Chapter 2: JavaScript Language Basics:

3. Let's start Coding:

index.html:

```
<!DOCTY....
   :

      <body>
          <h1> Section 2: JavaScript Language Basics </h1
      </body>

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

script.js:

```
    console.log('Hello World!!!');
```

4. A Brief Introduction to JavaScript:

What is JavaScript?

JavaScript is:

- A lightweight, cross-platform, object-oriented programming language.
- One of the three core technologies of web development.

JavaScript is used in:

- Client-side: It was traditionally used only in the browser.
- server-side: With NodeJS, we can use JavaScript on the server aswell.

With JavaScript:

- Dynamic effects & interactivity is possible.
- Frameworks like React / Angular are based on JavaScript

JavaScript Versions:

ES 5 → ES6/ES2015 → ES7/ES2016 → ES8/ES2017

5. Variables and Data Types:

Primitive JS datatypes:

→ Number: Floating point numbers, for decimals and integers.
→ String: Sequence of characters, used for text.
→ Boolean: Logical data type that can only be true or false.
→ Undefined: Data type of a variable that doesnot hav a value yet.

→ Null Also means 'non-existent'.

JavaScript has dynamic typing: data types are automatically assigned to variables.

Script.js:

```
         use camel cases in JS
var firstName = 'John';
console.log(firstName);

var lastName = 'Smith';
var age = 28;

var fullAge = true;
console.log(fullAge);

var job; → This is undefined
Console.log (job~~fullAge~~); But it is defined here.

job = 'Teacher';
console.log(job);
                → Variables names should always start with
var _3years = 3; an alphabet, underscore or $.
var johnMark = "John and Mark';
// var if = 23;
    ↳ Don't use reserved keywords as variable name
```

Output:

```
John

true

undefined

Teacher
```

6. Variable Mutation and Type Coercion:

Script.js:

```
/* Comment out the previously written code
*/

/* Variable mutation and type coercion
*/

var firstName = 'John';

var age = 28;

// Type coercion : This means all the output we are
// getting in the console is type coerced i.e, converted
// into a string
console.log(firstName + ' ' + age);

var job, isMarried;
job = 'teacher';
isMarried = false;
```

```
console.log ( firstName + ' is a ' + age + ' year old '
+ job + '. Is he married? ' + isMarried);

// variable mutation: Means to change the value of a
                                          variable.
age= 'twenty eight';
job= 'driver';

alert ( firstName + ' is a ' + age + ' year old ' +
job + '. Is he married? ' + isMarried);

var lastName = prompt ('What is his last Name?');
console.log ( firstName + ' ' + last Name);
```

7. Basic Operators:

: Comment out the above written code.

```
/*
Basic operators
*/

var year, yearJohn, yearMark ;

now = 2018;
ageJohn = 28;
ageMark = 33;
//Math operators
yearJohn = now - ageJohn;
```

```
yearMark = now - ageMark;

console.log(yearJohn);

console.log(now + 2);
console.log(now * 2);
console.log(now / 10);

//Logical operators
 var johnOlder = ageJohn < ageMark;
 console.log(johnOlder);

// typeof operator
 console.log(typeof johnOlder);
 console.log(typeof ageJohn);
 console.log(typeof 'Mark is older than John');
 var x;
 console.log(typeof x);.
```

Output:
```
1990
2020
4036
201.8
true
boolean
number
string
```

\boxed{IMP}

8. Operator Precedence:

You can find the precedence of different operators here: codingheroes.com → JavaScript Operator Precedence.

script.js:

comment out the previously written code.

```
:

/* Operator precedence
*/

var now= 2018;
var yearJohn = 1989;
var fullAge = 18;

//Multiple operators
var isFullAge= now - yearJohn >= fullAge //true
console.log (isFull Age);

// Grouping
var ageJohn = now - yearJohn;
var ageMark = 35;
var average = (ageJohn + ageMark) / 2;
console.log (average);
```

```
//Multiple assignments

var x,y ;

x = y = (3+5) * 4 - 6 ; // 8 *4 - 6 // 32-6 // 26

console.log (x , y);


//More operators

x *= 2 ;   → equivalent to :  x = x *2

console.log (x);

x += 10 ;   → equivalent to  x = x +10.

console.log (x);

x -- ;

console.log (x);
```

11. If else statements:

script.js:

```
/*....
* If /else statements
*/

var  firstName = 'John';

var  civilStatus = 'single';
                      Returns true or false.
if (civil Status === 'married') {
    console.log(firstName + ' is married!');
} else {
    console.log (firstName + ' will hopefully marry
  soon :)');
}

var isMarried = true;
if (isMarried) {
    console.log (firstName + ' is married!');
} else {
    console.log (firstName + ' will hopefully marry
    soon :)');
}

var massMark = 78;
var heightMark = 1.69;

var massJohn = 92;
var heightJohn = 1.95;

var BMIMark = massMark / (heightMark * heightMark
var BMIJohn = massJohn / (heightJohn * heightJohn).

