JavaScript Tutorial for Beginners: Learn Javascript Step by Step

By : James Hartman March 9, 2024

JavaScript Tutorial Summary

This online JavaScript guide is geared to make you a JavaScript pro! It will help you learn JavaScript step by step. You will learn all the JavaScript basics in this guide.

What is JavaScript?

JavaScript is an open-source and most popular client-side scripting language supported by all browsers. JavaScript is mainly used for enhancing the interaction of the webpage with users by making it more lively and interactive. It is also used for game development and mobile application development.

Javascript Syllabus

First Steps in Javascript Basics for Beginners

Lesson 1	What is JavaScript? — A Complete Introduction with Hello World! Example
Lesson 2	JavaScript Variables — Declare Variables in JavaScript, Assign a Value with Example
Lesson 3	JavaScript Array Methods — Create Array with Example in JavaScript
Lesson 4	JavaScript Loops — For, While and Do While LOOP in JavaScript (with Example)
Lesson 5	Conditional Statements — IF, Else, Else IF Conditional Statements in JavaScript
Lesson 6	JavaScript String Format — Methods with EXAMPLES

Javascript Advance Stuff!

Lesson 1	JavaScript Define & Call Functions — What is, How to Create with Example

Lesson 2	Cookies in JavaScript — Learn Javascrip Set, Get & Delete Cookie with Example
Lesson 3	JavaScript DOM Tutorial — Learn with Example
Lesson 4	OOJS Tutorial — Object Oriented JavaScript(OOJS) Tutorial with Example
Lesson 5	Internal & External JavaScript — Learn with Example
Lesson 6	Javascript Examples — Practical Code Examples using JavaScript
Lesson 7	JavaScript Unit Testing Frameworks — What is & Best Frameworks
Lesson 8	TypeScript Tutorial — What is, Types, Interfaces, Enums, Arrays, Example
Lesson 9	Typescript vs JavaScript — What's the Difference?
Lesson 10	Java vs JavaScript — Most Important Differences You Must Know
Lesson 11	QuickSort in JavaScript — Learn QuickSort Algorithm in JavaScript
Lesson 12	Difference Between =, ==, and === in JavaScript — Learn with Example

JavaScript Interview Questions, Tools, Books & Tutorial PDF

Lesson 1	JavaScript Courses — 90 Best JavaScript Certification Courses
Lesson 2	JavaScript Books — 14 Best JavaScript Books for Beginners and Experts
Lesson 3	BEST JavaScript IDE — List of Top 20 BEST JavaScript IDE
Lesson 4	JavaScript Interview Questions — Top 85 JavaScript Interview Questions and Answers
Lesson 5	JavaScript Tutorial PDF — Download Javascript Tutorial PDF for Beginners

Don't Miss:

- JavaScript DOM Tutorial with Example
- Object Oriented JavaScript(OOJS) Tutorial with Example
- Internal & External JavaScript: Learn with Example

What will you learn in this JavaScript Tutorial for Beginners?

In this JavaScript basics for beginners tutorial, you will learn about some fundamentals of JavaScript like Variables, Arrays, Loops, Conditional Statements, Cookies, etc., and some advanced JavaScript concepts like DOM, practical code examples, JavaScript Unit testing frameworks, algorithms, etc.

Are there any prerequisites for this JavaScript Tutorial?

Nothing! This is an absolute JavaScript beginner guide to learn JavaScript with examples. However, if you have some basic knowledge of HTML and CSS, it will help you learn faster and more efficiently.

Who is this JavaScript Tutorial for?

This JavaScript for beginners tutorial is for students who want to learn about Web application development and software development. This tutorial is also helpful for the professionals working in web application development to enhance their knowledge and skills.

Why learn JavaScript Programming Language?

JavaScript is the most popular client-side programming language which is widely used for web application development in every industry. There is a huge demand in the IT industry for candidates having knowledge of JavaScript. Therefore, learning JavaScript is beneficial for you to get a good job and also enhance your skills and knowledge as well.

How do JavaScript engines work?

JavaScript Engines are complicated. But it works on some simple basics:

- The engine reads (“parses:) the script.
- Then it converts or compiles the script to the machine language.
- After that machine code runs.

Here, JavaScript engine applies optimizations at each step of the process. It reads a compiled script and analyzes the data that passes in JavaScript engine. After that, it applies optimizations to the machine code from that acquired knowledge. When this process is completed, scripts run quite fast.

What can in-browser JavaScript do?

JavaScript's functionality depends on the environment it's running in. For example, Node.js supports functions which allows JavaScript to read and write arbitrary files, perform network requests, object-oriented, etc. The roles that JavaScript plays in both client-side (front end) and server-side (back end) development of applications can vary wildly.

In-browser JavaScript also allows you to perform webpage manipulation, interaction with the user and with the web server.

Javascript offer advantages like:

- Show dynamic content based on the user profile.
- React to user's operations, like mouse clicks events, key presses or pointer movements.
- Support features like auto-validated form entries and interactive drop-down menus.
- Send requests to remote servers, Upload and download files.
- JavaScript code can also create movement and sound
- Ask questions to the users, Get and set cookies, show messages, switch browser tabs.
- Allows the data to be stored in the local storage.

What can't in-browser JavaScript do?

JavaScript's capabilities in the browser are quite limited for the sake of the user's safety. It helps to prevent any unauthorized webpage from accessing private information.

Examples of such limitations are:

- JavaScript on a webpage may not allow you to copy, execute or read/write arbitrary files on the hard disk. It doesn't offer any access to Operating system functions.
- Many browsers allow it to work with files, but the access is very limited and only provided if the user is performing a specific action like, dropping a file into a browser window or selecting using <input> tag.
- JavaScript allows you to communicate over the net to the server where the current page came from. Although, it does not allow you to receive data from other sites/domains.

What makes JavaScript unique?

Here, are the three most important features which make JavaScript unique

- It offers full integration with HTML/CSS.
- Simple things are done quickly without any complication or following strict rules.
- Supported by all major browsers and JavaScript is enabled by default.

Alternatives to JavaScript

The syntax of JavaScript not suited for everyone as different project demands different features. However, some modern tools like a Coffee script, Typescript, and Dart allowing developers to code in another language and then auto-convert into the JavaScript code.

Where is JavaScript Today?

ECMAScript is a specification governed by ECMA international aimed at standardizing JavaScript. The latest version is ECMA9 also called JavaScript 9. It is supported by all major browsers like Chrome, Firefox, Internet Explorer, etc. Though each browser has an array of unique commands that are not part of the standards.

Report a Bug

Next →

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,
United States

Privacy Manager

What is JavaScript? Complete Introduction with Hello World! Example

By : James Hartman March 9, 2024

What is JavaScript?

JavaScript is a very powerful client-side scripting language. JavaScript is used mainly for enhancing the interaction of a user with the webpage. In other words, you can make your webpage more lively and interactive, with the help of JavaScript. JavaScript is also being used widely in game development and Mobile application development.

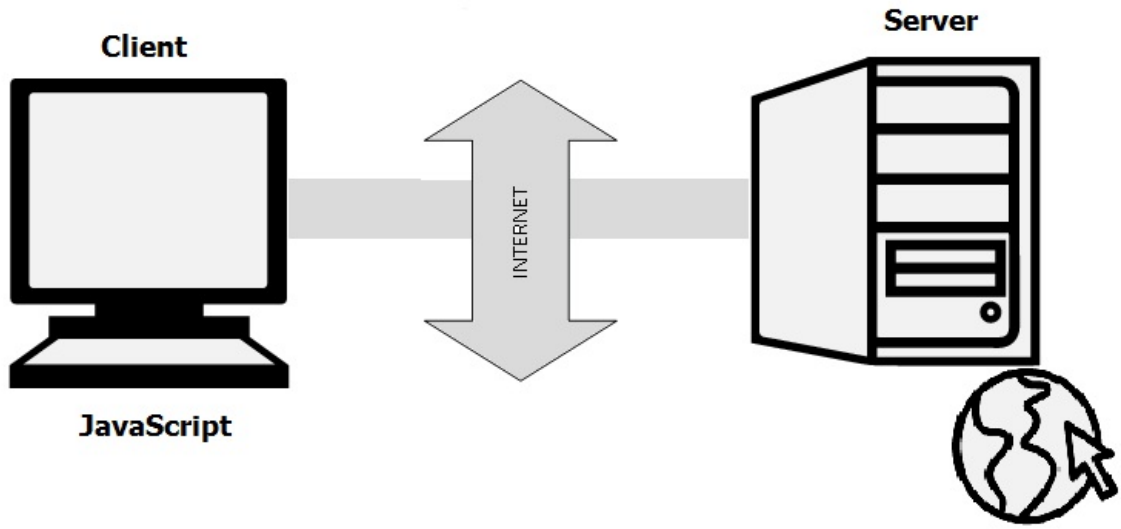


Table of Content:

Javascript History

JavaScript was developed by Brendan Eich in 1995, which appeared in Netscape, a popular browser of that time.

Brendan Eich - Creator of JavaScript

The language was initially called LiveScript and was later renamed JavaScript. There are many programmers who think that JavaScript and Java are the same. In fact, JavaScript and Java are very much unrelated. Java is a very complex programming language whereas JavaScript is only a scripting language. The syntax of JavaScript is

mostly influenced by the programming language C.

How to Run JavaScript?

Being a scripting language, JavaScript cannot run on its own. In fact, the browser is responsible for running JavaScript code. When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it is up to the browser to execute it. The main advantage of JavaScript is that all modern web browsers support JavaScript. So, you do not have to worry about whether your site visitor uses Internet Explorer, Google Chrome, Firefox or any other browser. JavaScript will be supported. Also, JavaScript runs on any operating system including Windows, Linux or Mac. Thus, JavaScript overcomes the main disadvantages of VBScript (Now deprecated) which is limited to just IE and Windows.

Tools You Need

To start with, you need a text editor to write your code and a browser to display the web pages you develop. You can use a text editor of your choice including Notepad++, Visual Studio Code, Sublime Text, Atom or any other text editor you are comfortable with. You can use any web browser including Google Chrome, Firefox, Microsoft Edge, Internet Explorer etc.

A Simple JavaScript Program

You should place all your JavaScript code within <script> tags (<script> and </script>) if you are keeping your JavaScript code within the HTML document itself. This helps your browser distinguish your JavaScript code from the rest of the code. As there are other client-side scripting languages (Example: VBScript), it is highly recommended that you specify the scripting language you use. You have to use the type attribute within the <script> tag and set its value to text/javascript like this:

```
<script type="text/javascript">
```

Hello World Example

```
<html>
<head>
        <title>My First JavaScript code!!!</title>
        <script type="text/javascript">
                alert("Hello World!");
        </script>
</head>
<body>
</body>
</html>
```

Note: type=”text/javascript” is not necessary in HTML5. Following code will work.

```
<html>
<head>
        <title>My First JavaScript code!!!</title>
        <script>
                alert("Hello World!");
        </script>
</head>
<body>
</body>
</html>
```

Summary

- JavaScript is a client-side scripting language developed by Brendan Eich.
- JavaScript can be run on any operating systems and almost all web browsers.
- You need a text editor to write JavaScript code and a browser to display your web page.

Don't Miss:

- TypeScript Tutorial: What is, Interface, Enum, Array with Example
- QuickSort Algorithm in JavaScript
- 15 Best FREE JavaScript Certification Courses Online (2024)

← Prev

Report a Bug

Next →

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,
United States

JavaScript Variable: Declare, Assign a Value with Example

By : James Hartman December 30, 2023

Variables are used to store values (name = “John”) or expressions (sum = x + y).

Declare Variables in JavaScript

Before using a variable, you first need to declare it. You have to use the keyword var to declare a variable like this:

```
var name;
```

Assign a Value to the Variable

You can assign a value to the variable either while declaring the variable or after declaring the variable.

```
var name = "John";
```

OR

```
var name;

name = "John";
```

Naming Variables

Though you can name the variables as you like, it is a good programming practice to give descriptive and meaningful names to the variables. Moreover, variable names should start with a letter and they are case sensitive. Hence the variables student name and studentName are different because the letter n in a name is different (n and N).

Try this yourself:

```
<html>
<head>
<title>Variables!!!</title>
<script type="text/javascript">
var one = 22;
var two = 3;
var add = one + two;
var minus = one - two;
var multiply = one * two;
var divide = one/two;
        document.write("First No: = " + one + "<br />Second No: = " + two + " <br />");
        document.write(one + " + " + two + " = " + add + "<br/>");
        document.write(one + " - " + two + " = " + minus + "<br/>");
```

```
        document.write(one + " * " + two + " = " + multiply + "<br/>");
        document.write(one + " / " + two + " = " + divide + "<br/>");
</script>
</head>
<body>
</body>
</html>
```

Don't Miss:

- TypeScript Tutorial: What is, Interface, Enum, Array with Example
- QuickSort Algorithm in JavaScript
- 15 Best FREE JavaScript Certification Courses Online (2024)

← Prev

Report a Bug

Next →

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,
United States

JavaScript Array Methods: Create with Example

By : James Hartman March 9, 2024

What is an Array?

An array is an object that can store a collection of items. Arrays become really useful when you need to store large amounts of data of the same type. Suppose you want to store details of 500 employees. If you are using variables, you will have to create 500 variables whereas you can do the same with a single array. You can access the items in an array by referring to its indexnumber and the index of the first element of an array is zero.

JavaScript Create Array

You can create an array in JavaScript as given below.

```
var students = ["John", "Ann", "Kevin"];
```

Here, you are initializing your array as and when it is created with values “John”, “Ann” and “Kevin”. The index of “John”, “Ann” and “Kevin” is 0, 1 and 2 respectively. If you want to add more elements to the students array, you can do it like this:

```
students[3] = "Emma";
students[4] = "Rose";
```

You can also create an array using Array constructor like this:

```
var students = new Array("John", "Ann", "Kevin");
```

OR

```
var students = new Array();

students[0] = "John";

students[1] = "Ann";

students[2] = "Kevin";
```

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().

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

Try this yourself:

```
<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>
```

Don't Miss:

- TypeScript Tutorial: What is, Interface, Enum, Array with Example
- QuickSort Algorithm in JavaScript
- 15 Best FREE JavaScript Certification Courses Online (2024)

← Prev

Report a Bug

Next →

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,
United States

For, While and Do While LOOP in JavaScript (with Example)

By : James Hartman March 9, 2024

How to use Loop?

Loops are useful when you have to execute the same lines of code repeatedly, for a specific number of times or as long as a specific condition is true. Suppose you want to type a 'Hello' message 100 times in 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.

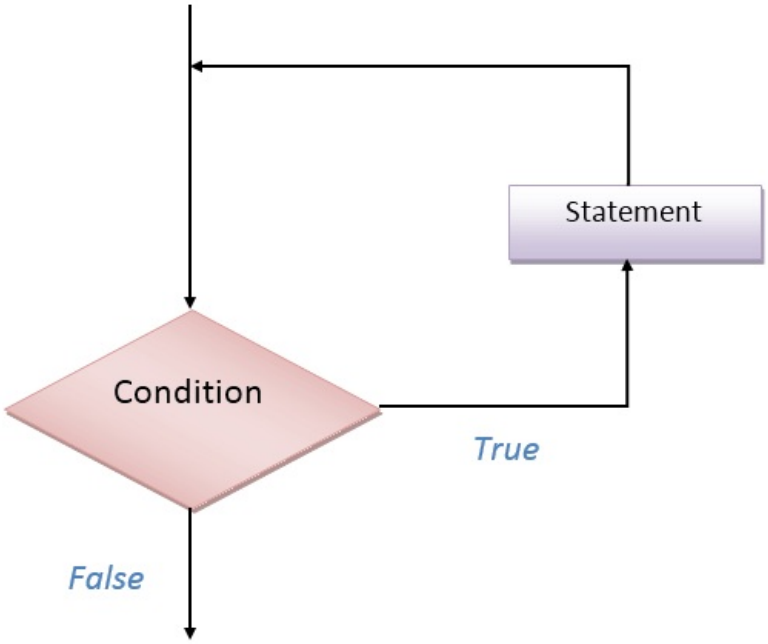


Table of Content:

Different Types of Loops

There are mainly four types of loops in JavaScript.

1. for loop
2. for/in a loop (explained later)
3. while loop
4. do…while loop

for loop

Syntax:

```
for(statement1; statement2; statment3)
{
lines of code to be executed
}
```

1. The statement1 is executed first even before executing the looping code. So, this statement is normally used to assign values to variables that will be used inside the loop.
2. The statement2 is the condition to execute the loop.
3. The statement3 is executed every time after the looping code is executed.

Try this yourself:

```
<html>
<head>
        <script type="text/javascript">
                var students = new Array("John", "Ann", "Aaron", "Edwin", "Elizabeth");
                document.write("<b>Using for loops </b><br />");
                for (i=0;i<students.length;i++)
                {
                document.write(students[i] + "<br />");
                }
        </script>
</head>
<body>
</body>
</html>
```

while loop

Syntax:

```
while(condition)
{
lines of code to be executed
}
```

The “while loop” is executed as long as the specified condition is true. Inside the while loop, you should include the statement that will end the loop at some point of time. Otherwise, your loop will never end and your browser may crash.

Try this yourself:

```
<html>
<head>
        <script type="text/javascript">
```

```
                document.write("<b>Using while loops </b><br />");
                var i = 0, j = 1, k;
                document.write("Fibonacci series less than 40<br />");
                while(i<40)
                {
                        document.write(i + "<br />");
                        k = i+j;
                        i = j;
                        j = k;
                }
        </script>
</head>
<body>
</body>
</html>
```

do…while loop

Syntax:

```
do
{
block of code to be executed
} while (condition)
```

The do…while loop is very similar to while loop. The only difference is that in do…while loop, the block of code gets executed once even before checking the condition.

Try this yourself:

```
<html>
<head>
        <script type="text/javascript">
                document.write("<b>Using do...while loops </b><br />");
                var i = 2;
                document.write("Even numbers less than 20<br />");
                do
                {
                        document.write(i + "<br />");
                        i = i + 2;
                }while(i<20)
        </script>
</head>
<body>
</body>
</html>
```

- TypeScript Tutorial: What is, Interface, Enum, Array with Example
- QuickSort Algorithm in JavaScript
- 15 Best FREE JavaScript Certification Courses Online (2024)

← Prev

Report a Bug

Next →

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,
United States

Conditional Statements in JavaScript: if, else, and else if

By : James Hartman February 24, 2024

JavaScript Conditional Statements

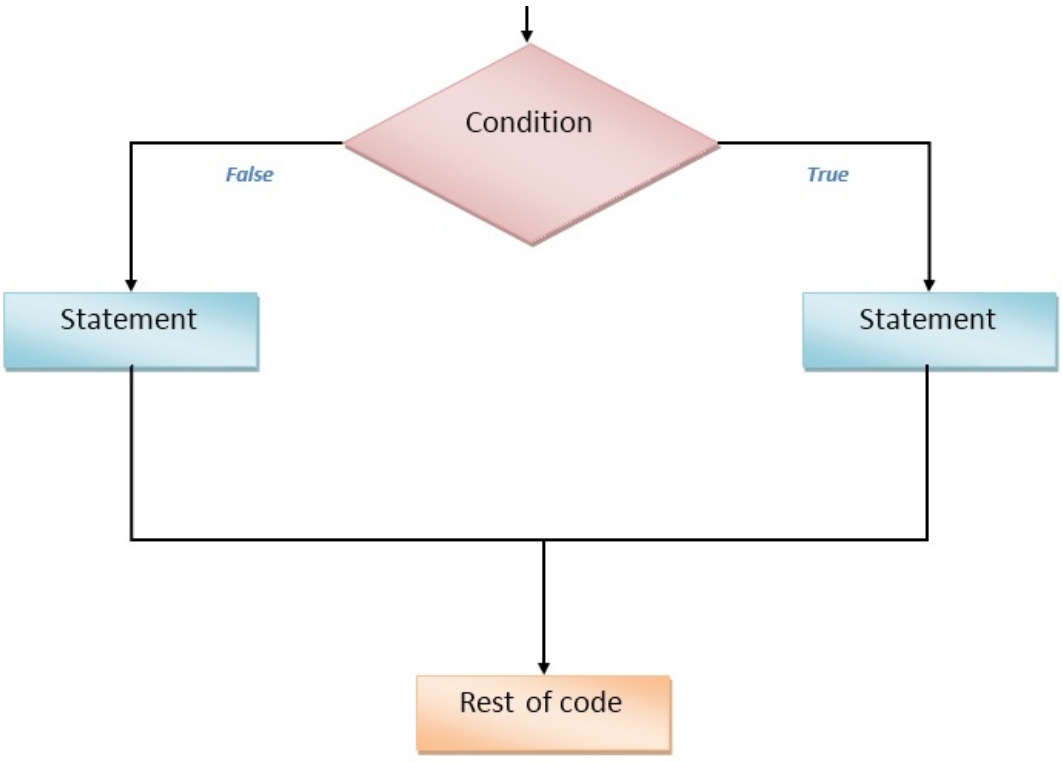
There are mainly three types of conditional statements in JavaScript.

1. if Statement: An 'if' statement executes code based on a condition.
2. if…else Statement: The if…else statement consists of two blocks of code; when a condition is true, it executes the first block of code and when the condition is false, it executes the second block of code.
3. if…else if…else Statement: When multiple conditions need to be tested and different blocks of code need to be executed based on which condition is true, the if…else if…else statement is used.

Table of Content:

How to use Conditional Statements

Conditional statements are used to decide the flow of execution based on different conditions. If a condition is true, you can perform one action and if the condition is false, you can perform another action.



If statement

Syntax:

```
if(condition)
{
        lines of code to be executed if condition is true
}
```

You can use `if` statement if you want to check only a specific condition.

Try this yourself:

```
<html>
<head>
        <title>IF Statments!!!</title>
        <script type="text/javascript">
                var age = prompt("Please enter your age");
                if(age>=18)
                document.write("You are an adult <br />");
                if(age<18)
                document.write("You are NOT an adult <br />");
        </script>
</head>
<body>
</body>
</html>
```

If…Else statement

Syntax:

```
if(condition)
{
        lines of code to be executed if the condition is true
}
else
{
        lines of code to be executed if the condition is false
}
```

You can use `If….Else` statement if you have to check two conditions and execute a different set of codes.

Try this yourself:

```
<html>
<head>
        <title>If...Else Statments!!!</title>
```

```
        <script type="text/javascript">
                // Get the current hours
                var hours = new Date().getHours();
                if(hours<12)
                document.write("Good Morning!!!<br />");
                else

                document.write("Good Afternoon!!!<br />");
        </script>
</head>
<body>
</body>
</html>
```

If…Else If…Else statement

Syntax:

```
if(condition1)
{
        lines of code to be executed if condition1 is true
}
else if(condition2)
{
        lines of code to be executed if condition2 is true
}
else
{
        lines of code to be executed if condition1 is false and condition2 is false
}
```

You can use `If….Else If….Else` statement if you want to check more than two conditions.

Try this yourself:

```
<html>
<head>
        <script type="text/javascript">
                var one = prompt("Enter the first number");
                var two = prompt("Enter the second number");
                one = parseInt(one);
                two = parseInt(two);
                if (one == two)
                        document.write(one + " is equal to " + two + ".");
                else if (one<two)
                        document.write(one + " is less than " + two + ".");
                else
                        document.write(one + " is greater than " + two + ".");
        </script>
</head>
<body>
</body>
</html>
```

Don't Miss:

- TypeScript Tutorial: What is, Interface, Enum, Array with Example
- QuickSort Algorithm in JavaScript
- 15 Best FREE JavaScript Certification Courses Online (2024)

← Prev Report a Bug Next →

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,
United States

JavaScript String Format: Methods with EXAMPLES

By : James Hartman March 9, 2024

What is JavaScript String Format?

