31/07

JAVA SCRIPT

- High level, interpreted programming language.
- Confirms to ECMAScript Specification.
- Multi-paradigm [many ways]
- Runs on client/browser as well as on the server (Node.js)

Q.] Why learn JAVAScript ?

- It is programming language of the browser.
- Build very interactive user interfaces with frameworks like REACT.
- Used in building very fast server side and full stack applications
- Used in Mobile development [React, Native, Ionic, ReactScript]

- Used in desktop development [Electron JS]

* TOPICS :-

- Variables and Datatypes
- Arrays
- Object literals
- Methods for strings, arrays, objects
- loops
- Conditionals
- Functions
- OOP
- Dom Selection
- Dom Manipulation
- Events
- Basic form Validation.

- Include Javascript into HTML file :-

<script src = "main.js" > </script >

* Methods :-

1.> log :-

console.log('Hello World');

2.> error :-

console.error('This is an error');

3.> Warn :-

console.warn('This is a warning')

* How to declare a variable :-

- 3 ways :-
  - var - globally scoped
  - let
  - const → Block level Scope (can reassign values)

eg. let age = 30;
age = 31;
console.log(age);

Output : 31

eg. const age : 30;
age = 31;
console.log(age);

Output: ERROR.

Note :- Use const unless you know you ~~are~~ ~~go~~ will reassign it will make the code robust.

eg. const score = 10.
console.log (score);

Output : 10

* Datatypes :-

String, Numbers, Boolean, null, undefined, Symbols.

```
const name = 'John';
const age = 30;
const isCool = true;
const rating = 4.5; → decimal/float
const x = null;
const y = undefined;
let z; [Here z is also undefined]

console.log(type of name);
```

⇒ Strings :-

1.] Concatenation :-

old way

```
const name = 'John';
const age = 30;

console.log('My name is ' + name +
            ' and I am ' + age);
```

⇒ Template String :- [Better way]

```
- const name = 'Yug';
  const age = 30;

  console.log(`My name is ${name}
              and I am ${age}`);
```

Back-ticks

```
- const name = 'Yug';
  const age = 30;

  const hello = `My name is ${name}
                and I am ${age}`;

  console.log(hello);
```

OUTPUT: My name is Yug and I am 30

Output is same for both

⇒ String properties & methods :-

(i) length property :-

```
const s = 'Hello World';
console.log(s.length);
```

Output: 11

(ii) Uppercase method :-

```
const s = 'Hello World';
console.log(s.toUpperCase());
```

Output: HELLO WORLD

Note :- If there is method there will always be paranthesis after them unlike property.

(iii) SubString :-

```
const s = 'Hello World';
console.log( s.substring(0, 6) );
```

Starting index ↑ ending index

OUTPUT: Hello W

```
const s = 'Hello World';
console.log( s.substring(0, 6).toUpperCase() );
```

OUTPUT: HELLO W

(iv) Split : [Splits a string by Array]

- const s = 'Hello World';
console.log(s.split(''));

↑ Splits letter seperately

NOT VERY HANDY

OUTPUT:-

0: 'H'
1: 'E'
2: 'L'
3: 'L'
4: 'O'
5: Space.
6: 'W'
7: 'O'
8: 'R'
9: 'L'
10: 'D'
length: 11

- const s = 'technology, computers, it, code';

console.log(s.split(', '));

↑ Space.

OUTPUT : 0 : 'technology'
1 : 'computers'
2 : 'it'
3 : 'code'
length : 4

* ARRAYS :-

- Basically are variables that hold multiple values.

```
const numbers = new Array(1,2,3,4,5);
```

Whenever we see new we come to know that we have used constructor to construct Array. Here Array() is our constructor.

```
console.log(numbers);
```

OUTPUT : 0 : 1
1 : 2
2 : 3
3 : 4
4 : 5
length : 5

- construct fruits = ['apples', 'oranges', '10', true, 'pears'];

```
console.log(fruits);
```

OUTPUT :-
```
0: 'apples'
1: 'oranges'
2: 10
3: true
4: 'pears'
length: 5
```

Here we can have different datatypes used in the arrays that will work in JS unlike other languages

- const fruits=['apples', 'oranges', 'pears'];

```
console.log(fruits[1]);
```

OUTPUT: Oranges.

- const fruits=['apples', 'oranges', 'pears'];

```
fruits[3]='grapes';
console.log(fruits);
```

OUTPUT: ["apples", "oranges", pears, grapes]

- const fruits=["apples", "oranges", "pears", "grapes"];

fruits.push('mangos');
fruits.unshift('strawberries');

console.log(fruits);

OUTPUT:
0: "strawberries"
1: "apples"
2: "oranges"
3: "pears"
4: "grapes"
5: "mangos"
length: 6

fruits.pop(); → It will remove last element from the Array.

console.log(fruits.indexof('oranges'));

It will show the position of oranges.

* Object literals :-

- Basically are key values pairs.

```
const person: {
    firstName: 'JOHN',
    LastName: 'Doe',
    age : 30,
    hobbies: [ music, 'movies', 'sports' ]
    address {
      street: '50 main st',
      city : 'Boston',
      State : 'MA',
    }
}

console.log (person );

console.log (person.Age, person. hobbi
```

(can access seperate values.

```
console.log (person. hobbies [1] );
console.log (person. address. city);
```

- If we want to make firstName, LastName, Age etc a variable. So after object literals is over...

```
const { firstName, lastName, address: {city} } = person;

console.log(city);
```

It will print Boston.

