JavaScript Interview Questions & Answers

ES6

JS

By Gowtham K

Microsoft MVP and Sr. Software Engineer

DotNetTricks

JavaScript/ES6 Interview Questions & Answers

Release History

- Initial Release 1.0 - 7th Feb 2018

About Dot Net Tricks

Dot Net Tricks is founded by Shailendra Chauhan (Microsoft MVP), in Jan 2010. Dot Net Tricks came into existence in form of a blog post over various technologies including .NET, C#, SQL Server, ASP.NET, ASP.NET MVC, JavaScript, Angular, Node.js and Visual Studio etc.

The company which is currently registered by a name of Dot Net Tricks Innovation Pvt. Ltd. came into the shape in 2015. Dot Net Tricks website has an average footfall on the tune of 300k+ per month. The site has become a cornerstone when it comes to getting skilled-up on .NET technologies and we want to gain the same level of trust in other technologies. This is what we are striving for.

We have a very large number of trainees who have received training from our platforms and immediately got placement in some of the reputed firms testifying our claims of providing quality training. The website offers you a variety of free study material in form of articles.

Dot Net Tricks Training Solutions

Dot Net Tricks provide you training in traditional as well as new age technologies, via various formats.

Master Courses (Instructor-led)

For a beginner who needs regular guidance, we have a fully packed Master Courses. They are almost equal to semester courses taught in engineering colleges when it comes to length, breadth of content delivery, the only difference instead of 5-6 months, they take approx. 16-weekend classes (2 months).

The detail about Master courses can be found here: https://www.dotnettricks.com/instructor-led-courses

Hands-On Learning (Learning to code)

Hands-On Learning courses give you the confidence to code and equally helpful to work in real-life scenarios. This course is composed of hands-on exercise using IDE or cloud labs so that you can practice each and everything by yourself. You can learn to code at your own pace, time and place.

The detail about Hands-On Learning courses can be found here: https://www.scholarhat.com

Skill Bootcamps (Instructor-led)

Professionals who don't have two months' time and want to get skilled up in least possible time due to some new project that their company has to take in very near future, we have designed Skill Bootcamps Concept, where you will get trained on consecutive days in a fast-paced manner, where our full focus is going to be on hands-on delivery of technological exercises.

The detail about Skill Bootcamps can be found here: https://www.dotnettricks.com/skill-bootcamp

Self-paced Courses (Video Courses)

Self-paced courses give you the liberty to study at your own pace, time and place. We understand everyone has their own comfort zone, some of you can afford to dedicate 2 hours a day, some of you not. Keeping this thing in mind, we created these self-paced courses. While creating these courses we have ensured that quality of courses doesn't get compromise at any parameter, and they also will be able to produce the same results as our other course formats, given the fact you will be able to put your own honest effort.

The detail about Self-paced courses can be found here: https://www.dotnettricks.com/self-paced-courses

Corporate Training (Online and Classroom)

Dot Net Tricks having a pool of mentors who help the corporate to enhance their employment skills as per changing technology landscape. Dot Net Tricks offers customized training programs for new hires and experienced employees through online and classroom mode. As a trusted and resourceful training partner, Dot Net Tricks helps the corporate to achieve success with its industry-leading instructional design and customer training initiatives.

The detail about Corporate Training can be found here: https://www.dotnettricks.com/corporate-training

Learning Platforms

We have very robust technology platforms to answer the needs of all our trainees, no matter in which program they have enrolled. We offer two self-intuitive Learning Management Systems (LMS), which help our learners to learn code by doing and evaluates their learning.

We offer the following two Learning Platforms for learning technologies:

1. Dot Net Tricks: https://www.dotnettricks.com
2. Scholar Hat: https://www.scholarhat.com

Apart from these, we also provide on-demand Skill bootcamps and personalized project consultation.

Dedication

I would like to say thanks to my mom Mrs. Indra and my brother Mr. Vikram for their support. They deserve their name on the cover as much as I do for all their support made this possible. I would like to say thanks to all my family members, friends, and the mentors who supported me throughout my carrier either directly or indirectly to achieve my goals.

-Gowtham k

Introduction

What Where Author Qualification to Write This Book

Gowtham K is awarded as MVP by Microsoft for his exceptional contribution in Microsoft technologies under the category "Developer Technologies" for the year 2016, 2017 and 2018. He has more than 5 years of experience on Microsoft technologies such as C#, ASP.NET MVC, ASP.NET Web API, ASP.NET Core, MS SQL Server, and Azure. and other technologies such as JavaScript, jQuery, HTML and CSS.

He is also a blogger and author of articles on various technologies. He is also a speaker and delivered talk on various technologies like ASP.NET MVC, Azure and Azure DevOps in the public events.

What This Book Is

JavaScript is the most popular language to create a web application with HTML and CSS, today JavaScript can execute not only in the browser but also on the server. In this book, we are going to learn about the fundamentals of JavaScript like DOM, String, Array, Objects, Events. Error handling and many more.

What You'll Learn

This book is for those who want to learn JavaScript and for those who are going to appear for the JavaScript interview to have a bright future in front-end technologies.

- Document Object Model
- How to Manipulate the HTML element
- Working with strings
- Working with Array in JavaScript
- Handling different type of events
- Working with Object
- Usage of the window object
- Working with Web Storage
- How to use the regular expression
- Working with error handling

Our best wishes always with you for your learning and growth!

About the Author

Gowtham K - An Author, Blogger, and Contributor

Gowtham K is awarded as MVP by Microsoft for his exceptional contribution in Microsoft technologies under the category "Developer Technologies" for the year 2016, 2017 and 2018. He has more than 5 years of experience on Microsoft technologies such as C#, ASP.NET MVC, ASP.NET WEB API, ASP.NET Core, MS SQL Server, and Azure. and other technologies such as JavaScript, jQuery, HTML and CSS.

He is also a blogger and author of articles on various technologies. He is also a speaker and delivered talk on various technologies like ASP.NET MVC, Azure and Azure DevOps in the public events.

He always tries to keep updated himself about new technologies and learning new skills and shared with other in simple manner.

He hopes that this e-book helps you to crack interview of JavaScript. This is the first edition of this book but not last. Please provide your feedback that helps us to improve this book quality.

How to Contact Us

Although the author of this book has tried to make this book as accurate as it possible but if there is something strikes you as odd, or you find an error in the book please drop a line via e-mail.

The e-mail addresses are listed as follows:

- mentor@dotnettricks.com
- info@dotnettricks.com

We are always happy to hear from our readers. Please provide your valuable feedback and comments!

You can follow us on YouTube, Facebook, Twitter, LinkedIn and Google Plus or subscribe to RSS feed.

Table of Contents

JavaScript/ES6 Interview Questions & Answers 1

Release History 1

About Dot Net Tricks 2

Dot Net Tricks Training Solutions 2

Dedication 4

Introduction 5

About the Author 6

How to Contact Us 7

Introducing JavaScript 13

Q1. What is JavaScript? 13

Q2. What is ECMAScript? 13

Q3. What are the data types supported by JavaScript? 13

Q4. What are the Primitive data types supported in JavaScript? 14

Q5. How to declare the variable in JavaScript? 14

Q6. What is “let” keyword in JavaScript? 14

Q7. What are the differences between var and let? 15

Q8. What is the difference between undefined and not defined? 15

Q9. What is DOM? 15

Q10. What is BOM? 17

Q11. What are the different ways to access HTML element in JavaScript? 17

Q12. How to write dynamic code in JavaScript? 18

Q13. What are the differences between attribute and property? 19

Q14. How to use external JavaScript? 19

Q15. What is the scope of the variables in JavaScript? 19

Q16. What is hoisting? 20

Q17. How to do comments in JavaScript? 20

Q18. What is variable typing in JavaScript? 20

Q19. What is the use of typeof operator? 20

Q20. What is the instanceof operator? 21
Q21. What is the difference between undefined and null? 21
Q22. What is the difference between == and === operator? 21
Q23. What is a cookie? 22
Q24. How to set a cookie? 22
Q25. How to read cookie? 22
Q26. How to delete cookie? 23
Q27. What is the web storage system? 23
Q28. What is JavaScript “this” keyword? 24
Q29. Is the JavaScript a case-sensitive language? 24
Q30. How to check a number equal to NaN? 24
Q31. What is JavaScript “const” keyword? 24
Q32. Can you change the property of a constant object? 24
Q33. What is the use of delete operator? 25
Q34. What is the looping statement in JavaScript? 25
Q35. What is strict mode? 26
Q36. Define Closure? 26