JavaScript String format refers to the approaches that can be used for manipulating and formatting strings in a defined form. It is a crucial feature of any programming language that permits you to have control over the text that appears in your application.

In your JavaScript code, string formatting can transform complex data in a simple way. It also improves the output readability and makes the code more maintainable. Additionally, properly formatted JavaScript strings can also play a role in enhancing the user experience of the application.

Table of Content:

How to Format Strings in JavaScript?

In order to format a string in JavaScript, follow the provided steps:

Step 1) First, it is required to determine the data that you want to include in the string.

Step 2) Then, select a method to format string in JS.

Step 3) Write the code for its implementation accordingly.

Step 4) Test the code to make sure that it generates the expected output.

Step 5) Update or modify the code to meet the additional formatting needs or requirements.

Step 6) Refactor the code for maintainability and readability.

Different String Formatting Methods in JavaScript

There exist multiple ways to format strings in JavaScript, including concatenation operators, template literals, regular expressions, and more.

In the next section, you will explore the following approaches:

- {} Brackets with backticks
- “+” Operator
- Custom function String Formatting
- Concatenation operator
- Template literals
- Regular expression
- Custom function
- Ternary operator

Method 1: Using {} Brackets with Backticks to Format String in JavaScript

In this method, you can use template literals by adding backticks (`) for enclosing a string. Then, insert variables of the string arguments in brackets {} preceded with a dollar sign “$”. These templates enable you to embed expressions within a string literal.

This operation helps to concatenate variables called placeholders and other defined values. Resultantly, the final formatted string is obtained by interpolating the values of placeholders.

For example, we have created two variables, “name” and “id,” and then enclosed them in backticks for generating the formatted output. Note that the “${}” syntax is used for embedding the expressions with the string.

Code:

```
const name = "Jenny";
const id = 1;
console.log (`Welcome ${name}, your ID is ${id}`);
```

It can be observed that the values of the placeholder’s “name” and “id” have been displayed, which means string interpolation has been performed successfully.

Output:

```
Welcome Jenny, your ID is 1
```

Don't Miss:

- Execute JavaScript Online
- TypeScript vs JavaScript – Difference Between Them
- QuickSort Algorithm in JavaScript

Method 2: Using + Operator to Format String in JavaScript

The second approach is to use the Concatenation operator “+” for basic formatting strings in JavaScript. This way involves concatenating the individual variables and strings together to generate the desired output.

For instance, we have now used the “+” operator to concatenate the same variable values, and the “console.log()” method will return a formatted string on the console.

Code:

```
const name = "Jenny";
const id = 1;
console.log("Welcome " + name + ", your ID is " + id);
```

Output:

```
Welcome Jenny, your ID is 1
```

Method 3: Custom function String Formatting in JavaScript

In your JavaScript program, you can also use the custom function string formatting approach. In this technique, the function accepts the desired variables and returns the formatted string. This format function also offers a lot of flexibility. You can also use it to fulfill custom or complex formatting requirements.

Here, we have created a custom format function named “welcomeMsg()” that accepts two variables “name” and “id” as arguments and outputs formatted strings by utilizing the template string literals.

After that, we defined the desired variables and passed them in the format custom formation function call.

Code:

```
function welcomeMsg(name, id) {
  return `Welcome ${name}, your ID is ${id}`;
}
const name = "Jenny";
const id = 1;
const msg = welcomeMsg(name, id);
console.log(msg);
```

Output:

```
Welcome Jenny, your ID is 1
```

Method 4: Using Concatenation for Format String in JavaScript

Concatenation is a simple method utilized for formatting strings in JavaScript. This approach involves combining string variables or multiple strings together using the concatenation operator “+”. You can also use this technique for simple string concatenation tasks.

In the provided example, the variables “firstName” and “lastName” will be concatenated with the help of the “+” operator to form the “fullName” string as follows.

Code:

```
var firstName = "Alex";
var lastName = "Edward";
var fullName = firstName + " " + lastName;
console.log(fullName);
```

Output:

```
Alex Edward
```

Method 5: Using Template Literals for Format String in JavaScript

Template strings which are also known as Template literals offer a more expressive and advanced way to format strings in JavaScript. They permit the direct embedding of expressions and variables within the string using the placeholder “${expression}”.

Moreover, template literal also provides support to the multi-line strings.

Code:

```
var name = "Alex";
var age = 30;
var msg = `Hi, ${name}! You are ${age} years old.`;
console.log(msg);
```

In the above example, we have directly interpolated the variable “name” and “age” within the string by utilizing the placeholders “${}” within the backticks (`) of the template literals.

Output:

```
Hi, Alex! You are 30 years old.
```

Method 6: Using Regular Expressions to Format String in JavaScript

In JavaScript code, Regular expressions are the robust tool used for string manipulation and pattern matching. Moreover, they are also utilized for finding the given pattern and replacing them with defined values. This operation can be done with the help of the “replace ()” method.

Check out the below program, the replace () method is used with two regular expressions that will replace the placeholders “[name]” and “[id]” with the respective value.

Code:

```
let name = "Jenny";
let id = 1;
let message = "Welcome [name], your ID is [id]".replace(/\[name\]/g, name).replace(/\[id\]/g, id);
console.log(message);
```

This results in a new string object that will be stored in the “message” variable logged on the console as output.

Output:

```
Welcome Jenny. Your ID is 1
```

Method 7: Using a Custom function for Format String in JavaScript

You can also create a more reusable and customized string format in JavaScript. To do so, define a new custom function that accepts a string and other parameters. You can call it to replace placeholders or apply custom formatting rules. This technique also supports customized formatting in complex formatting scenarios.

Here, we have created a new custom function format named “formatString” that accepts a “string” and “params” (array of parameters) as arguments.

Within the function body, we have invoked the “replace()” method with a regular expression for replacing placeholders “{0}, {1}” with their respective values from the parameters array.

Code:

```
function formatString(string, params) {
  return string.replace(/{(\d+)}/g, (match, index) => {
    return typeof params[index] !== 'undefined' ? params[index] : match;
  });
}
var name = "Alex";
var age = 30;
var formattedMsg = formatString("Hi, {0}! You are {1} years old.", [name, age]);
console.log(formattedMsg);
```

Output:

```
Hi, Alex! You are 30 years old.
```

Method 8: Using Ternary Operator to Format String in JavaScript

JavaScript Ternary operator is a shorthand way for writing conditional expressions on strings. However, it can be also used for string formatting by conditionally selecting one of two values according to the specified boolean expression and returning the string.

Syntax:

```
condition ? expressionIfTrue : expressionIfFalse
```

Have a look at this example code of using a ternary operator to format into a string object.

Code:

```
let name = "Jenny";
let isAdmin = true;
let msg = `Welcome ${name}${isAdmin ? ", you are an admin" : ""}`;
console.log(msg);
```

Output:

```
Welcome Jenny, you are an admin
```

Here, we have defined two variables “name” and “isAdmin” with the values “Jenny” and “true”, respectively.

After that, we utilized the ternary operator to conditionally add the string “, you are an admin” to the message string if the value of the optional parameter “isAdmin” is true, as stated above.

Comparison of JavaScript String Format Methods

Here, we have enlisted the advantages and disadvantages of the above-discussed JavaScript string format methods:

Method	Advantages	Disadvantages
Brackets {} with backticks (`)	• Easy to use. • Concise and readable code.	It may not be appropriate for more complex formatting requirements.
“+” operator	Easy to understand.	Can be inefficient for more complex formatting requirements.
Custom function for String formatting	Offers a lot of flexibility.	Needs more coding effort.
Concatenation Operator	Simple and widely supported.	Cumbersome for complex string formatting.
Template Literals	• Supports multi-line strings. • Readable and Intuitive syntax.	Limited control over formatting options.

Regular Expressions	Provide a lot of flexibility.	Different implementation and maintenance as compared to other methods.
Custom function	Provides a lot of flexibility.	Needs additional coding for creating and maintaining a custom function.
Ternary operator	Concise way for handling null or undefined values.	It might not be suitable for more complex formatting requirements.

Note that the selection of the method highly depends on the specific use case and the relevant requirements of your JavaScript code.

Conclusion

In JavaScript, you can use template literals, the ternary operator, "+" operator, regular expressions, and the custom function to format strings. These string format methods permit developers to manipulate and present textual data in an efficient manner.

Thus, select the right method based on readability, performance, and browser compatibility. It empowers developers to improve user experiences and achieve desired formatting standards.

Don't Miss:

- Execute JavaScript Online
- TypeScript vs JavaScript – Difference Between Them
- QuickSort Algorithm in JavaScript

← Prev

Report a Bug

Next →

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,
United States

Cookies in JavaScript: Set, Get & Delete Example

By : James Hartman March 9, 2024

What are Cookies?

A cookie is a piece of data that is 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 each and every time you try to login? If yes, then you are using cookies. Cookies are saved as key/value pairs.

Table of Content:

Why do you need a Cookie?

The communication between a web browser and server happens using a stateless protocol named HTTP. Stateless protocol treats each request independent. So, the server does not keep the data after sending it to the browser. But in many situations, the data will be required again. Here come cookies into a picture. With cookies, the web browser will not have to communicate with the server each time the data is required. Instead, it can be fetched directly from the computer.

Javascript Set Cookie

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

```
document.cookie = "cookiename=cookievalue"
```

You can even add expiry date to your cookie so that the particular cookie will be removed from the computer on the specified date. The expiry date should be set in the UTC/GMT format. If you do not set the expiry date, the cookie will be removed when the user closes the browser.

```
document.cookie = "cookiename=cookievalue; expires= Thu, 21 Aug 2014 20:00:00 UTC"
```

You can also set the domain and path to specify to which domain and to which directories in the specific domain the cookie belongs to. By default, a cookie belongs to the page that sets the cookie.

```
document.cookie = "cookiename=cookievalue; expires= Thu, 21 Aug 2014 20:00:00 UTC; path=/ "
```

//create a cookie with a domain to the current page and path to the entire domain.

JavaScript get Cookie

You can access the cookie like this which will return all the cookies saved for the current domain.

```
var x =  document.cookie
```

JavaScript Delete Cookie

To delete a cookie, you just need to set the value of the cookie to empty and set the value of expires to a passed date.

```
document.cookie = "cookiename= ; expires = Thu, 01 Jan 1970 00:00:00 GMT"
```

Try this Example yourself

Special instructions to make the code work … Press the run button twice

```
<html>
<head>
        <title>Cookie!!!</title>
        <script type="text/javascript">
                function createCookie(cookieName,cookieValue,daysToExpire)
        {
          var date = new Date();
          date.setTime(date.getTime()+(daysToExpire*24*60*60*1000));
          document.cookie = cookieName + "=" + cookieValue + "; expires=" + date.toGMTString();
        }
                function accessCookie(cookieName)
        {
          var name = cookieName + "=";
          var allCookieArray = document.cookie.split(';');
          for(var i=0; i<allCookieArray.length; i++)
          {
            var temp = allCookieArray[i].trim();
            if (temp.indexOf(name)==0)
            return temp.substring(name.length,temp.length);
          }
                return "";
        }
                function checkCookie()
        {
          var user = accessCookie("testCookie");
          if (user!="")
                alert("Welcome Back " + user + "!!!");
          else
```

```
          {
            user = prompt("Please enter your name");
            num = prompt("How many days you want to store your name on your computer?");
            if (user!="" && user!=null)
            {
            createCookie("testCookie", user, num);

            }
          }
        }
        </script>
</head>
<body onload="checkCookie()"></body>
</html>
```

Don't Miss:

- TypeScript Tutorial: What is, Interface, Enum, Array with Example
- QuickSort Algorithm in JavaScript
- 15 Best FREE JavaScript Certification Courses Online (2024)

← Prev

Report a Bug

Next →

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,
United States

Career Suggestion
SAP Career Suggestion Tool
Software Testing as a Career

Interesting
eBook
Blog
Quiz
SAP eBook
Privacy Manager

JavaScript DOM Tutorial with Example

By : James Hartman December 26, 2023

What is DOM in JavaScript?

JavaScript can access all the elements in a webpage making use of Document Object Model (DOM). In fact, the web browser creates a DOM of the webpage when the page is loaded. The DOM model is created as a tree of objects like this:

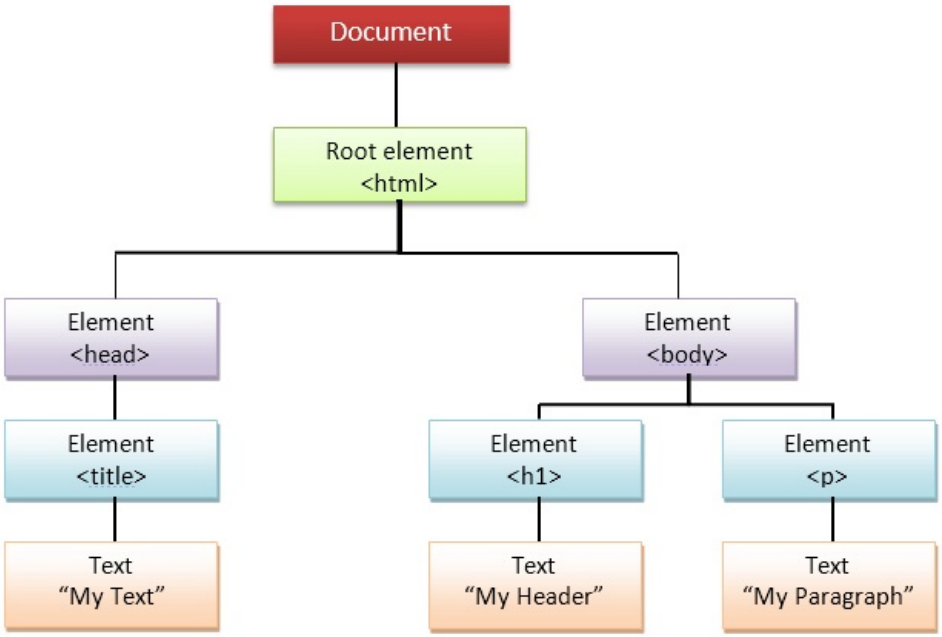


Table of Content:

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.

getElementById, innerHTML Example

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

Try this Example yourself:

```
<html>
<head>
        <title>DOM!!!</title>
</head>
<body>
  <h1 id="one">Welcome</h1>
  <p>This is the welcome message.</p>
  <h2>Technology</h2>
  <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>
```

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.

Try this Example yourself:

```
<html>
<head>
        <title>DOM!!!</title>
</head>
<body>
  <h1>Welcome</h1>
  <p>This is the welcome message.</p>
  <h2>Technology</h2>
  <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)
```

Try this Example yourself:

```
<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>
```

Don't Miss:

- TypeScript Tutorial: What is, Interface, Enum, Array with Example
- QuickSort Algorithm in JavaScript
- 15 Best FREE JavaScript Certification Courses Online (2024)

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,
United States

About
About Us
Advertise with Us
Write For Us
Contact Us

Career Suggestion
SAP Career Suggestion Tool
Software Testing as a Career

Interesting
eBook
Blog
Quiz
SAP eBook
Privacy Manager

Execute JavaScript Online

By : Krishna Rungta March 23, 2024

JavaScript Online Compiler (Editor)

Follow the simple steps below to compile and execute any JavaScript program online using your favourite browser, without having any setup on your local machine

Step-1 Type your source using available text editor

Step-2 Click Run to get Output

Note: Before Compilation you must know about JavaScript

```
<html>
<body>

<h1>My Web Page</h1>

<script>
document.write("My JavaScript");
</script>

</body>
</html>
```

Don't Miss:

- TypeScript Tutorial: What is, Interface, Enum, Array with Example
- QuickSort Algorithm in JavaScript
- 15 Best FREE JavaScript Certification Courses Online (2024)

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,
United States

Object Oriented JavaScript(OOJS) Tutorial with Example

By : James Hartman March 9, 2024

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 have many unique characteristics of their own. You can create properties and methods to your objects to make programming easier. If your object is a student, it will have properties like first name, last name, id etc and methods like calculateRank, changeAddress etc. If your object is a home, it will have properties like a number of rooms, paint color, location etc and methods like calculateArea, changeOwner etc.

Table of Content:

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} };
```

Access Object Properties and Methods

You can access properties of an object like this:

```
objectname.propertyname;
```

You can access methods of an object like this:

```
objectname.methodname();
```

Try this Example yourself:

```
<html>
<head>
        <title>Objects!!!</title>
        <script type="text/javascript">
      var student = new Object();
      student.fName = "John";
      student.lName = "Smith";
      student.id = 5;
      student.markE = 76;
      student.markM = 99;
      student.markS = 87;
      student.calculateAverage = function()
      {
        return (student.markE + student.markM + student.markS)/3;
      };
      student.displayDetails = function()
      {
        document.write("Student Id: " + student.id + "<br />");
        document.write("Name: " + student.fName + " " + student.lName + "<br />");
        var avg = student.calculateAverage();
        document.write("Average Marks: " + avg);
      };
                student.displayDetails();
        </script>
</head>
<body>
</body>
</html>
```

OOPS Constructor

But creating objects of this kind is not that useful because here also, you will have to create different objects for different students. Here comes object constructor into picture. Object constructor helps you create an object type which can be reused to meet the need of individual instance.

Try this Example yourself:

```
<html>
<head>
        <script type="text/javascript">
                function Student(first, last, id, english, maths, science)
        {
```

```
{
  this.fName = first;
  this.lName = last;
  this.id = id;
  this.markE = english;
  this.markM = maths;
  this.markS = science;
  this.calculateAverage = function()
  {
        return (this.markE + this.markM + this.markS)/3;
  }
         this.displayDetails = function()
         {
    document.write("Student Id: " + this.id + "<br />");
    document.write("Name: " + this.fName + " " + this.lName + "<br />");
    var avg = this.calculateAverage();
    document.write("Average Marks: " + avg + "<br /><br />");
         }
        }
var st1 = new Student("John", "Smith", 15, 85, 79, 90);
var st2 = new Student("Hannah", "Turner", 23, 75, 80, 82);
var st3 = new Student("Kevin", "White", 4, 93, 89, 90);
var st4 = new Student("Rose", "Taylor", 11, 55, 63, 45);
st1.displayDetails();
st2.displayDetails();
st3.displayDetails();
st4.displayDetails();
</script>
</head>
<body>
</body>
</html>
```

Loop Through the Properties of an Object

Syntax:

```
for (variablename in objectname)

{

lines of code to be executed

}
```

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 name of the object should be the same as that of an already existing object which you need to loop through.

Try this Example yourself:

```
<html>
<head>
<script type="text/javascript">
```

```
        <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>
```

Don't Miss:

- TypeScript Tutorial: What is, Interface, Enum, Array with Example
- QuickSort Algorithm in JavaScript
- 15 Best FREE JavaScript Certification Courses Online (2024)

← Prev

Report a Bug

Next →

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,

United States

Career Suggestion
SAP Career Suggestion Tool
Software Testing as a Career

Interesting
eBook
Blog
Quiz
SAP eBook
Privacy Manager

Internal & External JavaScript: Learn with Example

By : James Hartman March 9, 2024

You can use JavaScript code in two ways.

1. You can either include the JavaScript code internally within your HTML document itself
2. You can keep the JavaScript code in a separate external file and then point to that file from your HTML document.

What is Internal JavaScript?

We have been using Internal JS so far. Here is a sample –

```
<html>
<head>
  <title>My First JavaScript code!!!</title>
  <script type="text/javascript">
    // Create a Date Object
    var day = new Date();
    // Use getDay function to obtain todays Day.
    // getDay() method returns the day of the week as a number like 0 for Sunday, 1 for Monday,…..,
5
    // This value is stored in today variable
    var today = day.getDay();
    // To get the name of the day as Sunday, Monday or Saturday, we have created an array named
weekday and stored the values
    var weekday = new Array(7);
    weekday[0]="Sunday";
    weekday[1]="Monday";
    weekday[2]="Tuesday";
    weekday[3]="Wednesday";
    weekday[4]="Thursday";
    weekday[5]="Friday";
    weekday[6]="Saturday";
    // weekday[today] will return the day of the week as we want
    document.write("Today is " + weekday[today] + ".");
  </script>
</head>
<body>
</body>
</html>
```

What is External JavaScript?

You plan to display the current date and time in all your web pages. Suppose you wrote the code and copied into 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 in 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 the full path to your file in the src attribute.

How to link external JavaScript

```
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;
if (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)
```

```
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);
```

This is your currentdetails.js file. Don't worry seeing long lines of code. You will learn to code soon. Make changes to your HTML document like this:

```
<html>
        <head>
           <title>My External JavaScript Code!!!</title>
```

```
            <title>My External JavaScript Code!!!</title>
            <script type="text/javascript" src="currentdetails.js">
            </script>
        </head>
        <body>
        </body>
</html>
```

When to Use Internal and External JavaScript Code?

If you have only a few lines of code that is specific to a particular webpage, then it is better to keep your JavaScript code internally within your HTML document.

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

Don't Miss:

- TypeScript Tutorial: What is, Interface, Enum, Array with Example
- QuickSort Algorithm in JavaScript
- 15 Best FREE JavaScript Certification Courses Online (2024)

← Prev

Report a Bug

Next →

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,
United States

Write For Us
Contact Us

Career Suggestion
SAP Career Suggestion Tool
Software Testing as a Career

Interesting
eBook
Blog
Quiz
SAP eBook
Privacy Manager

Practical Code Examples using JavaScript

By : James Hartman March 9, 2024

Example#1: JavaScript Multiplication Table

Create a simple multiplication table asking the user the number of rows and columns he wants.

Solution:

```
<html>
<head>
  <title>Multiplication Table</title>
  <script type="text/javascript">
    var rows = prompt("How many rows for your multiplication table?");
    var cols = prompt("How many columns for your multiplication table?");
    if(rows == "" || rows == null)
                rows = 10;
    if(cols== "" || cols== null)
                cols = 10;
    createTable(rows, cols);
    function createTable(rows, cols)
    {
      var j=1;
      var output = "<table border='1' width='500' cellspacing='0'cellpadding='5'>";
      for(i=1;i<=rows;i++)
      {
        output = output + "<tr>";
        while(j<=cols)
        {
                  output = output + "<td>" + i*j + "</td>";
                  j = j+1;
                }
                  output = output + "</tr>";
                  j = 1;
    }
    output = output + "</table>";
    document.write(output);
    }
  </script>
</head>
<body>
</body>
</html>
```

Example#2: JS Forms Example:

Create a sample form program that collects the first name, last name, email, user id, password and confirms password from the user. All the inputs are mandatory and email address entered should be in correct format. Also, the values entered in the password and confirm password textboxes should be the same. After validating using JavaScript, In output display proper error messages in red color just next to the textbox where there is an error.

Solution with Source Code:

