WAD Module-3 Part-C

1Q. Is it possible to break JavaScript Code into several lines?

Ans.

There are two ways to break JavaScript code into several lines:

- **★ We can use the newline escape character i.e '\n', if we are working on the js file or not rendering the code to the html page.**
- **★ We can use the
 tag if we are working on the html file.**

Breaking JavaScript code into multiple lines is a process where the clumsiness of code is reduced.

```
// '\n'
<!DOCTYPE html>
<html>
<head>
    <title>Javascript breaking lines of code</title>
</head>
<body>
    <script>
    console.log(
        "Airtel \n Har Ek Friend Zaroori Hotha Hai"
    );
```

```
    </script>
</body>
</html>
//<br> tag
<!DOCTYPE html>
<html>
<head>
    <title>Javascript breaking lines of code</title>
</head>
<body>
    <script>
        document.write("<br> tag");
        document.write("<br>");
        document.write("<br>");
    </script>
</body>
</html>
```

2Q. What are undeclared and undefined variables?

Ans.

- **★ Undefined: It occurs when a variable has been declared but has not been assigned with any value. Undefined is not a keyword.**

★ **Undeclared: It occurs when we try to access any variable that is not initialized or declared earlier using var or const keyword. If we use 'typeof' operator to get the value of an undeclared variable, we will face the runtime error with return value as "undefined". The scope of the undeclared variables is always global.**

3Q. Write the code for adding new elements dynamically?

Ans.

```
<!DOCTYPE html>
<head>
    <style>
        body {
            height: 100% ;
            width: 100% ;
        }
        .button {
            display: flex;
            align-items: center;
            justify-content: center;
        }
        .tasks {
            display: flex;
```

```
            justify-content: center;
            align-items: center;
        }
    </style>
</head>
<body>
    <div class="button">
        <button id="addTask">Add task</button>
    </div>
    <div class="tasks"></div>
    <script type="text/javascript">
    let task = document.getElementsByClassName("tasks");
    let addTask = document.getElementById("addTask");
    addTask.addEventListener('click', function ()
    {
        for (let i = 0; i < task.length; i++)
        {
            let newDiv = document.createElement("div");
            newDiv.setAttribute("class", "list");
            newDiv.innerText = "New Div created";
            task[i].append(newDiv);
        }
    })
```

```
    </script>
</body>
</html>
```

4Q. What are global variables? How are these variables declared?

Ans.

A JavaScript global variable is declared outside the function or declared with a window object. It can be accessed from any function.

```
<script>
var value=50;//global variable
function a(){
alert(value);
}
function b(){
alert(value);
}
</script>
```

Global Variables are the variables that can be accessed from anywhere in the program. These are the variables that are declared in the main body of the source code and outside all the functions. These variables are available to every function to access.

Global variables are declared at the start of the block(top of the program)
Var keyword is used to declare variables globally.
Global variables can be accessed from any part of the program.

5Q. What is the working of timers in JavaScript?
Ans.
In JavaScript, a timer is created to execute a task or any function at a particular time. Basically, the timer is used to delay the execution of the program or to execute the JavaScript code in a regular time interval. With the help of a timer, we can delay the execution of the code. So, the code does not complete its execution at the same time when an event triggers or page loads.
The best example of the timer is advertisement banners on websites, which change after every 2-3 seconds. These advertising banners are changed at a regular interval on websites like Amazon. We set a time interval to change them.
JavaScript offers two timer functions setInterval() and setTimeout(), which helps to delay the execution of code and also allows to perform one or more operations repeatedly.

The setTimeout() function helps the users to delay the execution of code. The setTimeout() method accepts two parameters in which one is a user-defined function, and another is the time parameter to delay the execution. The time parameter holds the time in milliseconds (1 second = 1000 milliseconds), which is optional to pass.
The setInterval method is a bit similar to the setTimeout() function. It executes the specified function repeatedly after a time interval. Or we can simply say that a function is executed repeatedly after a specific amount of time provided by the user in this function. For example - Display updated time every five seconds.