String 28

Q1. Explain String in JavaScript? 28
Q2. How to find the length of the string? 28
Q3. How to find the character index in a string? 28
Q4. How to find the Unicode for the character? 28
Q5. How to handle the double quotes special character in the string? 28
Q6. What are the escape sequences available? 29
Q7. Is the String can be an object? 29
Q8. What are the differences between search() and indexOf() ? 29
Q9. What are the differences between indexOf() and lastIndexOf() ? 29
Q10. What are the differences between substr() and substring()? 30
Q11. What is the answer for below code? 30
Q12. What is the answer for below code? 30
Q13. What is the answer for below code? 30

Q14. What is the answer for below code? 31

Object 32

Q1. How to create an object? 32

Q2. How to read and write the JavaScript object properties? 32

Q3. How to copy the object? 32

Q4. What is JavaScript object prototype? 33

Q5. What is Prototypal Inheritance? 33

Q6. What is window object? 33

Q7. What are the window object methods? 33

Q8. What is a math object? 34

Q9. What is the output for below code? 34

Q10. Is it possible to change an object declared with const? 34

Array 35

Q1. What is an array and how to create it in JavaScript? 35

Q2. Is the Array being an object? 35

Q3. Can your store an object as an array element? 35

Q4. Can JavaScript Array have different types of elements? 35

Q5. How to find the length of the array? 36

Q6. How to read the array? 36

Q7. How to iterate the Array elements? 36

Q8. What are the differences between array and object? 36

Q9. How to recognize an Array? 37

Q10. How to convert Array to string? 37

Q11. How to add an item to an array? 37

Q12. How to remove the item from Array? 37

Q13. What is the use of the splice()? 37

Q14. How to sort an Array? 38

Q15. How to empty an Array? 38

Q16. What is the output of below code? 39

Q17. How to perform numeric sort in Array? 39

Functions 40

Q1. What are Anonymous Functions and Named Functions? ... 40
Q2. What is the Self-Executing Function? ... 40
Q3. What is Call back function? ... 40
Q4. What is arrow function? ... 41
Q5. What is time out function? ... 41
Q6. What is the use case of the slice()? ... 41
Q7. What is the use of valueOf()? ... 41
Q8. How to convert a string to lowercase or uppercase in Javascript? ... 41
Q9. How to read the character from a string based on the index? ... 42
Q10. What is the use of forEach()? ... 42

Regular Expression ... 43

Q1. How the RegExp is used in JavaScript? ... 43
Q2. What is the use of test()? ... 43
Q3. What is the use of exec()? ... 43
Q4. What is the flag in a regular expression? ... 43

Events ... 45

Q1. What are Events? ... 45
Q2. What is the use of addEventListener() ? ... 45
Q3. What is onclick Event? ... 46
Q4. What are most used Mouse Events? ... 46
Q5. What is onfocus() event in Javascript ? ... 47
Q6. What type of event is used when the control loses its focus? ... 48
Q7. What are the HTML DOM Keyboard events? ... 48

Window objects ... 50

Q1. How to find the browser which is running the web page? ... 50
Q2. How to find the online status of the application? ... 50
Q3. How to redirect the user to a new page? ... 50
Q4. How to open a web page in a new tab on the browser? ... 50
Q5. What is the use of localStorage? ... 50
Q6. What is the use of sessionStorage? ... 51

Error Handling and Code Snippets ... 52

Q1. What is the use of onerror()? 52
Q2. Can JavaScript method contains try, catch and finally block? 52
Q3. What is the output of below code? 53
Q4. What is the output of below code? 53
Q5. What is the output of below code? 53
Q6. What is the output of below code? 54
Q7. What is the output of below code? 54
Q8. What is the output of below code? 54
Q9. What is the output of below code? 54
Q10. What is the output of below code? 54
Q11. What is the output of below code? 54
Q12. What is the output of below code? 55
Q13. What is the output of below code? 55
References 56

1
Introducing JavaScript

Q1. What is JavaScript?

Ans. JavaScript is an object-based programming language, mostly it used as a client-side programming language with the HTML page to add some behaviour for it.

JavaScript initially created as a browser only language, but now it can be executed on the server or any client which has a JavaScript Engine. The product like Node.js, MongoDB, jaggery.js, ASP and many more uses server-side JavaScript.

In the browser, JavaScript can do many things as given below:

- Manipulating the HTML element.
- React to a user action, like running some event while user clicks on the mouse or by using the keyboard.
- Send request to the remote server.
- Downloading and uploading the files.
- Get and Set cookies and handling the client-side storage (local and session storage).

Major Advantage of using the JavaScript

- Full integration with HTML/CSS.
- Supported by all major browser which is enabled by default.

Q2. What is ECMAScript?

Ans. ECMAScript is a scripting language standardized by ECMA International in ECMA-262. Languages like ActionScript, JavaScript and many more scripting languages are used ECMAScript, among these JavaScript is a well know client-side scripting language and an implementation of ECMAScript, since the standard was published. The latest version is ECMAScript6

Q3. What are the data types supported by JavaScript?

Ans. JavaScript variables are dynamically typed, which means there is a data type but it will not bound to a particular type, example while initializing the variable it can be string type but later It can also assign to a numeric value.

The data types which are supported by JavaScript

- Undefined
- Null
- Boolean
- Object
- String
- Symbol
- Number

Q4. What are the Primitive data types supported in JavaScript?

Ans. The primitive data types in JavaScript are,

- Undefined
- Null
- Boolean
- String
- Number

Q5. How to declare the variable in JavaScript?

Ans. The **var** keyword is used to declare the variable in JavaScript. A variable can begin with a letter, $ or_

```
var _myvar = "Hello"
var x = true;
var $y = 1;
```

From the code snippet, you can observe the var keyword for _myvar is used to declare the string, x for boolean

And $y for integer.

Q6. What is "let" keyword in JavaScript?

Ans. let keyword came from ES2015, it is used to provide a block scope before ES2015 JavaScript had only global scope and function scope

```
  function iterate() {
                //a is *not* visible out here
                for (let a = 0; a < 5; a++) {
                    //a is only visible inside this for()
                    //and there is a separate variable for each iteration of the loop
                }
                //a is *not* visible out here
            }
```

Unlike var, the main benefit of let is to free up the memory when it is not in use.

Q7. What are the differences between var and let?

Ans. The differences between var and let keywords are given below:

Var	let
var is used to declare a variable in JavaScript. It is used from the beginning of the JavaScript	let keyword is introduced in ES6
It has a functional scope	It has block scope

`var a = 10;`
`console.log(window.a);//print 10 in console`
 	
var can be added as a property on the global window object
	`let x = 10;`
	`console.log(window.x);//undefined`
	
	let cannot be added as a property on global window object because of its block scope

`var x = 10;`
`add()`
`function add() {`
`var y = 15;`
`console.log(x + y); // return 25`
`}`
	`let x = 10;`
	`add()`
	`function add() {`
	`let y = 15;`
	`console.log(x + y); // gives error like x is already been declared`
	`}`

Q8. What is the difference between undefined and not defined?

Ans. Considers below example

```
var x;
console.log(x);
```

Now in the console, we will get a message x is '**undefined'** which means the variable is declared and memory is created but the value is not assigned to it.

```
console.log(y)
```

In this case, you will get a message like **'not defined'** because the variable y is not created, and memory is not allocated for it and we try to reference the variable.

Q9. What is DOM?

Ans. DOM is a W3C (World wide web consortium) standard, when the HTML page loads in the browser, the browser creates the DOM (Document object model). It defines the HTML element as an object and allows scripts to dynamically manipulate the content, and the structure of the document.