if (BMIMark > BMIJohn){
    Console.log('Mark\'s BMI is higher than John'

} else {
    console.log('John\'s BMI is higher than Mark\'s.'
}
```

Output:
```
John will hopefully marry soon :)
John is married!
Mark's BMI is higher than John's.
```

12. Boolean Logic:

Basic Boolean Logic: NOT, AND & OR.

var A

var B

AND	TRUE	FALSE
TRUE	T	F
FALSE	F	F

var A

var B

OR	TRUE	FALSE
TRUE	T	T
FALSE	T	F

→ AND (&&) ⇒ true if all are true

→ OR (||) ⇒ true if one is true

→ NOT (!) ⇒ inverts true/false value.

script.js:

```
var   firstName = 'John';

var   age = 20;

if (age < 13) {
    console.log(firstName + ' is a boy.');
} else if ( age >= 13 && age < 20) {
    console.log( firstName + ' is a teenager.');
} else if ( age >=20 && age < 30) {
    Console.log (firstName + ' is a young man!');
} else {
    console.log (firstName + ' is a man.');

}
```

Output: John is a young man.

13. The ternary Operator & Switch Statements :

Script.js:

```
var   firstName = 'John';

var   age = 14;

//Ternary operator
```

$$\overbrace{\texttt{age >= 18}}^{\text{if This is true}}\ \texttt{?}\ \overbrace{\texttt{console.log(firstName + ' drinks beer.'}}^{\text{This will be printed}}$$

$$\texttt{:}\ \underbrace{\texttt{console.log(first Name + 'drinks juice.');}}_{\text{else, this will be printed.}}$$

```
var drink = age >=18 ? 'beer' : 'juice';
console.log (drink);
```

Ternary operator is just a simpler way of writing simple if-else statements.

```
// Switch statement

var job= 'instructor';
switch (job) {
    case 'teacher':
    case 'instructor':
        console.log (firstName + ' teaches kids how
            to code.');
        break;

    case 'driver':
        console.log (firstName + ' drives an uber in
            Lisbon.');
        break;
    case 'designer':
        console.log ( firstName + ' designs beautiful
            websites.' );
        break;
    default: →this is else
        console.log ( firstName + 'does something else!
}
```

Switch statements are also similar to if-else statements.

```
// Boolean Logic code written in switch statements.

age= 56;
switch (true) {
    case age < 13:
        console.log ( firstName + 'is a boy.');
        break;
    case age >=13 && age < 20:
        console.log ( firstName + ' is a teenager.');
        break;
    case age >=20 && age < 30:
        console.log( first Name + ' is a young man.');
        break;
    default :
        Console.log (firstName + ' is a man.');
}
```

Output:
```
John drinks juice
juice
John teaches kids have to code.
John is a man.
```

14. Truthy and Falsy Values and Equality Operators:

Falsy values: undefined, null, 0, '', Not a Number (NaN)

Truthy values: which are not falsy values.

script.js:

```
var height;
height = 23;
if (height || height === 0) {
    console.log ('Variable is defined');
} else {
    console.log ('Variable has Not been defined');
}
```

```
// Equality operators

if (height === '23') {
    console.log('The == operator does type coercion!
}
```

The "===" operator is used to compare strict

But "==" operator just compares whether its

a string or a number.

If we use == operator, 23 = '23' will return

true and === operator returns 23 = '23' as

false. So, '== operator does type coercion.

It's better to use === operator to avoid bug

17. Functions:

script.js:

```
function calculateAge(birthYear) {
        return 2018 - birthYear;
}

var ageJohn = calculateAge(1990);
var  ageMike  =  calculateAge(1948);
var  ageJane  =  calculateAge (1969);
console.log (ageJohn, ageMike, ageJane);


function  yearsUntilRetirement (year, firstName) {
    var  age  =  calculateAge(year);
    var  retirement  =  65 - age;

    if  (retirement  > 0) {
        console.log (firstName + ' retires in' +
        retirement + ' years.');
    } else {
        console.log(firstName + ' is already
        retired.');
    }
}


yearsUntilRetirement (1990, 'John');
yearsUntilRetirement (1948, 'Mike');
yearsUntilRetirement (1969, 'Jane');
```

18. Function Statements & Expressions:

```
//This is a function declaration:

 function whatDoYouDo(job, firstName) { }
```

script.js

```
 //This is a function expression

  var   whatDoYouDo = function(job, firstName) {
     switch (job) {

          case 'teacher':

               return firstName + 'teaches kids how
                 to code';

          case 'driver':

               return firstName +' drives a cab in
                 Lisbon.';

          case 'designer':

               return firstName + ' designs beautifu
                 websites';

          default:

               return firstName +' does something
                 else';
          }
 }
```

when we use the return keyword, we return whatever we define after it and the function finished. So we don't need break here

```
console.log(whatDoYouDo('teacher', 'John'));
console.log(whatDoYouDo('designer', 'Jane'));
console.log(whatDoYouDo('retired', 'Mark'));
```

Output:

```
John teaches kids how to code
Jane designs beautiful websites
Mark does something else.
```

→ Usually anything that we do produces a result, it is an expression

For example: In console:

```
typeof 23 ↵                          }
"number" → This is the result        } So this is an expression
```

→ Statements do things but they don't provide immediate results.

Examples: if-else statements, while loop, function declaratio

For example: In console:

```
if (true) { console.log(23); } ↵
23
undefined ← This doesn't really return anything. Bu
            the '23' comes from console.log
```

→ Function expressions produce an immediate result.

→ Function declarations donot provide an immediat result.

19. Arrays:

script.js:

```
// Arrays are zero index elements. The elements in
// an array start from zero.

// Initialize new array

var names = ['John', 'Mark', 'Jane'];       We can defir
               0       1       2             an array in
var years = new Array(1990, 1969, 1948);     these two
                                             ways.

console.log(names[2]); → This returns Jane
console.log(names.length); → This returns '3' bcoz the length
                             of the array is 3.

// Mutate array data
names[1] = 'Ben'; → This replaces index 1 element ie, Mark
                    with Ben.
names[names.length] = 'Mary'; → This adds a new element
                                at position '3'.
console.log(names);
```

```
// Different data types

var john = ['John', 'Smith', 1990, 'designer', false

john.push('blue'); → This adds a new element at the
                     end of the array.
john.unshift('Mr.'); → This adds a new element
                       at the beginning of the array.
console.log(john);

john.pop(); → This removes the last element from
              the array.
john.pop(); → This again removes the last element
              from the array.
john.shift(); → This removes first element from
                the array.
console.log(john);

console.log(john.indexOf(23)); → As 23 is not present
                                 in the array, this
                                 returns -1.

var isDesigner = john.indexOf('designer') === -1 ?
'John is NOT a designer' : 'John IS a designer';

console.log(isDesigner);
```

22. Objects and Properties:

script.js:

```
// Object literal

var john = {
    firstName: 'John',
    lastName: 'Smith',
    birthYear: 1990,
    family: ['Jane', 'Mark', 'Bob', 'Emily'],
    job: 'teacher',
    isMarried: false
};
```

We can create an object in this way or new object way as shown below.

An object can hold an array or even other objects.

```
console.log(john.firstName);
console.log(john['lastName']);
```

→ We can print in this way or this

```
var x = 'birthYear';
console.log(john[x]);

john.job = 'designer';
```

→ This replaces teacher with designer.

```
john['isMarried'] = true;
```

→ This changes isMarried from false to true

```
console.log(john);
```

```
//new Object Syntax
var jane = new Object();
jane.firstName = 'Jane';
jane.birthYear = 1969;
jane['lastName'] = 'smith';
console.log(jane);
```

V.IMP

23. Objects and Methods:

script.js:

```
var john = {
    firstName: 'John',
    lastName: 'Smith',
    birthYear: 1992,
    family: ['Jane', 'Mark', 'Bob', 'Emily'],
    job: 'teacher',
    isMarried: false,
    calcAge: function() {
        this.age = 2018 - this.birthYear;
    }
};
john.calcAge();
console.log(john);
```

Only objects have methods. Arrays are also objects because they can also have methods

This function is a method of john

Instead of hard coding the age, we wrote this function

Here this means this object i.e, john.

V.IMP

26. Loops & Iteration:

script.js:

```
// for loop
         This prints from 1 to 20
for (var i=1; i<=20; i++) {
    console.log(i);
}

//What happens in this loop is:

/* i=0, 0<10 true, log i to the console, i++
   i=1, 1<10 true, log i to the console, i++
   :
   i=9, 9<10 true, log i to the console, i++
   i=10, 10<10 false, exit the loop.

   i++ is equivalent to i=i+1
   i+=2 is equivalent to i=i+2.
      :
*/

var john = ['John', 'Smith', 1990, 'designer', false,
  'blue'];

for (var i=0; i<john.length; i++) {
    console.log(john[i]);
}
```

This prints all the elements present in the array john.

```
// While loop

var i = 0;

while (i < john.length) {
    console.log(john[i]);
    i++;
}
```

This is an alternative of for loop.

```
//continue and break statements

var john = ['John', 'smith', 1990, 'designer', false,
 'blue'];

for (var i = 0; i< john.length; i++) {
    if (typeof john[i] !== 'string') continue;
    console.log(john[i]);
}
```

This prints all the available strings from the array

```
for (var i=0; i<john.length; i++) {
    if (typeof john[i] !== 'string') break;
    console.log(john[i]);
```

This prints only John and smith bcoz 1990 is not a string. So this breaks the loop when the condition is not satisfied.

// Looping backwards This prints the array from the back.

```
for (var i = john.length - 1; i>=0; i--){
    console.log(john[i]);
}.
```

27, 28, 29: Coding Challenges:

$\boxed{\text{V.V. Imp}}$