Shree Ganeshay Namah

Total-Questions: 500 JavaScript

Theory:-

1. **What is the use of isNaN function?**

Ans: isNan function returns true if the argument is not a number; otherwise, it is false.

2. **What is negative Infinity?**

Negative Infinity is a number in JavaScript which can be derived by dividing negative number by zero.

3. **What are undeclared and undefined variables?**

Undeclared variables are those that do not exist in a program and are not declared. If the program tries to read the value of an undeclared variable, then a runtime error is encountered.

Undefined variables are those that are declared in the program but have not been given any value. If the program tries to read the value of an undefined variable, an undefined value is returned.

4.Write the code for adding new elements dynamically?

<html>

<head>

<title>t1</title>

<script type="text/javascript">

function addNode ( {

var newP = document. createElement("p");

var textNode = document.createTextNode(" This is a new text node");

newP.appendChild(textNode);

document.getElementById("firstP").appendChild(newP);

}

</script> </head>

<body> <p id="firstP">firstP<p>

</body>

</html>

5. **What are global variables? How are these variable declared?**

Global variables are available throughout the length of the code so that it has no scope. The var keyword is used to declare a local variable or object. If the var keyword is omitted, a global variable is declared.

Example:

// Declare a global:

globalVariable = "Test";

6. **What is 'this' keyword in JavaScript?**

'This' keyword refers to the object from where it was called.

7. **What is the working of timers in JavaScript?**

Timers are used to execute a piece of code at a set time or repeat the code in a given interval. This is done by using the functions **setTimeout, setInterval,**and**clearInterval**.

The **setTimeout(function, delay)** function is used to start a timer that calls a particular function after the mentioned delay. The **setInterval(function, delay)** function repeatedly executes the given function in the mentioned delay and only halts when canceled. The **clearInterval(id)** function instructs the timer to stop.

Timers are operated within a single thread, and thus events might queue up, waiting to be executed.

8. **What is the difference between ViewState and SessionState?**

* 'ViewState' is specific to a page in a session.
* 'SessionState' is specific to user-specific data that can be accessed across all web application pages.

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

The parseInt() function is used to convert numbers between different bases. parseInt() takes the string to be converted as its first parameter. The second parameter is the base of the given string.

To convert 4F (or base 16) to integer, the code used will be -

parseInt ("4F", 16);

**10. Difference between "==" and "==="?**

"==" checks only for equality in value, whereas "===" is a stricter equality test and returns false if either the value or the type of the two variables are different.

**11. What do you mean by NULL in Javascript?**

The NULL value is used to represent no value or no object. It implies no object or null string, no valid boolean value, no number, and no array object.

**12. What is an undefined value in JavaScript?**

Undefined value means the

* Variable used in the code doesn't exist
* Variable is not assigned to any value
* Property does not exist.

**13. What are all the types of Pop up boxes available in JavaScript?**

* Alert
* Confirm and
* Prompt

**14. What is the use of Void (0)?**

Void(0) is used to prevent the page from refreshing, and parameter "zero" is passed while calling.

Void(0) is used to call another method without refreshing the page.

**15. How can a page be forced to load another page in JavaScript?**

The following code has to be inserted to achieve the desired effect:

<script language="JavaScript" type="text/javascript" >

<!-- location. href="http://newhost/newpath/newfile.html"; //--></script>

**16. What are JavaScript Cookies?**

Cookies are the small test files stored in a computer, and they get created when the user visits the websites, to store information that they need. Examples could be User Name details and shopping cart information from previous visits.

**17. What a pop()method in JavaScript is?**

The pop() method is similar to the shift() method, but the difference is that the Shift method works at the array's start. The pop() method takes the last element off of the given array and returns it. The array on which it is called is then altered.

Example:

var cloths = ["Shirt", "Pant", "TShirt"];

cloths.pop();

//Now cloth becomes Shirt,Pant

**18. Does JavaScript has concept level scope?**

No. JavaScript does not have concept-level scope. The variable declared inside the function has scope inside the function.

**19. What is break and continue statements?**

Break statement exits from the current loop.

Continue statement continues with next statement of the loop.

**20. What is the use of a type of operator?**

'Typeof' is an operator used to return a string description of the type of a variable.

**21. Which keywords are used to handle exceptions?**

Try… Catch---finally is used to handle exceptions in the JavaScript

Try{

Code

}

Catch(exp){

Code to throw an exception.

}

Finally{

Code runs either it finishes successfully or after catch

}

**22. Which keyword is used to print the text on the screen?**

Document. Write ("Welcome") is used to print the text–Welcome on the screen.

**23. What is the use of the blur function?**

Blur function is used to remove the focus from the specified object.

**24. What is variable typing?**

Variable typing assigns a number to a variable and then assigns a string to the same variable. An example is as follows:

i= 8;

i="john";

**25. What are the different types of errors in JavaScript?**

There are three types of errors:

* **Load time errors**: Errors that come up when loading a web page, like improper syntax errors, are known as Load time errors and generate the errors dynamically.
* **Runtime errors**: Errors that come due to misuse of the command inside the HTML language.
* **Logical Errors**: These are the errors that occur due to the bad logic performed on a function with a different operation.

**25. What is the 'Strict Mode in JavaScript, and how can it be enabled?**

* Strict Mode adds certain compulsions to JavaScript. Under the strict Mode, JavaScript shows errors for a piece of code, which did not show an error before, but might be problematic and potentially unsafe. Strict Mode also solves some mistakes that hamper the JavaScript engines from working efficiently.
* Strict mode can be enabled by adding the string literal "use strict" above the file. This can be illustrated by the given example:

function myfunction() {

"use strict;"

var v = "This is a strict mode function";

}

**26. What is the way to get the status of a Checkbox?**

The status can be acquired as follows -

alert(document.getElementById('checkbox1').checked);

If the CheckBox is checked, this alert will return TRUE.

