of for each can do

```
for ( let i = 0 , i < this.length ; i++) {
  newArray.push (callback [this [i]]);
}
```

this keyword gives us access to object we are calling myMap on.

Arrow function → () ⇒ return ("Hello World").

(a,b) ⇒ return a*b;

---

- ## filter function

useful Array function

Array.prototype.filter() or simply filter()

filter calls function on each element of an array and returns a new array containing only elements for which function returns true.

eg

```
const users = [
    { name: 'John', age: 30 },
    { name: 'Amy', age: 20 },
];
const Underage30 = user.filter(u => u.age < 30);
```

ns Underage30 = [ { name: 'Amy', age: 20 } ].

रविवार 10

- ## reduce function

```
const user = [
    { name: 'John', age: 30 },
    { name: 'Amy', age: 20 },
];
const ans = user.reduce((obj, users => {
    obj [users.name] = users.age;
    return obj;
}, { });
```

= { John : 30 , Amy : 20 }

## ⊙ every method

it works with array to check if every element pass
a particular test it returns true and false.

eg var num = [1, 5, 3, 0];
var x = num . every ( function (x) {
return x < 10;
});

→ true

x = true

## ⊙ some method

it works with array to check if
even any one element passes a particular test.

12 मंगलवार

## ⊙ Currying

The arity of a function is no. of arguments it requ
Currying a function means to convert function of
N arity into N functions of arity 1.

eg function unCurry (x, y) {
return x + y;
}

→

function curried (x) {
return function (y) {
return x + y;
}
}

called as curried (1) (2) → 3

# Regular Expression

Regular expressions are special strings that represent a search pattern. Also known as regex or regexp, they help programmers match, search & replace text.

1) If you want to find word "the" in string 'The dog chased the cat"
use regular expression /the/
one way to test regex is using .test() method.

eg Let testStr = "free CodeCamp";
let testRegex = /Code/;
testRegex.test(testStr);
↘ returns true.

but with /code/ → false

गुरुवार 14

2) using for more than one check
let string = "James has a pet cat".
let petRegex = /dog | cat | bird | fish /;
let result = petRegex.test(string);

3) ignore flag → i (ignore upper & lower case)
let petRegex = /cat/i
so true for cat, CAT, CAT etc.

## 4) Extract Matches

to extract actual match you find with .match

eg "Hello World!".match(/Hello/);
return ["Hello"]

Note .match has opposite syntax of .test
'string'.match(/regex/)
/regex/.test('string')

if regex present more than once & you want to extract
so /regex/g ← global flag.

16 शनिवार

- to use more than one flag
let startRegex = /twinkle/ig;

## 5) Wild card Period (.)

/hu./ → hug, huh, hut, hum

eg
x = /hu./.test("I will hug you")
x = true

/.un/ → run, fun, sun, bun

6) /b[aiu]g/ → "bag", "bug", "big"

/[a-e]at/ → "aat", "bat", "cat", "dat", "eat"

/[a-z0-9]/ig; → all number and alphabets.
↑ no space

eg

```
let quote = "What a game";
let myRegex = /a-z/ig;
let result = quote.match(myRegex);
```

result = ["W", "h", "a", "t", a, g, a, m, e]

---

सोमवार 18

7) not including a character → ^

eg

not include vowels & number

/[^aeiou0-9]/

8) more than one match (+)

/a+/g in "abc" → ["a"]

/a+/g in "aabc" → ["aa"]

/a+/g in "abab" → ["a", "a"]

9) + → one or more than ones
* → zero or more times

eg let soccer = "goooooaal";
let gPhrase = "gut feeling";
let oPhrase = "over the moon";
let goRegex = /go*/;

soccer.match(goRegex); // Return ["goooooaal"]
gPhrase.match(goRegex); // Return ["g"]
oPhrase.match(goRegex); // Return NULL

10) Lazy matching

20 बुधवार

/t[a-z]*i/ in titanic
↑ → returns ["titani"]
start with t end with i

if ? (Lazy matching)
/t[a-z]*?i/ in titanic
returns ["ti"]
? stops at first ti

11) only at Beginning (^)
Myregex = /^is/
gives true with "is this you"
false with "this is mine"
ue if is at start.

12) only at Ending ($)

/Story $/

true with " here is end of Story "

false with " new Story ends ".

13) ⊙ \w = [A-z a-z 0-9 _]

↑ underscore

So

Myregex = /\w+/; select all this

⊙ \W = [^A-z a-z 0-9 _]

match all except alpha numeric & underscore

⊙ \d = [0-9]

शुक्रवार 22

⊙ \D = [^0-9] not (all numbers)

14) white space

⊙ \s (matches all whitespace)

≡ [\r \t \f \n \v]

eg let s = " Whitspace . with ! "

let x = /\s/g;

s. match(x);

→ returns [" ", " "]

⊙ \S → non - white Space

15> upper & lower number of matches

{ }

eg /a{3,5}h/ → match only letter a appearing between 3 and 5 times

with aaah → true
aah → false

want true for
ohhhh no
ohhhhhh no ] 3 to 6

MyRegex = /oh{3,6}\s no/

24 रविवार

forspace

just all more than 3
/oh{3,}\s no/

Exacts 3 matches
/oh{3}\s no/

16> check for All or None

/colou?r/ → true for color
& true for colour

17) mixed grouping

/P(engu|umpK) in /g → Penguin
Pumpkin

anything in between .* → many times
↑ anything

eg /A.*r/ → Apple is r
Apr
Ar

18) \1 → captures first character of string

eg

find string that starts and ends with same character

मंगलवार 26

Const regex = /(.).*\1/;

(.) → captures any character
.* → any characters zero or more occurence
\1 → back reference to first capture group here = (.)

19 to include regex Mr. but dot (.) is operator
so use
/Mr\./

# Extra about javascript

⊙ to call a function again & again in let 1s.
→ setInterval ("showTime", 1000);

⊙ Template Strings (ES6)
for string interpolation

→ `string text` , `string text 1
string text 2`
`string text ${expression}`

1. '\'' === "'" True

→ before console.log ('fifteen is ' + (a+b) + ' this is ans (normal)
now console.log (`fifteen is ${a+b} this is ans`)

28 गुरुवार

using → function body_onload()
{
if (${count==0})
{
document.getElementsByClassName ("display");
}
}

or as
n.style.transform = `rotate (${temp}px} deg)`;

# Events

format 1

<element event = "some javascript" >

eg

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

or can do

<button onclick = "this.innerHTML = Date()" > The time is? </button>

Some events

onchange, onclick, onmouseover, onmouseout
onkeydown, onload.

शनिवार 30

function 2

<h1 onclick = "changeText(This)" > click me </h1>

```
<script>
function changeText(id){
    id.innerHTML = "oops!";
}
</script>
```

## 3) Events using DOM

```
<script>
document.getElementById("myBtn").onclick = displayDate;
</script>
```