```
<html>
<head>
  <title>Form Validation</title>

  <script type="text/javascript">
    var divs = new Array();
    divs[0] = "errFirst";
    divs[1] = "errLast";
    divs[2] = "errEmail";
    divs[3] = "errUid";
    divs[4] = "errPassword";
    divs[5] = "errConfirm";
    function validate()
        {
      var inputs = new Array();
      inputs[0] = document.getElementById('first').value;
      inputs[1] = document.getElementById('last').value;
      inputs[2] = document.getElementById('email').value;
      inputs[3] = document.getElementById('uid').value;
      inputs[4] = document.getElementById('password').value;
      inputs[5] = document.getElementById('confirm').value;
      var errors = new Array();
      errors[0] = "<span style='color:red'>Please enter your first name!</span>";
      errors[1] = "<span style='color:red'>Please enter your last name!</span>";
      errors[2] = "<span style='color:red'>Please enter your email!</span>";
      errors[3] = "<span style='color:red'>Please enter your user id!</span>";
      errors[4] = "<span style='color:red'>Please enter your password!</span>";
      errors[5] = "<span style='color:red'>Please confirm your password!</span>";
      for (i in inputs)
      {
        var errMessage = errors[i];
        var div = divs[i];
        if (inputs[i] == "")
                document.getElementById(div).innerHTML = errMessage;
        else if (i==2)
        {
          var atpos=inputs[i].indexOf("@");
          var dotpos=inputs[i].lastIndexOf(".");
          if (atpos<1 || dotpos<atpos+2 || dotpos+2>=inputs[i].length)
                document.getElementById('errEmail').innerHTML = "<span style='color: red'>Enter a
valid email address!</span>";
          else
                document.getElementById(div).innerHTML = "OK!";
        }
        else if (i==5)
        {
          var first = document.getElementById('password').value;
          var second = document.getElementById('confirm').value;
          if (second != first)
                document.getElementById('errConfirm').innerHTML = "<span style='color: red'>Your
passwords don't match!</span>";
          else
                document.getElementById(div).innerHTML = "OK!";
        }
        else
                document.getElementById(div).innerHTML = "OK!";
       }
```

```
     }
        function finalValidate()
        {
          var count = 0;
          for(i=0;i<6;i++)
          {
            var div = divs[i];
            if(document.getElementById(div).innerHTML == "OK!")
            count = count + 1;
          }
          if(count == 6)
                document.getElementById("errFinal").innerHTML = "All the data you entered is
correct!!!";
        }
   </script>
</head>
<body>
        <table id="table1">
      <tr>
        <td>First Name:</td>
        <td><input type="text" id="first" onkeyup="validate();" /></td>
        <td><div id="errFirst"></div></td>
      </tr>
      <tr>
        <td>Last Name:</td>
        <td><input type="text" id="last" onkeyup="validate();"/></td>
        <td><div id="errLast"></div></td>
      </tr>
      <tr>
        <td>Email:</td>
        <td><input type="text" id="email" onkeyup="validate();"/></td>
        <td><div id="errEmail"></div></td>
      </tr>
      <tr>
        <td>User Id:</td>
        <td><input type="text" id="uid" onkeyup="validate();"/></td>
        <td><div id="errUid"></div></td>
      </tr>
      <tr>
        <td>Password:</td>
        <td><input type="password" id="password" onkeyup="validate();"/></td>
        <td><div id="errPassword"></div></td>
      </tr>
      <tr>
        <td>Confirm Password:</td>
        <td><input type="password" id="confirm" onkeyup="validate();"/></td>
        <td><div id="errConfirm"></div></td>
      </tr>
      <tr>
        <td><input type="button" id="create" value="Create" onclick="validate();finalValidate();"/>
</td>
        <td><div id="errFinal"></div></td>
      </tr>
        </table>
</body>
</html>
```

Example#3: POPUP Message using Event:

Display a simple message “Welcome!!!” on your demo webpage and when the user hovers over the message, a popup should be displayed with a message “Welcome to my WebPage!!!”.

Solution:

```
<html>
    <head>
        <title>Event!!!</title>
        <script type="text/javascript">
            function trigger() {
                document.getElementById("hover").addEventListener("mouseover", popup);
                function popup() {
                    alert("Welcome to my WebPage!!!");
                }
            }
        </script>
        <style>
            p{
                font-size: 50px;
                position: fixed;
                left: 550px;
                top: 300px;
            }
        </style>
    </head>
    <body onload="trigger();">
        <p id="hover">Welcome!!!</p>
    </body>
</html>
```


Don't Miss:

- TypeScript Tutorial: What is, Interface, Enum, Array with Example
- QuickSort Algorithm in JavaScript
- 15 Best FREE JavaScript Certification Courses Online (2024)

← Prev Report a Bug Next →

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,
United States

Best JavaScript Unit Testing Frameworks

By : James Hartman March 9, 2024

In this JavaScript Unit Testing tutorial, we will learn:

What is JavaScript?

JavaScript is a programming language which is defined as high level, dynamic and interpreted language used with HTML web applications. JavaScript is also used for other than web documents such as PDFs and desktop widgets and became popular for server-side web application. JavaScript is object-based script and follows the prototype.

Table of Content:

JavaScript Unit Testing

JavaScript Unit Testing is a testing method in which JavaScript test code written for a web page or web application module 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.

Each and every suite contains a number of 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 in JavaScript Unit Testing

There are certain problems one can find while performing Unit Testing in JavaScript such as:

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

To avoid such issues what you can do is:

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

Best JavaScript Unit Testing Frameworks

Following is a curated list of popular JavaScript Unit Testing Frameworks and Tools which are widely used:

1. Unit.js: It is known as an open source assertion library running on browser and Node.js. It is extremely compatible with other JavaScript Unit Testing framework like Mocha, Karma, Jasmine, QUnit, Protractor, etc. Provides the full documented API of assertion list

2. QUnit: It is used for both client-side as well as server-side JavaScript Unit Testing. This Free JavaScript testing framework is used for jQuery projects. It follows Common JS unit testing Specification for unit testing in JavaScript. It supports the Node Long-term Support Schedule.

3. Jasmine: Jasmine is the behavior-driven development framework to unit test JavaScript. It is used for testing both synchronous and asynchronous JavaScript Code. It does not require DOM and comes with the easy syntax that can be Written for any test.

4. Karma: Karma is an open source productive testing environment. Easy workflow control Running on the command line. Offers the freedom to write the tests with Jasmine, Mocha, and QUnit. You can run the test on real devices with easy debugging.

5. Mocha: Mocha runs on Node.js and in the browser. Mocha performs asynchronous Testing in a simpler way. Provides accuracy and flexibility in reporting. Provides tremendous support of rich features such as test-specific timeouts, JavaScript APIs etc.

6. Jest: Jest is used by Facebook so far to test all of the JavaScript code. It provides the 'zero-configuration' testing experience. Supports independent and non-interrupting running test without any conflict. Do not require any other setup configuration and libraries.

7. AVA: AVA is simple JavaScript Unit Testing Framework. Tests are being run in parallel and serially. Parallel tests run without interrupting each other. AVA Supports asynchronous testing as well. AVA uses subprocesses to run the unit test JavaScript.

Summary

- JavaScript Unit Testing may become tedious and tricky sometimes as it is performed for the front-end basically. One can use the JS libraries to for adding little ease. The challenge might become bigger as JavaScript is getting incorporated with Node.js and TypeScript.
- You should keep three things in mind while performing the test such as; The feature that needs to be tested, the final output and the expected output. Some tools and JavaScript testing framework may help you in performing this task. Above mentioned tool lists is mentioned with most popular and useful frameworks used for Unit Testing JavaScript.
- More than these with upcoming challenges in performing testing there, some more powerful frameworks and tools may get evolved in future.

- QuickSort Algorithm in JavaScript
- 15 Best FREE JavaScript Certification Courses Online (2024)

← Prev

Report a Bug

Next →

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,
United States

TypeScript Tutorial: What is, Interface, Enum, Array with Example

By : James Hartman March 9, 2024

What is TypeScript?

TypeScript is a superset of JavaScript. TypeScript is pure object-oriented programming language that supports classes, interfaces, etc. It is an open-source language developed by Microsoft which statically compiles the code to JavaScript. It can easily run in a browser or Nodejs.

All the latest features released for ECMAScript are supported in TypeScript and in addition to it TypeScript has its own object-oriented features like interfaces, ambient declaration, class inheritance, etc. which helps in developing a large application which otherwise would be difficult to do in JavaScript.

Table of Content:

How to Download and Install TypeScript

Here is the step by step process to download and install TypeScript:

Step 1) Download and Install Nodejs

Go to the official site of nodejs : https://nodejs.org/en/download/ and download and install nodejs as per your operating system. The detailed instruction on how to download nodejs is available here: https://www.guru99.com/download-install-node-js.html

Step 2) Check Nodejs and npm version

To check if nodejs and npm is installed just check the version in your command prompt.

```
D:\typeproject>node --version
V10.15.1

D:\typeproject>npm --version
6.4.1
```

So you have nodejs v10 and npm 6 installed.

Step 3) TypeScript Installation

Create your project directory typeproject/ and run npm init, as shown in the command below:

```
npm init
```

Step 4) Start the Installation

Now, we will create package .json which will store the dependencies for our project.

Once done install TypeScript as follows:

```
npm -g install typescript
```

The above command will take care of installing TypeScript. Adding “-g” to npm install will install TypeScript globally. The advantage of using -g is that you will be able to use TypeScript tsc command from any directory as it is installed globally. In case you don’t want to install TypeScript globally use below command:

```
npm --save install typescript
```

Create src/ folder in your project directory and in src/ folder create TypeScript file test.ts and write your code.

Example : test.ts

```
function add(x:number, y:number) {
        return x+y;
}

let sum = add(5,10);
console.log(sum);
```

Compile TypeScript code to Javascript

To compile above code use following command:

If TypeScript is installed globally use below command:

```
tsc test.ts
```

If TypeScript is installed local to your project you need to use the path of TypeScript from node_modules as shown:

```
node_modules/typescript/bin/tsc test.ts
```

The above command will create a test.js file and will be have code compiled to javascript.

Example : test.js

```
function add(x, y) {
    return x + y;
}
var sum = add(5, 10);
console.log(sum);
```

Execute Javascript using Nodejs

In this TypeScript tutorial, we will execute test.js in nodejs as follows:

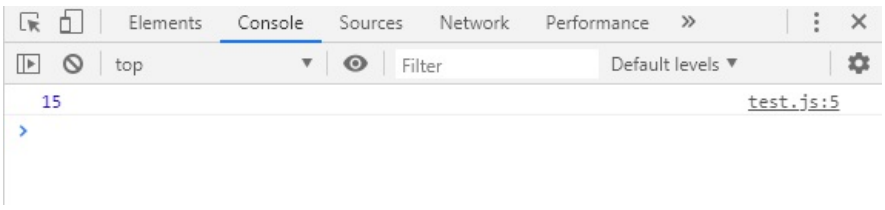
```
D:\typeproject\src>node test.js
15
```

The value consoled is displayed on the execution of test.js

Execute JavaScript in Browser

Example:

```
<html>
<head></head>
<body>
<script type="text/javascript" src="test.js"></script>
</body>
</html>
```



Compile TypeScript code to Javascript using EcmaScript version

TypeScript supports all the Ecmascript features released, and developers can use the same while coding. But not all the new features support on older browsers, due to which you need to compile javascript to an older version of Ecmascript. TypeScript provides compiler options which can do so.

Example : test.ts

```
var addnumbers = (a, b) => {
```

```
    return a+b;
}

addnumbers(10, 20);
```

To compile to the ES version of your choice, you can use the target or t option in your command as follows:

```
tsc --target ES6  test.ts

OR

tsc -t ES6 test.ts
```

By default, the target is ES3.In case you want to change it, you can use the above command.

At present we will use ES6 in this TypeScript tutorial as the target:

```
tsc --target ES6  test.ts
```

test.ts to test.js

```
var addnumbers = (a, b) => {
    return a+b;
}

addnumbers(10, 20);
```

The code remains as it is, as the arrow function you have used is a ES6 feature and the same when compiled to ES6 is not changed.

By default the target is ES3 so without target you get test.js as :

```
var addnumbers = function (a, b) {
    return a + b;
};
addnumbers(10, 20);
```

So over here, the fat arrow is changed to a normal anonymous function.

Don't Miss:

- Execute JavaScript Online
- QuickSort Algorithm in JavaScript
- 15 Best FREE JavaScript Certification Courses Online (2024)

Variables in TypeScript

Variables are used to store values, and the value can be a string, number, Boolean, or an expression. When it comes to variables in TypeScript, they are similar to JavaScript. So let's learn to declare and assign value to variables in TypeScript.

Variables cannot be used in code without defining. To declare a variable you can use

var keyword,

let keyword

const keyword

Working with variables in TypeScript is similar to javascript, and users familiar with javascript will find it very easy. Only variables like let and const are not much used in comparison to var.

Declaring variables using var

Syntax:

```
var firstname = "Roy";
```

Let us take a look at few TypeScript examples to understand the working of var keyword and also the scope of variables declared using var keyword.

Example 1:

```
var  k = 1; // variable k will have a global scope

function test() {
    var c = 1; // variable c is local variable and will be accessible inside function test, it will
not be available outside the function.
    return k++;
}

test(); // output as  1
test(); // output as 2
alert(c); // will throw error , Uncaught ReferenceError: c is not defined
```

Example 2:

```
var t = 0; // variable t is declared in global scope.
function test() {
    var t = 10; //variable t is again redeclared inside function with value 10, so here t is local
to the function and changes done to it will remain inside the function.
    return t;

}
test(); // will return 10.
console.log(t); // will console 0.
```

Example 3:

```
var i = 0;
function test() {
    if (i>0) {
      var t = 1;
    }
    return t;
}

test(); // the value returned will be undefined. The if-block has the variable which gets executed
when I> 0. Over here the if-block is not expected but you are still having a reference to the
variable t, and it returns undefined, this is because var defined variables once defined inside a
function will have reference to it inside the function.
i++; // here value of i is incremented.
test(); // since i >0 the if block is executed and value returned is 1.
```

Declaring variables using let

The TypeScript syntax for let is as given below:

Syntax:

```
let name="Roy";
```

The working of let variable is almost same as var, but with a small difference and will understand the same using an TypeScript example.

Example:

```
let i = 1;
function test() {
    if (i>0) {
          let t = 1;
    }
    return t;
}

test(); // throws an error : Uncaught ReferenceError: t is not defined.
```

Above TypeScript example throws an error, but the same would have worked fine if it was with the var keyword. Variables using let are available within the block scope declared, for example, the variable t is available only inside the if-block and not to the entire function.

Also if you happen to declare a variable inside any function, or for-loop, while-loop, TypeScript switch block, it will be available to you only inside that block and no reference to it outside the block, and it will throw an error if the variable is used outside the block. This is the main difference between var and let keyword declared variables.

Declaring variables using const

Const means constant variables. They are similar to let variables, with the only difference, is that once a value is assigned to it cannot be changed.

Syntax:

```
const name;
```

Example:

```
const age = "25";
age="30"; // will throw an error : Uncaught TypeError: Assignment to constant variable.
```

So users can use const variables only in cases when they know they don't have to change the values assigned to it.

Types in TypeScript

TypeScript is a strongly typed language, whereas javascript is not. A variable which has a value defined as a string can be changed to a number without any issues in Javascript. The same is not tolerated in TypeScript. In TypeScript, the type to a variable is defined at the start only and through the execution, it has to maintain the same type any changes to it will lead to a compile-time error during compilation to javascript.

Following are the types :

- Number
- String
- Boolean
- Any
- Void

Number

Takes only integers, floats, fractions, etc.

Syntax:

```
let a :number = 10;
let marks :number = 150;
let price :number = 10.2;
```

Here are some important methods which can be used on Number types:

toFixed() – it will convert the number to a string and will keep decimal places given to the method.

toString() – this method will convert number to a string.

valueOf() – this method will give back the primitive value of the number.

toPrecision() – this method will format the number to a specified length.

Example : with all String methods

```
let _num :number = 10.345;
_num.toFixed(2); // "10.35"
_num.valueOf(); // 10.345
_num.toString(); // "10.345"
_num.toPrecision(2); //"10"
```

String

String: only string values

Syntax:

```
let str :string = "hello world";
```

Here are some important methods which can be used on String types:

- split() – this method will split the string into an array.
- charAt() – this method will give the first character for the index given.
- indexOf() – this method will give the position of the first occurrence for the value given to it.
- Replace () – this method takes 2 strings, first the value to search in the string and if present will replace it will the 2nd one and will give a new string back.
- Trim () – this method will remove white spaces from both sides of the string.
- substr() – this method will give a part of the string which will depend on the position given as start and end.
- substring() – this method will give a part of the string which will depend on the position given as start and end. The character at the end position will be excluded.
- toUpperCase() -will convert the string to uppercase
- toLowerCase() – will convert the string to lowercase.

Example:

```
let _str:string = "Typescript";

_str.charAt(1); // y
_str.split(""); //["T", "y", "p", "e", "s", "c", "r", "i", "p", "t"]
_str.indexOf("s"); //4 , gives -1 is the value does not exist in the string.
_str.replace("Type", "Coffee"); //"Coffeescript"
_str.trim(); //"Typescript"
_str.substr(4, _str.length); //"script"
_str.substring(4, 10); //"script"
_str.toUpperCase();//"TYPESCRIPT"
```

```
_str.toLowerCase();//"typescript"
```

Boolean

Will accept logical values like true, false, 0, and 1.

Syntax:

```
let bflag :boolean = 1;
let status :boolean = true;
```

Any

Syntax:

```
let a :any = 123
a = "hello world"; // changing type will not give any error.
```

Variables declared using any type can take the variable as a string, number, array, boolean or void. TypeScript will not throw any compile-time error; this is similar to the variables declared in JavaScript. Make use of any type variable only when you aren't sure about the type of value which will be associated with that variable.

Void

Void type is mostly used as a return type on a function which does not have anything to return.

Syntax:

```
function testfunc():void{
 //code here
}
```

TypeScript Array

An Array in TypeScript is a data type wherein you can store multiple values. Let's learn how to declare and assign values for Array operations in TypeScript.

Since TypeScript is a strongly typed language, you have to tell what will be the data type of the values in an array. Otherwise, it will consider it as of type any.

Declare and Initialize an Array

Syntax:

```
let nameofthearray : Array<typehere>
```

Example

```
let months: Array<string> = ["Jan", "Feb", "March", "April", "May", "June", "July", "Aug", "Sept",
"Oct", "Nov", "Dec"]; //array with all string values.


let years: Array<number> = [2015, 2016, 2017, 2018, 2019]; //array will all numbers

let month_year: Array<string | number> = ["Jan", 2015, "Feb", 2016]; //array with string and numbers
mixed.

let alltypes: Array<any> = [true, false, "Harry", 2000, { "a": "50", "b": "20" }]; //array of all
types boolean, string , number , object etc.
```

Different Ways to access elements from an Array

To get the elements from an array, the values starts from index 0 to the length of the array.

Example:

```
let years: Array<number> = [ 2016, 2017, 2018, 2019]; //array will all numbers
years[0]; // output will be 2016
years[1]; // output will be 2017
years[2]; // output will be 2018
years[3]; // output will be 2019
```

You can also get the elements from an array using TypeScript for loop as shown below:

Using TypeScript for loop

```
let years: Array<number> = [ 2016, 2017, 2018, 2019];
for (let i=0;i<=years.length; i++) {
     console.log(years[i]);
}
Output:
2016
2017
2018
2019
```

Using for-in loop

```
let years: Array<number> = [ 2016, 2017, 2018, 2019];
for (let i in years) {
     console.log(years[i])
}

Output:
2016
2017
2018
2019
```

Using for-of loop

```
let years: Array<number> = [ 2016, 2017, 2018, 2019];
for (let  i of years) {
     console.log(i)
}
Output:
2016
2017
2018
2019
```

Using foreach loop

```
let years: Array<number> = [ 2016, 2017, 2018, 2019];
years.forEach(function(yrs, i) {
  console.log(yrs);
});
Output:
2016
2017
2018
2019
```

TypeScript Array Methods

TypeScript 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 array name.method().

length property

=> If you want to know the number of elements in an array, you can use the length 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 value as the last item of the array.

concat method

=> You can join two arrays into one array element.

Example for length property

```
let months: Array<string> = ["Jan", "Feb", "March", "April", "May", "June", "July", "Aug", "Sept",
"Oct", "Nov", "Dec"]; //array with all string values.

console.log(months.length);  // 12
```

Example for reverse method

```
let months: Array<string> = ["Jan", "Feb", "March", "April", "May", "June", "July", "Aug", "Sept",
"Oct", "Nov", "Dec"]; //array with all string values.

console.log(months.reverse());  //  ["Dec", "Nov", "Oct", "Sept", "Aug", "July", "June", "May",
"April", "March", "Feb", "Jan"]
```

Example for sort method

```
let months: Array<string> = ["Jan", "Feb", "March", "April", "May", "June", "July", "Aug", "Sept",
"Oct", "Nov", "Dec"]; //array with all string values.

console.log(months.sort()); // ["April", "Aug", "Dec", "Feb", "Jan", "July", "June", "March", "May",
"Nov", "Oct", "Sept"]
```

Example for pop method

```
let months: Array<string> = ["Jan", "Feb", "March", "April", "May", "June", "July", "Aug", "Sept",
"Oct", "Nov", "Dec"]; //array with all string values.

console.log(months.pop()); //Dec
```

Example for shift method

```
let months: Array<string> = ["Jan", "Feb", "March", "April", "May", "June", "July", "Aug", "Sept",
"Oct", "Nov", "Dec"]; //array with all string values.

console.log(months.shift()); // Jan
```

Example for push method

```
let years: Array<number> = [2015, 2016, 2017, 2018, 2019]; //array will all numbers
console.log(years.push(2020));
years.forEach(function(yrs, i) {
  console.log(yrs); // 2015 , 2016,2017, 2018, 2019,2020
```

```
});
```

Example for concat method

```
let array1: Array<number> = [10, 20, 30];
let array2: Array<number> = [100, 200, 300];
console.log(array1.concat(array2)); //[10, 20, 30, 100, 200, 300]
```

Class in TypeScript

TypeScript is a superset of JavaScript, so whatever possible to do in JavaScript is also possible in TypeScript. Class is a new feature added from ES6 onward, so earlier in JavaScript the class type functionality was tried using a function with prototype functionality to reuse code. Using class, you can have our code almost close to languages like java, c#, python, etc., where the code can be reused. With the feature of class in TypeScript/JavaScript, it makes the language very powerful.

Defining a Class in TypeScript

Here is a basic class syntax in TypeScript:

```
class nameofclass {
     //define your properties here

    constructor() {
     // initialize your properties here
    }

   //define methods for class
}
```

Example: A working example on Class

```
class Students {
    age : number;
    name : string;
    roll_no : number;

    constructor(age: number, name:string, roll_no: number) {
        this.age = age;
        this.name = name;
        this.roll_no = roll_no;
    }

    getRollNo(): number {
        return this.roll_no;
    }

    getName() : string {
        return this.name;
    }

    getAge() : number {
```

```
    getAge() : number {
        return this.age;
    }
}
```

In the above example, you have a class called Students. It has properties age, name, and roll_no.

Constructor in a TypeScript Class

The class Students example we have defined above, it has a constructor as shown below :

```
constructor(age: number, name:string, roll_no: number) {
        this.age = age;
        this.name = name;
        this.roll_no = roll_no;
    }
```

The constructor method has params age, name, and roll_no. The constructor will take care of initializing the properties when the class is called. The properties are accessed using this keyword. Example this.age to access age property, this.roll_no to access roll_no, etc. You can also have a default constructor, as shown below:

```
constructor () {}
```

Methods inside a TypeScript Class

The class Students example there are methods defined for example getRollNo(), getName(), getAge() which are used to give details of properties roll_no, name and age.