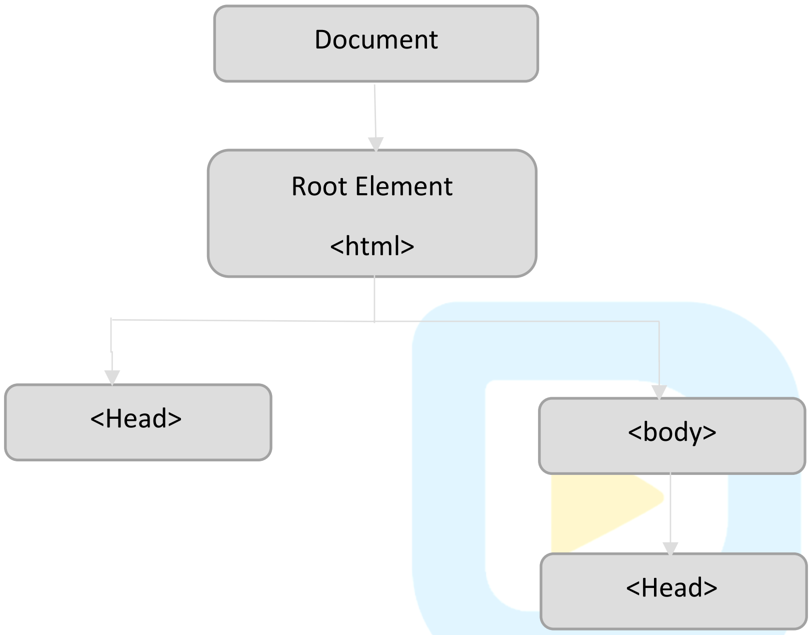
HTML:

```
<!DOCTYPE html>
<html lang="en">
```

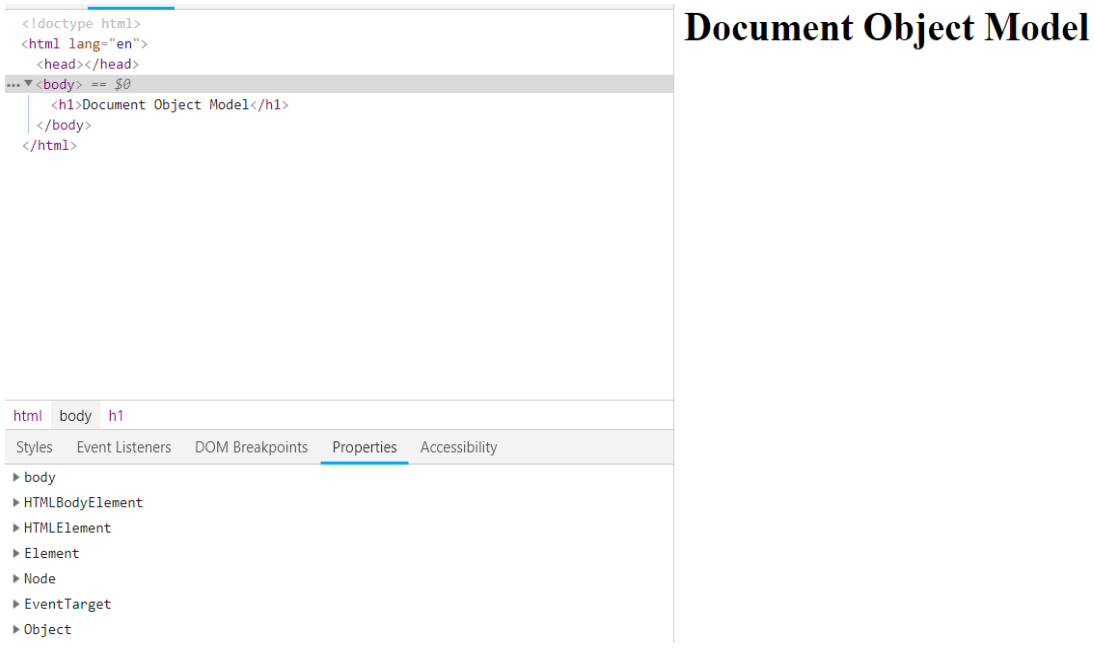
```
<body>
   <h1>Document Object Model</h1>
</body>
</html>
```

In DOM, every HTML is an object, Nested tag are “children”, the text inside a <h1> is an object as well

The DOM Tree of objects



The DOM represents HTML as a tree structure of tags. Here’s how it looks in browser inspect element



```
<!doctype html>
<html lang="en">
  <head></head>
··· ▼<body> == $0
    <h1>Document Object Model</h1>
  </body>
</html>
```

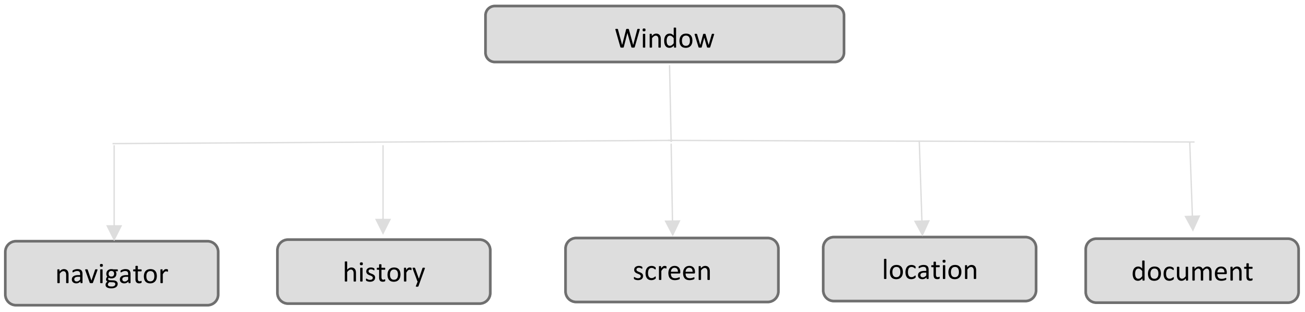
html body h1
Styles Event Listeners DOM Breakpoints Properties Accessibility

```
▸ body
▸ HTMLBodyElement
▸ HTMLElement
▸ Element
▸ Node
▸ EventTarget
▸ Object
```

Document Object Model

Q10. What is BOM?

Ans. BOM (Browser Object Model) which provides an interaction with the browser, the default object of the browser is a window. Various property provided by windows is a document, history, screen, location, navigator



Q11. What are the different ways to access HTML element in JavaScript?

Ans. The following DOM Methods are used to capture the HTML element and manipulate it**.**

1. **getElementById('idname') - >** This function is used to select the HTML element based on ID

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title></title>
</head>
<body>
    <label id="myelement"></label>
    <script>
        document.getElementById('myelement').innerHTML = '<h3> Welcome </h3>'
    </script>
</body
</html>
```

2. **getElementsByClassName('className') - >** This function is used to select the HTML elements based on the class name in DOM, it will return all matched HTML element with respect to the class name.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title></title>
</head>
<style>
    .lblMsg {
        color: #000;
    }
</style>
<body>
```

```
    <label id="myelement" class="lblMsg"></label>
    <script>
        document.getElementsByClassName('lblMsg')[0].innerHTML = '<h3> Welcome
</h3>'
    </script>
</body>
</html>
```

3. **getElementsByTagName('HTMLtagname') - >** This function is used to select the HTML elements based on the Tag name in DOM, it will return all matched HTML element with respect to the Tag name.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title></title>
</head>
<style>
    .lblMsg {
        color: #000;
    }
</style>
<body>
    <label id="myelement" class="lblMsg"></label>
<script>
        document.getElementsByTagName('label')[0].innerHTML = '<h3> Welcome
</h3>'
</script>
</body>
</html>
```

Q12. How to write dynamic code in JavaScript?

Ans. We can manipulate the data dynamically using the DOM methods.

getElementById DOM method is used to capture the HTML element as an object, from this object we can dynamically control the element based on its property.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title></title>
</head>
<body>
    <label id="myelement"></label>
    <script>
        document.getElementById('myelement').innerHTML = '<h3> Welcome </h3>'
    </script>
```

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

Q13. What are the differences between attribute and property?

Ans. Most of the time people will think both property and attribute are the same, but there is a difference between the two.