**27. How can a value be appended to an array?**

A value can be appended to an array in the given manner -

arr[arr.length] = value;

**28. What is the difference between .call() and .apply()?**

The function .call() and .apply() are very similar in their usage except a little difference. .call() is used when the number of the function's arguments are known to the programmer, as they have to be mentioned as arguments in the call statement. On the other hand, .apply() is used when the number is not known. The function .apply() expects the argument to be an array.

The basic difference between .call() and .apply() is in the way arguments are passed to the function. Their usage can be illustrated by the given example.

var someObject = {

myProperty : 'Foo',

myMethod : function(prefix, postfix) {

alert(prefix + this.myProperty + postfix);

}

};

someObject.myMethod('<', '>'); // alerts '<Foo>'

var someOtherObject = {

myProperty : 'Bar.'

};

someObject.myMethod.call(someOtherObject, '<', '>'); // alerts '<Bar>'

someObject.myMethod.apply(someOtherObject, ['<', '>']); // alerts '<Bar>'

**29. What is event bubbling?**

JavaScript allows DOM elements to be nested inside each other. In such a case, if the handler of the child is clicked, the handler of the parent will also work as if it were clicked too.

**30. Write the point of difference between a web garden and a web farm?**

Both web-garden and web-farm are web hosting systems. The only difference is that web-garden is a setup that includes many processors in a single server. At the same time,web-farm is a larger setup that uses more than one server.

**31. How are object properties assigned?**

Assigning properties to objects is done in the same way as a value is assigned to a variable. For example, a form object's action value is assigned as 'submit' in the following manner - Document. form.action="submit"

**32. What is the method for reading and writing a file in JavaScript?**

This can be done by Using JavaScript extensions (runs from JavaScript Editor), for example, for the opening of a file -

fh = fopen(getScriptPath(), 0);

**33. What are the various functional components in JavaScript?**

The different functional components in JavaScript are-

* **First-class functions:** Functions in JavaScript are utilized as first-class objects. This usually means that these functions can be passed as arguments to other functions, returned as values from other functions, assigned to variables, or can also be stored in data structures.
* **Nested functions:** The functions, which are defined inside other functions, are called Nested functions. They are called 'every time the main function is invoked.

**34. Write about the errors shown in JavaScript?**

JavaScript gives a message as if it encounters an error. The recognized errors are -

* **Load-time errors:** The errors shown at the time of the page loading are counted under Load-time errors. The use of improper syntax encounters these errors and is thus detected while the page is getting loaded.
* **Runtime errors:** This is the error that comes up while the program is running. For example, illegal operations cause the division of a number by zero or access a non-existent area of the memory.
* **Logic errors:** It is caused by syntactically correct code, which does not fulfill the required task—for example, an infinite loop.

**35. What are Screen objects?**

Screen objects are used to read the information from the client's screen. The properties of screen objects are -

* AvailHeight: Gives the height of the client's screen
* AvailWidth: Gives the width of the client's screen
* ColorDepth: Gives the bit depth of images on the client's screen
* Height: Gives the total height of the client's screen, including the taskbar
* Width: Gives the total width of the client's screen, including the taskbar

**36. What is the unshift() method?**

This method is functional at the starting of the array, unlike the push(). It adds the desired number of elements to the top of an array. For example -

var name = [ "john" ];

name.unshift( "charlie" );

name.unshift( "joseph", "Jane" );

console.log(name);

The output is shown below:

[" joseph ,"," Jane ,", " charlie ", " john "]

**37. What is unescape() and escape() functions?**

The escape () function is responsible for coding a string to transfer the information from one computer to the other across a network.

For Example:

<script>

document.write(escape("Hello? How are you!"));

</script>

Output: Hello%3F%20How%20are%20you%21

The unescape() function is very important as it decodes the coded string.

It works in the following way. For example:

<script>

document.write(unescape("Hello%3F%20How%20are%20you%21"));

</script>

Output: Hello? How are you!

**38. What are the decodeURI() and encodeURI()?**

EncodeURl() is used to convert URL into their hex coding. And DecodeURI() is used to convert the encoded URL back to normal.

<script>

var url="my test.asp?name=ståle&car=saab";

document.write(encodeURI(url)+ "<br>");

document.write(decodeURI(url));

</script>

Output -

my%20test.asp?name=st%C3%A5le&car=saab

my test.asp?name=ståle&car=saab

**39. Why you should not use innerHTML in JavaScript?**

innerHTML content is refreshed every time and thus is slower. There is no scope for validation in innerHTML. Therefore, it is easier to insert rogue code in the document and make the web page unstable.

**40. What does the following statement declare?**

var myArray = [[[]]];

It declares a three-dimensional array.

**41. How are JavaScript and ECMA Script related?**

ECMA Script is like rules and guidelines, while Javascript is a scripting language used for web development.

**42. What is namespacing in JavaScript, and how is it used?**

Namespacing is used for grouping the desired functions, variables, etc., under a unique name. It is a name that has been attached to the desired functions, objects, and properties. This improves modularity in the coding and enables code reuse.

**43. How can JavaScript codes be hidden from old browsers that do not support JavaScript?**

For hiding JavaScript codes from old browsers:

Add "<!--" without the quotes in the code just after the <script> tag.

Add "//-->" without the quotes in the code just before the <script> tag.

Old browsers will now treat this JavaScript code as a long HTML comment. While a browser that supports JavaScript will take the "<!--" and "//-->" as one-line comments.