- Adding the property.

After we complete object literal type.

```
person.email = 'joh@gmail.com';
console.log(person);
```

* ARRAYS of OBJECT:

```
const todos = [
    {
        id: 1,
        text: 'Take out Trash',
        isCompleted: true
    },
```

keys →

```
{
  id: 2,
  text: 'meting with boss',
  isCompleted: true.
},
{
  id: 3,
  text: 'Dentist appt',
  isCompleted: false
}
];

console.log(todos);
```

* JSON :-

- Dataform used within API
  In API data to Server.
- Similiar to Object literals.

⇒ Difference b/w JSON and Object literals :-

- Double quotes around keys and strings.

  eg. "text" ; "meting with Bos

If we want to pass JSON string into our data literals.

```
const todoJSON = JSON.stringify(todos);
console.log(todoJSON);
```

↑ method for JSON

This will pas JSON string into our object literals

* for loops :-

```
for(let i=0; i<10; i++){
  console.log(i);
}
```

OUTPUT: 0 1 2 3 4 5 6 7 8 9

* while loops :-

```
let i: 0;
while (i<10) {
console.log(`while loop Number: ${i}`)
  i++;
}
```

* loop through Arrays :-

Taking the todo object literal after it ends type ....

```
for (let i=0; i<todos.length; i++){
  console.log (todos[i].text);
}
```

NOT the best way.

- This could be any variable

```
for(let todo of todos) {
  console.log (todo);
}
```

of-for loop

- forEach, map, Filter :-

(i) forEach :-

```
todos.forEach(function(todo){
  console.log (todo.text);
});
```

(ii) Map :-

```
const todoText = todos.map(function(todo) {
    return todo.text;
});

console.log(todoText);
```

(iii) filter :-

```
const todoCompleted = todos.filter(function(todo) {
    return todo.isCompleted === true;
});

console.log(todoCompleted);
```

* filter and map together :-

```
const todoCompleted = todos.filter(function(todo) {
    return todo.isCompleted === true;
}).map(function(todo) {
    return todo.text;
});
```

```
console.log(todoCompleted);
```

* Conditional Statements :-

```
const x = 10;
if (x == 10) {
    console.log('x is 10');
}
```

Triple equals to is used to match the data types to whom you use if statement.

* Turnery operator :-

```
const x = 10;
const color = x > 10 ? 'red' : 'blue';
console.log(color)
```

then

else

* Switch :-

```
const x = 10;
const color = x > 10 ? 'red' : 'blue';

switch(color) {
    case 'red':
        console.log('color is red');
        break;

    case 'blue':
        console.log('color is blue');
        break;

    default:
        console.log('color is NOT B(R)');
        break;
}
```

* functions :-

```
function addNums(num1, num2) {
    console.log(num1 + num2);
}

addNums(5, 4);
```

* Arrow function :-

- const addNum = (num1 = 1, num2 = 1)
  => {
  return num1 + num2;
  }

- console.log(addnums());

- const addNums = (num1, num2)
  => num1 + num2

console.log(addNums(5, 5));

* Object Oriented Programing :-

- Constructor functions with prototypes.
- ES6 classes.

(i) Constructor functions with prototypes :-

```
function Person(firstName, lastName,
    dob) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.dob = dob;
}
```

constructor function

you can add as many person as you want

```
const Person1 = new Person('John',
        'Doe', '4-3-1980');
console.log(person1);
```

Initiate object

⇒ Prototype :-

```
function person(firstName, lastName, dob)
  {
    this.firstName = firstName;
    this.lastName = lastName;
    this.dob = new Date(dob);
    this.getFullName = function() {
      return `${this.firstName}
        ${this.lastName}`;
    }
}
```

```
person.prototype.getBirthYear = function()
{
    return this.dob.getFullYear();
}

person.prototype
console.log(person1);
```

⇒ In ES6 :-

Creating a class :-

```
class Person {
    constructor(firstName, lastName,
                dob) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.dob = new Date(dob);
    }

    getBirthYear()
    {
        return this.dob.getFullYear();
    }

    getFullName(){
        return `${this.firstName}
                ${this.lastName}`;
    }
}
```

Meditation is the best mode of worship.

- Adding methods to classes which will result to show in prototype

eg. BirthYear & FullName.

* selector elements :-

```
console.log(window);
```

↑ window is an object it is the parent object of the browser.

*(i) Single element :-

```
console.log(document.getElementbyId('my-form'));
```

OR

```
const form = document.getElementbyId
                ('my-form');
console.log(form);
```

- console.log(document.~~getElement~~querySelector('.container'));

(ii) multiple element :-

console.log(document.querySelectorAll('.item'));

Use this it can take classes also

OR

console.log(document.getElementByClassName('item'));

- const item = document.querySelectorAll('.item');

items.forEach((item) => console.log(item

- const ul = document.querySelector('.ite

ul.remove();

↓

method to remove ul tag

* Event :-

```
const btn = document.querySelector('.btn');

btn.addEventListener('click', (e) => {
    e.preventDefault();
document.querySelector('#my-form').
style.background = '#ccc';
```