Attribute	Property
The attribute will be in HTML itself	JS DOM object will contain the property of the HTML element
The attribute is always a string	The property will return multiple types of data

Note: If there is a property it's always recommended to use the property of the element instead of the attribute.

Q14. How to use external JavaScript?

Ans. Assume my JS file name is module.js, then we can refer this js file from the HTML page using <script> tag

```
<script type="text/javascript" src="module.js"> </script>
```

In src attribute, we need to give the path of the JS file

Q15. What is the scope of the variables in JavaScript?

Ans. There are two types of the variable based on scope, that is a local and global variable

1. Global variable – It has a scope everywhere in the JavaScript code.
2. Local variable – It has a scope within the function where it is defined.

```
var x = 10;
add()
Mul()
function add()
   {
      var y = 15;
      console.log(x + y);
   }
function Mul()
   {
      var z = 10;
      console.log(x * z);
   }
```

From the above code, **x** is a global variable where the scope will be entire JavaScript code so that you can use in a different function block, and **y** is local variable where the scope of it will be within **add()**

Q16. What is hoisting?

Ans. In JavaScript the variable can be declared after it has been used.

```
x = 5; // Assign 5 to x
console.log(x);
var x; // Declare x
```

It is a default behavior of JavaScript. Consider below example,

```
var x = 5; // Initialize x
console.log(x)// Display x
console.log(y)//Display y
var y = 7; // Initialize y
```

This will print **x** as **5** and **y** as **undefined** because the y is used before it is initialized, to avoid this we need to declare all variables at beginning of every scope.

Q17. How to do comments in JavaScript?

Ans. // is used to comment single line, /**/ is used to comment multiple lines

```
// single line comment
/*
   Multi-line comment
*/
```

Q18. What is variable typing in JavaScript?

Ans. Consider below example,

```
var x = 10;
x = 'hello';
```

From the above example, you can notice the variable x is first assigned with the integer and the same variable is assigned to a string, this is called variable typing.

Q19. What is the use of typeof operator?

Ans. The **typeof** is a unary operator which means it takes a single operand in a statement or expression, it is used to check the data type of its operand in the form of a string.

```
var x=10;
console.log(typeof (x))
```

It will print **number** in console

```
var x = 10;
console.log(typeof (x) == 'number')
```

From the above code if the typeof x is number, so from the expression it will print **true** in the console.

Q20. What is the instanceof operator?

Ans. Instanceof operator checks whether the object is an instance of a class or not.

```
function Country(name){this.name=name};
var country = new Country("India");
console.log(country instanceof Country)      // return true
```

It will also consider inheritance

```
let arr = ['apple', 'orange', 'grapes'];
console.log(arr instanceof Array); //prints true in console
console.log(arr instanceof Object); //prints true in console
```

arr is an array, but it also belongs to object, because array prototypal inherits from object.

Q21. What is the difference between undefined and null?

Ans. When the variable is declared, and the value is not defined then we call the variable as undefined and JS will also return undefined when we use typeof operator, where null is manually done.

1. undefined

```
var x;
typeof(x); // Undefined
```

2. null

```
var y = null;
typeof(y) //object with null
```

Q22. What is the difference between == and === operator?

Ans. == is used to check the value

```
var x = "10"
var y = 10;
if (x == y) {
//Control Will enter into this block, since == operator will check only the value
            }
```

=== is used to check both value and type

```
var x = "10"
var y = 10;
if (x === y) {
//Control Will not enter into this block, since === operator will check both the
value and type of the variable
            }
```

From the above code, you can notice the value of x and y are 10, but the data type of x is string and y is number since we have used === operator it will return **false**

Q23. What is a cookie?

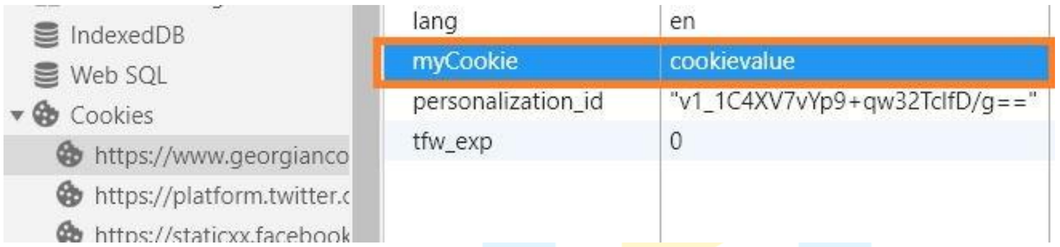
Ans. Cookies is a client-side storage system, which is used to store the data in the browser as a key and value pair, it can be accessed by both web browser and web server. It can also call as an HTTP cookie, browser cookie or web cookie. When we run a website, the website sends the cookie to our computer, where it will be stored in a file located inside our web browser.

Q24. How to set a cookie?

Ans. Cookie can be created using the document object.

```
document.cookie = 'myCookie = cookievalue';
```

You can check the cookie using the developer tool in the browser, as shown in below Image, in chrome, it will be under the application section.



lang	en
myCookie	cookievalue
personalization_id	"v1_1C4XV7vYp9+qw32TclfD/g=="
tfw_exp	0

Q25. How to read cookie?

Ans. The cookie is basically just the value of the object in the document which can be easily read using document object. The value of the cookie is just the string value.

```
var x = document.cookie; // This will return all the cookies which are used in
the web application which is running
```

To read the particular cookie value use below generic function

```
function getCookie(cookiename) {
        var name = cookiename + "=";
        var decodedCookie = decodeURIComponent(document.cookie);
        var ca = decodedCookie.split(';');
 for (var i = 0; i < ca.length; i++)
 {
       var c = ca[i];
       while (c.charAt(0) == ' ')
       {
           c = c.substring(1);
       }
       if (c.indexOf(name) == 0)
       {
           return c.substring(name.length, c.length);
       }
 }
                  return "";
      }

        console.log(getCookie("myCookie"));// prints cookievalue in console
```

We can read a cookie value by passing a cookie name to the getCookie function.

Q26. How to delete cookie?

Ans. To delete the cookie, we need to set the value of the cookie to empty

```
document.cookie ='cookiename = ';
```

```
var deletecookie = function (name)
     {
       document.cookie = name + '=;expires=Thu, 01 Jan 1970 00:00:01 GMT;';
     };
deletecookie('myCookie');
```

The generic function deletecookie() is used to delete cookie based on cookie Name.

Q27. What is the web storage system?

Ans. The concept of local and session storage in a web application was introduced in HTML 5. Before HTML 5, the local data in client side was stored in the cookies. Unlike cookies, now the web application can store the data locally within the browser which can't be transferred to the server.

There are two types of web storage.

1. Local Storage
2. Session Storage

Local Storage

Local storage stores the data, the data persists until the user manually clears the browser cache or programmatically clears the storage.

Session Storage

Unlike local storage the data in session storage will persist only until the window or the tab in the browser is closed, which means the data will be available only for the session.

Between Local and session storage the difference will be only in the persistence of data other than that functionality and functions used in both objects will remain the same.

Q28. What is JavaScript “this” keyword?

Ans. It refers to the global object if there is no current object.

```
var employee = {
         firstName: "John",
         lastName: "Davis",
         fullName: function () {
                   return this.firstName + " " + this.lastName; // Now "refers" a
current object employee
                }
           };
```

```
window.Hello = "Hello World"
function globalObjectTest()
          {
               console.log(this.Hello);// Now "this" refers a global object
          }
```

Q29. Is the JavaScript a case-sensitive language?

Ans. **Yes**, JavaScript is a case-sensitive language, which means the variable, keyboard, function names, and any other identifier should be in the same case when we are using in the JavaScript language.

```
var x = 10;
console.log(X);//X is undefined
```

Q30. How to check a number equal to NaN?

Ans. **NaN** represent a value that is “Not a Number”. **isNaN()** is used to check a number is equal to NaN or not.

```
var x = "10";
var result = isNaN(x);
console.log(result)// false because x is number
```

Q31. What is JavaScript “const” keyword?

Ans. **const** came from ES2015, it behaves like **let** but it can’t be reassigned.

```
const x = 10;
console.log(x)// print 10
x = 11; // this will give error
```

x is a const variable, initially, 10 is assigned to x, when 11 is reassigned to x it will give you an error because const variable can’t be reassigned.

Q32. Can you change the property of a constant object?

Ans. **Yes**, you can change the property of a constant object, but you cannot reassign the object.

```
const employee = { name: "Ram", age: 25 }
```

```
employee.name = "Raju" // it is possible

const employee = { name: "Ram", age: 25 }
employee = { name: "Raju", age: 25 }// will give error
```

Q33. What is the use of delete operator?

Ans. The delete operator is used to delete the property of the object, on successful deletion, it will return true or else false.

```
var employee = { name: "Ram", age: 25 }
delete employee.age;
console.log(employee);
```

Q34. What is the looping statement in JavaScript?

Ans. The looping statement in JavaScript are

- for
- for in
- while
- do-while loops

for

```
var message = "";
for (i = 0; i < 5; i++) {
message += "The number is " + i + "<br>";
   }
```

As like other programming language for loop in JavaScript contains 3 statement which is separated by semi colon. All the three statement are optional in for loop.

for in

for in will loops through the property of an object

```
var employee = { firstName: "John", lastName: "Doe" };
var message = "";
for (var x in employee) {
    message += employee[x];
 }
```

while

It will loop through a block till the condition is true.

```
var i = 0;
var serialNumber = "";
while (i < 10) {
  serialNumber +=  i;
```

```
   i++;
  }
```

do while

do while loop executes at least once, this is because the code block is executed before the condition check.

```
do
{
   text += "The number is " + i;
   i++;
}
   while (i < 10);
```

Q35. What is strict mode?

Ans. “use strict” is not a statement but a literal expression which is supported by ECMAScript version 5. This statement instruct browser to use the strict mode, which is a safer future in JavaScript. It will eliminate some JavaScript silent errors.

Example

```
"use strict";
 x = 10; // this will give error
```

The above statement will give an error because in strict mode the variable should be declared before it is used.

The “use strict” expression can be in global scope as well as local scope

Global scope

```
const employee = { name: "Ram", age: 25 }
employee.name = "Raju" // it is possible
use strict";
x = 10;  // this will give error

```

local scope

```
x = 10;       // This will not give error.
myFunction();
function myFunction() {
 "use strict";
  y = 15;   // This will give error
 }
```

Q36. Define Closure?

Ans. The closure is needed when a variable which is defined outside the scope in reference is accessed from some inner scope. In JavaScript, the closure is created when the function is created. Simply we can say the closure is an inner function that has access to the outer function’s variable.

```
function showName (firstName, lastName) {
    var message = "Your name is ";
        // Inner function has access to the outer function variables
    function FullName () {
            return message + firstName + " " + lastName;
        }

        return FullName ();
      }
        showName ("John", "Davis");
```

The closure has access to variables that are declared and defined in the parent function scope. The closure has access to the variables in following scopes.

- In its own scope
- Parent function scope
- Global scope

```
var globalStr = "Welcome, " // global scope
function showName(firstName, lastName) {
     var message = "Your name is ";
                // Inner function has access to the outer function variables
      function FullName() {
          var str = "Hello "//local scope
          return globalStr + str + message + firstName + " " + lastName;
       }

       return FullName();
     }
 showName("John", "Davis");
```

As per the above statement and from the code you can observe globalStr, message, str are the variable with global scope, parent function scope and local scope respectively, which can be accessed through closure.

2 String

Q1. Explain String in JavaScript?

Ans. The group of character or textual data is called string, in JavaScript, there is no separate type for the character, even a single character will be stored as a string. In JavaScript, the string can be enclosed with single quotes or double quotes

```
var str = 'hello';
console.log(str);//print hello
```

Q2. How to find the length of the string?

Ans. Length property is used to find the length of the string

```
var str = 'hello';
console.log(str.length);
```

Q3. How to find the character index in a string?

Ans. indexOf() is used to find the index of particular character from the string , it will return -1 if it is not found.

```
var str = 'hello'
str.indexOf(e)// it will return 1.
```

Q4. How to find the Unicode for the character?

Ans. charCodeAt() is used to find the Unicode for given character.

```
var temp = "hello";
console.log(temp.charCodeAt(2));// it will print 101 in console.
```

The above code will print **101** in the console since the Unicode of 'e' is 101

Q5. How to handle the double quotes special character in the string?

Ans. The double quote special character can be handled using the escape sequence **\”**

```
var x = 'Use of \"this"\ Keyword in JavaScript';
console.log(x);// it will print Use of “this” keyword in Java in the console.
```

Q6. What are the escape sequences available?

Ans. Following are the escape sequences available.

Code	Result
\b	Backspace
\f	Form Feed
\n	New Line
\r	Carriage Return
\t	Horizontal Tabulator
\v	Vertical Tabulator

Q7. Is the String can be an object?

Ans: **Yes**, Strings can be an object.

```
var temp = new string('Raj');
```

Note: JavaScript strings are primitive values which are created from literals.

Q8. What are the differences between search() and indexOf() ?

Ans: The differences between search and indexOf methods are given below:

search()	indexOf()
It is used to find a specified value and returns the position of the match, the value can be a string or regular expression	It is used to find a specified value and returns the position of the match, the value should be a string, it won't accept a regular expression

`var m = /e/;`
`var str = "apple";`
`str.search(m)//return 4`
	`var m = /e/;`
	`var str = "apple";`
	`str.indexOf(m)//return -1`
	
	Since indexOf() will not support the regular expression it will return -1
	
	`var m = e;`
	`var str = "apple";`
	`str.indexOf(m)//return 4`

Q9. What are the differences between indexOf() and lastIndexOf() ?

Ans: The differences between indexOf and lastIndexOf methods are given below:

indexOf()	lastIndexOf()
It will return the index of the first occurrence of specific text in a string	It will return the index of the last occurrence of specific text in a string

`var str = 'Hello find me test me';`
`str.indexOf('me') // return 11`
	`var str = 'Hello find me test me';`
	`str.lastIndexOf('me') // return 19`

Q10. What are the differences between substr() and substring()?

Ans: The differences between substr and substring methods are given below:

substr()	**substring()**
It is used to return the characters in a string beginning at the specified index and returns the number of characters based on length provided	It is used to return the characters in a string beginning at the specified index and returns the number of characters based on length provided**-1**

`var x = "hello";`
`console.log((x.substr(1, 4) == "ello"))`
 	
It will print **true** in the log
	`var x = "hello";`
	`console.log((x.substring(1, 4) == "ello"))`
	
	It will print **false** in the log
	
	`var x = "hello";`
	`console.log((x.substring(1, 5) == "ello"))//print true in console`

Q11. What is the answer for below code?

```
var x = "Raj";
var y = new String("Raj");
var result = (x == y)
console.log(result)
```

Ans. **true** // because x and y have same value

Q12. What is the answer for below code?

```
var x = "Raj";
var y = new String("Raj");
var result = (x === y)
console.log(result)
```

Ans. **false** // because x and y have different types x- string, y-object

Q13. What is the answer for below code?

```
var x = new String("Raj");
var y = new String("Raj");
var result = (x === y)
console.log(result)
```

Ans. **false** // because x and y are different objects.

Q14. What is the answer for below code?

```
var x = new String("Raj");
var y = new String("Raj");
var result = (x == y)
console.log(result)
```

Ans. **false** // because x and y are different objects.

3 Object

Q1. How to create an object?

Ans. JavaScript support to use the object concept. The object is used to store a collection of data with different types and even more complex entities. An object can be created with the brackets {…} and a property list which is optional. The property is a key and value pair. Example as given below

```
var employee = {name: 'Raj', age: 25, company: 'ABC'}
```

Using the new keyword

```
var employee = new Object ()
employee.name ='Raj';
employee.company ='ABC'
```

Q2. How to read and write the JavaScript object properties?

Ans. Using the dot notation, we can read and write the properties in the object

```
var customer = { name: 'Raj', age: 25, company: 'ABC' }// set the value
customer.name = 'Arun'; //getting the value
customer.name;
customer.age;
```

Q3. How to copy the object?

Ans. We can copy the data from one object to another using the looping statement with an independent object because by copying the object will create one more reference to the same object.

```
let employee = {
            name: "David",
            age: 30
        };
let newObject = {}; // the new empty object
     // let's copy all user properties into it
     for (let key in employee) {
            newObject[key] = employee[key];
    }
        newObject.name = "Alex";
```

Q4. What is JavaScript object prototype?

Ans. The properties and methods of the JavaScript objects are inherited from a prototype. We can add properties and method to objects using prototype property.

```
function Employee(name, company) {
            this.name = name;
            this.company = company;
        }

        Employee.prototype.Age = 25;
        Employee.prototype.getname = function () {
            return this.name;
        };
        var employee = new Employee('Ram', 'XYZ');
```

Q5. What is Prototypal Inheritance?

Ans. Each object in the JavaScript has a property called Prototype, we can add a method to it, as well as we can add another object for it, in this case, it will also inherit the parent object properties, this process is called prototypal inheritance.

Date object inherits from Date.prototype , Array object inherit from Array.prototype, The Employee object inherit from Employee.prototype as given in below code,

```
function Employee(name, company) {
            this.name = name;
            this.company = company;
    }
  Employee.prototype.Age = 25;
  Employee.prototype.getname = function () {
                        return this.name;
                  };
  var employee = new Employee('Ram', 'XYZ');
```

Q6. What is window object?

Ans: Window object is not a JavaScript object, it is a browser object, and it will be created by the browser that represents a window of a browser.

Q7. What are the window object methods?

Ans. Some of the frequently used window object methods are listed below

Method	Description
alert()	Display the alert box containing the message with an Ok button
prompt()	This box is used to get the input from the user
confirm()	Display confirm box containing the message with ok and cancel button
open()	opens the new window
close()	close the current window

Q8. What is a math object?

Ans. It is a build-in object which has a mathematical property and methods.

```
console.log(Math.PI) // 3.141592653589793
console.log(Math.abs(1.0)); // 1
```

Q9. What is the output for below code?

```
let x = {};
let y = {};
console.log((x == y));
```

Ans. **false,** because x and y are two independent objects even though both the value are same

Q10. Is it possible to change an object declared with const?

Ans. **Yes,** the const will protect only the variable itself from changing, consider below example

```
const employee = {
firstName: "David"
       };
 employee.firstName = "Alex"; // yes, this is possible
```

From the above example, you can notice the firstName property from the employee object is got changed even though the object is constant, at same case consider below example,

```
const employee = {
            firstName: "David"
  };
 employee = "hello";// this won't work
```

Modifying the constant variable will not work.

4
Array

Q1. What is an array and how to create it in JavaScript?

Ans. Array is a variable which can store multiple value in a single variable as an ordered collection, example as given below:

```
var fruits = ['Apple', 'Orange', 'Grapes']
console.log(fruits);
```

We can also create an array, and assign a value to it by below statement

```
var fruits = new Array('Apple', 'Orange', 'Grapes');
```

Array elements are numbered, which starts with zero

```
var fruits = new Array('Apple', 'Orange', 'Grapes');
console.log(fruits[0])//Apple
```

Q2. Is the Array being an object?

Ans. **Yes**, it is a special type of object. The typeof operator will return the array as an object type.

The difference between array and object is the array will use numbers to access its elements where object use name to access its members

```
var fruits = new Array('Apple', 'Orange', 'Grapes');
console.log(typeof (fruits));//object
```

Q3. Can your store an object as an array element?

Ans. **Yes**, the array element can be an object

```
var myArray = [];
myArray[0] = Date.now;
```

Note: Array element can also have a functions and objects

Q4. Can JavaScript Array have different types of elements?

Ans. **Yes**, In JavaScript we can have array with different type

```
var myArray = [];
myArray[0] = Date.now;
myArray[1] = 1;
```

```
myArray[2] = "hello";
```

Q5. How to find the length of the array?

Ans. The length property is used to find the length of the array

```
var fruits = ['Apple', 'Orange', 'Grapes']
console.log(fruits.length)
```

Q6. How to read the array?

Ans. We can read the array using its index and to read the complete list in the array, we can do using looping so that we can iterate through each element in the array.

```
var fruits = ['Apple', 'Orange', 'Grapes']
for (var i = 0; i < fruits.length; i++)
    {
       console.log(fruits[i])
    }
```

Q7. How to iterate the Array elements?

Ans. We can loop the array in JavaScript using the various looping statement. Best way is using the for loop

```
var fruits = new Array('Apple', 'Orange', 'Grapes');
for (var i = 0; i < fruits.length; i++)
    {
          console.log(fruits[i])
    }
```

We can also use the forEach method,

```
var fruits = new Array('Apple', 'Orange', 'Grapes');
fruits.forEach(myFunction);
function myFunction(value)
    {
            console.log(value)
    }
```

Q8. What are the differences between array and object?

Ans. The differences between array and object are given below:

Array	**object**
The array uses the numbered indexes to access the element in it	The object uses the named indexes to access the members in it
You should use an array when you want the element name to be number	You should use an object when you want the element name to be a string
It is an ordered collection	It is a collection of unorder properties

Q9. How to recognize an Array?

Ans. The typeof operator will return the array as an object type, then how we can find a variable is an array? The simple solution is to use the Array.isArray().

```
var fruits = new Array('Apple', 'Orange', 'Grapes');
Array.isArray(fruits) // it will returns true
```

Note: Array.isArray() came from ECMAScript 5 which is not supported in the old browser

Q10. How to convert Array to string?

Ans. toString() method is used to convert the array to string.

```
var fruits = new Array('Apple', 'Orange', 'Grapes');
console.log(fruits.toString()); // output : Apple,Orange,Grapes
```

Q11. How to add an item to an array?

Ans. push() is used to add the item in an array.

```
var fruits = new Array('Apple', 'Orange', 'Grapes');
fruits.push('Papaya');
console.log(fruits.toString()); // output : Apple,Orange,Grapes,Papaya
```

Q12. How to remove the item from Array?

Ans. pop() method is used to remove the last item/element from an array.

```
var fruits = new Array('Apple', 'Orange', 'Grapes');
fruits.pop();
console.log(fruits.toString()); // output : Apple,Orange
```

Q13. What is the use of the splice()?

Ans . splice() is used to add/remove an element from the Array, it is like replace function

```
var fruits = new Array('Apple', 'Orange', 'Grapes');
fruits.splice(1, 0, 'Papaya')
```

first parameter ‘1’ represents the index of the array where the element needs to be added

second parameter ‘0’ is used to define how many elements should be removed from the list

the third parameter is the element which should be added

```
console.log(fruits.toString()); // Apple,Papaya,Orange,Grapes
```

To remove the element

```
var fruits = new Array('Apple', 'Orange', 'Grapes');
```

```
fruits.splice(1, 1)
console.log(fruits.toString()); // Apple, Grapes
```

Q14. How to sort an Array?

Ans. sort() is used to sort the Array based on the values as string

```
var fruits = new Array('Apple', 'Orange', 'Grapes');
fruits.sort();
console.log(fruits.toString());
```

Q15. How to empty an Array?

Ans. We have multiple methods to empty an array

Method 1

```
var arr = [1, 2, 3, 4, 5]
arr=[]
```

We can reassign the array with empty list, but by doing this the original referenced array will remain unchanged, as shown in below code.

```
 var arr = [1, 2, 3, 4, 5]
 var newArrayList = arr;
 arr = []
 console.log(newArrayList)//[1, 2, 3, 4, 5]
 console.log(arr)//[]
```

Method 2

We can clear the array list by making the array length as 0, this method is useful when we want to update all reference variable pointing to array list.

```
var arr = [1, 2, 3, 4, 5]
var newArrayList = arr;
arr.length=0
console.log(newArrayList)//[]
console.log(arr)//[]
```

Method 3

We can clear the array list using the splice (), as like method 2 this method is useful when we want to update all reference variable pointing to array list.

```
var arr = [1, 2, 3, 4, 5]
var newArrayList = arr;
arr.length=0
console.log(newArrayList)//[]
console.log(arr)//[]
```

Q16. What is the output of below code?

```
var numbers = [1, 2, 3, 4, 5];
delete numbers[3];
console.log(numbers.length);
```

Ans. It will print **5** in console. When we use the delete operator to delete the array element, the array element length will not get affected, but the value will be updated as **empty** in chrome and **<1 empty slot>** in Firefox.

```
var numbers = [1, 2, 3, 4, 5];
delete numbers[3];
console.log(numbers);// (5) [1, 2, 3, empty, 5]
```

Q17. How to perform numeric sort in Array?

Ans. sort() will sort the array based on string, for numeric sorting we need to provide a compare function as a call back function to sort().

```
var numArr = [4, 9, 8, 2, 1]
var resultAsc = numArr.sort(function (a, b) { return a - b }); // Ascending
var resultDesc = numArr.sort(function (a, b) { return b - a }); // Descending
```

5
Functions

Q1. What are Anonymous Functions and Named Functions?

Ans. The function which exists only after it is called is named as Anonymous function, whereas named function will exist even when it is not called.

Named function – Function contains the name at time of the definition

```
function myfunc()
{
   alert('Named function');
}
```

Anonymous function – It is declared dynamically at runtime

```
var display = function ()
{
   alert('anonymous function')
}
```

Q2. What is the Self-Executing Function?

Ans. The self-executing function will execute right after it has been defined. The advantage of using it is, it will execute the code without declaring any global. Mostly it will be used to attach event listeners to DOM elements and another initialization work.

```
 (function ()
    {
            //function body
    })();
```

Q3. What is Call back function?

Ans. Passing function as an argument to another function is called as call back functions.

```
Myfunction('onclick', function ()
 {
     //call back function body
 });
```

Q4. What is arrow function?

Ans. The arrow function will support in JavaScript only after ES6 or above, it is a short way to write function expression. The conventional way of writing a function

```
function add(a, b) {
            return a + b;
        }
console.log(add(1, 2));//3
```

Using arrow function

```
add = (a, b) => { return a + b }
console.log(add(1, 2));//3
```

Q5. What is time out function?

Ans. The timeout function can be used to set a particular time for the function execution, for example, if you need to execute a function after some delay once it is called, then this will do the trick.

```
setTimeout(function ()
{
    alert("Hello");
 }, 3000);
```

The above function will be executed after 3 secs from its call time.

Q6. What is the use case of the slice()?

Ans. We can copy array simply by assigning to another variable in JavaScript as given below.

```
var arr1 = ['a', 'b', 'c'];
var arr2 = arr1;
```

But the problem in above code is both arr1 and arr2 will have same reference, we can use the slice() to return the reference of the new array.

```
var arr1 = ['a', 'b', 'c'];
var arr2 = arr1.slice();
```

Q7. What is the use of valueOf()?

Ans. valueOf() is used to get the primitive value of string

```
var temp ='Hello'
console.log(temp.valueOf())
```

Q8. How to convert a string to lowercase or uppercase in Javascript?

Ans. We can convert the string to lower or upper case using toUpperCase() and toLowerCase()

toLowerCase()

```
var temp = "Hello";
console.log(temp.toLowerCase());//print hello
```

toUpperCase()

```
var temp = "Hello";
console.log(temp.toUpperCase());
```

Q9. How to read the character from a string based on the index?

Ans. charAt() is used to get the character from the string based in index

```
var temp = "hello";
temp.charAt(1)//return "e"
```

Q10. What is the use of forEach()?

Ans. forEach() is used to call a function for each element in the array

```
var fruits = new Array('Apple', 'Orange', 'Grapes');
fruits.forEach(myFunction);
function myFunction(value)
    {
              console.log(value)
    }
```

The above code will print each element in the array list.

6

Regular Expression

Q1. How the RegExp is used in JavaScript?

Ans. Regular Expression will describe the pattern of characters. It is an object. RegEx is a universal concept. It is the best way of finding the pattern which is used to search and replace inside a string. In JavaScript, expressions are defined using the objects of RegExp class which is built-in.

```
var str = "burger";
var regex = /ge/;
str.search(regex)// return 3
```

Q2. What is the use of test()?

Ans. test() is a regular expression method. It searches a string for a specific pattern. It will return true or false as a result.

```
var patternVar = /o/;
var result = patternVar.test('Hello World');
console.log(result) //It will print true in browser console
```

Q3. What is the use of exec()?

Ans. exec() is a regular expression method. It searches a string for a specific match. It will return the text as an object, if there is no match it will return empty object.

```
var patternVar = /o/;
var result = patternVar.exec('Hello World');
console.log(result) // It will print the result as an object
```

```
0: "o"
groups: undefined
index: 4
input: "Hello World"
length:
```

Q4. What is the flag in a regular expression?

Ans. There are 5 flags in JavaScript regular expression that affects the search

g- This indicates that the regular expression should be tested against all possible matches in a string

```
var str = "Test hello Test";
var regex = /Test/g;
var result = str.match(regex);
console.log(result); // return all "Test" found in a string in the form of
array
```

i- This flag is the used in search string pattern by ignoring the case

```
var str = "burGEr";
var regex = /ge/;
str.search(regex)// return -1 without flag
```

```
var str = "burGEr";
var regex = /ge/i;
str.search(regex)// return 3 with i flag
```

m- do a multiline search for test at the beginning of each line in a string

```
var str = "\nTest hello \nTest";
var regex = /^Test/gm;
var result = str.match(regex);
console.log(result);
```

u - It enables the Unicode support, treats the pattern as a sequence of Unicode code points

y- This makes the search should find an exact match at the position specified by the property regexp.lastIndex

```
let str = "Hello from Dot Net Tricks";
let reg = /Dot/iy;
console.log(reg.lastIndex); // 0 (default)
console.log(str.match(reg)); // null, not found at position 0
reg.lastIndex = 11;
console.log(str.match(reg)) // Dot, the word start at the position 11
```

7 Events

Q1. What are Events?

Ans. JavaScript interacts with HTML through the events. Different types of event will be fired when the HTML page loads, buttons click, input field change and much more.

List of most used DOM events,

Keyboard events:

- Keydown – This event will be invoked when the user presses down the keys in the keyboard
- keyup - This event will be invoked when the user releases the keys in the keyboard

Mouse events:

- click – this event will be invoked when the mouse clicks on an element
- mouseover/mouseout- this event will be invoked when the mouse cursor moves over/ leaves an element
- mousedown/mouseup – when the mouse button is pressed/ released over an element
- mousemove- When the mouse is moved
- contextmenu – When the mouse right-clicks on the element

Form events:

- submit – this will invoke when the form is submitted
- focus- the will be invoked when the user focuses on the input field.

Q2. What is the use of addEventListener() ?

Ans. addEventListner() is the DOM element which is used to attach the event to the control, for example when we need to execute a function when the button is clicked we can use this method as written in below code.

```
  <!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Untitled</title>
</head>
```

```
        <body>
        <input type="button" id="myBtn" value="Click Me" />
        <script>
         document.getElementById("myBtn").addEventListener("click", function () {
                      sayHello()//click event call back
                  });

                  function sayHello()
                     {
                      alert('Hello')
                     }
        </script>
        </body>
        </html>
```

Note: This method will not support in IE 8 and earlier version

Q3. What is onclick Event?

Ans. The onclick event is most used event type. This event will execute when the user clicks on left button from mouse.

```
        <!DOCTYPE html>
        <html>
        <head>
            <meta charset="utf-8">
            <title>Untitled</title>
        </head>
        <body>
        <input type="button" onclick="sayHello()" value="Click Me" />
        <script>
            function sayHello()
            {
                alert('Hello')
            }
        </script>
        </body>
        </html>
```

Mouse click event only within HTML

```
<input type="button" onclick="alert('Hello')" value="Button">
```

Q4. What are most used Mouse Events?

Ans. **onmouseover** and **onmouseout** are the two most commonly used mouse events. The mouseover event will trigger when you mouse over the HTML element, on the other hand, the onmouseout event will fire when you leave the HTML element.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Untitled</title>
</head>
<body>
<div onmouseover="mouseover()" onmouseout="mouseout()">
    <h2 id="txt"> Hello Welcome ! </h2>
</div>
<script>
    function mouseover() {
        document.getElementById("txt").innerHTML = "Mouse Over";
    }

    function mouseout() {
        document.getElementById("txt").innerHTML ="Mouse Out"
    }
</script>
</body>
</html>
```

mouseover() this method is fired when the user mouse over the div element.

mouseout() this method is fired when the user leaves the div element.

Q5. What is onfocus() event in Javascript ?

Ans. This event will be fired when the control gets a focus, mostly this event is used with the input control.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Untitled</title>
</head>
<body>
<<input type="text" id="text1"  onfocus="focusFunction()"/>
<script>
function focusFunction()
   {
 console.log("I'm focus event")
    }
</script>
</body>
</html>
```

We can also pass the event object so that we can manipulate the control.

```
<input type="text" id="text1"  onfocus="focusFunction(this)"/>
```

```
       function focusFunction()
          {
       e.style.background = "Aqua";// this is dynamically change the text box
       color when we focus the textbox
        }
```

Q6. What type of event is used when the control loses its focus?

Ans. **onblur()** event will get fired when the control losses it's focus, basically this event is used to the input control. It is just opposite to onfocus().

```
       <!DOCTYPE html>
       <html>
       <head>
           <meta charset="utf-8">
           <title>Untitled</title>
       </head>
       <body>
       <<input type="text" id="text1"  onblur="blurFunction()"/>
       <script>
       function blurFunction()
          {
        console.log("I'm blur event")
           }
       </script>
       </body>
       </html>
```

Q7. What are the HTML DOM Keyboard events?

Ans. List of HTML DOM Keyboard events which are used more frequently

- onkeydown.
- onkeypress
- onkeyup

onkeydown:

This event will get fired when user pressing a key

```
<!DOCTYPE html>
<html>
<head>
      <meta charset="utf-8">
      <title>Untitled</title>
</head>
<body>
    <input type="text" onkeydown="keyDown()">
</body>
<script>
```

```
        function keyDown()
          {
               alert("I'm key Down Event ")
           }
</script>
</html>
```

onkeyup:

This event will get fired when user releases a key

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8">
   <title>Untitled</title>
</head>
<body>
    <input type="text" onkeyup="keyUp()">
</body>
<script>
      function keyUp()
        {
               alert("I'm key Up Event")
           }
</script>
</html>
```

onkeypress:

This event will get fired when the user presses a key

```
<!DOCTYPE html>
<html>
<head>
     <meta charset="utf-8">
     <title>Untitled</title>
</head>
<body>
      <input type="text" onkeypress=" keyPress()">
</body>
<script>
      function keyPress()
        {
               alert("I'm key Press Event")
         }
</script>
</html>
```

8
Window objects

Q1. How to find the browser which is running the web page?

Ans. The window object navigator is used to find the browser which is currently running the web application.

```
var browsername = navigator.appName;
console.log(browsername);
```

Q2. How to find the online status of the application?

Ans. The online property of the navigator is used to find the application in online or offline

```
var browsername = navigator.onLine;
console.log(browsername);
```

Q3. How to redirect the user to a new page?

Ans. We can use the window object location to redirect the user to new page

```
window.location="https://www.dotnettricks.com/"
```

Q4. How to open a web page in a new tab on the browser?

Ans. We can use the window method open to launch the web application in a new tab on the browser

```
window.open("https://www.dotnettricks.com/", '_blank')
```

Q5. What is the use of localStorage?

Ans. localStorage is the window object which is used to save the data in the local storage of the browser.

Local storage stores the data, the data persists until the user manually clears the browser cache or programmatically clears the storage.

setItem() is used to store the data in localStorage.

```
localStorage.setItem('name', 'value2')
```

getItem() is used to get the data in local storage

```
localStorage.getItem('name')
```

removeItem() is used to remove the data in local storage

```
localStorage.removeItem('name')
```

Q6. What is the use of sessionStorage?

Ans. sessionStorage is the window object which is used to save the data in session storage of browser.

Unlike local storage the data in session storage will persist only until the window or the tab in the browser is closed, which means the data will be available only for the particular session.

setItem() is used to store the data in sessionstorage.

```
sessionStorage.setItem('name', 'value2')
```

getItem() is used to get the data in session storage

```
sessionStorage.getItem('name')
```

removeItem() is used to remove the data in session storage

```
sessionStorage.removeItem('name')
```

9
Error Handling and Code Snippets

Q1. What is the use of onerror()?

Ans. onerror() is the event handler which is fired whenever the exception occurs on the page. It is a window object method.

```
<html>
<head>
    <script type="text/javascript">
     window.onerror = function (msg, url, line)
    {
    alert("Message : " + msg + ',' + "url : " + url + ',' + "Line number : " +
line);
}
    </script>
</head>
<body>
    <p>Click the following to see the result:</p><input type="button"
value="Click Me" onclick="onErrorMessage();" />
 </body>
 </html>
```

From the above example, you can notice onErrorMessage() is called when the user clicks on the button, but there is no definition for onErrorMessage() so it will enter into onerror window object method.

Q2. Can JavaScript method contains try, catch and finally block?

Ans. **Yes**, we can have try, catch and finally block in javascript

```
<html>
<head>
<script>
          function keyDown() {
              try {
                  alert("Hello")
              }
              catch (e) {
                    //catch block
                }
                finally {
                    //final block
                }
            }
</script>
</head>
<body>
    <input type="text" onkeydown="keyDown()">
</body>
</html>
```

Q3. What is the output of below code?

```
var num = "10";
(function () {
            console.log("Original Number " + num);
            var num = "50";
            console.log("New Number " + num);
    })();
```

Ans. Original Number undefined

New Number 50

Reason: You will expect the original number will take the value from the outer scope, but the salary value was undefined, because of hoisting.

Q4. What is the output of below code?

```
var x = typeof null;
console.log(x)
```

Ans. It will print **object**

Q5. What is the output of below code?

```
(function () {
            var x = y = 5;
        })();
        console.log(y);
        console.log(x);
```

Ans. It will print 5 for y and undefined error for x

Reason: var x=y=5 is shorthand for,

y=5

```
x=y ;
```

so, y ends up as a global variable, it will have the scope outside of the block. This is a common behaviour of JavaScript, by using the strict mode, we can avoid these kinds of bugs.

Q6. What is the output of below code?

```
console.log(typeof NaN === "number");
```

Ans. true

Reason: NaN property represents the value which is 'Not a Number', this is because of the result of the operation is not numeric, so NaN means "Not a Number", it's type is Number;

Q7. What is the output of below code?

```
for (var i = 0; i < 5; i++) {
        setTimeout(function () { console.log(i); }, 10);
}
```

Ans: It will print 5,5,5,5,5 because each function which is executed within the loop will execute after the entire loop has completed, therefore the last value stored in I will get printed. To avoid this we should use closures concept.

Q8. What is the output of below code?

```
console.log(true == '1')
console.log(true === '1')
```

Ans. It will print true, false. Since === operator will check the type of the operand along with value it will print as false.

Q9. What is the output of below code?

```
for (let i = 0; i < 5; i++) {
            setTimeout(function () { console.log(i); }, 10);
}
```

Ans. It will print 0,1,2,3,4 because we use let instead of var

Q10. What is the output of below code?

```
console.log(typeof typeof true)
```

Ans. It will print **a string,** type of true will return **"boolean"** and type of **"boolean" is a string**

Q11. What is the output of below code?

```
  var obj = { a: 1 };
  var output = (function ()
        {
```

```
                    delete obj.a;
                    return obj.a;
         })();

   console.log(output);// undefined
```

Ans. The output will be 10 because the delete operator will delete the property of an object.

Q12. What is the output of below code?

```
var a = 10;
      var output = (function ()
      {
          delete a;
          return a;
      }
      )();
console.log(output);
```

Ans. The output will be 10 because the delete operator is used to delete the property of an object, here a is not an object

Q13. What is the output of below code?

```
for (; ;)
{
   console.log(1);
}
```

Ans. The application will get crashed because of this infinite loop, it's always recommended to use the break statement if there is no conditional check in for a loop.

References

This book has been written by referring to the following sites:

1. https://developer.mozilla.org - JavaScript MDN
2. https://stackoverflow.com/questions/tagged/javascript - Stack Overflow - JavaScript
3. https://www.dotnettricks.com/learn/javascript - Dot Net Tricks - JavaScript