**44. How to use Loop in JavaScript?**

Loops are useful when you repeatedly execute the same lines of code a specific number of times or as long as a specific condition is true. Suppose you want to type a 'Hello' message 100 times on your webpage. Of course, you will have to copy and paste the same line 100 times. Instead, if you use loops, you can complete this task in just 3 or 4 lines.

**45. What are the important JavaScript Array Method explain with example?**

**JavaScript Array Methods**

The Array object has many properties and methods which help developers to handle arrays easily and efficiently. You can get the value of a property by specifying arrayname.property and the output of a method by specifying arrayname.method().

* **length property** --> If you want to know the number of elements in an array, you can use the length property.
* **prototype property** --> If you want to add new properties and methods, you can use the prototype property.
* **reverse method** --> You can reverse the order of items in an array using a reverse method.
* **sort method -->** You can sort the items in an array using sort method.
* **pop method** --> You can remove the last item of an array using a pop method.
* **shift method** --> You can remove the first item of an array using shift method.
* **push method** --> You can add a value as the last item of the array.

<html>

<head>

<title>Arrays!!!</title>

<script type="text/javascript">

var students = new Array("John", "Ann", "Aaron", "Edwin", "Elizabeth");

Array.prototype.displayItems=function(){

for (i=0;i<this.length;i++){

document.write(this[i] + "<br />");

}

}

document.write("students array<br />");

students.displayItems();

document.write("<br />The number of items in students array is " + students.length + "<br />");

document.write("<br />The SORTED students array<br />");

students.sort();

students.displayItems();

document.write("<br />The REVERSED students array<br />");

students.reverse();

students.displayItems();

document.write("<br />THE students array after REMOVING the LAST item<br />");

students.pop();

students.displayItems();

document.write("<br />THE students array after PUSH<br />");

students.push("New Stuff");

students.displayItems();

</script>

</head>

<body>

</body>

</html>

**46.** **What is OOPS Concept in JavaScript?**

Many times, variables or arrays are not sufficient to simulate real-life situations. JavaScript allows you to create objects that act like real-life objects. A student or a home can be an object that has many unique characteristics of its own. You can create properties and methods for your objects to make programming easier. If your object is a student, it will have properties like the first name, last name, id, etc., and methods like calculating rank, change address, etc. If your object is a home, it will have properties like a number of rooms, paint color, location, etc. The methods like to calculate area, change owner, etc.

How to Create an Object

You can create an object like this:

var objName = new Object();

objName.property1 = value1;

objName.property2 = value2;

objName.method1 = function()

{

line of code

}

OR

var objName= {property1:value1, property2:value2, method1: function()

{ lines of code} };

**47. What is Loop through the Properties of an Object?**

The for/in a loop is usually used to loop through the properties of an object. You can give any name for the variable, but the object's name should be the same as an already existing object you need to loop through.

**Syntax:**

for (variablename in objectname)

{

lines of code to be executed

}

**Example:**

<html>

<head>

<script type="text/javascript">

var employee={first:"John", last:"Doe", department:"Accounts"};

var details = "";

document.write("<b>Using for/in loops </b><br />");

for (var x in employee)

{

details = x + ": " + employee[x];

document.write(details + "<br />");

}

</script>

</head>

<body>

</body>

</html>

**48.** **What is JavaScript Unit Testing, and what are the challenges in JavaScript Unit Testing?**

JavaScript Unit Testing is a testing method in which JavaScript tests code written for a web page or web application module. It is combined with HTML as an inline event handler and executed in the browser to test if all functionalities work fine. These unit tests are then organized in the test suite.

Every suite contains several tests designed to be executed for a separate module. Most importantly, they don't conflict with any other module and run with fewer dependencies on each other (some critical situations may cause dependencies).

**Challenges of JavaScript Unit Testing:**

Here are important challenges of JavaScript Unit Testing:

* Many other languages support unit testing in browsers, in the stable as well as in runtime environment, but JavaScript can not
* You can understand some system actions with other languages, but this is not the case with JavaScript
* Some JavaScript are written for a web application that may have multiple dependencies.
* JavaScript is good to use in combination with HTML and CSS rather than on the web
* Difficulties with page rendering and DOM manipulation
* Sometimes you find an error message on your screen regarding 'Unable to load example.js' or any other JavaScript error regarding version control. These vulnerabilities come under Unit Testing JavaScript

**Solutions of JavaScript Unit Testing:**

To avoid such issues, what you can do is;

* Do not use global variables.
* Do not manipulate predefined objects.
* Design core functionalities based on the library.
* Try to create small pieces of functionalities with lesser dependencies.

**49.** **How to use DOM and Events?**

Using DOM, JavaScript can perform multiple tasks. It can create new elements and attributes, change the existing elements and attributes and even remove existing elements and attributes. JavaScript can also react to existing events and create new events in the page.

1. getElementById, innerHTML Example
2. getElementById: To access elements and attributes whose id is set.
3. innerHTML: To access the content of an element.

<html>

<head>

<title>DOM!!!</title>

</head>

<body>

<h3 id="one">Welcome</h3>

<p>This is the welcome message.</p>

<h3>Technology</h3>

<p>This is the technology section.</p>

<script type="text/javascript">

var text = document.getElementById("one").innerHTML;

alert("The first heading is " + text);

</script>

</body>

</html>

### 2.getElementsByTagName Example

getElementsByTagName: To access elements and attributes using tag name. This method will return an array of all the items with the same tag name.