```
getRollNo(): number {
        return this.roll_no;
}

getName() : string {
        return this.name;
}

getAge() : number {
        return this.age;
}
```

Creating Instance of Class in TypeScript

Example:

In TypeScript to create an instance of a class you need to use the new operator. When we create an instance of a class using new operator we get the object which can access the properties and methods of the class as shown below:

```
let student_details = new Students(15, "Harry John", 33);
student_details.getAge(); // 15
student_details.getName(); // Harry John
```

Compiling TypeScript Class to JavaScript

You can use tsc command as shown below to compile to Javascript.

```
Command: tsc  Students.ts
```

The output of Javascript code on compilation is as shown below:

```
var Students = /** @class */ (function () {
    function Students(age, name, roll_no) {
        this.age = age;
        this.name = name;
        this.roll_no = roll_no;
    }
    Students.prototype.getRollNo = function () {
        return this.roll_no;
    };
    Students.prototype.getName = function () {
        return this.name;
    };
    Students.prototype.getAge = function () {
        return this.age;
    };
    return Students;
}());
```

In Javascript, the Class is converted into a self invoked function.

Class Inheritance

Classes can be inherited using the extend keyword in TypeScript.

Class Inheritance Syntax:

```
class A {
     //define your properties here

    constructor() {
     // initialize your properties here
    }

   //define methods for class

}

class B extends A {
 //define your properties here
```

```
    constructor() {
     // initialize your properties here
    }

   //define methods for class


}
```

class B will be able to share class A methods and properties.

Here is a working example of a class using Inheritance

```
class Person {
    name: string;
    age: number;

    constructor(name: string, age: number) {
        this.name = name;
        this.age = age;
    }

    getName(): string {
        return this.name;
    }

    getAge(): number {
        return this.age;
    }
}

class Student extends Person {
    tmarks: number;
    getMarks(): number {
        return this.tmarks;
    }

    setMarks(tmarks) {
        this.tmarks = tmarks;
    }
}

let _std1 = new Student('Sheena', 24);
_std1.getAge(); // output is 24
_std1.setMarks(500);
_std1.getMarks(); // output is 500
```

You have two classes, Person and Student. Student class extends Person, and the object created on Student is able to access its own methods and properties as well as the class it has extended.

Now let us add some more changes to the above class.

Example:

```
class Person {
    name: string;
    age: number;

    constructor(name: string, age: number) {
        this.name = name;
        this.age = age;
    }

    getName(): string {
        return this.name;
    }

    getAge(): number {
        return this.age;
    }
}

class Student extends Person {
    tmarks: number;
    constructor(name: string, age: number, tmarks: number) {
        super(name, age);
    }
    getMarks(): number {
        return this.tmarks;
    }

    setMarks(tmarks) {
        this.tmarks = tmarks;
    }
}

let _std1 = new Student('Sheena', 24, 500);
_std1.getAge(); // output is 24
_std1.getMarks(); // output is 500
```

The changes that you have added in comparison to the previous example is there is a constructor defined in class Student. The constructor has to take the same params as the base class and add any additional params of its own if any.

In TypeScript you need to call super will all the params as the bases params in it. This has to be the first thing to be done inside the constructor. The super will execute the constructor of the extended class.

Access Modifiers in TypeScript

TypeScript supports public, private, and protected access modifiers to your methods and properties. By default, if access modifiers are not given the method or property is considered as public, and they will be easily accessible from the object of the class.

In case of private access modifiers, they are not available to be accessed from the object of the class and meant to be used inside the class only. They are not available for the inherited class.

In case of protected access modifiers, they are meant to be used inside the class and the inherited class and will not be accessible from the object of the class.

Example:

```
class Person {
    protected name: string;
    protected age: number;

    constructor(name: string, age: number) {
        this.name = name;
        this.age = age;
    }

    private getName(): string {
        return this.name;
    }

    getDetails(): string {
        return "Name is "+ this.getName();
    }
}

class Student extends Person {
    tmarks: number;
    constructor(name: string, age: number, tmarks: number) {
        super(name, age);
        this.tmarks = tmarks;
    }
    getMarks(): number {
        return this.tmarks;
    }

    getFullName(): string {
        return this.name;
    }

    setMarks(tmarks) {
        this.tmarks = tmarks;
    }
}

let _std1 = new Student('Sheena', 24, 500);
_std1.getMarks(); // output is 500
_std1.getFullName(); // output is Sheena
_std1.getDetails(); // output is Name is Sheena
```

- Private: properties or methods cannot be accessed by the object of the class and also the derived class, they are meant to be used internally inside the class.
- Protected: properties and methods also cannot be accessed by the object created. They are accessible from inside the class and available to the class extending it.
- Public: properties and methods are declared without any keyword. They are easily accessed using the object of the class from outside.

Interface in TypeScript

One of the core features of TypeScript is interfaces. The interface is a set of a rule defined which needs to be implemented by the entity using it. The entity can be a class, function, or variable. An interface can be made up of properties and methods. You can define properties as optional using “?” syntax for that property or method. The interface adds a strong type check for any function, variable, or class implementing the interface.

Syntax of an Interface in TypeScript

```
interface Dimension {
    width: string;
    height: string;
}
```

You have defined an interface named as Dimension which has properties width and height, and both have type as a string.

Now this interface can be implemented by a variable, a function, or a class. Here is the example of variable implementing the interface Dimension.

Example:

```
interface Dimension {
    width: string;
    height: string;
}

let _imagedim: Dimension = {
    width: "100px",
    height: "200px"
};
```

The signature of the interface Dimension has width and height, and both are mandatory. In case while implementing the interface, any of the property is missed, or the type is changed, it will give a compile time error while compiling the code to javascript.

The above code, when compiled to javascript, looks as follows:

```
var _imagedim = {
    width: "100px",
    height: "200px"
};
```

Let us now see how to use an interface with a function.

Using Interface on a function as a return type

Example:

```
interface Dimension {
    width: string;
    height: string;
}

function getDimension() : Dimension {
    let width = "300px";
    let height = "250px";
    return {
        width: width,
        height: height
    }
}
```

In the above example, the interface Dimension is implemented on the function getDimension() as the return type. The return type of getDimension() has to match with the properties and type mentioned for Interface Dimension.

The compiled code to Javascript will be as follows:

```
function getDimension() {
    var width = "300px";
    var height = "250px";
    return {
        width: width,
        height: height
    };
}
```

During compilation, if the return type does not match with the interface, it will throw an error.

Interface as function parameter

```
interface Dimension {
    width: string;
    height: string;
}

function getDimension(dim: Dimension) : string {
    let finaldim  = dim.width +"-"+ dim.height;
    return finaldim;
}

getDimension({width:"300px", height:"250px"}); // will get "300px-250px"
```

So above example, you have used Interface Dimension as a parameter to the function getDimension(). When you

call the function, you need to make sure the parameter passed to it matches to the Interface rule defined.

The compiled code to Javascript will be as follows:

```
function getDimension(dim) {

    var finaldim = dim.width + "-" + dim.height;
    return finaldim;
}
getDimension({ width: "300px", height: "250px" });
```

Class implementing Interface

To make use of interface with a class, you need to use the keyword implements.

Syntax for Class implementing an interface:

```
class NameofClass implements InterfaceName {
}
```

Following example shows working of interface with class.

```
interface Dimension {
    width : string,
    height: string,
    getWidth(): string;
}

class Shapes implements Dimension {
    width: string;
    height: string;
    constructor (width:string, height:string) {
        this.width = width;
        this.height = height;
    }
    getWidth() {
        return this.width;
    }
}
```

In the above example, you have defined interface Dimension with properties width and height both of type string and a method called getWidth() which has return value as a string.

The compiled code to Javascript will be as follows:

```
var Shapes = /** @class */ (function () {
    function Shapes(width, height) {
        this.width = width;
        this.height = height;
    }
    Shapes.prototype.getWidth = function () {
        return this.width;
```

```
        return this.width;
    };
    return Shapes;
}());
```

Functions in TypeScript

Functions are set of instructions performed to carry out a task. In Javascript, most of the code is written in the form of functions and plays a major role. In TypeScript, you have class, interfaces, modules, namespaces available, but still, functions play an important role. The difference between the function in javascript and TypeScript function is the return type available with TypeScript function.

JavaScript function:

```
function add (a1, b1) {
   return a1+b1;
}
```

TypeScript function:

```
function  add(a1 : number, b1: number) : number {
    return a1 + b1;
}
```

In the above functions, the name of the function is added, the params are a1, and b1 both have a type as a number, and the return type is also a number. If you happen to pass a string to the function, it will throw a compile-time error while compiling it to JavaScript.

Making call to the function: add

```
let  x = add(5, 10) ;  // will return 15
let  b = add(5); // will throw an error : error TS2554: Expected 2 arguments, but got 1.
let c = add(3,4,5); // will throw an error : error TS2554: Expected 2 arguments, but got 3.
let t = add("Harry", "John");// will throw an error :  error TS2345: Argument of type '"Harry"' is
not assignable to parameter of type 'number'.
```

The params a1 and b1 are mandatory parameters and will throw an error if not received in that manner. Also, the param type and return type is very important and cannot change once defined.

Optional parameters to a function

In javascript, all the parameters to the functions are optional and considered as undefined if not passed. But same is not the case with TypeScript, once you define the params you need to send them too, but in case you want to keep any param optional, you can do so by using? against the param name as shown below:

```
function getName(firstname: string, lastname?: string): string {
    return firstname + lastname;
}
```

```
let a = getName("John"); // will return Johnundefined.
let b = getName("John", "Harry"); // will return JohnHarry
let c = getName("John", "H", "Harry"); // error TS2554: Expected 1-2 arguments, but got 3.
```

Please note that the optional params are to be defined in a function at the last only, you cannot have the first param as optional and second param as mandatory.When you call the function with one param compiler will throw an error. So it is necessary to keep the optional params at the end.

Assign Default Values to Params

You can assign default values to params as shown below:

```
function getName(firstname: string, lastname = "Harry"): string {
    return firstname + lastname;
}

let a = getName("John"); // will return JohnHarry
let b = getName("John", "H"); // will return JohnH
```

Similar to optional params, here too default initialized params has to be kept at the end in a function.

Rest Parameters

You have seen how TypeScript handles mandatory params, optional params, and the default value initialized params. Now, will take a look at rest params. Rest params are a group of optional params defined together, and they are defined using three dots (…) followed by the name of the param, which is an array.

Syntax for Rest params:

```
function testFunc(a: string, ...arr: string[]) :string {
    return a + arr.join("");
}
```

As shown above, the rest params are defined using (…param-name); the rest param is an array prefixed by three dots. The array will have all the params passed to it. You can call the function, as shown in the example below:

Example:

```
let a = testFunc("Monday", "Tuesday", "Wednesday", "Thursday"); // will get output as
MondayTuesdayWednesdayThursday
```

Arrow Functions

An arrow function is one of the important features released in ES6, and it is available in TypeScript too. Arrow function syntax has a fat arrow in it due to which the function is called an arrow function.

Arrow function Syntax:

```
var nameoffunction = (params) => {
 // code here
}
```

What is the use of Arrow Function?

Let’s take a look at the example to understand the use case of Arrow function:

Example:

```
var ScoreCard = function () {
    this.score = 0;

    this.getScore = function () {
        setTimeout(function () {
            console.log(this.score);    // gives undefined.
        }, 1000);
    }
}

var a = new ScoreCard();
a.getScore();
```

You have created an anonymous function which has a property this. Score initialize to 0 and a method getScore which internally has a setTimeout, and in 1 second it consoles this.score. The consoled value gives undefined though you have this.score defined and initialized. The issue here is with this keyword. The function inside setTimeout has its own this, and it tries to refer the score internally, and since it is not defined, it gives undefined.

The same can be taken care using Arrow function as shown below:

```
var ScoreCard = function () {
    this.score = 0;

    this.getScore = function () {
        setTimeout(()=>{
            console.log(this.score);   // you get  0
        }, 1000);
    }
}

var a = new ScoreCard();
a.getScore();
```

You have changed the function inside setTimeout to a arrow function as shown below:

```
setTimeout(()=>{
            console.log(this.score);   // you get  0
        }, 1000);
```

An arrow function does not have its own this defined and it shares its parent this, so variables declared outside are easily accessible using this inside an arrow function. They are useful because of the shorter syntax as well as for callbacks, event handlers, inside timing functions, etc.

TypeScript Enums

TypeScript Enum is an object which has a collection of related values stored together. Javascript does not support enums. Most of the programming language like Java, C, C++ supports TypeScript Enum and it also available with TypeScript too. Enums are defined using keyword enum.

How to declare an Enum?

Syntax:

```
enum NameofEnum {
   value1,
   value2,
    ..
}
```

Example: Enum

```
enum Directions {
North,
South,
East,
West
}
```

In the above example, you have defined an enum called Directions. The value given is North, South, East, West. The values are numbered from 0 for the first value in the enum and subsequently increments by 1 for the next value.

Declare An Enum with a numeric value

By default, if an enum is not given any value, it considers it a number starting from 0. Following example shows an enum with a numeric value.

```
enum Directions {
North = 0,
South = 1,
East =2,
West =3
}
```

You may also assign a start value to the enum and the next enum values will get the incremented values. For example:

```
enum Directions {
North = 5,
```

```
South, // will be 6
East, // 7
West // 8
}
```

Now the enum value North starts with 5, so South will get value as 6, East = 7 and West = 8.

You may also assign values of your choice instead of taking the default ones. For Example:

```
enum Directions {
North = 5,
South = 4,
East = 6,
West = 8
}
```

How to access an Enum?

Following example shows how to make use of Enum in your code:

```
enum Directions {
    North,
    South,
    East,
    West
}

console.log(Directions.North); // output is  0
console.log(Directions["North"]); // output is 0
console.log(Directions[0]); //output is North
```

The compiled code to javascript is as follows:

```
var Directions;
(function (Directions) {
    Directions[Directions["North"] = 0] = "North";
    Directions[Directions["South"] = 1] = "South";
    Directions[Directions["East"] = 2] = "East";
    Directions[Directions["West"] = 3] = "West";
})(Directions || (Directions = {}));
console.log(Directions.North);
console.log(Directions["North"]);
console.log(Directions[0]);
```

Since Javascript does not support enums, it converts the enum into a self invoked functions as shown above.

Declare An Enum with a string value

You can assign string values of your choice, as shown in the example below:

Example:

```
enum Directions {

    North = "N",
    South = "S",
    East = "E",
    West = "W"
}

console.log(Directions.North); // output is N
console.log(Directions["North"]); // output is N
console.log(Directions[0]); // output is North
```

The compiled code to javascript is as follows:

```
var Directions;
(function (Directions) {
    Directions["North"] = "N";
    Directions["South"] = "S";
    Directions["East"] = "E";
    Directions["West"] = "W";
})(Directions || (Directions = {}));
console.log(Directions.North);
console.log(Directions["North"]);
console.log(Directions[0]);
```

What are the Modules in TypeScript?

The files created in TypeScript have global access, which means that variables declared in one file are easily accessed in another file. This global nature can cause code conflicts and can cause issues with execution at run-time. You have export and import module functionality which can be used to avoid global variable, function conflicts. This feature is available in JavaScript with ES6 release and also supported in TypeScript.

Why do you need Modules in TypeScript?

Following example shows the issue without modules:

Example test1.ts

```
let age : number = 25;
```

You have defined a variable age of type number in test1.ts.

Example test2.ts

In test2.ts file you are easily able to access the variable age defined in test1.ts and also modify it as shown below:

```
age = 30; // changed from 25 to 30.
let _new_age = age;
```

So the above case can create a lot of problems as the variables are globally available and can be modified.

With Modules, the code written remains locale to the file and cannot be accessed outside it. To access anything from the file, it has to be exported using the export keyword. It is used when you want the variable, class, function, or interface to be used in another file. Import is used when you want to access the exported variable, class, or interface or function too. Doing so the code is written remains intact within the file, and even if you define same variable names, they are not mixed up and behave local to the file where they are declared.

Using Export and Import

There are many ways to export and import. So will discuss syntax here which is mostly used.

The syntax for import and export 1:

```
export  nameofthevariable or class name or interface name etc

//To import above variable or class name or interface you have to use import as shown below:


Import {nameof thevariable or class name or interfacename} from "file path here without.ts"
```

Here is a working example using export and import.

Example:

test1.ts

```
export let age: number = 25;
```

Export keyword is used to share age variable in another file.

test2.ts

```
import { age } from "./test1"
let new_age :number = age;
```

Import keyword is used to access the age variable, and you need to specify the file location as shown above.

Syntax for import and export 2:

There is another way to export, and import and the syntax for the same is as shown below:

```
export = classname;

import classname = require(“file path of modulename”)
```

When you are using export = to export your module, the import has to use require(“file path of modulename”) to import it.

Here is a working example showing the above case:

Customer.ts

```
class Customer {
    name: string;
    age: number;

    constructor(name: string, age: number) {
        this.name = name;
        this.age = age;
    }

    getName(): string {
        return this.name;
    }
}

export = Customer;
```

testCustomer.ts

```
import Customer = require("./Customer");

let a = new Customer("Harry", 30);
alert(a.getName());
```

Module Loader

Modules cannot work on its own, so you need module loader to locate the import dependencies as you have seen in TypeScript examples shown above. The module loader available is CommonJS for nodejs and Require.js to run in the browser.

To compile code using CommonJS module use following command:

```
tsc --module commonjs testCustomer.ts
```

To compile code using Requirejs module use following command:

```
tsc --module amd testCustomer.ts
```

The dependent files will get converted to js file with above command.

Example testCustomer.ts to testCustomer.js using Requirejs

```
define(["require", "exports", "./Customer"], function (require, exports, Customer) {
    "use strict";
    exports.__esModule = true;
    var a = new Customer("Harry", 30);
    alert(a.getName());
});
```

Example Customer.ts to Customer.js using Requirejs

```
define(["require", "exports"], function (require, exports) {
    "use strict";
    var Customer = /** @class */ (function () {
        function Customer(name, age) {
            this.name = name;
            this.age = age;
        }
        Customer.prototype.getName = function () {
            return this.name;
        };
        return Customer;
    }());
    return Customer;
});
```

To test it using require.js, you need to create a file called main.js, which has reference to the dependencies as shown.

Here is the folder structure:

```
src/
    Customer.js
    testCustomer.js
    main.js
    require.js  // you can get this file from github or npm install requirejs
    test.html
```

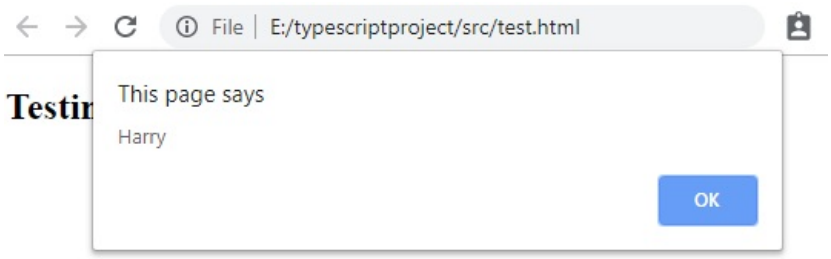
main.js

```
define(function (require) {
    var customer = require("./Customer");
    var testCustomer = require("./testCustomer");
});
```

test.html

```
<!DOCTYPE html>
```

```
<html>
<head>
    <title>TypeScript Module testing using Requirejs</title>
    <script data-main="main" src="require.js"></script>
</head>
<body>

    <h3>Testing modules using Requirejs</h3>
</body>
</html>
```



Namespaces in TypeScript

Namespace is basically has collection of classes, interfaces, variables, functions together in one file.

Namespace Syntax

```
namespace name{

export class {
}

export interface {
}

export const constname;

}
```

The code related in available under one namespace.

Namespace working example: testnamespace.ts

```
namespace StudentSetup {

    export interface StudDetails {
        name: string;
        age: number;
    }

    export function addSpace(str) { // will add space to the string given
        return str.split("").join(" ");
    }

    export class Student implements StudDetails {
```

```
        name: string;
        age: number;

        constructor(studentdetails: StudDetails) {
            this.name = studentdetails.name;
            this.age = studentdetails.age;

        }

        getName(): string {
            return this.name;
        }
    }
}
```

The name of the namespace is StudentSetup, you have added a interface StudDetails , function addSpace and a class called Student.

Accessing Namespace

Following is the code where you are using the namespace StudentSetup.

testStudentSetup.ts

```
let a = new StudentSetup.Student({ name: "Harry", age: 20 });

console.log("The name is :" + StudentSetup.addSpace(a.getName()));
```

The class, interface, a function available inside a namespace has to be referred using the name of the namespace example StudentSetup.addSpace to access the function, StudentSetup.Student to access the class.

You can compile both files into one js as shown below:

```
tsc --outFile namespace.js testnamespace.ts  testStudentSetup.ts
```

Check the output in command prompt using below command:

```
node namespace.js
```

It will display output as :

```
The name is: H a r r y
```

Ambient Declarations in TypeScript

TypeScript allows you to use third-party javascript files using ambient declaration. The advantage of this feature is that you don’t have to rewrite and yet use all the features of the library in TypeScript.

Ambient Syntax

To declare ambient module:

```
declare module moduleName {
   //code here
}
```

The ambient file has to saved as :

```
filename.d.ts
```

To use the file filename.d.ts in your .ts you need to refer it as:

```
/// <reference path="filename.d.ts"/>
```

The ambient type declaration in TypeScript will have a reference to the third party library and will re-declare the functions required with its own type. For example, consider you have a small javascript library, as shown below:

Third Party JavaScript file: testString.js

Example: testString.js

```
var StringChecks = {
    isString: function (str) {
        return typeof str === "string";
    },

    convertToUpperCase: function (str) {
        return str.toUpperCase();
    },

    convertToLowerCase: function (str) {
        return str.toLowerCase();
    },

    convertToStringBold: function (str) {
        return str.bold();
    }
};
```

You have an object called StringChecks which has functions like isString, convertToUpperCase, convertToLowerCase, and converToStringBold.

Creating of Ambient Module in TypeScript

Now will create an ambient module which will have reference to above js functions and also add type check as per our requirements.

Filename : tstring.d.ts

```
declare module TestString {

    export interface StringsFunc {

        isString(str: string): boolean;
        convertToUpperCase(str: string): string;
        convertToLowerCase(str: string): string;
        convertToStringBold(str: string): string;
    }
}

declare var StringChecks: TestString.StringsFunc;
```

You have to define a module name as TestString and have exported interface StringsFunc.

isString(str: string): boolean

=> This will take param as a string and the return type will be boolean. When using in your .ts file in case you happen to pass the param as a number or anything other than string it will give you a compile type error.

convertToUpperCase(str:string): string

=> This will take argument as string and return a string.Same goes for convertToLowerCase(str: string)
: string; and convertToStringBold(str: string): string
;

Since in the javascript file you have the object name as StringChecks, finally we need to refer the same in the .d.ts file which is done as :

```
declare var StringChecks: TestString.StringsFunc;
```

Using Ambient module in TypeScript

Now here is the test.ts file where will use ambient file tstring.d.ts

Example: test.ts