6Q. What are all the looping structures in JavaScript?
Ans.
JavaScript supports different kinds of loops:

- **★ for - loops through a block of code a number of times**
- **★ for/in : loops through the properties of an object**
- **★ for/of : loops through the values of an iterable object**
- **★ while : loops through a block of code while a specified condition is true**
- **★ do/while : also loops through a block of code while a specified condition is true**

7Q. How can you convert the string of any base to an integer in JavaScript?

Ans.

In JavaScript parseInt() function (or a method) is used to convert the passed in string parameter or value to an integer value itself. This function returns an integer of base which is specified in the second argument of parseInt() function. parseInt() function returns Nan(not a number) when the string doesn't contain a number.

```
<script>
    function convertStoI() {
        var a = "100";
        var b = parseInt(a);
        document.write("Integer value is : " + b);
        var d = parseInt("3 11 43");
        document.write("</br>");
        document.write("Integer value is : " + d);
    }
convertStoI();
</script>
```

8Q. What is an undefined value in JavaScript?

Ans.

The undefined is a primitive type in JavaScript. The undefined type has exactly one value that is undefined.

```
let counter;
console.log(counter); // undefined
let counter = 0;
console.log(counter); // 0
let x;
if (x === "undefined") {
  text = "x is undefined";
}
else {
  text = "x is defined";
}
```

9Q. How can a page be forced to load another page in JavaScript?

Ans.

We can use the “window.location” property inside the script tag to forcefully load another page in Javascript. It is a reference to a location object that represents the current location of the document. We can change the URL of a window by accessing it.

```
<script>
   window.location = <Path / URL>
```

</script>

Below is the step by step implementation:

Step 1: Create a file named index.html. Add a heading and two buttons to it. One button forcefully loads a page with a live URL and the other button loads a local HTML page. In the <script> tag we have two functions, one loads the URL's home page, and the second loads a local HTML page using window.location property.

Step 2: Create a file named newPage.html. This is the local HTML page that would be loaded by Javascript.

10Q. What is the difference between an alert box and a confirmation box?

Ans.

Alert box

1. Alert box is used if we want the information to come through to the user.

2. You need to click 'OK' to proceed when an alert box pops up.

3. window.alert("Hogwarts");

4. It always returns true. We always need to click on 'OK' to proceed further.

5. The alert box takes the focus away from the current window and forces the browser to read the message.

Confirmation box

1. Confirm box is used if we want the user to verify or accept something.

2. We need to click either “OK” or “Cancel” to proceed when a confirmation box pops up.

3. window.confirm(“Hogwarts”);

4. It return true if we click “OK”

5. It returns false if we don't click on “OK” (“Cancel”)

11Q. What are JavaScript Cookies?

Ans.

A cookie is an amount of information that persists between a server-side and a client-side. A web browser stores this information at the time of browsing.

A cookie contains the information as a string generally in the form of a name-value pair separated by semicolons. It maintains the state of a user and remembers the user's information among all the web pages.

12Q. What a pop() method in JavaScript is?

Ans.

The pop() method removes (pops) the last element of an array.

The pop() method changes the original array.

The pop() method returns the removed element.

```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Arrays</h1>
<h2>The pop() Method</h2>
<p>pop() returns the element it removed:</p>
<p id="demo"></p>
<script>
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let removed = fruits.pop();
document.getElementById("demo").innerHTML = removed;
</script>
</body>
</html>
```

13Q. Does JavaScript have concept level scope?

Ans.

No, JavaScript does not have block level scope, all the variables declared inside a function possess the same level of scope unlike c,c++,java.

14Q. What are the disadvantages of using innerHTML in JavaScript?

Ans.

Disadvantages of using innerHTML property in JavaScript:

- ★ **<u>The use of innerHTML is very slow:</u> The process of using innerHTML is much slower as its contents are slowly built, also already parsed contents and elements are also re-parsed which takes time.**
- ★ **<u>Preserves event handlers attached to any DOM elements:</u> The event handlers do not get attached to the new elements created by setting innerHTML automatically. To do so one has to keep track of the event handlers and attach it to new elements manually. This may cause a memory leak on some browsers.**
- ★ **<u>Content is replaced everywhere:</u> Either you add, append, delete or modify contents on a webpage using innerHTML, all contents is replaced, also all the DOM nodes inside that element are reparsed and recreated.**
- ★ **<u>Appending to innerHTML is not supported:</u> Usually, += is used for appending in JavaScript. But on appending to an Html tag using innerHTML, the whole tag is re-parsed.**

15Q. How can generic objects be created?

Ans.

Generics means parameterized types. The idea is to allow type (Integer, String, ... etc., and user-defined types) to be a parameter to methods, classes, and interfaces. Using Generics, it is possible to create classes that work with different data types. An entity such as class, interface, or method that operates on a parameterized type is a generic entity.

Since js is not a strictly typed language , so here we don't need to write generic objects. We use generic Objects in TypeScript or Java etc.

Suppose you have written a function that return the value passed to it as an argument like

function show(val){

return val;}

The returned value of the function must match the data-type of the function show() .

16Q. Which keywords are used to handle exceptions ?

Ans.

JavaScript handles exceptions through the "try..catch...finally" statement.

```
try {
  // Attempt to execute this code
} catch (exception) {
  // This code handles exceptions
} finally {
  // This code always gets executed
}
```

A throw statement is used to raise an exception. It means when an abnormal condition occurs, an exception is thrown using throw.

17Q. What are the different types of errors in JavaScript?

Ans.

Types of Errors

While coding, there can be three types of errors in the code:

- ★ **<u>Syntax Error:</u> When a user makes a mistake in the predefined syntax of a programming language, a syntax error may appear.**
- ★ **<u>Runtime Error:</u> When an error occurs during the execution of the program, such an error is known as Runtime error. The codes which create runtime errors are known as Exceptions. Thus, exception handlers are used for handling runtime errors.**

★ **Logical Error:** **An error which occurs when there is any logical mistake in the program that may not produce the desired output, and may terminate abnormally. Such an error is known as Logical error.**

18Q. What is the way to get the status of a CheckBox?

Ans.

A checkbox has two states: checked and unchecked.

To get the state of a checkbox, you follow these steps:

★ **First, select the checkbox using a DOM method such as getElementById() or querySelector().**

★ **Then, access the checked property of the checkbox element. If its checked property is true, then the checkbox is checked; otherwise, it is not.**

```
function check() {
   document.getElementById("myCheck").checked = true;
}

function uncheck() {
   document.getElementById("myCheck").checked = false;
}
```

19Q. How can the OS of the client machine be detected?

Ans.

To detect the operating system on the client machine, one can simply use navigator.appVersion or navigator.userAgent property.

The Navigator appVersion property is a read-only property and it returns a string which represents the version information of the browser.

Syntax : navigator.appVersion

```
<!DOCTYPE html>
<html>
<head>
    <title>
        How to detect operating system on the
        client machine using JavaScript ?
    </title>
</head>
<body style="text-align:center;">
    <h1 style="color:dark blue;">WizardingWorld</h1>
    <button ondblclick="operatingSytem()">
        Return Operating System Name
    </button>
```

```
<p id="OS"></p>
<script>
function operatingSytem() {
    var OSName="Unknown OS";
    if (navigator.appVersion.indexOf("Win")!=-1)
        OSName="Windows";
    if (navigator.appVersion.indexOf("Mac")!=-1)
        OSName="MacOS";
    if (navigator.appVersion.indexOf("X11")!=-1)
        OSName="UNIX";
    if (navigator.appVersion.indexOf("Linux")!=-1)
        OSName="Linux";
    document.getElementById("OS").innerHTML = OSName;
    }
</script>
</body>
</html>
```

20Q. What is a window.onload and onDocumentReady?

Ans.

“window.onload” will execute code when the browser has loaded the DOM tree and all other resources like images, objects, etc.

onDocumentReady executes when the DOM tree is built, without waiting for other resources to load. This allows executing the code against the DOM faster with onDocumentReady.
Another difference is that window.onload is not cross-browser compatible while using something like jQuery's document.ready() will work nicely on all browsers.