<html>

<head>

<title>DOM!!!</title>

</head>

<body>

<h3>Welcome</h3>

<p>This is the welcome message.</p>

<h3>Technology</h3>

<p id="second">This is the technology section.</p>

<script type="text/javascript">

var paragraphs = document.getElementsByTagName("p");

alert("Content in the second paragraph is " + paragraphs[1].innerHTML);

document.getElementById("second").innerHTML = "The orginal message is changed.";

</script>

</body>

</html>

### Event handler Example

1. createElement: To create new element
2. removeChild: Remove an element
3. you can add an **event handler** to a particular element like this

document.getElementById(id).onclick=function()

{

lines of code to be executed

}

OR

document.getElementById(id).addEventListener("click", functionname)

**Example:**

<html>

<head>

<title>DOM!!!</title>

</head>

<body>

<input type="button" id="btnClick" value="Click Me!!" />

<script type="text/javascript">

document.getElementById("btnClick").addEventListener("click", clicked);

function clicked()

{

alert("You clicked me!!!");

}

</script>

</body>

</html>

**50. What is External JavaScript?**

You plan to display the current date and time on all your web pages. Suppose you wrote the code and copied it in to all your web pages (say 100). But later, you want to change the format in which the date or time is displayed. In this case, you will have to make changes to all the 100 web pages. This will be a very time-consuming and difficult task.

So, save the JavaScript code in a new file with the extension .js. Then, add a line of code in all your web pages to point to your .js file like this:

<script type="text/javascript," src="/currentdetails.js,">

Note**:** It is assumed that the .js file and all your web pages are in the same folder. If the external.js file is in a different folder, you need to specify your file's full path in the src attribute.

**Example:**

var currentDate = new Date();

var day = currentDate.getDate();

Var month = currentDate.getMonth() + 1;

var monthName;

var hours = currentDate.getHours();

var mins = currentDate.getMinutes();

var secs = currentDate.getSeconds();

var strToAppend;

It (hours >12 )

{

hours1 = "0" + (hours - 12);

strToAppend = "PM";

}

else if (hours <12)

{

hours1 = "0" + hours;

strToAppend = "AM";

}

else

{

hours1 = hours;

strToAppend = "PM";

}

if(mins<10)

mins = "0" + mins;

if (secs<10)

secs = "0" + secs;

switch (month)

{

case 1:

monthName = "January";

break;

case 2:

monthName = "February";

break;

case 3:

monthName = "March";

break;

case 4:

monthName = "April";

break;

case 5:

monthName = "May";

break;

case 6:

monthName = "June";

break;

case 7:

monthName = "July";

break;

case 8:

monthName = "August";

break;

case 9:

monthName = "September";

break;

case 10:

monthName = "October";

break;

case 11:

monthName = "November";

break;

case 12:

monthName = "December";

break;

}

var year = currentDate.getFullYear();

var myString;

myString = "Today is " + day + " - " + monthName + " - " + year + ".<br />Current time is " + hours1 + ":" + mins + ":" + secs + " " + strToAppend + ".";

document.write(myString);

**51.** **When to Use Internal and External JavaScript Code?**

Suppose you have only a few lines of code that is specific to a particular webpage. In that case, it is better to keep your JavaScript code internal within your HTML document.

On the other hand, if your JavaScript code is used in many web pages, you should consider keeping your code in a separate file. If you wish to make some changes to your code, you have to change only one file, making code maintenance easy. If your code is too long, it is better to keep it in a separate file. This helps in easy debugging.

**52. What are Cookies in JavaScript?**

A cookie is a piece of data stored on your computer to be accessed by your browser. You also might have enjoyed the benefits of cookies knowingly or unknowingly. Have you ever saved your Facebook password so that you do not have to type it every time you try to login? If yes, then you are using cookies. Cookies are saved as key/value pairs.

Javascript Set-Cookie:

You can create cookies using document. cookie property like this.

### 53 .Give an example of an Arrow function in ES6? List down its advantages.

Arrow function provides us a more accurate way of writing the [functions in JavaScript](https://www.javatpoint.com/javascript-function). They allow us to write smaller function syntax.

The context within the arrow functions is lexically or statically scoped. Arrow functions do not include any prototype property, and cannot be used with the new keyword.

You can learn more about arrow functions by clicking on this link [ES6 Arrow Function](https://www.javatpoint.com/es6-arrow-function).

**Example**

1. var myfun = () => {
2. console.log("It is an Arrow Function");
3. };
4. myfun();

**Output**

It is an Arrow Function

**Advantages of Arrow Function**

The advantages of the arrow function are listed below:

* It reduces code size.
* The return statement is optional for a single line function.
* Lexically bind the context.
* Functional braces are optional for a single-line statement.

### 54) Discuss spread operator in ES6 with an example.

The spread operator is represented by three dots (...) to obtain the list of parameters. It allows the expansion of an iterable such as array or string in places where more than zero arguments are expected.

The spread operator syntax is similar to the rest operator, but functionality is entirely opposite to it. It is also used to combine or to perform the concatenation between arrays. Let's understand it by an example.

**Example**

1. let num1 = [40,50,60];
3. let num2 = [10,20,30,...num1,70,80,90,100];
5. console.log(num2);

**Output**

[

10, 20, 30, 40, 50,

60, 70, 80, 90, 100

]

### 55) Discuss the Rest parameter in ES6 with an example.

It is introduced in ES6 that improves the ability to handle the parameters. With rest parameters, it is possible to represent indefinite parameters as an array. By using the rest parameter, we can call a function with any number of arguments.

**Example**

1. function show(...args) {
2. let sum = 0;
3. **for** (let i of args) {
4. sum += i;
5. }
6. console.log("Sum = "+sum);
7. }
9. show(10, 20, 30);

**Output**

Sum = 60

### 56. What are the template literals in ES6?

Template literals are a new feature introduced in ES6. It provides an easy way of creating multiline strings and perform string interpolation.

Template literals allow embedded expressions and also called as string literals.