```
/// <reference path="tstring.d.ts"/>
let str1 = StringChecks.isString("Hello World");
console.log(str1);
let str2 = StringChecks.convertToUpperCase("hello world");
console.log(str2);
let str3 = StringChecks.convertToLowerCase("HELLO");
console.log(str3);
let str4 = StringChecks.convertToStringBold("Hello World");
console.log(str4);
```

Compile TypeScript tsc test.ts to test.js

```
/// <reference path="tstring.d.ts"/>
var str1 = StringChecks.isString("Hello World");
console.log(str1);
var str2 = StringChecks.convertToUpperCase("hello world");
console.log(str2);

var str3 = StringChecks.convertToLowerCase("HELLO");
console.log(str3);
var str4 = StringChecks.convertToStringBold("Hello World");
console.log(str4);
```

Now you can use test.js in html file and also the library file testString.js

```
<html>
<head>
    <title>Test TypeScript Ambient</title>
    <script src="testStrings.js"></script>
    <script src="test.js"></script>
</head>
<body>
</body>
</html>
```

This is the output seen in the console:

```
true
HELLO WORLD
hello
<b>Hello World</b>
```

TypeScript History

Let see important landmarks from the history of TypeScript:

- After two years of internal development at Microsoft. TypeScript 0.9, released in 2013
- Additional support for generics TypeScript 1.0 was released at Build 2014
- In July 2014, a new TypeScript compiler came which is five times faster than it's the previous version.
- In July 2015, support for ES6 modules, namespace keyword, for, of support, decorators.
- In November 2016, an added feature like key and lookup types of mapped types, and rest.
- On March 27, 2018, conditional types, the improved key with intersection types supports added in the TypeScript.

Why use TypeScript?

Here, are important pros/benefits of using TypeScript

- Big and complex project in JavaScript are difficult to code and maintain.
- TypeScript helps a lot in code organization and yet gets rid of most of the errors during compilation.
- TypeScript supports JS libraries & API Documentation
- It is optionally typed scripting language

- TypeScript Code can be converted into plain JavaScript Code
- Better code structuring and object-oriented programming techniques
- Allows better development time tool support
- It can extend the language beyond the standard decorators, async/await

Who uses TypeScript?

Here, are some most common applications of TypeScript:

- The angular team uses TypeScript.
- NodeJS and NPM Installation
- TypeScript Installation
- Compile TypeScript code to Javascript
- Execute Code using Nodejs
- Execute Javascript in Browser
- Compile TypeScript code to Javascript using EcmaScript version
- You can easily compile code written in TypeScript to JavaScript using NodeJS.
- So to work with TypeScript you need first to download and install NodeJS.

Summary

- TypeScript is a superset of JavaScript. TypeScript is pure object-oriented programming language that supports classes, interfaces, etc.
- TypeScript supports all the Ecmascript features released, and developers can use the same while coding.
- Variables are used to store values, and the value can be a string, number, Boolean, or an expression.
- In TypeScript, the type to a variable is defined at the start only and through the execution, it has to maintain the same type any changes to it will lead to a compile-time error during compilation to javascript.
- An Array in TypeScript is a data type wherein you can store multiple values.
- Class is a new feature added from ES6 onward, so earlier in JavaScript the class type functionality was tried using a function with prototype functionality to reuse code.
- TypeScript supports public, private, and protected access modifiers to your methods and properties.
- One of the core features of TypeScript is interfaces. The interface is a set of a rule defined which needs to be implemented by the entity using it.
- Functions are set of instructions performed to carry out a task.
- TypeScript Enum is an object which has a collection of related values stored together.

Don't Miss:

- Execute JavaScript Online
- QuickSort Algorithm in JavaScript
- 15 Best FREE JavaScript Certification Courses Online (2024)

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,
United States

TypeScript vs JavaScript – Difference Between Them

By : James Hartman March 9, 2024

Key Difference Between TypeScript and JavaScript

- JavaScript is a scripting language which helps you create interactive web pages whereas Typescript is a superset of JavaScript.
- Typescript code needs to be compiled while JavaScript code doesn't need to compile.
- Comparing TypeScript and JS, Typescript supports a feature of prototyping while JavaScript doesn't support this feature.
- Typescript uses concepts like types and interfaces to describe data being used whereas JavaScript has no such concept.
- Typescript is a powerful type system, including generics & JS features for large size project whereas JavaScript is an ideal option for small size project.

Table of Content:

What is JavaScript?

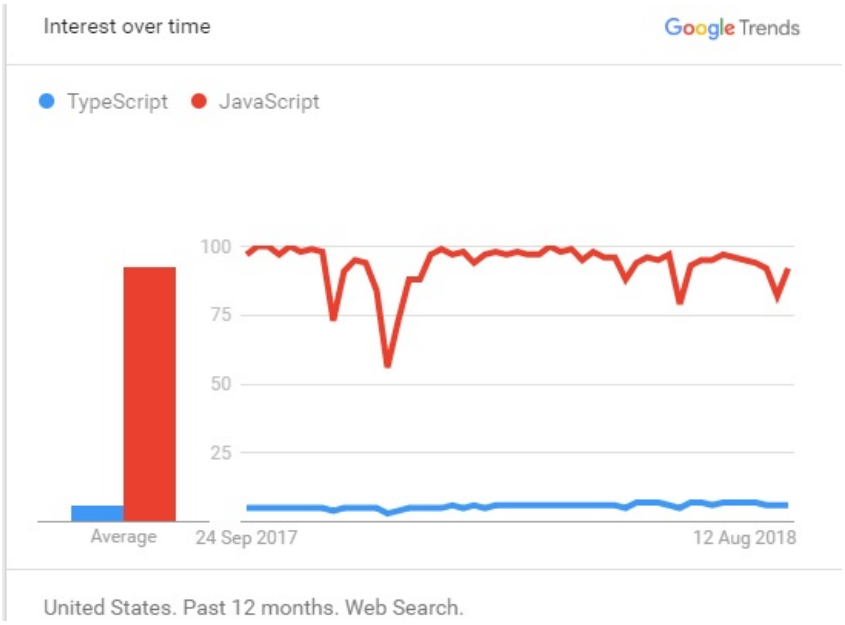
JavaScript is a scripting language which helps you create interactive web pages. It followed rules of client-side programming, so it runs in the user's web browser without the need of any resources forms the web server. You can also use Javascript with other technologies like REST APIs, XML, and more.

The idea behind developing this script is to make it a complementary scripting language like Visual Basic was to C++ in Microsoft's language families. However, JavaScript is not designed for large complex applications. It was developed for applications with a few hundred lines of code!

What is TypeScript?

TypeScript is a modern age JavaScript development language. It is a statically compiled language to write clear and simple JavaScript code. It can be run on Node js or any browser which supports ECMAScript 3 or newer versions.

TypeScript provides optional static typing, classes, and interface. For a large JavaScript project adopting Typescript can bring you more robust software and easily deployable with a regular JavaScript application.



Don't Miss:

- Execute JavaScript Online
- QuickSort Algorithm in JavaScript
- 15 Best FREE JavaScript Certification Courses Online (2024)

Why JavaScript?

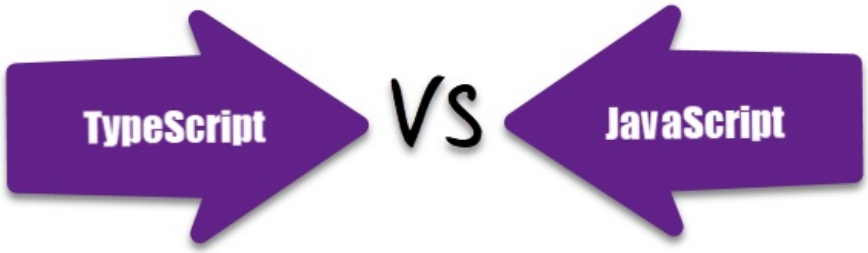
- Open source project with Microsoft's patronage
- Specially designed tool for small scripts
- Supports classes, interfaces & modules.
- Compiled JavaScript runs in any browser
- Allows cross-compilation
- You can extend JavaScript for writing large apps
- Adds support for classes, interfaces, and modules

Why TypeScript?

- TypeScript supports JS libraries & API Documentation
- It is a superset of JavaScript
- It is optionally typed scripting language
- TypeScript Code can be converted into plain JavaScript Code
- Better code structuring and object-oriented programming techniques
- Allows better development time tool support
- It can extend the language beyond the standard decorators, async/await

Difference Between TypeScript and JavaScript

Following is the main TypeScript and JavaScript difference:



Typescript Vs JavaScript

Parameter	Typescript	JavaScript
What is	Powerful type system, including generics & JS features	Lightweight, interpreted, object-oriented language with first-class functions
Data Binding	TypeScript uses concepts like types and interfaces to describe data being used.	No such concept is available with JavaScript.
Ecosystem	The Ecosystem is quite powerful and intuitive. Thus, it allows you to statically type various type of idiomatic JavaScript features like union types, intersection, discriminated union.	JavaScript offers the option to explore and create code without a build step.
Npm package	With Typescript, many npm packages either come with static type definitions or have an external one that is easy to install.	JavaScript offers the option to explore and create code without a build step.
Learning curve	Stiff learning curve. Requires prior scripting knowledge.	Flexible and easy to learn, scripting language.
Prototyping	Typescript has a feature of prototyping.	JavaScript doesn’t have this feature.
Community	Typescript does not have a large community of developers.	The JavaScript has a huge community of developers
Compilation	TypeScript code needs to be compiled	No need to compile JavaScript.
Annotation	To get the most out of TypeScript features, developers should constantly annotate their code.	No Annotations Required is need for JavaScript.

Famous Company using the Technology	Asana, Clever, Screen award	Airbnb, Codecademy, Instagram
Salary	The average salary for “Typescript developer” ranges from approximately $148,027 per year in United States	The average salary for a JavaScript Developer is $110,777 per year in the United States.

History of JavaScript

Netscape Communications Corporation programmer Brendan Eich created JavaScript. It was meant to working Netscape navigator. However, after becoming a popular scripting tool, it is had become LiveScript. Later on, it was renamed as JavaScript to reflect Netscape’s support of Java within its browser.

Let see an important landmark in the history of JavaScript:

- It was launched in September 1995, and It took just ten days to develop this e scripting language which was initially called Mocha
- In November 1996, Netscape submitted JavaScript to ECMA (European Computer Manufacturers Association) International
- ECMAScript 2 was released in 1998
- ECMAScript 3 was released in 1999
- In 2005, Eich and Mozilla join ECMA to develop E4X Java script
- In January 2009, the CommonJS project was launched with the aim of defining a common standard library
- In June 2011, ECMAScript 5.1 was released
- In June 2015. ECMAScript 2016 was released
- The current version is ECMAScript 2017 which was released in June 2017

History of TypeScript

Let see important landmarks from the History of TypeScript:

- The typescript was first made public in the year 2012
- After two years of internal development at Microsoft. TypeScript 0.9, released in 2013
- Additional support for generics TypeScript 1.0 was released at Build 2014
- In July 2014, a new TypeScript compiler came which is five times faster then it’s previous version
- In July 2015, support for ES6 modules, namespace keyword, for, of support, decorators
- In November 2016, an added feature like key and lookup types mapped types, and rest
- On March 27, 2018, conditional types, the improved key with intersection types supports added in the Typescript.

Features of JavaScript

- It’s a cross-platform language
- It’s used for client side and server side
- It’s easy to learn and to start with
- It’s a dynamic language: flexible and powerful
- You have ‘the great freedom’ to do whatever you want with any object
- Strong Testing Workflow
- Added Dependencies
- Framework Unsupported

Features of Typescript

- Maintainability
- Offered great productivity for developers
- Code navigation and bug prevention
- Code ‘discoverability’ & refactoring
- Optional Static Type Annotation / Static Typing
- Additional Features for Functions
- Supports ES6
- Supports interfaces, sub-interfaces, classes, and subclasses
- Scalable HTML5 client-side development
- Rich IDE available with autocomplete and code navigation features.
- Class-based object-oriented with the inheritance of private members and interfaces.

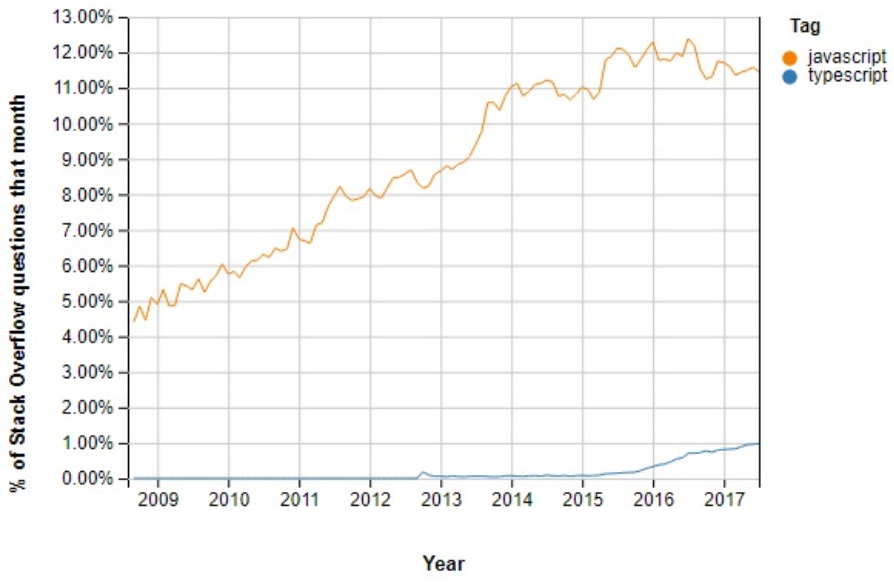
Summary

What is the Difference Between TypeScript and JavaScript?

TypeScript is a modern age JavaScript development language whereas JavaScript is a scripting language which helps you create interactive web pages. TypeScript uses concepts like types and interfaces to describe data being used whereas no such concept is available with JavaScript.

JavaScript VS TypeScript: Which is better?

In the end of this JavaScript and TypeScript difference tutorial, we can say that if an experienced developer is working on relatively small coding projects, then JavaScript is ideal. However, if you have knowledge and expertise development team, then Typescript is a most preferred option.



Typescript vs JavaScript

Don't Miss:

- Execute JavaScript Online
- QuickSort Algorithm in JavaScript
- 15 Best FREE JavaScript Certification Courses Online (2024)

Guru99's Headquarters

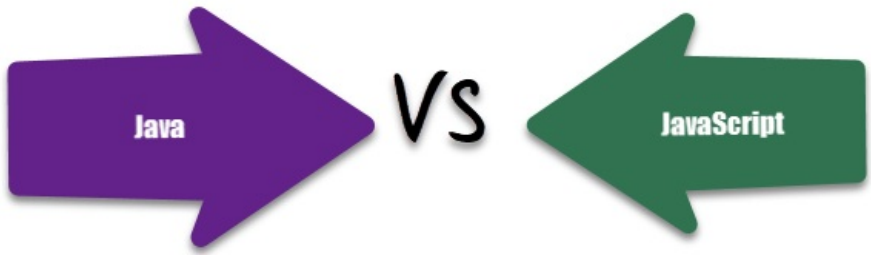
4023 Kennett Pike #50286,
Wilmington, Delaware,
United States

Java vs JavaScript – Difference Between Them

By : James Hartman March 9, 2024

Key Difference between Java and JavaScript

- Java is a multi-platform, object-oriented, and network-centric programming language, whereas JavaScript is a scripting language that helps you create interactive web pages.
- Java is a strongly typed language, while JavaScript is a weakly typed language.
- Java has a file extension ".Java", whereas Javascript has the file extension ".js"
- With Java, you write code once and run it on almost any computing platform, on the other hand, Javascript is a cross-platform language.
- Java is compiled on the server before execution on the client, while JavaScript is interpreted by the client.
- Java is a static language, while JavaScript is a dynamic language.



Java vs JavaScript

Table of Content:

What is Java?

Java is a multi-platform, object-oriented, and network-centric, programming language. It is among the most used programming language. It also used as a computing platform, and it was first released by Sun Microsystem in 1995. It was later acquired by Oracle Corporation.

What is JavaScript?

JavaScript is a scripting language that helps you create interactive web pages. It follows the rules of client-side programming, so it runs in the user's web browser without the need for any resources form the web server. You can also use JavaScript with other technologies like REST APIs, XML, and more. Nowadays JavaScript also using

technologies like Node js.

Java vs JavaScript – Difference Between Them

A major difference Java and JavaScript is that Java is compiled and interpreted language while JavaScript code is directly executed by the browser

Parameters	Java	JavaScript
Variable Definition	Java is a strongly typed language, so the variable should be declared first before using in the program.	JavaScript is a weakly typed language, so its variable can be declared where they are used.
Type of language	It is an object-oriented programming language.	It is an object-based scripting language
Type of object	Objects of Java are class-based, so you can't create any program in java without developing a class.	Objects are prototype-based.
Extension	It has a file extension ".Java".	It has file extension ".js"
Compilation process	It is interpreted as well as complied. Java translates source code into bytecodes. It is executed by JVM(Java Virtual Machine).	All browser has the JavaScript interpreter, which allows you to execute JavaScript code.
Process	Compiled on the server before execution on the client.	Interpreted (not compiled) by the client.
Code type	Object-oriented. Applets consist of object classes with inheritance.	It is object-based. Code uses built-in, extensible objects but not uses any classes or inheritance.
Syntax	Data types must be declared.	Data types not declared.
Type of language	Static	Dynamic
Key Features	• Great libraries • Widely used • Excellent tooling	• Can be used on frontend/backend • It's everywhere • Lots of great frameworks
Famous Company using the Technology	Airbnb, Uber Technologies, Netflix, Instagram.	Reddit, eBay, Coursera.

Code	`class A {`
	`public static void main(String args[]){`
	`System.out.println("Hello World");`
	`}`
	`}`
		`<html>`
		`<head>`
		`<title>My First JavaScript code!!!</title>`
		`<script>`
		`alert("Hello World!");`
		`</script>`
		`</head>`
		`<body>`
		`</body>`
		`</html>`

Salary	The average salary for a Java Developer is $103,464 per year in the US.	The average salary for a JavaScript Developer is $113,615 per year in the US.
TOBIE Rating	1	6

Don't Miss:

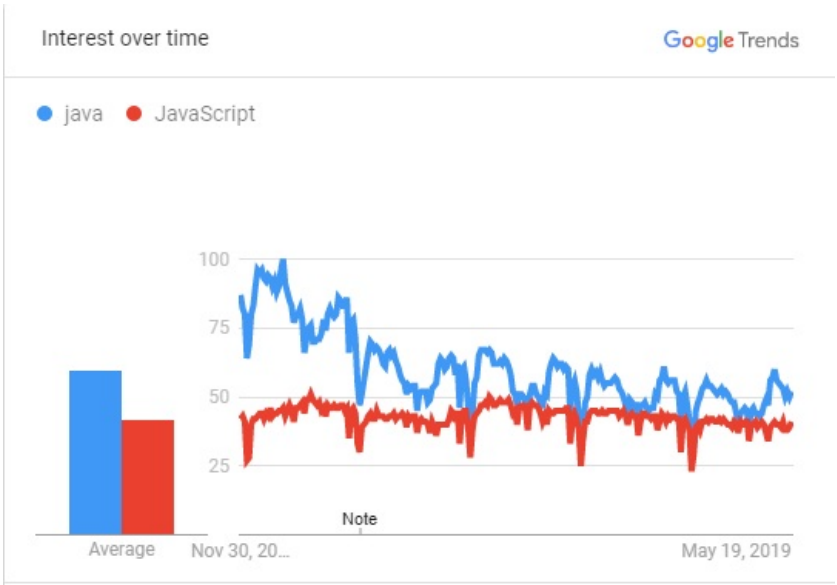
- Execute JavaScript Online
- TypeScript vs JavaScript – Difference Between Them
- QuickSort Algorithm in JavaScript

Features of Java

Here are the important features of Java.

- Write code once and run it on almost any computing platform
- It is designed for building object-oriented applications.
- It is a multithreaded language with automatic memory management
- Facilitates distributed computing as its network-centric

Features of JavaScript

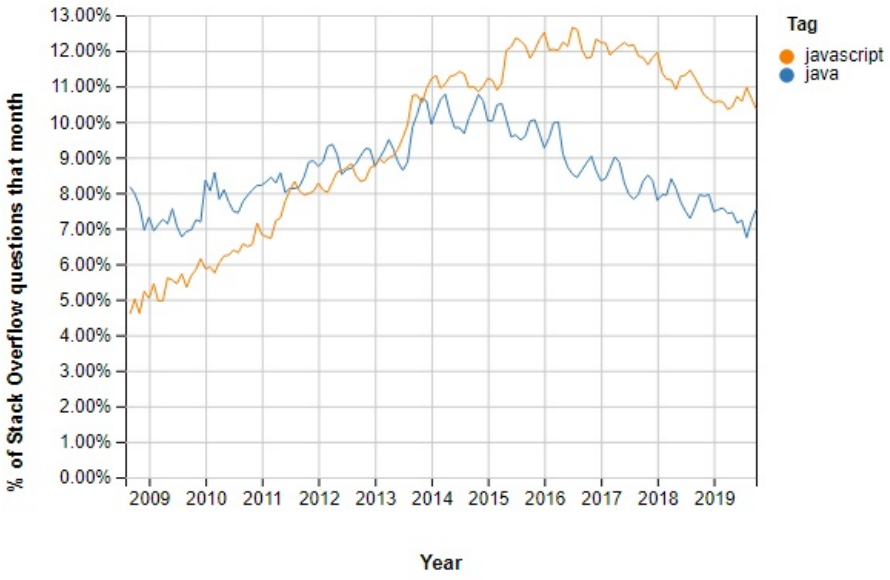


Google trends Java vs. Java script

Here are important features of Java-script:

- It’s a cross-platform language
- It’s widely used for client-side and server-side
- Strong Testing Workflow
- It’s easy to learn and to start coding with
- Added dependencies

Application of Java



Stack Overflow Questions JAVA vs Java Script

Here, are important applications of Java language:

To develop:

- Android Apps
- Enterprise Software
- Scientific Computing Applications
- Big Data Analytics
- Java Programming of Hardware devices
- Used for Server-Side Technologies like Apache, JBoss, GlassFish, etc.

Application of JavaScript

Here, are some important applications of JavaScript:

- Dynamic Single-Page Applications (SPAs)
- Front-End technologies like jQuery, AngularJS, Ember.js, ReactJS are based on Java Script
- Server-Side technologies like Node.js, Express.js, MongoDB are based on Java Script.
- Mobile App Development using PhoneGap, React Native, etc.

Advantage of Java

Here, are benefits/ pros of using Java

- Detailed documentation is available.
- A large pool of skilled developers available
- Huge array of 3rd party libraries
- It allows you to form standard programs and reusable code.
- It is a multi-threaded environment that allows you to perform many tasks at the same time in a program.
- Excellent performance
- Easy to navigate libraries

Advantages of JavaScript

Here, are pros/benefits of using JavaScript

- It is an open-source project with Microsoft's patronage
- Specially designed tool for small scripts
- Supports classes, interfaces, & modules.
- Compiled JavaScript runs in any browser
- Allows cross-compilation
- You can extend JavaScript for writing large apps
- You can use JavaScript to store and retrieve information on the user's computer
- Immediate feedback to the visitors
- It allows you to create interfaces that react when the user hovers using mouse.

Disadvantages of Java

Here, are cons/drawback of using Java language

- JIT compiler makes the program comparatively slow.
- Java has high memory and processing requirements. Therefore, hardware cost increases.
- It does not provide support for low-level programming constructs like pointers.
- You don't have any control over garbage collection as Java does not offer functions like delete(), free().

Disadvantages of JavaScript

Here, are drawbacks/cons of using JavaScript

- Client-side JavaScript does not allow the reading or writing of files. It has been kept for security reasons.
- JavaScript can't be used for networking applications because there is not much support available.
- JavaScript doesn't have any multithreading or multiprocessor features.

Don't Miss:

- Execute JavaScript Online
- TypeScript vs JavaScript – Difference Between Them
- QuickSort Algorithm in JavaScript

← Prev

Report a Bug

Next →

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,
United States

QuickSort Algorithm in JavaScript

By : James Hartman March 9, 2024

What is Quick Sort?

Quick Sort algorithm follows Divide and Conquer approach. It divides elements into smaller parts based on some condition and performing the sort operations on those divided smaller parts.

Quick Sort algorithm is one of the most used and popular algorithms in any programming language. But, if you are a JavaScript developer, then you might of heard of sort() which is already available in JavaScript. Then, you might have been thinking what the need of this Quick Sort algorithm is. To understand this, first we need what is sorting and what is the default sorting in JavaScript.

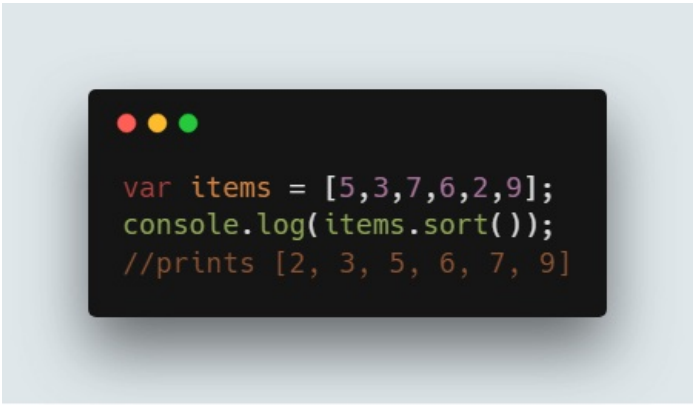
Table of Content:

What is Sorting?

Sorting is nothing but, arranging elements in the order we want. You might have come across this in your school or college days. Like arranging numbers from smaller to greater (ascending) or from greater to smaller (descending) is what we saw till now and is called sorting.

Default sorting in JavaScript

As mentioned earlier, JavaScript has sort(). Let us take an example with few array of elements like [5,3,7,6,2,9] and want to sort this array elements in ascending order. Just call sort() on items array and it sorts array elements in ascending order.



Code:

```
var items = [5,3,7,6,2,9];
console.log(items.sort());
//prints [2, 3, 5, 6, 7, 9]
```

What is the reason to choose Quick sort over default sort() in JavaScript

Though sort() gives the result we want, problem lies with the way it sorts the array elements. Default sort() in JavaScript uses insertion sort by V8 Engine of Chrome and Merge sort by Mozilla Firefox and Safari.

But, other this is not suitable if you need to sort large number of elements. So, the solution is to use Quick sort for large dataset.

So, to understand completely, you need to know how Quick sort works and let us see that in detail now.

What is Quick sort?

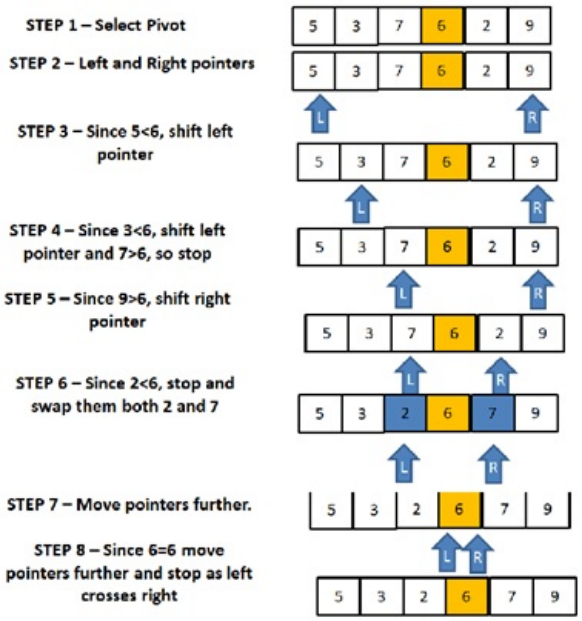
Quick sort follows Divide and Conquer algorithm. It is dividing elements in to smaller parts based on some condition and performing the sort operations on those divided smaller parts. Hence, it works well for large datasets. So, here are the steps how Quick sort works in simple words.

1. First select an element which is to be called as pivot element.
2. Next, compare all array elements with the selected pivot element and arrange them in such a way that, elements less than the pivot element are to it's left and greater than pivot is to it's right.
3. Finally, perform the same operations on left and right side elements to the pivot element.

So, that is the basic outline of Quick sort. Here are the steps which need to be followed one by one to perform Quick sort.

How does QuickSort Work

1. First find the "pivot" element in the array.
2. Start the left pointer at first element of the array.
3. Start the right pointer at last element of the array.
4. Compare the element pointing with left pointer and if it is less than the pivot element, then move the left pointer to the right (add 1 to the left index). Continue this until left side element is greater than or equal to the pivot element.
5. Compare the element pointing with right pointer and if it is greater than the pivot element, then move the right pointer to the left (subtract 1 to the right index). Continue this until right side element is less than or equal to the pivot element.
6. Check if left pointer is less than or equal to right pointer, then swap the elements in locations of these pointers.
7. Increment the left pointer and decrement the right pointer.
8. If index of left pointer is still less than the index of the right pointer, then repeat the process; else return the index of the left pointer.



So, let us see these steps with an example. Let us consider array of elements which we need to sort is [5,3,7,6,2,9].

Determine Pivot element

But before going forward with the Quick sort, selecting the pivot element plays a major role. If you select the first element as the pivot element, then it gives worst performance in the sorted array. So, it is always advisable to pick the middle element (length of the array divided by 2) as the pivot element and we do the same.

Here are the steps to perform Quick sort that is being shown with an example [5,3,7,6,2,9].

STEP 1: Determine pivot as middle element. So, 7 is the pivot element.

STEP 2: Start left and right pointers as first and last elements of the array respectively. So, left pointer is pointing to 5 at index 0 and right pointer is pointing to 9 at index 5.

STEP 3: Compare element at the left pointer with the pivot element. Since, 5 < 6 shift left pointer to the right to index 1.

STEP 4: Now, still 3 <6 so shift left pointer to one more index to the right. So now 7 > 6 stop incrementing the left pointer and now left pointer is at index 2.

STEP 5: Now, compare value at the right pointer with the pivot element. Since 9 > 6 move the right pointer to the left. Now as 2 < 6 stop moving the right pointer.

STEP 6: Swap both values present at left and right pointers with each other.

STEP 7: Move both pointers one more step.

STEP 8: Since 6 = 6, move pointers to one more step and stop as left pointer crosses the right pointer and return the index of the left pointer.

So, here based on the above approach, we need to write code for swapping elements and partitioning the array as mentioned in above steps.

Code to swap two numbers in JavaScript


```
function swap(items, leftIndex, rightIndex){
    var temp = items[leftIndex];
    items[leftIndex] = items[rightIndex];
    items[rightIndex] = temp;
}
```

Code to perform the partition as mentioned in above steps

```
function partition(items, left, right) {

    var pivot   = items[Math.floor((right + left) / 2)],//middle element
        i       = left, //left pointer
        j       = right; //right pointer


    while (i <= j) {

        while (items[i] < pivot) {
            i++;
        }

        while (items[j] > pivot) {
            j--;
        }

        if (i <= j) {
            swap(items, i, j); //swap two elements
            i++;
            j--;
        }
    }

    return i;
}
```

```
function partition(items, left, right) {
    var pivot   = items[Math.floor((right + left) / 2)], //middle element
        i       = left, //left pointer
        j       = right; //right pointer
    while (i <= j) {
        while (items[i] < pivot) {
            i++;
        }
        while (items[j] > pivot) {
            j--;
        }
        if (i <= j) {
            swap(items, i, j); //swap two elements
            i++;
            j--;
        }
    }
    return i;
}
```

Perform the recursive operation

Once you perform above steps, index of the left pointer will be returned and we need to use that to divide the array and perform the Quick sort on that part. Hence, it is called Divide and Conquer algorithm.

So, Quick sort is performed until all elements on the left array and right array are sorted.

Note: Quick sort is performed on the same array and no new arrays are created in the process.

So, we need to call this partition() explained above and based on that we divide the array in to parts. So here is the code where you use it,

```
function quickSort(items, left, right) {

    var index;

    if (items.length > 1) {

        index = partition(items, left, right); //index returned from partition

        if (left < index - 1) { //more elements on the left side of the pivot
            quickSort(items, left, index - 1);
        }

        if (index < right) { //more elements on the right side of the pivot
            quickSort(items, index, right);
        }

    }

    return items;
}


// first call to quick sort
var result = quickSort(items, 0, items.length - 1);
```

```
function quickSort(items, left, right) {
    var index;
    if (items.length > 1) {
        index = partition(items, left, right); //index returned from partition
        if (left < index - 1) { //more elements on the left side of the pivot
            quickSort(items, left, index - 1);
        }
        if (index < right) { //more elements on the right side of the pivot
            quickSort(items, index, right);
        }
    }
    return items;
}
// first call to quick sort
```

```
var result = quickSort(items, 0, items.length - 1);
```

Complete Quick sort code

```
var items = [5,3,7,6,2,9];
function swap(items, leftIndex, rightIndex){
    var temp = items[leftIndex];
    items[leftIndex] = items[rightIndex];
    items[rightIndex] = temp;
}
function partition(items, left, right) {
    var pivot   = items[Math.floor((right + left) / 2)], //middle element
        i       = left, //left pointer
        j       = right; //right pointer
    while (i <= j) {
        while (items[i] < pivot) {
            i++;
        }
        while (items[j] > pivot) {
            j--;
        }
        if (i <= j) {
            swap(items, i, j); //sawpping two elements
            i++;
            j--;
        }
    }
    return i;
}

function quickSort(items, left, right) {
    var index;
    if (items.length > 1) {
        index = partition(items, left, right); //index returned from partition
        if (left < index - 1) { //more elements on the left side of the pivot
            quickSort(items, left, index - 1);
        }
        if (index < right) { //more elements on the right side of the pivot
            quickSort(items, index, right);
        }
    }
    return items;
}
// first call to quick sort
var sortedArray = quickSort(items, 0, items.length - 1);
console.log(sortedArray); //prints [2,3,5,6,7,9]
```

```
var items = [5,3,7,6,2,9];

function swap(items, leftIndex, rightIndex){
    var temp = items[leftIndex];
    items[leftIndex] = items[rightIndex];
    items[rightIndex] = temp;
}

function partition(items, left, right) {
    var pivot   = items[Math.floor((right + left) / 2)], //middle element
        i       = left, //left pointer
        j       = right; //right pointer
    while (i <= j) {
        while (items[i] < pivot) {
            i++;
        }
        while (items[j] > pivot) {
            j--;
        }
        if (i <= j) {
            swap(items, i, j); //sawpping two elements
            i++;
            j--;
        }
    }
    return i;
}

function quickSort(items, left, right) {
    var index;
    if (items.length > 1) {
        index = partition(items, left, right); //index returned from partition
        if (left < index - 1) { //more elements on the left side of the pivot
            quickSort(items, left, index - 1);
        }
        if (index < right) { //more elements on the right side of the pivot
            quickSort(items, index, right);
        }
    }
    return items;
}


// first call to quick sort
var sortedArray = quickSort(items, 0, items.length - 1);
console.log(sortedArray); //prints [2,3,5,6,7,9]
```

NOTE: Quick sort runs with the Time Complexity of O(nlogn).

Don't Miss:

- Execute JavaScript Online
- TypeScript vs JavaScript – Difference Between Them
- 15 Best FREE JavaScript Certification Courses Online (2024)

← Prev

Report a Bug

Next →

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,
United States

Difference Between =, ==, and === in JavaScript [Examples]

By : James Hartman March 9, 2024

What is = in JavaScript?

Equal to (=) is an assignment operator, which sets the variable on the left of the = to the value of the expression that is on its right. This operator assigns lvalue to rvalue.

For example, Writing a=10 is fine. If we write 10=10, ‘a’ = 10 or ‘a’ = ‘a’, it will result in a reference error.

Table of Content:

What is == in JavaScript?

Double equals (==) is a comparison operator, which transforms the operands having the same type before comparison.

So, when you compare string with a number, JavaScript converts any string to a number. An empty string is always converts to zero. A string with no numeric value is converts to NaN (Not a Number), which returns false.

What is === in JavaScript?

=== (Triple equals) is a strict equality comparison operator in JavaScript, which returns false for the values which are not of a similar type. This operator performs type casting for equality. If we compare 2 with “2” using ===, then it will return a false value.

Why use = in JavaScript?

Here are the important uses of = in JavaScript:

= JavaScript operator assigns a value to the left operand depends on the value of operand available on the right side. The first operand should be a variable.

The basic assignment operator is =, that assigns the value of one operand to another. That is, a = b assigns the value of b to a.

Why use == in JavaScript?

Here are the important uses of == in JavaScript:

The == operator is an equality operator. It checks whether its two operands are the same or not by changing expression from one data type to others. You can use == operator in order to compare the identity of two operands even though, they are not of a similar type.

How === Works Exactly?

- Strict equality === checks that two values are the same or not.
- Value are not implicitly converted to some other value before comparison.
- If the variable values are of different types, then the values are considered as unequal.
- If the variable are of the same type, are not numeric, and have the same value, they are considered as equal.
- Lastly, If both variable values are numbers, they are considered equal if both are not NaN (Not a Number) and are the same value.

Example of =

In the below program, there are two variables “a” and “b”. We are adding and printing their values using a third variable, “c”. The sum of the value of variable “a” and “b” is 7. Therefore, the output is 7.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Operators</h2>

 <p>a = 2, b = 5, calculate c = a + b, and display c:</p>

<p id="demonstration"></p>

<script>
var a = 2;
var b = 5;
var c= a + b;
document.getElementById("demonstration").innerHTML = c;
</script>

</body>
</html>
```

Don't Miss:

- Execute JavaScript Online
- TypeScript vs JavaScript – Difference Between Them
- QuickSort Algorithm in JavaScript

Output:

a = 2, b = 5, calculate c = a + b, and display c:

7

Example of ==

In the below program, we have declared one variable “a” having value 10. Lastly, the statement a == 20 returns false as the value of a is 10.

```
<!DOCTYPE html>
<html>
<body>

<p id="demonstration"></p>

<script>
  var a = 10;
  document.getElementById("demonstration").innerHTML = (a == 20);
</script>

</body>
</html>
```

Output:

false

Example of ===

In the below program, the value of variable x is 10. It is compared to 10 written in double-quotes, which is considered as a string, and therefore, the values are not strictly the same. The output of the program is false.

```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>
```

```
    var x = 10;
    document.getElementById("demo").innerHTML = (x === "10");

</script>

</body>
</html>
```

Output:

false

= Vs == VS === in JavaScript

Here are the important differences between =, ==, and ===

=	==	===
= in JavaScript is used for assigning values to a variable.	== in JavaScript is used for comparing two variables, but it ignores the datatype of variable.	=== is used for comparing two variables, but this operator also checks datatype and compares two values.
It is called as assignment operator	It is called as comparison operator	It is also called as comparison operator
The assignment operator can evaluate to the assigned value	Checks the equality of two operands without considering their type.	Compares equality of two operands with their types.
It does not return true or false	Return true if the two operands are equal. It will return false if the two operands are not equal.	It returns true only if both values and data types are the same for the two variables.
= simply assign one value of variable to another one.	== make type correction based upon values of variables.	=== takes type of variable in consideration.
== will not compare the value of variables at all.	The == checks for equality only after doing necessary conversations.	If two variable values are not similar, then === will not perform any conversion.

KEY DIFFERENCES

- = is used for assigning values to a variable, == is used for comparing two variables, but it ignores the datatype of variable whereas === is used for comparing two variables, but this operator also checks datatype and compares two values.
- = is called as assignment operator, == is called as comparison operator whereas It is also called as comparison operator.
- = does not return true or false, == Return true only if the two operands are equal while === returns true only if both values and data types are the same for the two variables.

Don't Miss:

- Execute JavaScript Online
- TypeScript vs JavaScript – Difference Between Them
- QuickSort Algorithm in JavaScript

← Prev

Report a Bug

Next →

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,

United States

15 Best FREE JavaScript Certification Courses Online (2024)

By : James Hartman July 13, 2024

JavaScript is the most popular client-side scripting language supported by all browsers. It is used for wide-ranging browser functionalities like validating forms, creating cookies, and more. Moreover, it is considered a high-demanding programming language in the current job market. Decent knowledge of JavaScript programming language not only helps you get a good job but also gives you an edge in your current job as well.

You can learn JavaScript classes online from JavaScript certification courses and advance your career. We have compiled some of the best JavaScript certification courses from top industry leaders and the best online learning websites like Udemy, Coursera, Udacity, Edx, and.

Here, we have selected the best JavaScript classes online for beginners and experts. We have also identified courses geared for intermediate and expert professionals who want to upskill and advance their careers. This list has many online JavaScript courses for free as well as paid ones.

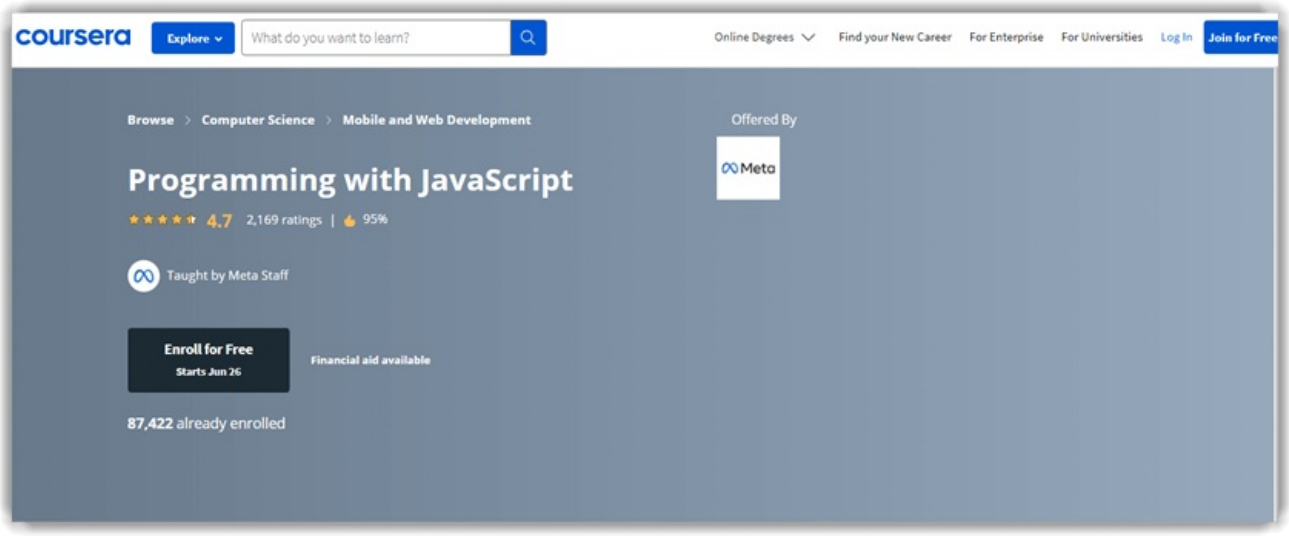
Best Online JavaScript Certification Courses

Name	Provider	Price	Key Topics	Ratings	Duration	Link
Programming with JavaScript	Coursera	Free	Creating simple JavaScript codes.	4.7	42 Hours	Learn More
Modern JavaScript From The Beginning	Udemy	$11.99	Lean Javascript from scratch.	4.7	21.5 hours on-demand video	Learn More
The Complete JavaScript Course: From Zero to Expert!	Udemy	$11.99	How to become a confident and modern JavaScript developer.	4.8	69 hours on-demand video	Learn More
Intro to JavaScript	Udacity	Free	What is JavaScript	4.5	Approx. 2 Weeks	Learn More
Full Stack JavaScript Developer	Udacity	Free	Full-stack JavaScript development.	4.5	4 Months	Learn More

1) Programming with JavaScript (Coursera)

Specs: Duration: 42 hours | Price/Fee: Free | Certification: Yes | Level: Beginner

Programming with JavaScript is an online learning course that teaches you all the skillsets required to get comfortable with JavaScript. This JavaScript course helps you learn the basics of JavaScript with real-world applications. This JavaScript training course teaches you how to write JavaScript in browser and code editor, manipulate web forms, read data, check inputs for errors, and use various JavaScript functions.



Key topics:

- Introduction to Javascript
- The Building Blocks of a Program
- Programming Paradigms
- End-of-Course Graded Assessment
- Better User Experience with an API

Features:

- Flexible deadlines
- Shareable Certificate
- Digital certificate on completion

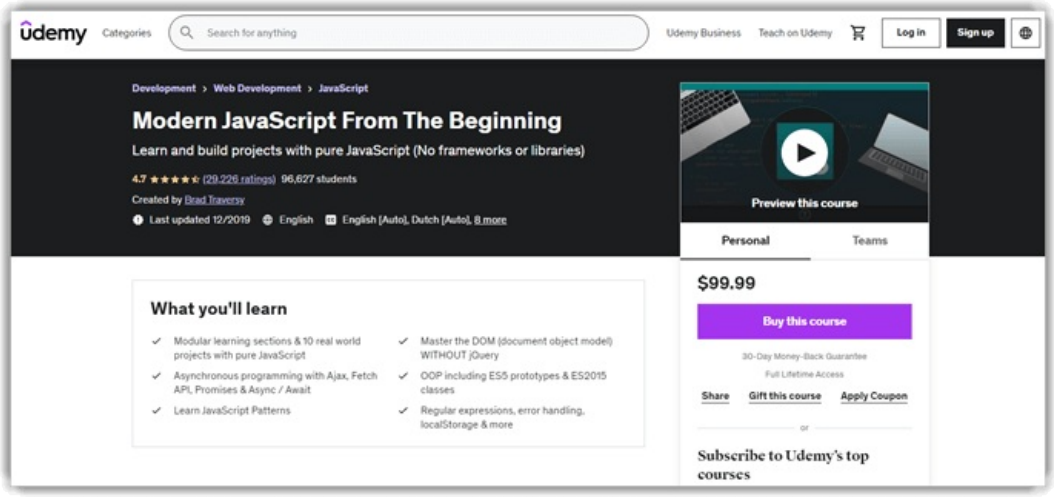
Enroll Now >>

---

2) Modern JavaScript From The Beginning (Udemy)

Specs: Rating: 4.7 | Duration: 21.5 hours on-demand video | Certification: Yes | Prerequisites: Anyone who wants to learn modern JavaScript from beginner to advance level without libraries and frameworks

Modern JavaScript from The Beginning is an online course for beginners that helps everybody learn JavaScript from scratch. This course starts with the fundamentals and teaches advanced JavaScript programming without depending on frameworks or libraries. You will learn various JavaScript types, irrespective of whether you are a beginner or an established JS programmer.



