Introduction To JavaScript

PAGE NO. 1
DATE / /

- JavaScript is light weight, Just In Time (JIT) programming language.

FAQ: What is JIT?

-> There are two types of compilation techniques

① JIT -> Just In Time
② AOT -> Ahead Of Time

JIT [Loaded then Compiled into Browser]

- Compiles logic in browser

AOT [Compiled in application then loaded in browser]

- Compiles login at application level

* JavaScript supports various programming techniques and approaches

a) Functional programming
b) Structural programming
c) Imperative programming
d) Object oriented programming

* JavaScript is not an OOP Language
* It supports only few features of OOP

* Javascript is used

a) Client side : HTML
b) Server side : Node.js
c) Database : Mongo DB
d) Animations : Action Script, Flash

* Java script with HTML [Client Side]

- Javascript is used to reduce burden on server.
- Javascript can reduce burden on server by

1) Handling Validations
2) Handling Interactions
3) DOM Manipulations

a) Adding elements
b) Removing elements
c) Rendering new data into elements
d) Update data in elements

FAQ: What is role of JavaScript in HTML?

-> DOM Manipulations

* Integrating Javascript into page

① Inline

ex. <button onclick = "window.print()">
Print </button>

② Embeded

ex <script type = "text/javascript">
function name () {
// logic
}
</script>

③ External

ex.

① Create new file
Printing.js
function printPage () {
window.print();
}

② Link to HTML page

<script src = "path/printing.js">
</script>

* How Javascript refers HTML Elements?

→ There are various ways for referring HTML elements which are as follows

1) JavaScript can refer elements by using DOM Hierarchy.

Example:

```
<script>
    function bodyload(){
        window.document.images[0].src = "path";
        window.document.forms[0].elements[1]
        .value = "Register";
        window.document.forms[1].elements[1]
        .value = "Login";
    }
</script>
```

Advantage:

It is faster in rendering

Disadvantage:

If you change the position of any element then every time you have to update its position in code.

2) Javascript can refer element by using "name"

Example:

```
<script>
    function bodyload(){
        pic.src = "path";
        frmRegister.btnRegister.value = "Register";
    }
</script>

<body onload = "bodyload()">
    <img width = "100" name = "pic" height = "200" />
    <Form name = "frmRegister">
        <button name = "btnRegister">
        </button>
    </Form>
</body>
```

- Advantage:

  - You can access any element directly by using its reference name.

- Disadvantage:

  - You can not access child elements DIRECTLY

∴ You have to refer both, child and parent hierarchy

3) You can refer by using "ID"

- Using document.getElementById()

```
<script>
    function bodyload(){
        document.getElementById("pic").src="path";
        document.getElementById("btnRegister").
            value="Register";
    }
</script>
```

Disadvantages

- Every element can have only one ID
- ID have conflict with CSS reference
- ID is not required if you are accessing any direct document element.

(v) Javascript can refer any element by using CSS selectors

- Using method document.querySelector()

Example :

```
<script>
    function bodyload() {
        document.querySelector("img").src = "path";
        document.querySelector(".hfh-register")
        .value = "Register";
        document.querySelector("#login")
        .value = "Login";
    }
</script>
```

Disadvantages

- Queryselector can apply effects only to first element

⇒ Advantages:

- However it can handle data for multiple elements
- It can also handle styles for multiple elements

Syntax :

```
document.querySelector("h2")
document.querySelector("#pic")
document.querySelector(".pic");
```

5) JavaScript can refer all elements having common name

- Using document.getElementsByName()

Example:

```
<script>
    function bodyload () {

result = document.getElementsByName ("pay");
alert ("Total no. of payment methods :" +
        result.length );

    }
</script>
```

6) JavaScript can refer all elements having common class name

- Using document.getElementByClassName()

Example:

```
function bodyload () {

    result = document.getElementsByClassName (".frm");
    alert ("Total no. of forms :" + result.length);
}
```

PAGE No. 9
DATE / /

7) Javascript can refer elements by using Tag name

- Using document.getElementsByTagName()

* JavaScript Output Techniques

1) alert() : Mandatory Confirmation
(Display only "ok" as option)
2) confirm() : Optional Confirmation
(Displays both "ok" and "cancle" as options)
3) console.[log(), warn(), info(), error()]
: Used while testing
Displays output on "Console" tab
4) inner HTML : Inserts code into specified tag
5) outer HTML :
6) document.write() : Display message on same window

* JavaScript Input Techniques

1) prompt() : Like alert with input
2) form input elements

1) prompt() :

It is like alert() message
But it has "input".

Syntax:

```
prompt("Your Message", "Default Value");
```

Prompt Returns

Value Event

null on Cancle

"" [Empty] on OK without value

value on OK with value

Example :

```
<html> <script>
    function createClick() {
        foldername = prompt("Enter Foldername",
                            "New Folder");

        if(foldername == null) {
            document.write("You cancled");
        } else if (foldername == "") {
            document.write("Please provide
                            : Foldername");
        } else {
            document.querySelector("p")
            .innerHTML += "Folder Created :"
                        + foldername;
        }
    </script>
```

* Form - Input Elements

- You can use form elements like textbox, password, number, email, radio etc.
- Every form element must have reference ID

ex: <input type="text" id="txtName" />
<select id="lstCities" />

- You can access element and use its properties

Ex document.getElementById("txtName").value;
document.getElementById("lstCities").value;
document.getElementById("optStock").checked == true | false;

For coding Standards refer site PMD Code Analyzer

FAQ: What is strict mode of JavaScript?

Ans: Strict mode for Javascript allows to reduce code in consistancy
So that developers have to follow coding standards

ex:

```
<script>
"use strict";

var x=10;
document.write("x="+x);

</script>
```

FAQ: How to write JavaScript for Legacy Browser? [Old version]

Ans: By enclosing code in HTML Comments

Syntax:

```
<!--
    Code
-->
```

FAQ: How to add Java Script coments

Ans: // Single line
/* */ multi line
<!-- --> xml-Comments

JavaScript Language Basics

* Variables

- Variables are storage locations in memory where you can store value and use it as part of any expression

- JavaScript allows to use variables directly if it is not in strict mode

- In strict mode variable configuration comprises 3 phases

a) Declaration	var x;
b) Assignment	x = 10;
c) Initialization	var y = 20;

a) Declaration :
Declaring is defining scope and name for variable
ex. var username;

b) Assigning :
Rendering a value into variable after declaration
ex. username = "john";

c) Initialization
Rendering value into variable ~~after~~ during Declaration
ex. var username = "John";

Note:

Declaration or Initialization is mandatory if JavaScript is in Strict mode

Variables in JS can be declared by using

a) var
b) let
c) const

a) var :

- Defines function scope variable
- You can declare in any block of function and access from any another block in function
- "var" allows declarations, assignment and initialization.

~~Syntax~~ : "var" allows shadowing

FAQ: What is shadowing ?

Ans: Redeclaring same name identifier within the scope

- "var" allows hoisting

FAQ: What is hoisting ?

Ans: Compiling technique where compiler can find declaration of variable before using it

So we can use "var" before declaration

Syntax:

b) let:

- Used to define block scope variable
- It is accessible only in the block where it is declared or to its inner blocks
- It allows declaration, assignment and initialization
- It will not allow shadowing
- It will not allow hoisting

Syntax :

c) const :

- It is block scope variable
- Allows only Initialization
- No assigning
- No Declaration
- No Shadowing
- No Hoisting

Note :

We can change value of const
Const is use to restrict the user
to put some value in the variable [mandatory]

8.

PAGE No. 19
DATE / /

FAQ: Why we need const?

Ans: const is required to initialize memory

At the time of loading application or component memory is initialized with some default value

Note:

If initialization is missing then by default value will be "undefined"

FAQ: Can const change its value?

Ans: Dynamically yes.

* Global scope for variable

- You can declare variable in module scope so that it is global and accessible to all functions in module

FAQ: Can we declare global variable inside function?

Ans: If Yes, By using browser "window" object

Syntax: function f1 () {
window.y = 20;

```
function f2() {
    document.write("y : " +y);
}
```

* Variable Naming :

① Variable name must start with alphabet or underscore (_)

username - Valid
_username - Valid
2021Sales - Invalid

The special character (_) is used to indicate that variable is not implemented It requires further implementation

② Variable name can't exceed more than 255 characters

③ Don't use language key words for variable names

④ Always variable must speek what it is

⑤ Special characters and blank space is are not allowed [Except underscore "_"]

* Datatypes :

- It defines datastructure
- Datatypes determine the size and type of data
- Datatypes are classified into 2 types

1) Primitive datatypes
2) Non-Primitive datatypes

1) Primitive data type

- Primitive datatypes are immutable datatypes
  - That is their structure will not change
  - They have fixed range of values
  - The range can't change
- They uses memory stack
  - Stack uses LIFO [Last In First Out]

- JavaScript primitive datatypes are

a) number
b) string
c) boolean
d) null
e) ~~unidentified~~ undefined

Note: JavaScript is not Strongly typed
It is Implicitly typed

ex. var x = 10 // x is number

~~var~~

x = "John"; // x is string

a) number types :

In JavaScript number type refers to

①	signed integer	-10
②	unsigned integer	10
③	floating point	24.53
④	double	145.563
⑤	decimal	45.5695 [29 Decimal]
⑥	exponent	2e3 = 2000
⑦	binary	0b0101
⑧	octal	00761
⑨	hexa	0xfd

- JavaScript uses "isNaN()" method to verify the number type

Syntax: if (isNaN(value)).
{

This method returns "true" if value is not a number

- JavaScript can not identify numeric value in string format you have to explicitly convert into number by using following methods

a) parseInt()
b) parseFloat()

Syntax:

```
var age = "20"
document.write(age+1); //201
document.write(parseInt(age)+1); //21
```

b) String type

String is a literal with group of characters enclosed in

a) single quotes ('')
b) double quotes ("")
c) Back tick (``)

Example:

```
var msg1 = "Hello!" + username + "you will be"
    + (parseInt(age)+1) + "old";

var msg2 = 'Hello' + username + 'you will be'
    + (parseInt(age)+1) + 'old';
```

var msg3 = `Hello ! ${username} you will be ${parseInt(age)+13} old`;

- Single and double quote are used to configure inner and outer string combination
- Back tick allows a string with embedded expression : ${3}"

${3} → Data Binding Expression

Note:

Escape Sequence Issues

- Special characters in a string can escape printing
- You have to print the non printable characters by using "\"

String Handling in JavaScript

- JavaScript provides set of methods and properties to format and manipulate string

- String formatting methods [Changing Appearience]
  - bold()
  - italic()
  - sup()
  - sub()
  - fontcolor()
  - fontsize()
  - toUpperCase()
  - toLowerCase()

String property

- length :: Returns the total no of characters [Count of characters]

Syntax:

```
var msg = "Welcome to JS";
msg.bold().italics().fontcolor("green");
msg.toUpperCase
```

Events :

- onkeyup : Action to perform when key is released
- onblur : Actions to perform when control is blured [last focus]
- onclick : Actions to perform when clicked
- onload : Actions to perform on page or image load

Note:
You can define styles and classes dynamically to any element

ex. document.querySelector("p").style.color = "red";
document.querySelector("button").className = "btn btn-primary";

* String Manipulations

- charAt () : Returns character at specified index
- charCodeAt() : Returns character ASCII code present at specified index
- slice () : It can extract characters betn specified index
- substr() : It can return specified no of characters from specified index
- substring() : It can return characters from specific index in any direction

FAQ : What is charCodeAt() ?

-> charAt () returns character at specified index
charCodeAt() returns its character code as per UTF standards

A = 65
Z = 90

FAQ : What is difference between slice(), substr() and substring()?

Ans :

- slice() : It can read the chars between specified start and end index. If end is not defined then it will read upto end of string. End index must be greater than start index.

```
ex. string.slice(0,7)   // 0 to 7
    string.slice(7)     // 7 to end
    string.slice(7,0)   // Invalid
```

Syntax : String.slice(startIndex, [endIndex]);
↑ optional

startIndex <= endIndex

- Substr() : It can read specified no. of characters from given index no.

Syntax : String.substr(startIndex, countOf(chars));

```
Ex.  string.substr(7,3);
     // From 7 it will read 3 chars
     // i.e. chars at indices 7, 8, 9
     // will be returned.
```

- substring: It can read from specified index to any direction

Syntax: String.substring(startIndex, endIndex);

Note: endIndex can be less than startIndex

Ex: string.substring (0,7) // 0 to 7 chars
string.substring (7) // 7 to end
string.substring (7,0) // 7 to start[0]

* indexOf() and lastIndexOf()

- These are the functions which can find any character in a string and return its index position

- If character not found then it returns "-1"

- indexOf() function will return first occurance index
- lastIndexOf() function will return last occurance index

Ex: welcome.indexOf("e"); // 1
welcome.lastIndexOf("e"); // 6

* startsWith() and endsWith() :

- These functions are used to verify starting and ending chars in a string.

- These functions return boolean true when string is starting or ending with specified characters

Syntax: String.startsWith("chars");
String.endsWith("chars");

~~FAQ:~~

* match() :

- It is used to verify and compare given string with any regular expression
- It returns true if string format is as per regular expression

- In Javascript regular expression is enclosed in "/ /"

Syntax: var regExp = /pattern/;
var string = "";

if (string.match(regExp))
{

}

* trim() & split()

- trim() is used to remove the leading spaces in string

- split() is used to split the string at specified delimeter and return an array

c) boolean type :

- boolean types are used in decision making
- boolean type in JS can handle 2 values
  - a) true
  - b) false
- boolean conditions JavaScript can use 1 for true and 0 for false

true = 1
false : 0

d) undefined type

It specifies that value is not supplied into reference

- Undefined type is verified by using "undefined" keyword
- It indicates that value is not given into given reference

FAQ: What is difference betn undefined and not-defined ?

Ans: undefined : reference is there but value is not defined
notdefined : reference is not defined

Ex.

```
<script>
    var x;
    document.write('x = ${x} <br>
                    y = ${y} ');
</script>
```

x = undefined
y = not defined

e) Null type

- It is an exception type
- Exceptions occures at runtime
- Null indicates that value is not supplied during run time
- You can verify null tippe using null keyword.

Summary : Primitive Datatypes

- number : numeric values
- string : literals
- boolean : true / false
- null : no value at run time
- undefined : no value at compile time

2) Non-Primitive types

- They are mutable types
- They can change structure according to state and situation
- There is no fixed range of value
- Value range varies according to memory available
- They are stored in heap memory

In Javascript non-primitive types are

a) Array
b) Object
c) Map

a) Array type:

- Array means organizing in order and accessing in random order

- Stack means organizing in order and accessing the last in first out

- Queue means organizing in order and accessing the first in first out

What is purpose of array?

-> Array is used in computer programming to reduce overhead and complexity.

Arrays will reduce overhead by storing values in sequential order

Arrays will reduce complexity by storing multiple values under one name

Arrays can handle any type of value since JS is implicitly typed.

Array's size can be changed dynamically

Note:

Few technologies can't allocate various types of memory in sequential order hence they restrict array to same type of values and size can't be changed dynamically [C, C++, Java etc]

* Configure array

① Declaring array
② Initialization of memory for Array

① Declaring array:

Declaration of an arry is same as declaration of variables

ex. var x;

② Initialization of memory for Array:

For initialization of array we can use any of the following two methods

(i) ~~var~~ Datatype VarName = new Array(int);

(ii) Datatype VarName = [data, data ...];

Storing values into Array :

- Values are stored and accessed from an array by using property
- Property maps to Indexes in memory

Syntax :

```
let values = [];
values[0] = 10; // valid
values["1"] = 20; // valid
```

Reading values from Array

- You can use reference of property

ex
```
value[0]
value["1"]
```

- By using Array methods
  1. toString()
  2. join()
  3. slice()
  4. filter()
  5. find()
  6. map()