Prior to ES6, template literals were referred to as **template strings**. Template literals are enclosed by the **backtick (` `) character**. Placeholders in template literals are represented by the dollar sign and the curly braces **(${expression})**. If we require to use an expression within the backticks, then we can place that expression in the **(${expression})**.

To learn more about template literals in ES6, follow this link [ES6 Template Literals.](https://www.javatpoint.com/es6-template-literals)

**Example**

1. let str1 = "Hello";
3. let str2 = "World";
5. let str = `${str1} ${str2}`;
6. console.log(str);

**Output**

Hello World

### 57) Discuss Destructuring Assignment in ES6.

Destructuring is introduced in ECMAScript 2015 or ES6 to extract data from objects and arrays into place it in to separate variables. It allows us to extract smaller fragments from objects and arrays.

To learn more about array destructuring in ES6, follow this link [ES6 Array Destructuring](https://www.javatpoint.com/es6-array-destructuring).

To learn more about object destructuring in ES6, follow this link [ES6 Object Destructuring](https://www.javatpoint.com/es6-object-destructuring).

**Example**

1. let fullname =['Alan','Rickman'];
2. let [fname,lname] = fullname;
3. console.log (fname,lname);

**Output**

Alan Rickman

### 58) How to create a class in ES6?

This keyword is used for creating the class. We can include the classes in our code either by using class expression or by class declaration. A class definition can only include **functions** and **constructors**. These components are together called as data members of the class.

Constructors in classes allocate the memory to the objects of the class. Functions in a class are responsible for performing the actions to the objects.

To learn more about classes in ES6, follow this link [ES6 Classes](https://www.javatpoint.com/es6-classes).

Let us see the syntax for creating classes.

**Syntax: In ES5**

1. var var\_name = **new** class\_name {
2. }

**Syntax: In ES6 (Using class keyword)**

1. **class** class\_name{
2. }

### 59) What do you understand by Generator function?

A generator provides us a new way to work with iterators and functions. The generator is a special kind of function that may be paused in the middle either one or many times and can be resumed later. The declaration **function\* (function keyword followed by an asterisk)** is used to define a generator function.

When the generator gets called, it does not run its code. Instead, it returns a special object, which is called a **Generator object** to manage the execution. Let us see an example of generators in ES6.

To learn more about Generators in ES6, follow this link [ES6 Generators](https://www.javatpoint.com/es6-generators).

**Example**

1. function\* gen()
2. {
3. yield 100;
4. yield;
5. yield 200;
6. }
7. // Calling the Generator Function
8. var mygen = gen();
9. console.log(mygen.next().value);
10. console.log(mygen.next().value);
11. console.log(mygen.next().value);

**Output**

100

undefined

200

### 60) What are the default parameters?

By using the default parameters, we can initialize named parameters with default values if there is no value or **undefined** is passed.

**Example**

1. var show = (a, b=200) => {
2. console.log(a + " " + b);
3. }
4. show(100);

**Output**

100 200

### 61) What do you mean by IIFE (Immediately invoked function expressions)?

IIFE is a function in JavaScript that runs as soon as it is defined. It is also called as the **Self-Executing Anonymous Function**. It includes two major parts that are as follows:

* The first part is an anonymous function that has a lexical scope (static scope), which is enclosed within the **Grouping operator ()**.
* The second part creates the IIFE by which the [JavaScript](https://www.javatpoint.com/javascript-tutorial) engine will interpret the function directly.

You can learn more about arrow functions by clicking on this link [ES6 IIFE](https://www.javatpoint.com/es6-immediately-invoked-function-expression).

**Example**

1. (function()
2. {
3. console.log("Hello World");
4. })();

**Output**

Hello World

### 62) Discuss the for...in loop.

It is similar to for loop that iterates through the properties of an object. It is useful when we require to visit the properties or keys of the object.

**Example**

1. function Mobile(model\_no){
2. **this**.Model = model\_no;
3. **this**.Color = 'White';
4. **this**.RAM = '4GB';
5. }
6. var Samsung = **new** Mobile("Galaxy");
7. **for**(var props in Samsung)
8. {
9. console.log(props+ " : " +Samsung[props]);
10. }

**Output**

Model: Galaxy

Color: White

RAM: 4GB

### 63) Discuss the for...of loop.

This loop is used for iterating the iterables (arrays, string, etc.).

**Example**

1. var fruits = ['Apple', 'Banana', 'Mango', 'Orange'];
2. **for**(let value of fruits)
3. {
4. console.log(value);
5. }

**Output**

Apple

Banana

Mango

Orange

64) Define Map.

Prior to ES6, when we require the mapping of keys and values, we often use an object. **Map object** is a new collection type, which is introduced in ES6. It holds the key-value pairs in which any type of values can be used as either keys or values.

A map object always remembers the actual insertion order of the keys. Maps are ordered, so they traverse the elements in their insertion order.

To learn more about Map in ES6, follow this link [ES6 Maps](https://www.javatpoint.com/es6-map).

65) What do you understand by Weakset?

Using weakset, it is possible to store weakly held objects in a collection. As similar to set, weakset cannot store duplicate values. Weakset cannot be iterated.

Weakset only includes **add(value), delete(value)** and **has(value)** methods of the set object.

66) What do you understand by Weakmap?

Weak maps are almost similar to maps, but the keys in weak maps must be objects. It stores each element as a key-value pair where keys are weakly referenced. Here, the keys are objects, and the values are arbitrary.

A weak map object iterates the element in their insertion order. It only includes **delete(key), get(key), has(key)** and **set(key, value)** method.s

### 67) Explain Promises in ES6.

ES6 promises are the easiest way to work with asynchronous programming in JavaScript. Asynchronous programming includes running of processes individually from the main thread, and it notifies the main thread when it gets complete.

Prior to ES6, there is the use of **Callbacks** for performing asynchronous programming. Promises are used to overcome the problem of **Callback hell**.

To learn more about promises, follow this link: [ES6 Promises](https://www.javatpoint.com/es6-promises).

### 68) What are the states of promises in ES6?

Promises have mainly three states that are as follows:

* **Pending:** It is the initial state of every promise. It represents that the result has not been computed yet.
* **Fulfilled:** It represents the completion of an operation.
* **Rejected:** It represents the failure that occurs during computation.

Once the promise is fulfilled or rejected, then it will be immutable. The **Promise()** constructor takes two arguments that are **rejected** function and a **resolve** function. Based on the asynchronous operation, it returns either the first argument or the second argument.

### 69) What do you understand by Callback and Callback hell in JavaScript?

**Callback:** It is used to handle the execution of function after the completion of the execution of another function. A callback would be helpful in working with events. In the callback, a function can be passed as an argument to another function. It is a great way when we are dealing with basic cases such as minimal asynchronous operations.

**Callback hell:** When we develop a web application that includes a lot of code, then working with callback is messy. This excessive Callback nesting is often referred to as **Callback hell**.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 69) What is ES6 or ECMAScript 2015? ES6 was released in June 2015, which is stated as the sixth edition of the language. Initially, it was named **ES6** and later renamed to ECMAScript 2015. This edition includes several new features that are modules, iterators, class, arrow functions, for...of loop, promises, and many more. Brendan Eich developed it. 70) Define ECMAScript. It is the specification that is defined in the ECMA-262 standard to create a general-purpose scripting language. 71) What are the new features introduced in ES6? The new features that are introduced in ES6 are listed as follows:   * Let and const keywords. * Default Parameters. * Arrow functions. * Template Literals. * Object Literals. * Rest and spread operators. * Destructuring assignment. * Modules, Classes, Generators, and iterators. * Promises, and many more.  72) Define let and const keywords. **let:** The variables declared using **let** keyword will be mutable, i.e., the values of the variable can be changed. It is similar to **var** keyword except that it provides block scoping.  **const:** The variables declared using **the const** keyword are immutable and block-scoped. The value of the variables cannot be changed or re-assigned if they are declared by using **the const** keyword. 73) What is the arrow function, and how to create it? Arrow functions are introduced in [ES6](https://www.javatpoint.com/es6). Arrow functions are the shorthand notation to write [ES6 functions](https://www.javatpoint.com/es6-functions). The definition of the arrow function consists of parameters, followed by an arrow (=>) and the body of the function.  An Arrow function is also called as **'fat arrow'** function. We cannot use them as constructors.  **Syntax**   1. **const** functionName = (arg1, arg2, ...) => { 2. //body of the function 3. }  74) Give an example of an Arrow function in ES6? List down its advantages. Arrow function provides us a more accurate way of writing the [functions in JavaScript](https://www.javatpoint.com/javascript-function). They allow us to write smaller function syntax.  The context within the arrow functions is lexically or statically scoped. Arrow functions do not include any prototype property, and cannot be used with the new keyword.  You can learn more about arrow functions by clicking on this link [ES6 Arrow Function](https://www.javatpoint.com/es6-arrow-function).  **Example**   1. var myfun = () => { 2. console.log("It is an Arrow Function"); 3. }; 4. myfun();   **Output**  It is an Arrow Function  **Advantages of Arrow Function**  The advantages of the arrow function are listed below:   * It reduces code size. * The return statement is optional for a single line function. * Lexically bind the context. * Functional braces are optional for a single-line statement.  75) Discuss spread operator in ES6 with an example. The spread operator is represented by three dots (...) to obtain the list of parameters. It allows the expansion of an iterable such as array or string in places where more than zero arguments are expected.  The spread operator syntax is similar to the rest operator, but functionality is entirely opposite to it. It is also used to combine or to perform the concatenation between arrays. Let's understand it by an example.  **Example**   1. let num1 = [40,50,60]; 3. let num2 = [10,20,30,...num1,70,80,90,100]; 5. console.log(num2);   **Output**  [  10, 20, 30, 40, 50,  60, 70, 80, 90, 100  ] 76) Discuss the Rest parameter in ES6 with an example. It is introduced in ES6 that improves the ability to handle the parameters. With rest parameters, it is possible to represent indefinite parameters as an array. By using the rest parameter, we can call a function with any number of arguments.  **Example**   1. function show(...args) { 2. let sum = 0; 3. **for** (let i of args) { 4. sum += i; 5. } 6. console.log("Sum = "+sum); 7. } 9. show(10, 20, 30);   **Output**  Sum = 60 77) What are the template literals in ES6? Template literals are a new feature introduced in ES6. It provides an easy way of creating multiline strings and perform string interpolation.  Template literals allow embedded expressions and also called as string literals.  Prior to ES6, template literals were referred to as **template strings**. Template literals are enclosed by the **backtick (` `) character**. Placeholders in template literals are represented by the dollar sign and the curly braces **(${expression})**. If we require to use an expression within the backticks, then we can place that expression in the **(${expression})**.  To learn more about template literals in ES6, follow this link [ES6 Template Literals.](https://www.javatpoint.com/es6-template-literals)  **Example**   1. let str1 = "Hello"; 3. let str2 = "World"; 5. let str = `${str1} ${str2}`; 6. console.log(str);   **Output**  Hello World 78) Discuss Destructuring Assignment in ES6. Destructuring is introduced in ECMAScript 2015 or ES6 to extract data from objects and arrays into separate variables. It allows us to extract smaller fragments from objects and arrays.  To learn more about array destructuring in ES6, follow this link [ES6 Array Destructuring](https://www.javatpoint.com/es6-array-destructuring).  To learn more about object destructuring in ES6, follow this link [ES6 Object Destructuring](https://www.javatpoint.com/es6-object-destructuring).  **Example**   1. let fullname =['Alan','Rickman']; 2. let [fname,lname] = fullname; 3. console.log (fname,lname);   **Output**  Alan Rickman 79) How to create a class in ES6? This keyword is used for creating the class. We can include the classes in our code either by using class expression or by class declaration. A class definition can only include **functions** and **constructors**. These components are together called as data members of the class.  Constructors in classes allocate the memory to the objects of the class. Functions in a class are responsible for performing the actions to the objects.  To learn more about classes in ES6, follow this link [ES6 Classes](https://www.javatpoint.com/es6-classes).  Let us see the syntax for creating classes.  **Syntax: In ES5**   1. var var\_name = **new** class\_name { 2. }   **Syntax: In ES6 (Using class keyword)**   1. **class** class\_name{ 2. }  80) What do you understand by Generator function? A generator provides us a new way to work with iterators and functions. The generator is a special kind of function that may be paused in the middle either one or many times and can be resumed later. The declaration **function\* (function keyword followed by an asterisk)** is used to define a generator function.  When the generator gets called, it does not run its code. Instead, it returns a special object, which is called a **Generator object** to manage the execution. Let us see an example of generators in ES6.  To learn more about Generators in ES6, follow this link [ES6 Generators](https://www.javatpoint.com/es6-generators).  **Example**   1. function\* gen() 2. { 3. yield 100; 4. yield; 5. yield 200; 6. } 7. // Calling the Generator Function 8. var mygen = gen(); 9. console.log(mygen.next().value); 10. console.log(mygen.next().value); 11. console.log(mygen.next().value);   **Output**  100  undefined  200 81) What are the default parameters? By using the default parameters, we can initialize named parameters with default values if there is no value or **undefined** is passed.  **Example**   1. var show = (a, b=200) => { 2. console.log(a + " " + b); 3. } 4. show(100);   **Output**  100 200 82) What do you mean by IIFE (Immediately invoked function expressions)? IIFE is a function in JavaScript that runs as soon as it is defined. It is also called as the **Self-Executing Anonymous Function**. It includes two major parts that are as follows:   * The first part is an anonymous function that has a lexical scope (static scope), which is enclosed within the **Grouping operator ()**. * The second part creates the IIFE by which the [JavaScript](https://www.javatpoint.com/javascript-tutorial) engine will interpret the function directly.   You can learn more about arrow functions by clicking on this link [ES6 IIFE](https://www.javatpoint.com/es6-immediately-invoked-function-expression).  **Example**   1. (function() 2. { 3. console.log("Hello World"); 4. })();   **Output**  Hello World 83) Discuss the for...in loop. It is similar to for loop that iterates through the properties of an object. It is useful when we require to visit the properties or keys of the object.  **Example**   1. function Mobile(model\_no){ 2. **this**.Model = model\_no; 3. **this**.Color = 'White'; 4. **this**.RAM = '4GB'; 5. } 6. var Samsung = **new** Mobile("Galaxy"); 7. **for**(var props in Samsung) 8. { 9. console.log(props+ " : " +Samsung[props]); 10. }   **Output**  Model: Galaxy  Color: White  RAM: 4GB 84) Discuss the for...of loop. This loop is used for iterating the iterables (arrays, string, etc.).  **Example**   1. var fruits = ['Apple', 'Banana', 'Mango', 'Orange']; 2. **for**(let value of fruits) 3. { 4. console.log(value); 5. }   **Output**  Apple  Banana  Mango  Orange 85) Define set. A set is a data structure that allows us to create a collection of unique values. It is a collection of values that are similar to arrays, but it does not include any duplicates. It supports both object references and primitive values.  To learn more about Sets in ES6, follow this link [ES6 Sets](https://www.javatpoint.com/es6-set).  **Example**   1. let colors = **new** Set(['Green', 'Red', 'Orange', 'Yellow', 'Red']); 2. console.log(colors);   **Output**  Set { 'Green', 'Red', 'Orange', 'Yellow' } 86) Define Map. Prior to ES6, when we require the mapping of keys and values, we often use an object. **Map object** is a new collection type, which is introduced in ES6. It holds the key-value pairs in which any type of values can be used as either keys or values.  A map object always remembers the actual insertion order of the keys. Maps are ordered, so they traverse the elements in their insertion order.  To learn more about Map in ES6, follow this link [ES6 Maps](https://www.javatpoint.com/es6-map). 87) What do you understand by Weakset? Using weakset, it is possible to store weakly held objects in a collection. As similar to set, weakset cannot store duplicate values. Weakset cannot be iterated.  Weakset only includes **add(value), delete(value)** and **has(value)** methods of the set object. 88) What do you understand by Weakmap? Weak maps are almost similar to maps, but the keys in weak maps must be objects. It stores each element as a key-value pair where keys are weakly referenced. Here, the keys are objects, and the values are arbitrary.  A weak map object iterates the element in their insertion order. It only includes **delete(key), get(key), has(key)** and **set(key, value)** method. 89) Explain Promises in ES6. ES6 promises are the easiest way to work with asynchronous programming in JavaScript. Asynchronous programming includes running of processes individually from the main thread, and it notifies the main thread when it gets complete.  Prior to ES6, there is the use of **Callbacks** for performing asynchronous programming. Promises are used to overcome the problem of **Callback hell**.  To learn more about promises, follow this link: [ES6 Promises](https://www.javatpoint.com/es6-promises). 90) What are the states of promises in ES6? Promises have mainly three states that are as follows:   * **Pending:** It is the initial state of every promise. It represents that the result has not been computed yet. * **Fulfilled:** It represents the completion of an operation. * **Rejected:** It represents the failure that occurs during computation.   Once the promise is fulfilled or rejected, then it will be immutable. The **Promise()** constructor takes two arguments that are **rejected** function and a **resolve** function. Based on the asynchronous operation, it returns either the first argument or the second argument. 91) What do you understand by Callback and Callback hell in JavaScript? **Callback:** It is used to handle the execution of function after the completion of the execution of another function. A callback would be helpful in working with events. In the callback, a function can be passed as an argument to another function. It is a great way when we are dealing with basic cases such as minimal asynchronous operations.  **Callback hell:** When we develop a web application that includes a lot of code, then working with callback is messy. This excessive Callback nesting is often referred to as **Callback hell**. 92) List the comparisons between ES5 and ES6. ES5 and ES6 are similar in their nature, but there are some differences between them. The comparison between ES5 and ES6 are tabulated as follows:   |  |  |  | | --- | --- | --- | | **Based on** | **ES5** | **ES6** | | **Definition** | ES5 is the fifth edition of the ECMAScript (a trademarked scripting language specification defined by ECMA International) | ES6 is the sixth edition of the ECMAScript (a trademarked scripting language specification defined by ECMA International). | | **Release** | It was introduced in 2009. | It was introduced in 2015. | | **Data-types** | ES5 supports primitive data types that are **string, boolean, number, null,** and **undefined**. | In ES6, there are some additions to JavaScript data types. It introduced a new primitive data type **'symbol'** for supporting unique values. | | **Defining Variables** | In ES5, we could only define the variables by using the **var** keyword. | In ES6, there are two new ways to define variables that are **let** and **const**. | | **Performance** | As ES5 is prior to ES6, there is a non-presence of some features, so it has a lower performance than ES6. | Because of new features and the shorthand storage implementation ES6 has a higher performance than ES5. | | **Support** | A wide range of communities supports it. | It also has a lot of community support, but it is lesser than ES5. | | **Object Manipulation** | ES5 is time-consuming than ES6. | Due to destructuring and speed operators, object manipulation can be processed more smoothly in ES6. | | **Arrow Functions** | In ES5, both **function** and **return** keywords are used to define a function. | An arrow function is a new feature introduced in ES6 by which we don't require the **function** keyword to define the function. | | **Loops** | In ES5, there is a use of **for** loop to iterate over elements. | ES6 introduced the concept of **for...of** loop to perform an iteration over the values of the iterable objects. |   To learn more about the difference between ES5 and ES6, follow this link: [ES5 v/s ES6](https://www.javatpoint.com/es5-vs-es6) 92) Define Modules in JavaScript. Modules are the piece of JavaScript code written in a file. By using Modules, it is easy to maintain the code, debug the code, and reuse the code. Each module is a piece of code that gets executed once it is loaded. 93) What do you understand by the term Hoisting in JavaScript? It is a JavaScript's default behavior, which is used to move all the declarations at the top of the scope before the execution of code. It can be applied to functions as well as on variables. It allows the JavaScript to use the component before its declaration. It does not apply to scripts that run in strict mode. 94) List the new Array methods introduced in ES6? There are many array methods available in ES6, which are listed below:   * Array.of() * Array.from() * Array.prototype.copyWithin() * Array.prototype.find() * Array.prototype.findIndex() * Array.prototype.entries() * Array.prototype.keys() * Array.prototype.values() * Array.prototype.fill()   To learn more about the above array methods, follow this link: [ES6 Array methods](https://www.javatpoint.com/es6-array-methods). 95) What are the new String methods introduced in ES6? There are four string methods introduced in ES6 that are listed as follows:   * string.startsWith() * string.endsWith() * string.includes() * string.repeat()   To learn more about the strings, follow this link: [ES6 Strings.](https://www.javatpoint.com/es6-strings) 96) Define Babel. Babel is one of the popular transpilers of JavaScript. It is mainly used for converting the ES6 plus code into the backward-compatible version of JavaScript that can be run by previous JavaScript engines. 97) Define Webpack. It is an open-source JavaScript module bundler that takes modules with dependencies. It allows us to run an environment that hosts Babel. |

98. Event loop : The Event Loop has one simple job — to monitor the Call Stack and the Callback Queue. If the Call Stack is empty, the Event Loop will take the first event from the queue and will push it to the Call Stack, which effectively runs it. Such an iteration is called a tick in the Event Loop

99. spread rest : The **spread operator** allows us to spread the value of an array (or any iterable) across zero or more arguments in a function or elements in an array (or any iterable). The **rest parameter** allows us to pass an indefinite number of parameters to a function and access them in an array.

100. Event bubbling: - **Event bubbling** directs an **event** to its intended target, it works like this: A button is clicked and the **event** is directed to the button. If an **event** handler is set for that object, the **event** is triggered. If no **event** handler is set for that object, the **event bubbles** up (like a **bubble** in water) to the objects parent.

4.lifecycle method

5.class function which one best

6.was class in es5 no.

7.event propagation

8.redux.

9.Reg ex

Time –Table :

Morning: 6:00 –7:00 :- Wake Up ,fresh, Asana

Day : 7:00 – 9:30 :- Study.

Day : 9:30-10:30 :- Bath Pooja

Day : 10:30 -8:30 :-Job.

Night : 9:00 to 11:30 :- Study

Night : 11:30-6:00 :- Sleep

1 week roaster : 22 july to 31st july

//Javascript : - 7:00 to 9:30

Unit 9 : String Unit -10,11 Full.

//React : 9:00 to 11:30

Unit -3, 4, 5,6 full