Key topics:

- Modular learning sections & 10 real-world projects with pure JavaScript
- Master the DOM (Document Object Model) without jQuery
- OOP including ES5 prototypes & ES2015 classes
- Regular expressions, error handling, localStorage & more

Features:

- 2 articles
- 111 downloadable resources
- Full lifetime access
- Access on mobile and TV

Enroll Now >>

---

3) The Complete JavaScript Course: From Zero to Expert! (Udemy)

Specs: Rating: 4.8 | Duration: 69 hours on-demand video | Certification: Yes

The Complete JavaScript Course: From Zero to Expert is an online course where you will learn about modern JavaScript from the beginning. It is one of the best JavaScript certification courses and is full of practical and fun code examples for an important theory about how JavaScript works behind the scenes.

You will also learn how to think like a developer, architect your code, and debug code. It has a lot of other real-world skills that you will need for a web development job.

This JavaScript Bootcamp contains beginner, intermediate, advanced, and even expert-level topics. So you do not have to buy any other course to master JavaScript from the ground up.

Key topics:

- Offers real-world projects for your portfolio (not boring toy apps)
- Problem-solving, researching, and workflows.
- JavaScript fundamentals: variables, operators, Boolean logic, functions, objects, loops, etc.
- Asynchronous JavaScript: Event loop, async/await, AJAX calls, and APIs
- Architect your code using flowcharts and common patterns.

Features:

- 20 articles
- 18 downloadable resources
- Full lifetime access
- Access on mobile and TV

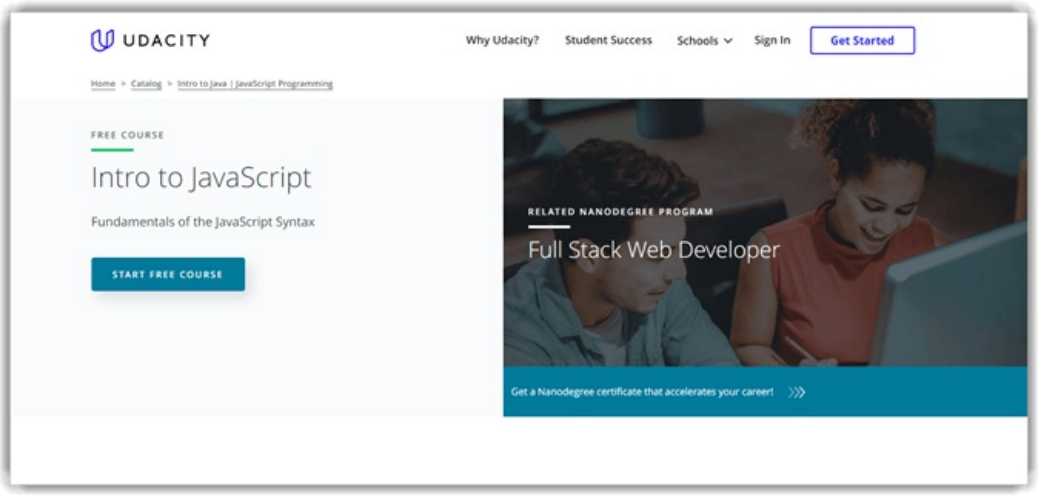
Enroll Now >>

---

4) Intro to JavaScript (Udacity)

Specs: Duration: Approx. 2 Weeks | Price/Fee: Free | Certification: Yes | Level: Beginner | Prerequisites: No prior experience is necessary for this course.

Intro to JavaScript is an online course that is ideal for beginners looking to add a new programming language to their skillset. JavaScript is a foundational programming language for developers who want to begin a career in web development.



Key topics:

- What is JavaScript
- Data Types & Variables
- Conditions
- Learning Loops
- Functions and Arrays

Features:

- Rich Learning Content
- Interactive Quizzes
- Self-Paced Learning
- Taught by Industry Pros

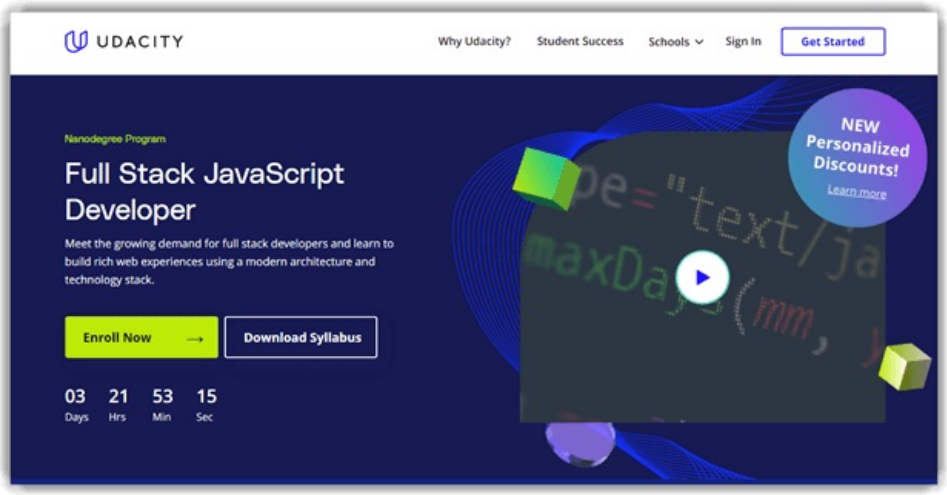
Enroll Now >>

---

5) Full Stack JavaScript Developer (Udacity)

Specs: Duration: 4 Months | Price/Fee: Free | Certification: Yes | Level: Intermediate | Prerequisites: Knowledge of HTML, CSS, Basic JavaScript, and JSON.

Full Stack JavaScript Developer is an online course. It this class, you will be able to master the skills necessary to become a successful JavaScript full-stack developer. You will also learn how to create APIs and server-side business logic and develop the persistence layer to store, process, and retrieve data using JavaScript.



Key topics:

- Backend Development with Node.js
- Creating an API with PostgreSQL and Express JS
- Angular Fundamentals
- Deployment Process

Features:

- Rich Learning Content
- Offers real-world projects from industry experts
- Self-Paced Learning
- Taught by Industry Pros

Enroll Now >>

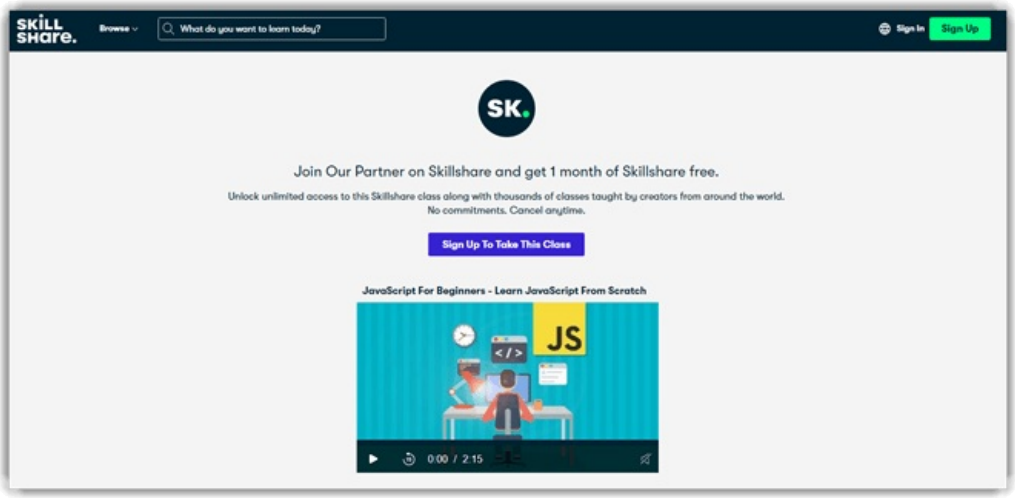
---

6) JavaScript For Beginners – Learn JavaScript From Scratch (Skillshare)

Specs: Duration: Self-placed | Certification: Yes | Level: Beginner

JavaScript For Beginners is a beginner JavaScript programming course that helps you learn and understand all JavaScript concepts. It also helps you to effectively learn how to learn the JavaScript programming language.

This JavaScript basics course will show you, as a complete beginner, how to begin creating programs using JavaScript. This JavaScript online class helps anyone to learn web development who only knows HTML, CSS, or any other programming language and never worked with JavaScript before.



Key topics:

- What You Will Be Learning
- Creating Your Very First JavaScript Program
- What is JavaScript?
- JavaScript Console Challenge
- What is a Variable?
- Combining Strings Together
- The Variable Coding Challenge

Features:

- Unlimited access to every class
- Supportive online creative community
- Supported platforms: iOS and Google Play

Enroll Now >>

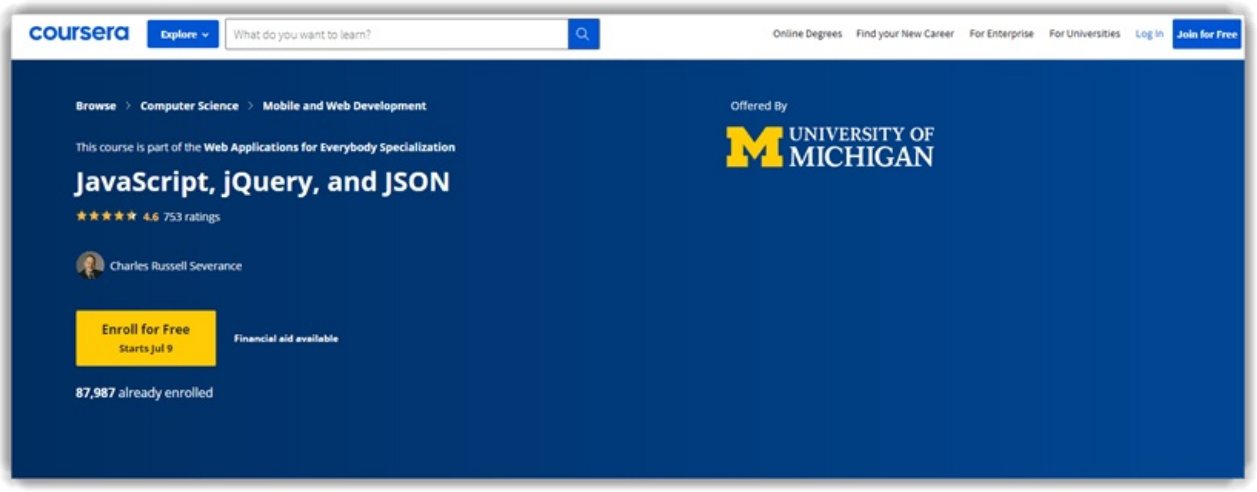
---

7) JavaScript, jQuery, and JSON (Coursera)

Specs: Duration: Approx. 26 hours to complete | Price/Fee: Free | Certification: Yes

JavaScript, jQuery, and JSON is an online JavaScript course. You will learn how JavaScript supports the Object-Oriented pattern, focusing on the unique aspect of how JavaScript approaches OOPS concepts. The course also briefly introduces the jQuery library.

In this JavaScript Bootcamp, you will learn more about JavaScript Object Notation (JSON), which is used as a syntax to exchange data between code running on the server (i.e., in PHP) and code running in the browser.



Key topics:

- Introduction to JavaScript
- JavaScript Objects
- How to use JQuery
- JSON – JavaScript Object Notation

Features:

- Shareable course certificate
- Self-paced learning option
- Course videos & readings
- Graded assignments with peer feedback
- Graded quizzes with feedback
- 100% online course
- Flexible schedule
- Graded programming assignments

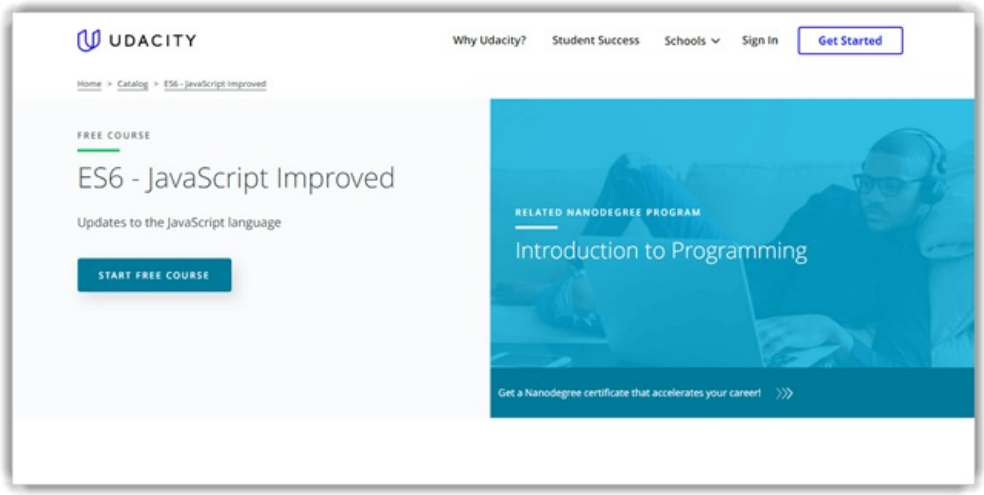
Enroll Now >>

---

8) ES6 – JavaScript Improved (Udacity)

Specs: Rating: 4.4 | Duration: Approx. 4 Weeks | Price/Fee: Free | Certification: Yes | Prerequisites: Knowledge of basic JavaScript

ECMAScript 6, or ES6, has brought about a ton of changes to the JavaScript programming language. In this course, you will explore those changes to learn about the latest features and improvements to the JavaScript language, including new keywords, arrow functions, the Class syntax, Promises, and much more.



Key topics:

- Learning about Syntax
- JavaScript Functions
- Built-ins
- Professional development

Features:

- Instructor videos
- Learn by doing exercise
- Rich Learning Content
- Interactive Quizzes
- Taught by Industry Pros
- Self-Paced Learning

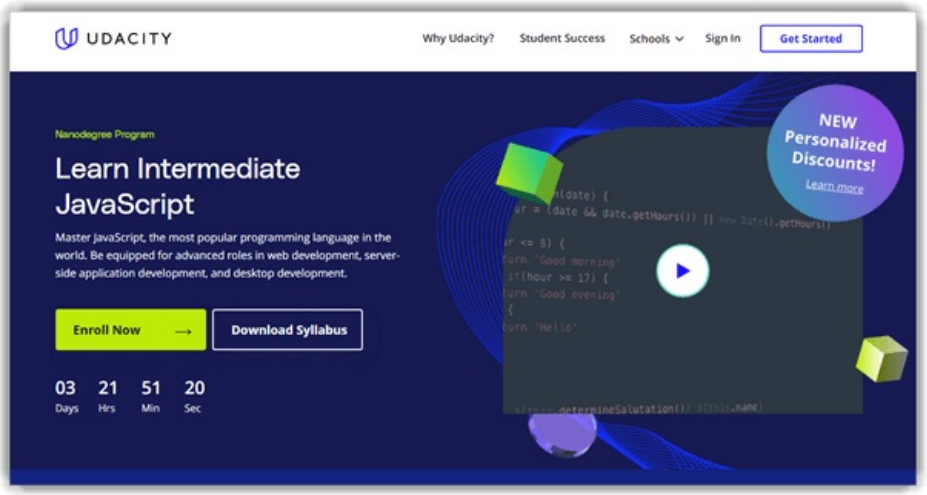
Enroll Now >>

---

9) Learn Intermediate JavaScript (Udacity)

Specs: Duration: 3 months | Price/Fee: Free | Certification: Yes

Learn Intermediate JavaScript is an online course that helps students learn and prepare for web development, server-side application development, and desktop development that need a more advanced set of JavaScript skills. This online course also provides the skills required to use JavaScript frameworks like AngularJS, ReactJS, and VueJS.



Key topics:

- Object-Oriented JavaScript
- Functional Programming
- Asynchronous Programming in JavaScript

Features:

- Real-world projects from industry experts
- Technical mentor support
- Flexible learning program

Enroll Now >>

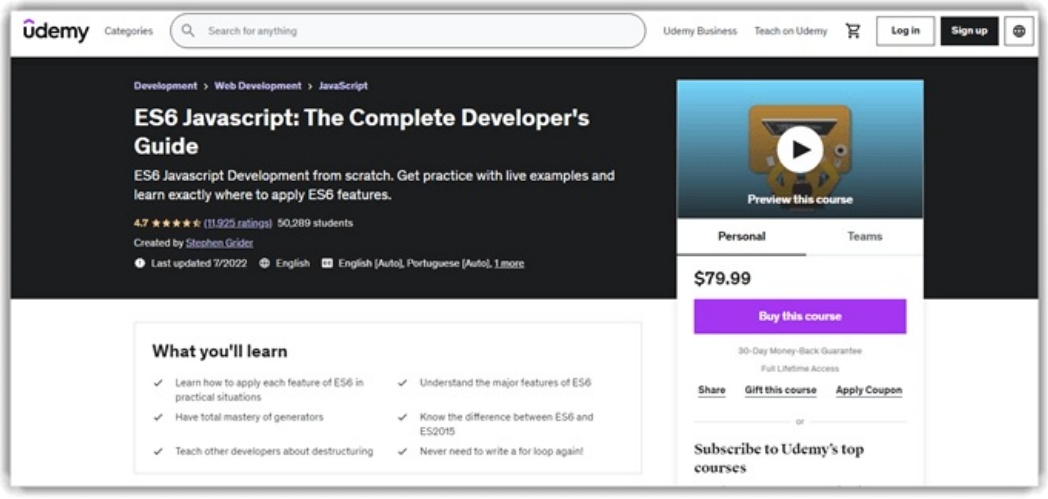
---

10) ES6 JavaScript: The Complete Developer's Guide (Udemy)

Specs: Rating: 4.7 | Duration: 6 hours on-demand video | Certification: Yes

ES6 JavaScript: The Complete Developer's Guide is an online JavaScript course. It will get you up and running quickly and teach you the core knowledge you need to understand for building applications using each new piece of JavaScript syntax introduced with ES6.

After introducing Array helpers, this JavaScript training course also teaches you advanced ES6, covering topics like enhanced object literals, all the default function arguments, and classes. All these topics include multiple live code exercises and exams to ensure that you understand each new concept easily.



Key topics:

- How you can apply each feature of ES6 in practical situations
- Understand the major features of ES6
- Know the difference between ES6 and ES2015
- Teach other developers about destructring

Features:

- 2 articles
- 36 coding exercises
- Full lifetime access
- Access on mobile and TV

Enroll Now >>

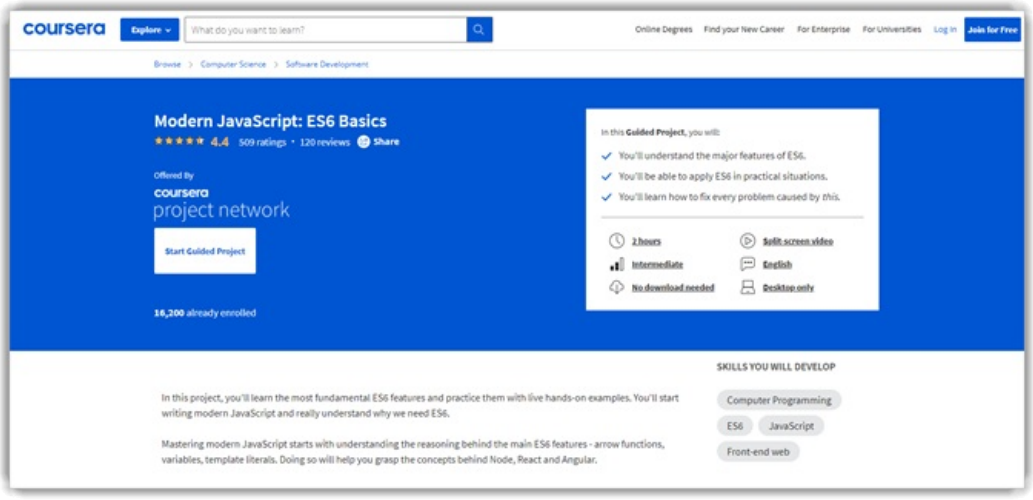
---

11) Modern JavaScript: ES6 Basics (Coursera)

Specs: Rating: 4.4 | Duration: 2 hours | Price/Fee: Free Trial | Certification: Yes

Modern JavaScript: ES6 Basics is an online JavaScript course where you can learn the fundamental ES6 features and practice them with live hands-on examples. You will be able to write modern JavaScript applications and understand the importance of ES6.

This one of the best JavaScript certification courses helps you learn the reasoning behind the main ES6 features like arrow functions, variables, templates, literals, etc.



Key topics:

- Variables (let) and Scoping.
- Variables (const) and Immutability.
- Extracting data with Destructuring.
- Strings and Interpolation.
- Learn about the major features of ES6.
- Arrow functions basics.
- How to apply ES6 in practical situations.
- You will learn how to fix every problem.

Features:

- Course video tutorials
- Self-paced
- No download is needed
- Graded quizzes with feedback
- 100% online course
- Flexible schedule
- Graded programming assignments

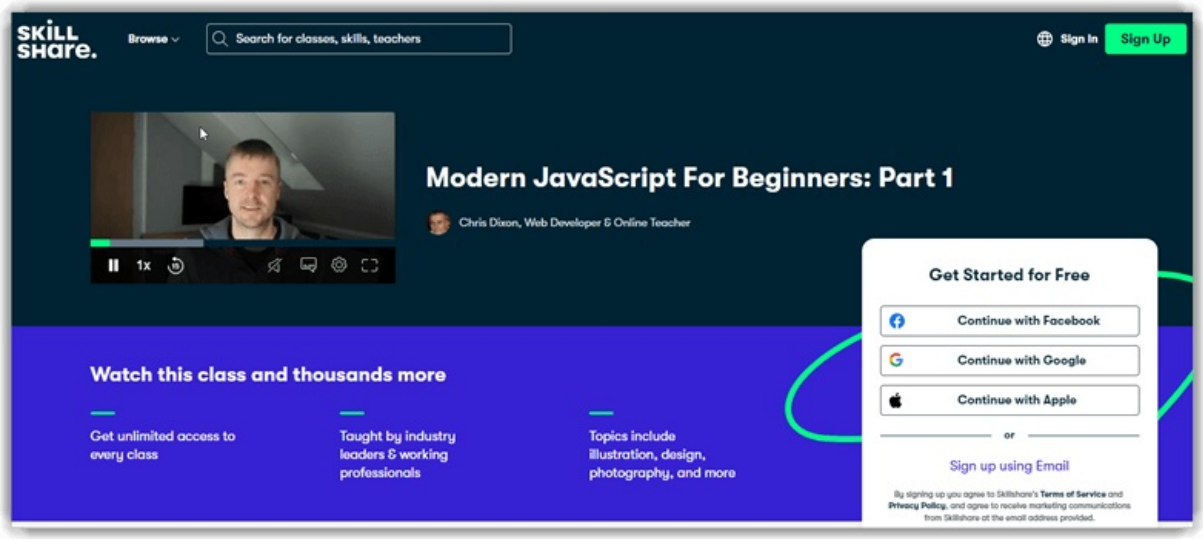
Enroll Now >>

---

12) Modern JavaScript For Beginners: Part 1 (Skillshare)

Specs: Duration: 5 Hours 29 Minutes | Price/Fee: 1-Month Free Trial | Certification: Yes | Level: Beginner

Modern JavaScript For Beginners is a course that helps you to improve your skillsets and job prospects by getting comfortable with JavaScript. It helps you learn the core principles of JavaScript like variables, data types, conditions, and advanced topics like loops, DOM scripting, etc. This course will also help you understand JavaScript programming through practical examples and mini-projects.



Key topics:

- Where To Add Javascript
- Mixing Strings With Variables
- Mixing Strings With Variables
- Data Types: Boolean, Null & Undefined
- Operators: Assignment & Comparison
- Introduction To Objects
- Introduction To Properties, Methods & The Prototype

Features:

- 1 project assignment for practice
- Access to tablet and phone
- Digital certificate on completion

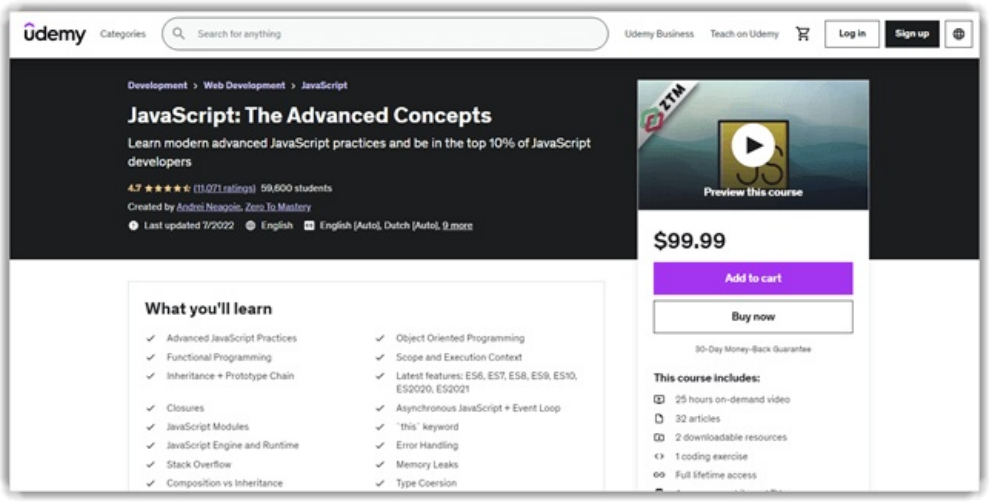
Enroll Now >>

---

13) JavaScript: The Advanced Concepts (Udemy)

Specs: Rating: 4.7 | Duration: 25 hours on-demand video | Certification: Yes

JavaScript: The Advanced Concepts is an online course where you will learn beyond just basics like other JavaScript online courses. You will learn patterns, methods, and best practices for JavaScript. This online course also helps you become a JavaScript developer by going beyond the simple basics that many courses cover. This JavaScript training helps you become a confident advanced JavaScript developer.



Key topics:

- Advanced JavaScript Practices
- Object-Oriented Programming
- Functional Programming
- Scope and Execution Context
- Inheritance + Prototype Chain
- Latest features: ES6, ES7, ES8, ES9, ES10, ES2020
- Asynchronous JavaScript + Event Loop
- JavaScript Modules
- JavaScript Engine and Runtime
- Error Handling
- Stack Overflow
- Composition vs. Inheritance
- Type Coercion
- Pass By Reference vs. Pass by Value
- Higher-Order Functions
- Interpreter/ Compiler/ JIT Compiler
- Garbage Collection
- JavaScript best practices

Features:

- 32 articles
- 2 downloadable resources
- 1 coding exercise
- Access on mobile and TV

Enroll Now >>

---

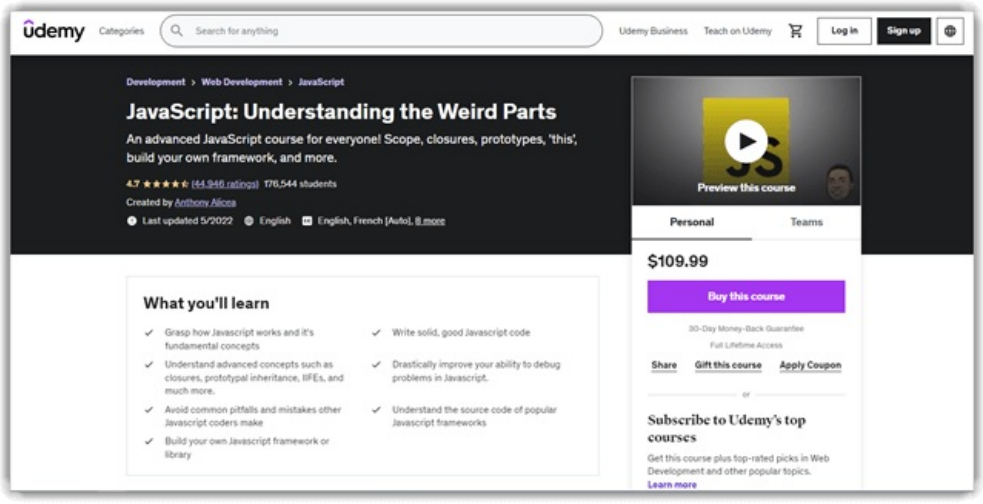
14) JavaScript: Understanding the Weird Parts (Udemy)

Specs: Rating: 4.7 | Duration: 12 hours on-demand video | Certification: Yes | Prerequisites: Those with basic Javascript skills who wish to improve

JavaScript: Understanding the Weird Parts is an online programming course. It helps you to gain a deep understanding of JavaScript, learn how JavaScript works, and how your JavaScript knowledge helps you avoid

specific issues and improve your ability to debug problems.

This JavaScript course covers advanced concepts like objects and object literals, functional expressions, prototypical inheritance, functional programming, scope chains, apply, bind, and more.



Key topics:

- Write solid, good JavaScript code
- How to avoid common issues and mistakes that other JavaScript coders make.
- Understand advanced concepts like prototypal inheritance closures, IIFEs, etc.
- You can improve your ability to debug problems in JavaScript.
- How to build your JavaScript framework or library.

Features:

- 7 articles
- 50 downloadable resources
- Full lifetime access
- Access on mobile and TV

Enroll Now >>

Other Useful Programming Resources you may like to explore

- Best FREE Udemy Courses (100% Off Coupon)
- 10+ Best FREE Online Google Courses With Certification
- 25+ Best Free Online Education Sites
- 200 Best FREE Coursera Courses with Certificates
- 160 Best Udacity Free Online Courses
- 65+ BEST Udemy Courses Online
- 30 Best FREE edX Courses with Certificates
- 35 Best LinkedIn Learning Courses with Certifications
- 30 BEST Online Coding Courses to Learn Code for Free/Paid

FAQ:

? Do I get a Printable Certificate?

Yes, you will get a printable certificate in many courses. Some course providers will ship a hard copy of the

certificate to your desired address.

⚡ What eligibility is required to join a JavaScript course?

For most of the courses:

- No previous JavaScript experience is necessary.
- Basic knowledge of HTML and CSS.
- Basic computer skills and Internet access.
- Desire to learn about JavaScript.

What if I miss a class?

All the classes are recorded and can be replayed later.

What if I do not like a JavaScript course I purchased?

There are many courses that come with a 30-day return policy or have a 7-day free trial.

How can I ask my doubts or questions?

Most courses have a forum that allows you to raise questions that are frequently answered by course authors.

❗ Why Learn JavaScript?

Here are some prominent reasons to learn JavaScript:

JavaScript is easy to use:

The most prominent reason to use JavaScript by new developers is that it is a higher-level language. So many complexities of your code are done by the machine instead of in your own.

JavaScript offers a simple syntax that resembles the English language, making it easy to build strong mental connections to your covering programming concepts.

JavaScript is versatile:

JavaScript helps you add many interactive features to a website, from video players to functional buttons and images that change when clicked.

So-called “vanilla JavaScript” (where you only use JavaScript) is only part of coding in JavaScript. So if you have mastered the basics of JavaScript, you can extend your skills and use them to master a library like Angular, React.js, and Vue.js. These technologies are all widely used for web development.

You can also use the skills which you have mastered while learning JavaScript with a framework like Node.js to build back-end web apps. Or you can use your JavaScript skills with React Native or other mobile frameworks to build a mobile app.

JavaScript developers are in high demand:

Whether or not you are looking to start a career in the development field. It is always comforting to know that your JavaScript skills can be easily transformed when needed. JavaScript is a crucial part of the Internet that many companies are constantly looking for talented developers with JavaScript proficiency.

What are the best JavaScript Certifications?

Here are some of the best JavaScript Certifications:

- JavaScript for Web Designers
- Modern JavaScript From The Beginning
- The Complete JavaScript Course 2022: From Zero to Expert!
- Intro to JavaScript
- Full Stack JavaScript Developer
- JavaScript For Beginners – Learn JavaScript From Scratch
- JavaScript, jQuery, and JSON
- ES6 – JavaScript Improved
- Learn Intermediate JavaScript
- ES6 JavaScript: The Complete Developer's Guide
- Modern JavaScript: ES6 Basics
- JavaScript Essential Training
- JavaScript: The Advanced Concepts
- JavaScript: Understanding the Weird Parts

? What are the prerequisites for the JavaScript course?

Here are the prerequisites:

- Basic knowledge of HTML and CSS.
- Knowledge of basic JavaScript.

Can a person without prior experience in programming learn JavaScript?

Yes, JavaScript can be learned without any prior experience.

← Prev

Report a Bug

Next →

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,
United States

About
About Us
Advertise with Us
Write For Us
Contact Us

Career Suggestion
SAP Career Suggestion Tool
Software Testing as a Career

Interesting
eBook
Blog
Quiz
SAP eBook
Privacy Manager

Top 100 JavaScript Interview Questions and Answers (2024)

By : James Hartman April 29, 2024

Here are JavaScript interview questions and answers for fresher as well as experienced candidates to get their dream job.

Table of Contents:

JavaScript Interview Questions for Freshers

1. What is JavaScript?

JavaScript is a very powerful client-side scripting language. JavaScript is used mainly for enhancing the interaction of a user with the webpage. In other words, you can make your webpage more lively and interactive, with the help of JavaScript. JavaScript is also being used widely in game development and Mobile application development.

2. Enumerate the differences between Java and JavaScript?

Java is a complete programming language. In contrast, JavaScript is a coded program that can be introduced to HTML pages. These two languages are not at all inter-dependent and are designed for different intent. Java is an object-oriented programming (OOPS) or structured programming languages like C++ or C, whereas JavaScript is a client-side scripting language.

3. What are JavaScript Data Types?

Following are the JavaScript Data types:

- Number
- String
- Boolean
- Object
- Undefined

4. What is the use of isNaN function?

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

5. Which is faster between JavaScript and an ASP script?

JavaScript is faster. JavaScript is a client-side language,, and thus it does not need the assistance of the webserver to execute. On the other hand, ASP is a server-side language and hence is always slower than JavaScript. Javascript now is also a server-side language (nodejs).

6. What is negative Infinity?

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

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

Breaking within a string statement can be done by using a backslash, '\,' at the end of the first line.

Example:

```
document. Write ("This is \a program,");
```

And if you change to a new line when not within a string statement, then javaScript ignores the break in the line.

Example:

```
var x=1, y=2,
z=
x+y;
```

The above code is perfectly fine, though not advisable as it hampers debugging.

Don't Miss:

- Execute JavaScript Online
- QuickSort Algorithm in JavaScript

- 15 Best FREE JavaScript Certification Courses Online (2024)

8. Which company developed JavaScript?

Netscape is the software company that developed JavaScript.

9. 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.

10. 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>
```

11. 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”;

The problems faced by using global variables are the clash of variable names of local and global scope. Also, it is difficult to debug and test the code that relies on global variables.

12. What is a prompt box?

A prompt box is a box that allows the user to enter input by providing a text box. A label and box will be provided to enter the text or number.

---

13. What is ‘this’ keyword in JavaScript?

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

---

14. 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.

---

15. Which symbol is used for comments in Javascript?

// for Single line comments and

/* Multi

Line

Comment

*/

---

16. 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.

---

17. What is === operator?

=== is called a strict equality operator, which returns true when the two operands have the same value without

conversion.

---

18. How you can submit a form using JavaScript?

To submit a form using JavaScript use

```
document.form[0].submit();
document.form[0].submit();
```

---

19. Does JavaScript support automatic type conversion?

Yes, JavaScript does support automatic type conversion. It is the common way of type conversion used by JavaScript developers

---

20. How can the style/class of an element be changed?

It can be done in the following way:

```
document.getElementById("myText"). style. fontSize = "20";
```

or

```
document. getElementById ("myText"). className = "anyclass";
```

---

21. How to read and write a file using JavaScript?

There are two ways to read and write a file using JavaScript

- Using JavaScript extensions
- Using a web page and Active X objects

---

22. What are all the looping structures in JavaScript?

Following are looping structures in Javascript:

- For
- While
- Do-while loops

---

23. What is called Variable typing in Javascript?

Variable typing is used to assign a number to a variable. The same variable can be assigned to a string.

Example:

```
i = 10;
i = "string;"
```

This is called variable typing.

---

24. 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);
```

---

25. 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.

JavaScript Interview Questions for Experienced

26. What would be the result of 3+2+”7″?

Since 3 and 2 are integers, they will be added numerically. And since 7 is a string, its concatenation will be done. So the result would be 57.

---

27. How to detect the operating system on the client machine?

In order to detect the operating system on the client machine, the navigator. Platform string (property) should be used.

---

28. 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.

29. What is the function of the delete operator?

The delete keyword is used to delete the property as well as its value.

Example

```
var student= {age:20, batch:"ABC"};
Delete student. age;
```

30. 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.

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

- Alert
- Confirm and
- Prompt

32. 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.

33. 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="https://www.guru99.com/javascript-interview-questions-answers.html"; //-->
</script>
```

34. What is the data type of variables in JavaScript?

All variables in JavaScript are object data types.

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

An alert box displays only one button, which is the OK button.

But a Confirmation box displays two buttons, namely OK and cancel.

36. What are escape characters?

Escape characters (Backslash) is used when working with special characters like single quotes, double quotes, apostrophes, and ampersands. Place backslash before the characters to make it display.

Example:

```
document. write "I m a "good" boy."
document. write "I m a \"good\" boy."
```

37. 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.

38. 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
```

39. 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.

40. What are the disadvantages of using innerHTML in JavaScript?

If you use innerHTML in JavaScript, the disadvantage is

- Content is replaced everywhere
- We cannot use it like “appending to innerHTML
- Even if you use +=like “innerHTML = innerHTML + ‘html'” still the old content is replaced by html
- The entire innerHTML content is re-parsed and builds into elements. Therefore, it’s much slower
- The innerHTML does not provide validation, and therefore we can potentially insert valid and broken HTML in the document and break it

---

41. What is break and continue statements?

Break statement exits from the current loop.

Continue statement continues with next statement of the loop.

---

42. What are the two basic groups of data types in JavaScript?

- They are as—Primitive
- Reference types

Primitive types are number and Boolean data types. Reference types are more complex types like strings and dates.

---

43. How can generic objects be created?

Generic objects can be created as:

```
var I = new object();
```

---

44. 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.

---

45. 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
}
```

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

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

47. What is the use of the blur function?

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

48. 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";
```

49. How to find an operating system in the client machine using JavaScript?

The ‘Navigator. the app version is used to find the operating system’s name in the client machine.

50. 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.

JavaScript Interview Questions for 5 Years Experience

51. What is the use of the Push method in JavaScript?

The push method is used to add or append one or more elements to an Array end. Using this method, we can append multiple elements by passing multiple arguments.

52. What is the unshift method in JavaScript?

Unshift method is like the push method, which works at the beginning of the array. This method is used to prepend one or more elements to the beginning of the array.

53. What is the difference between JavaScript and Jscript?

Both are almost similar. Netscape and Jscript develop JavaScript was developed by Microsoft.

54. How are object properties assigned?

Properties are assigned to objects in the following way –

```
obj ["class"] = 12;
or

obj.class = 12;
```

---

55. 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";
}
```

---

56. 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.

---

57. How can the OS of the client machine be detected?

The navigator. appVersion string can be used to detect the operating system on the client machine.

---

58. What is a window.onload and onDocumentReady?

The onload function is not run until all the information on the page is loaded. This leads to a substantial delay before any code is executed.

onDocumentReady loads the code just after the DOM is loaded. This allows early manipulation of the code.

---

59. How closures work in JavaScript?

The closure is a locally declared variable related to a function that stays in memory when it has returned.

For example:

```
function greet(message) {
    console.log(message);
}
function greeter(name, age) {

    return name + " says howdy!! He is " + age + " years old";
}
// Generate the message
var message = greeter("James", 23);
// Pass it explicitly to greet
greet(message);
This function can be better represented by using closures
function greeter(name, age) {
    var message = name + " says howdy!! He is " + age + " years old";
    return function greet() {
        console.log(message);
    };
}
// Generate the closure
var JamesGreeter = greeter("James", 23);
// Use the closure
JamesGreeter();
```

---

60. 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;
```

---

61. What is for-in loop in Javascript?

The for-in loop is used to loop through the properties of an object.

The syntax for the for-in loop is –

```
for (variable name in object){
    statement or block to execute
}
```

In each repetition, one property from the object is associated with the variable name. The loop is continued till all the properties of the object are depleted.

---

62. What are the important properties of an anonymous function in JavaScript?

A function that is declared without any named identifier is known as an anonymous function. In general, an anonymous function is inaccessible after its declaration.

Anonymous function declaration –

```
var anon = function() {
    alert('I am anonymous');
};
anon();
```

---

63. 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>'
```

---

64. 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.

---

65. Is JavaScript case sensitive? Give its example.

Yes, JavaScript is case-sensitive. For example, a function parseInt is not the same as the function Parseint.

---

66. What boolean operators can be used in JavaScript?

The 'And' Operator (&&), 'Or' Operator (||), and the 'Not' Operator (!) can be used in JavaScript.

*Operators are without the parenthesis.

67. How can a particular frame be targeted, from a hyperlink, in JavaScript?

This can be done by including the name of the required frame in the hyperlink using the ‘target’ attribute.

```
<a href="/newpage.htm" target="newframe">>New Page</a>
```

68. What is the role of break and continue statements?

The break statement is used to come out of the current loop. In contrast, the continue statement continues the current loop with a new recurrence.

69. 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.

70. 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”

71. 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);
```

72. How are DOM utilized in JavaScript?

DOM stands for Document Object Model and is responsible for how various objects in a document interact with each other. DOM is required for developing web pages, which includes objects like paragraphs, links, etc. These objects can be operated to include actions like add or delete. DOM is also required to add extra capabilities to a web page. On top of that, the use of API gives an advantage over other existing models.

73. How are event handlers utilized in JavaScript?

Events are the actions that result from activities, such as clicking a link or filling a form by the user. An event handler is required to manage the proper execution of all these events. Event handlers are an extra attribute of the object. This attribute includes the event’s name and the action taken if the event takes place.

74. What is the role of deferred scripts in JavaScript?

The HTML code’s parsing during page loading is paused by default until the script has not stopped executing. If

the server is slow or the script is particularly heavy, then the web page is delayed.

While using Deferred, scripts delays execution of the script till the time the HTML parser is running. This reduces the loading time of web pages, and they get displayed faster.

---

75. 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.

---

76. 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.

---

77. 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

---

78. 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 "]
```

79. 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!

80. 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 uri="my test.asp?name=ståle&car=saab";

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

    document.write(decodeURI(uri));
</script>
```

Output –

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

my test.asp?name=ståle&car=saab

JavaScript Interview Questions for 10+ Years Experience

81. 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.

---

82. What does the following statement declare?

```
var myArray = [[[]]];
```

It declares a three-dimensional array.

---

83. How are JavaScript and ECMA Script related?

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

---

84. 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.

---

85. 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.

---

86. 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.

---

87. How to use Loops in Javascript?

There are mainly four types of loops in JavaScript.

for loop

for/in a loop (explained later)

while loop

do…while loop

for loop

Syntax:

```
for(statement1; statement2; statment3)

{

lines of code to be executed

}
```

1. Statement1 is executed first, even before executing the looping code. So, this statement is normally used to assign values to variables used inside the loop.
2. The statement2 is the condition to execute the loop.
3. The statement3 is executed every time after the looping code is executed.

```
<html>
<head>
    <script type="text/javascript">
        var students = new Array("John", "Ann", "Aaron", "Edwin", "Elizabeth");
        document.write("<b>Using for loops </b><br />");
        for (i=0;i<students.length;i++)
        {
        document.write(students[i] + "<br />");
        }
    </script>
</head>
<body>
</body>
</html>
```

while loop

Syntax:

```
while(condition)

{

lines of code to be executed

}
```

The “while loop” is executed as long as the specified condition is true. Inside the while loop, you should include the statement that will end the loop at some point in time. Otherwise, your loop will never end, and your browser may crash.

do…while loop

Syntax:

```
<pre>
do

{

block of code to be executed

} while (condition)
```

The do…while loop is very similar to the while loop. The only difference is that in do…while loop, the block of code gets executed once even before checking the condition.

Example:

```
<html>
<head>
    <script type="text/javascript">
        document.write("<b>Using while loops </b><br />");
        var i = 0, j = 1, k;
        document.write("Fibonacci series less than 40<br />");
        while(i<40)
        {
            document.write(i + "<br />");
            k = i+j;
            i = j;
            j = k;
        }
    </script>
</head>
<body>
</body>
</html>
```

---

88. 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>
```

```
<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>
```

---

89. 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} };
```

---

90. What is Loop Though 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>
```

---

91. 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.

---

92. What are some important JavaScript Unit Testing Frameworks?

Following is a curated list of popular JavaScript Unit Testing Frameworks and Tools that are widely used :

Unit.js: It is known as an open-source assertion library running on browser and Node.js. It is extremely compatible with other JavaScript Unit Testing frameworks like Mocha, Karma, Jasmine, QUnit, Protractor, etc. Provides the full documented API of assertion list.

QUnit: It is used for both client-side and server-side JavaScript Unit Testing. This Free JavaScript testing framework is used for jQuery projects. It follows Common JS unit testing Specification for unit testing in JavaScript. It supports the Node Long-term Support Schedule.

Jasmine: Jasmine is the behavior-driven development framework to unit test JavaScript. It is used for testing both synchronous and asynchronous JavaScript codes. It does not require DOM and comes with an easy syntax that can be written for any test.

Karma: Karma is an open-source productive testing environment. Easy workflow control running on the command line. Offers the freedom to write the tests with Jasmine, Mocha, and QUnit. You can run the test on real devices with easy debugging.

Mocha: Mocha runs on Node.js and in the browser. Mocha performs asynchronous testing more simply. Provides accuracy and flexibility in reporting. Provides tremendous support of rich features such as test-specific timeouts, JavaScript APIs.

Jest: Facebook uses jest so far to test all the JavaScript code. It provides the 'zero-configuration testing experience. Supports independent and non-interrupting running tests without any conflict. Do not require any other setup configuration and libraries.

AVA: AVA is a simple JavaScript Unit Testing Framework. Tests are being run in parallel and serially. Parallel tests run without interrupting each other. This testing framework supports asynchronous testing as well. AVA uses subprocesses to run the unit test JavaScript.

---

93. What is QuickSort Algorithm in JavaScript?

Quick Sort algorithm follows Divide and Conquer approach. It divides elements into smaller parts based on some conditions and performing the sort of operations on those divided smaller parts.

Quick Sort algorithm is one of the most used and popular algorithms in any programming language. If you are a JavaScript developer, you might have heard of sort() which is already available in JavaScript. