Module -3

Part A:-

1) **What is the significance of, and reason for, wrapping the entire content of a JavaScript source file in a function block?**

Ans: The purpose of wrapping is to a namespace and control the visibility of member functions. It wraps the code inside a function scope and decreases clashing with other libraries. This is what we call Immediately Invoked Function Expression (IIFE) or Self Executing Anonymous Function.
There are two main purposes for the use of a function block:

- *To avoid polluting the global scope.*
- *To avoid overriding already existing variables.*

Let's imagine you have different JavaScript files within your HTML, they would probably have global functions and variables that could be essential for the correct execution of that particular code, but if you by accident declare an already existing variable inside your own JavaScript file, you will end up overriding them. It won't matter if that variable exists in another file, because at the end, the browser will merge all JavaScript files into one. And you will then have a bug or failure in your code that will be very difficult to detect.

All up until ES5, the only way that existed to prevent something from living in the global object was through the use of a function that encapsulated the entire code.

The use of functions will also make your code be a little safer and less exposed to vulnerabilities by not letting outsiders get access to properties or methods that shouldn't be accessible.
The main reason is that this create a local variable scope and avoid use of global scope: by this way, you can be sure that your variables aren't accessible from outside, and other scripts running on same web page cannot override them (and reciprocally)....The other reason is to be able to assign the function execution to an event listener such as 'load', to delay the script execution until this event was raised: in code playground, the JS tab is loaded as linked at end of <head>, sp you often require to start your script after DOM becomes available ^^

---

2)Create "native" methods using javascript?

Ans: There are many fun and useful methods in JavaScript that aren't widely common.
Confusion Between isNaN and Number.isNaN in JavaScript.

1. Number.isFinite — MDN.
2. isFinite — MDN.
3. Math.trunc — MDN.

4. Math.floor — MDN.
5. Array.prototype.indexOf — MDN.
6. Array.prototype.includes — MDN.
7. String.prototype.repeat — MDN.

Incomplete Answer

---

4)Write a sum method which will work properly when invoked using either syntax below.
console.log(sum(2,3));
// Outputs 5

Ans:

Source code:-

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>JavaScript program to compute the sum of the two given integers. If the two values are same, then return triple their sum</title>
</head>
<body>
</body>
</html>
```

Javascript file:

```
function sumTriple (x, y) {
  if (x == y) {
    return 3 * (x + y);
    }
   else
   {
    return (x + y);
   }
 }
console.log(sumTriple(2,3));
```

---

4)How do you decode or encode a URL in JavaScript?

Ans:

Encoding and Decoding URI and URI components is a usual task in web development while making a GET request to API with query params. Many times construct a URL string with query params and in order to understand it, the response server needs to decode this URL. Browsers automatically encode the URL i.e. it converts some special characters to other reserved characters and then makes the request.

For eg: Space character ” ” is either converted to + or %20.

Example:

Open www.google.com and write a search query “javascript course”.

After search results appear, observe the browser’s URL bar. The browser URL will consist %20 or + sign in place of space.

The URL will be displayed be like: https://www.google.com/search?q=javascript%20for%course or https://www.google.com/search?q=javascript+course

1. encodeURI function: The encodeURI() function is used to encode complete URI. This function encode the special character except (, / ? : @ & = + $ #) characters.

Syntax:

encodeURI(complete_uri_string)

Source code:-

```
<script>
const url = "https://www.google.com/search?q=geeks for geeks";
const encodedURL = encodeURI(url);
document.write(encodedURL)
</script>
```

escape() function: This function takes a string as a single parameter & encodes the string that can be transmitted over the computer network which supports ASCII characters. Encoding is the process of converting plain text into ciphertext.

Syntax:

escape(string)

```
<script>
const url = "https://www.google.com/search?q=geeks for geeks";
const encodedURL = encodeURI(url);// encoding using encodeURI
document.write(encodedURL)
document.write("<br>"+escape(url)); //encoding using escape
</script>
```

---

5)Explain what a callback function is and provide a simple example?

Ans:

A callback function is a function passed into another function as an argument, which is then invoked inside the outer function to complete some kind of routine or action.

```
function greeting(name) {
  alert('Hello ' + name);
```

```
}


function processUserInput(callback) {
  var name = prompt('Please enter your name.');
  callback(name);
}
processUserInput(greeting);
```

The above example is a synchronous callback, as it is executed immediately.

Note, however, that callbacks are often used to continue code execution after an asynchronous operation has completed — these are called asynchronous callbacks. A good example is the callback functions executed inside a .then() block chained onto the end of a promise after that promise fulfills or rejects. This structure is used in many modern web APIs, such as fetch().

---

6)How to empty an array in JavaScript?

Ans:

1) Assigning it to a new empty array

This is the fastest way to empty an array:

```
a = [];
```

This code assigned the array `a` to a new empty array. It works perfectly if you do not have any references to the original array.

See the following example:

```
let b = a;
a = [];
console.log(b); // [1,2,3]
```

In this example, first, the `b` variable references the array `a`. Then, the `a` is assigned to an empty array. The original array still remains unchanged.

2) Setting its length to zero

The second way to empty an array is to set its length to zero:

```
a.length = 0;
```

The `length` property is read/write property of an `Array` object. When the `length` property is set to zero, all elements of the array are automatically deleted.

3) Using splice() method

The third way to empty an array is to remove all of its elements using the `splice()` method as shown in the following example:

```
a.splice(0,a.length);
```

In this solution, the `splice()` method removed all the elements of the `a` array and returned the removed elements as an array.

4) Using pop() method

The fourth way to empty an array is to remove each element of the array one by one using the `while` loop and `pop()` method:

```
while(a.length > 0) {
    a.pop();
}
```

This solution is quite trivial and is the slowest one in terms of performance.

---

7)How would you use a closure to create a private counter?.

Ans:You can create a function within an outer function (a closure) that allows you to update a private variable but the variable wouldn't be accessible from outside the function without the use of a helper function.

```
function counter() {
  var _counter = 0;
  // return an object with several functions that allow you
  // to modify the private _counter variable
  return {
    add: function(increment) { _counter += increment; },
    retrieve: function() { return 'The counter is currently at: ' + _counter; }
  }
}

// error if we try to access the private variable like below
// _counter;

// usage of our counter function
var c = counter();
c.add(5);
c.add(9);

// now we can access the private variable in the following way
c.retrieve(); // => The counter is currently at: 14
```

---

8)How does the this keyword work? Provide some code examples?

Ans: the value of this is assigned in different scenarios. The best way to digest the content of this article is by quickly executing the code snippet in your browser's console. Follow the below steps to launch the console in your Chrome browser:

- Open new tab in Chrome
- Right click on page, and select "inspect element" from the context menu
- Go to the console panel
- Start executing the JavaScript code

Objects are the basic building blocks in JavaScript. There's one special object available in JavaScript, the this object. You can see the value of this at every line of JavaScript execution. The value of this is decided based on how the code is being executed.Before getting started with this, we need to understand a little about the JavaScript runtime environment and how a JavaScript code is executed.

```
function foo () {
    console.log("Simple function call");
    console.log(this === window);
}
```

```
foo();    //prints true on console
console.log(this === window) //Prints true on console.
```

---

9)How would you create a private variable in JavaScript?

Ans: Alternatively, we may also use the “this” keyword to make method (function) calls to stick to the main method itself which thus makes the variables private. The main idea for using the “this” keyword is just to make things directly visible that is making methods directly accessible.

```
<script>
        function carDetails() {
                let kms = 0;
                let speed = 0;
                this.speedUp = (intialSpeed) => {
                        speed += intialSpeed;
                        kms += speed;
                };
                this.totalkmsDriven = () => kms;
        }

        let car_object = new carDetails();
        car_object.speedUp(7);
        car_object.speedUp(9);
        console.log(car_object.totalkmsDriven());

        // Undefined, since it is made private:
        console.log(car_object.kms);
</script>
```

---

10)Write a recursive function that performs a binary search?

Ans: Recursive Approach :

BASE CONDITION: If starting index is greater than ending index return false.

Compute the middle index.

Compare the middle element with number x. If equal return true.

If greater, call the same function with ending index = middle-1 and repeat step 1.

If smaller, call the same function with starting index = middle+1 and repeat step 1.

Below is the implementation of Binary Search (Recursive Approach) in JavaScript:

```
<script>
let recursiveFunction = function (arr, x, start, end) {

	// Base Condition
	if (start > end) return false;

	// Find the middle index
	let mid=Math.floor((start + end)/2);

	// Compare mid with given key x
	if (arr[mid]===x) return true;

	// If element at mid is greater than x,
	// search in the left half of mid
	if(arr[mid] > x)
		return recursiveFunction(arr, x, start, mid-1);
	else

		// If element at mid is smaller than x,
		// search in the right half of mid
		return recursiveFunction(arr, x, mid+1, end);
}

// Driver code
let arr = [1, 3, 5, 7, 8, 9];
let x = 5;

if (recursiveFunction(arr, x, 0, arr.length-1))
	document.write("Element found!<br>");
else document.write("Element not found!<br>");

x = 6;

if (recursiveFunction(arr, x, 0, arr.length-1))
	document.write("Element found!<br>");
else document.write("Element not found!<br>");
</script>
```

Part B:-

1)explain the various JavaScript data types?

Ans:There are eight basic data types in JavaScript. They are:

Data Types	Description	Example
`String`	represents textual data	`'hello'`, `"hello world!"` etc
`Number`	an integer or a floating-point number	`3`, `3.234`, `3e-2` etc.
`BigInt`	an integer with arbitrary precision	`900719925124740999n`, `1n` etc.
`Boolean`	Any of two values: true or false	`true` and `false`
`undefined`	a data type whose variable is not initialized	`let a;`
`null`	denotes a `null` value	`let a = null;`
`Symbol`	data type whose instances are unique and immutable	`let value = Symbol('hello');`
`Object`	key-value pairs of collection of data	`let student = { };`

---

2)Name the types of functions?

Ans: A Function is a block of code that is designed to perform a task and executed when it is been called or invoked.

There are 3 ways of writing a function in JavaScript:

1)Function Declaration

2)Function Expression

3)Arrow Function

1. Function Declaration: Function Declaration is the traditional way to define a function. It is somehow similar to the way we define a function in other programming languages. We start declaring using the keyword “function”. Then we write the function name and then parameters.

Below is the example that illustrate the use of Function Declaration.

// Function declaration

```
function add(a, b) {
        console.log(a + b);
}

// Calling a function
add(2, 3);
```

2. Function Expression: Function Expression is another way to define a function in JavaScript. Here we define a function using a variable and store the returned value in that variable.
Below is the example that illustrate the use of Function Expression.

```
// Function Expression
const add = function(a, b) {
        console.log(a+b);
}

// Calling function
add(2, 3);
```

3. Arrow Functions: Arrow functions are been introduced in the ES6 version of JavaScript. It is used to shorten the code. Here we do not use the "function" keyword and use the arrow symbol.
Below is the example that illustrate the use of Arrow Function.

```
// Single line of code
let add = (a, b) => a + b;

console.log(add(3, 2));
```

---

3)If we want to return the character from a specific index which method is used?
Ans: The charAt() method returns the character at a specified index (position) in a string.
The index of the first character is 0, the second 1, ...
The index of the last character is string length - 1 (See Examples below).
See also the charCodeAt() method.
Syntax:
***string*.charAt(*index*)**

---

4)What are the key differences between Java and JavaScript? / How is JavaScript different from Java?
Ans: The JavaScript programming language, developed by Netscape, Inc., is not part of the Java platform.
JavaScript does not create applets or stand-alone applications. In its most common form, JavaScript resides inside HTML documents, and can provide levels of interactivity to web pages that are not achievable with simple HTML.

Key differences between Java and JavaScript:

- Java is an OOP programming language while Java Script is an OOP scripting language.
- Java creates applications that run in a virtual machine or browser while JavaScript code is run on a browser only.
- Java code needs to be compiled while JavaScript code are all in text.
- They require different plug-ins.

For additional information about JavaScript, visit Mozilla.org

---

5)What is BOM?

Ans: The Browser Object Model (BOM):There are no official standards for the Browser Object Model (BOM).Since modern browsers have implemented (almost) the same methods and properties for JavaScript interactivity, it is often referred to, as methods and properties of the BOM.

---

6)What is DOM? What is the use of document object?

Ams: The Document Object Model (DOM) is a programming interface for web documents. It represents the page so that programs can change the document structure, style, and content. The DOM represents the document as nodes and objects; that way, programming languages can interact with the page.

A web page is a document that can be either displayed in the browser window or as the HTML source. In both cases, it is the same document but the Document Object Model (DOM) representation allows it to be manipulated. As an object-oriented representation of the web page, it can be modified with a scripting language such as JavaScript.

For example, the DOM specifies that the querySelectorAll method in this code snippet must return a list of all the <p> elements in the document:

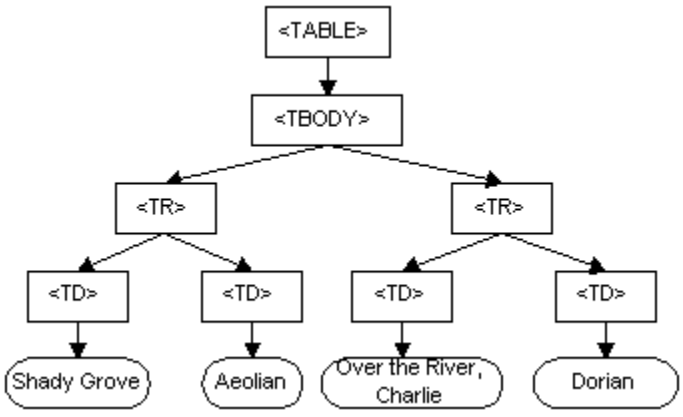
```
const paragraphs = document.querySelectorAll("p");

// paragraphs[0] is the first <p> element

// paragraphs[1] is the second <p> element, etc.

alert(paragraphs[0].nodeName);
```

What the Document Object Model is

The DOM is a programming API for documents. It closely resembles the structure of the documents it models. For instance, consider this table, taken from an HTML document:

```
<TABLE>
<TBODY>
<TR>
<TD>Shady Grove</TD>
<TD>Aeolian</TD>
</TR>
<TR>
<TD>Over the River, Charlie</TD>
<TD>Dorian</TD>
</TR>
</TBODY>
</TABLE>
```

The DOM represents this table like this:



In the DOM, documents have a logical structure which is very much like a tree; to be more precise, it is like a "forest" or "grove", which can contain more than one tree. However, the DOM does not specify that documents must be implemented as a tree or a grove, nor does it specify how the relationships among objects be implemented. The DOM is a logical model that may be implemented in any convenient manner. In this specification, we use the term structure model to describe the tree-like representation of a document; we specifically avoid terms like "tree" or "grove" in order to avoid implying a particular

implementation. One important property of DOM structure models is structural isomorphism: if any two Document Object Model implementations are used to create a representation of the same document, they will create the same structure model, with precisely the same objects and relationships.The name "Document Object Model" was chosen because it is an "object model" in the traditional object oriented design sense: documents are modeled using objects, and the model encompasses not only the structure of a document, but also the behavior of a document and the objects of which it is composed. In other words, the nodes in the above diagram do not represent a data structure, they represent objects, which have functions and identity. As an object model, the DOM identifies:

the interfaces and objects used to represent and manipulate a document

the semantics of these interfaces and objects - including both behavior and attributes

the relationships and collaborations among these interfaces and objects

---

7)What is the use of window object? Explain?

Ans:The Window Object

The window object is supported by all browsers. It represents the browser's window.

All global JavaScript objects, functions, and variables automatically become members of the window object.

Global variables are properties of the window object.

Global functions are methods of the window object.

Even the document object (of the HTML DOM) is a property of the window object:

document

is the same as:

getElementById

Window Size

Two properties can be used to determine the size of the browser window.

Both properties return the sizes in pixels:

- window.innerHeight - the inner height of the browser window (in pixels)
- window.innerWidth - the inner width of the browser window (in pixels)

The browser window (the browser viewport) is NOT including toolbars and scrollbars.

Other Window Methods

Some other methods:

- window.open() - open a new window
- window.close() - close the current window
- window.moveTo() - move the current window
- window.resizeTo() - resize the current window

---

8)How to create a function in JavaScript? define with example?

Ans: JavaScript Function Syntax

A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ().Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

The parentheses may include parameter names separated by commas:

(parameter1, parameter2, ...)

The code to be executed, by the function, is placed inside curly brackets: {}

```
function name(parameter1, parameter2, parameter3) {
  // code to be executed
}
```

Example

Calculate the product of two numbers, and return the result:

```
let x = myFunction(4, 3);   // Function is called, return value will end up in x

function myFunction(a, b) {
  return a * b;             // Function returns the product of a and b
}
```

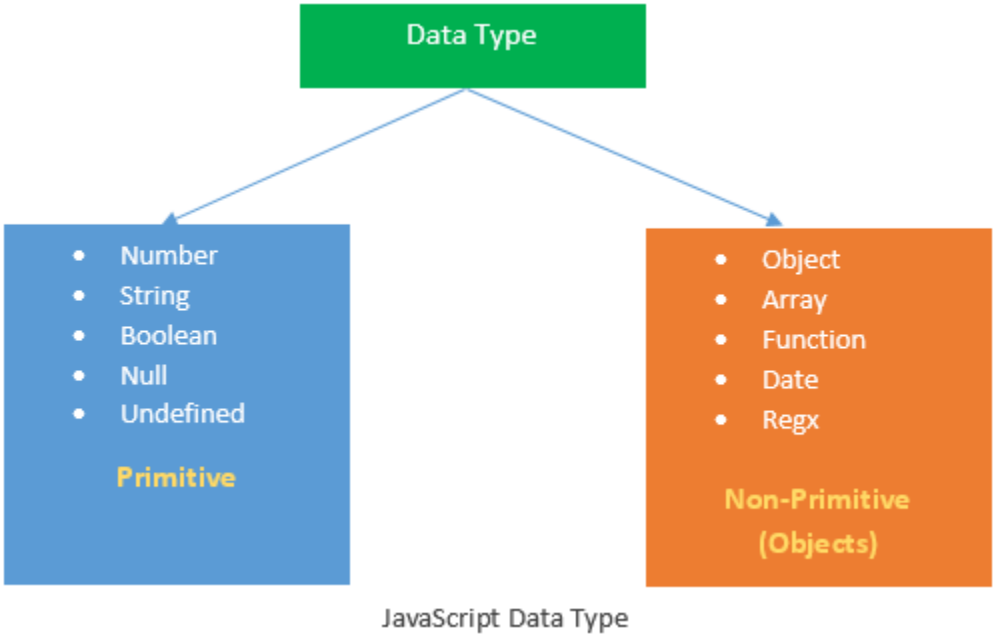
The result in x will be:

```
12
```

---

9)What are the different data types present in JavaScript?

Ans:



JavaScript Data Type

---

10)How to create objects in JavaScript?

Ans:Creating a JavaScript Object

With JavaScript, you can define and create your own objects.

There are different ways to create new objects:

- **Create a single object, using an object literal.**
- **Create a single object, with the keyword `new`.**
- **Define an object constructor, and then create objects of the constructed type.**
- **Create an object using `Object.create()`.**

Using an Object Literal

This is the easiest way to create a JavaScript Object.

Using an object literal, you both define and create an object in one statement.

An object literal is a list of name:value pairs (like age:50) inside curly braces {}.

The following example creates a new JavaScript object with four properties:

Example

```
const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

Using the JavaScript Keyword new

The following example create a new JavaScript object using `new Object()`, and then adds 4 properties:

Example

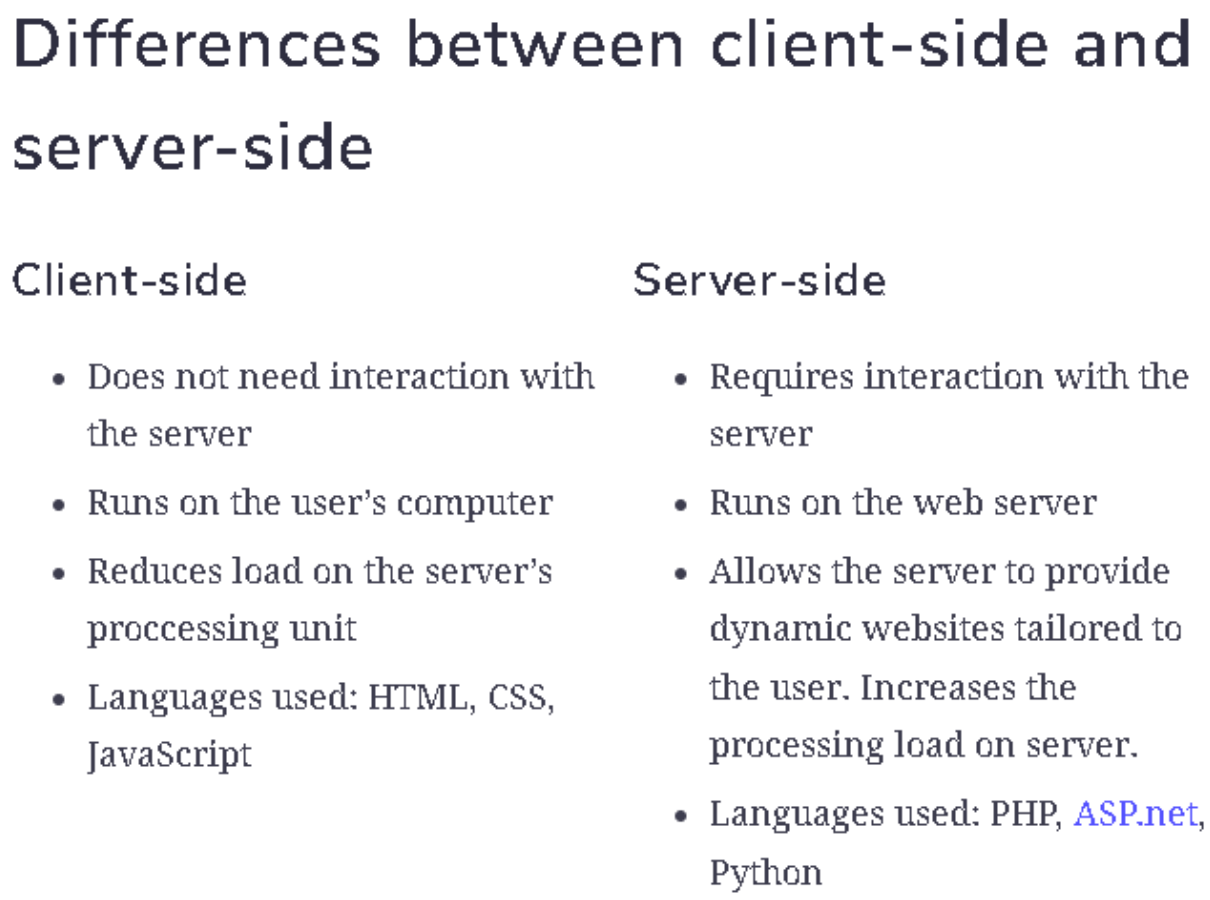
```
const person = new Object();
person.firstName = "John";
person.lastName = "Doe";
person.age = 50;
person.eyeColor = "blue";
```

---

11)Difference between Client side JavaScript and Server side JavaScript?

Ans:Client-side means that the processing takes place on the user's computer. It requires browsers to run the scripts on the client machine without involving any processing on the server.

Server-side means that the processing takes place on a web server.



---

12)In which location cookies are stored on the hard disk?

Ans:-

This depends on the user's browser and OS.

In the case of Netscape with Windows OS,all the cookies are stored in a single file called

cookies.txt

c:\Program Files\Netscape\Users\username\cookies.txt

In the case of IE,each cookie is stored in a separate file namely username@website.txt.

c:\Windows\Cookies\username@Website.txt

---

13)What's the difference between event.preventDefault() and event.stopPropagation() methods in JavaScript?

Ans:`preventDefault()`, `stopPropagation()`, and `return false` statements can all be used in jQuery *event handler* functions.

> **`preventDefault()` prevents the default browser behavior for a given element.**
>
> **`stopPropagation()` stops an event from bubbling or propagating up the DOM tree.**

Whereas,

> **`return false` is a combination of both `preventDefault()` and `stopPropagation()`.**

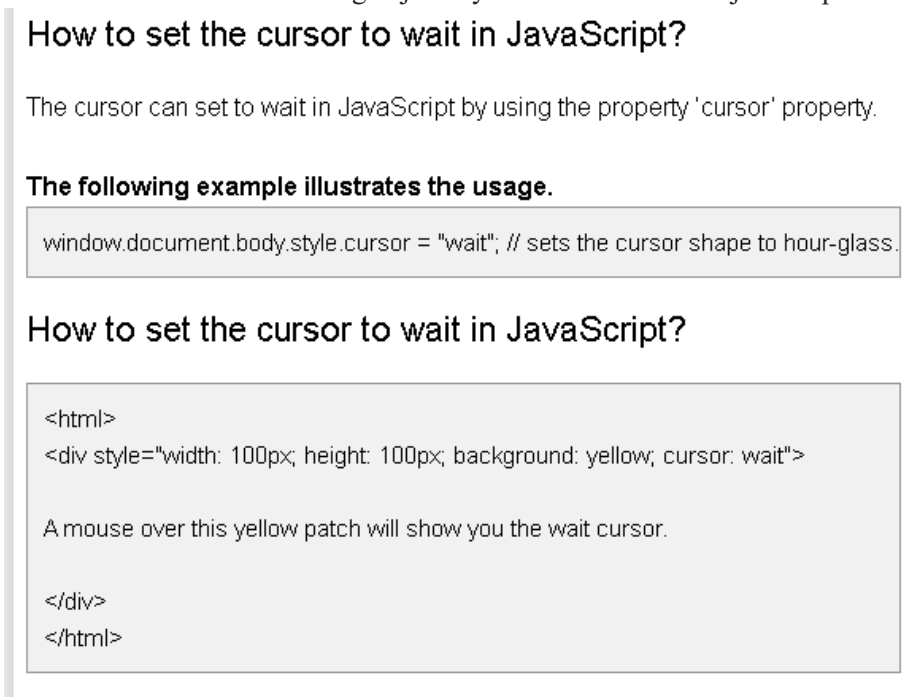
event.PreventDefault Vs. event.stopPropagation

The event.preventDefault() prevents the browsers default behaviour, but does not stop the event from bubbling up the DOM. The event.stopPropagation() prevents the event from bubbling up the DOM, but does not stop the browsers default behaviour

---

14)How to set the cursor to wait in JavaScript?What is this [[[[]]?

Ans: We can use the wait property with the mouse cursor in JS and CSS throughout the webpage. We can set the cursor to wait using object.style.cursor = “wait” in javascript.



How to set the cursor to wait in JavaScript?

The cursor can set to wait in JavaScript by using the property 'cursor' property.

The following example illustrates the usage.

```
window.document.body.style.cursor = "wait"; // sets the cursor shape to hour-glass.
```

How to set the cursor to wait in JavaScript?

```
<html>
<div style="width: 100px; height: 100px; background: yellow; cursor: wait">

A mouse over this yellow patch will show you the wait cursor.

</div>
</html>
```

---

15)What is the difference between View state and Session state?

Ans:

ViewState	SessionState
Maintained at page level only.	Maintained at session level.
View state can only be visible from a single page and not multiple pages.	Session state value availability is across all pages available in a user session.
It will retain values in the event of a postback operation occurring.	In session state, user data remains in the server. Data is available to user until the browser is closed or there is session expiration.
Information is stored on the client's end only.	Information is stored on the server.
used to allow the persistence of page-instance-specific data.	used for the persistence of user-specific data on the server's end.
ViewState values are lost/cleared when new page is loaded.	SessionState can be cleared by programmer or user or in case of timeouts.

---

16)What are the pop-up boxes available in JavaScript?

Ans:

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

Alert Box

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

window.alert("sometext");

The window.alert() method can be written without the window prefix.

Confirm Box

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax

```
window.confirm("sometext");
```

The `window.confirm()` method can be written without the window prefix.

---

17)How to submit a form using JavaScript by clicking a link?

Ans:n this article, we have submitted a form using JavaScript by clicking a link. In the body tag, created an HTML form and specify the id, method, and action of the form. In the form, specify an anchor tag with an event onclick. Create a function for JavaScript that will get executed when the link is clicked. When we click on the link, the function submitForm() will get executed. This function will get the element object using DOM getElementById() method by passing the form id to this method, then the form will be submitted by using submit() method.

Example: Create a form and submit it using the above approach. It is required for the form structure where the user will provide his/her details.

```
<!DOCTYPE html>
<html>

<body>
    <h2 style="color:green">GeeksforGeeks</h2>
    <b>Submit form details</b>

    <form id="form__submit" action="form.php" method="post">
        <label>NAME: </label><br />
        <input type="text" name="name" /><br />
        <label>AGE: </label><br />
        <input type="number" name="age" /><br />
        <label>CITY: </label><br />
        <input type="text" name="city" /><br /><br />
        <a href="#" onclick="submitForm()">Submit Here</a>
    </form>

    <script>
        function submitForm() {
            let form = document.getElementById("form__submit");
            form.submit();
        }
    </script>
</body>

</html>
```

18)How to validate a form in JavaScript?

Ans: Form Validation

HTML form validation can be done by JavaScript.

If a form field (fname) is empty, this function alerts a message, and returns false, to prevent the form from being submitted:

JavaScript Example

```
function validateForm() {
  let x = document.forms["myForm"]["fname"].value;
  if (x == "") {
    alert("Name must be filled out");
    return false;
  }
}
```

---

19)How to validate email in JavaScript?

Ans: Email validation

Validating email is a very important point while validating an HTML form. In this page we have discussed how to validate an email using JavaScript :

An email is a string (a subset of ASCII characters) separated into two parts by @ symbol. a "personal_info" and a domain, that is personal_info@domain. The

length of the personal_info part may be up to 64 characters long and domain name may be up to 253 characters.

The personal_info part contains the following ASCII characters.

- **Uppercase (A-Z) and lowercase (a-z) English letters.**
- **Digits (0-9).**
- **Characters ! # $ % & ' * + - / = ? ^ _ ` { | } ~**
- **Character . (period, dot or fullstop) provided that it is not the first or last character and it will not come one after the other.**

The domain name [for example com, org, net, in, us, info] part contains letters, digits, hyphens, and dots.

Example of valid email id

- **mysite@ourearth.com**
- **my.ownsite@ourearth.org**
- **mysite@you.me.net**

JavaScript code to validate an email id

```
function ValidateEmail(mail) { if
(/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/.test(myForm.e
mailAddr.value)) { return (true) } alert("You have entered an
invalid email address!") return (false) }
```

---

20)What is the requirement of debugging in JavaScript?

Ans: JavaScript Debugging

Sometimes a code may contain certain mistakes. Being a scripting language, JavaScript didn't show any error message in a browser. But these mistakes can affect the output.

The best practice to find out the error is to debug the code. The code can be debugged easily by using web browsers like Google Chrome, Mozilla Firebox.

Using debugger keyword

In debugging, generally we set breakpoints to examine each line of code step by step. There is no requirement to perform this task manually in JavaScript.

JavaScript provides debugger keyword to set the breakpoint through the code itself. The debugger stops the execution of the program at the position it is applied. Now, we can start the flow of execution manually. If an exception occurs, the execution will stop again on that particular line.

```
<script>
x = 10;
y = 15;
z = x + y;
debugger;
document.write(z);
document.write(a);
</script>
```

Output: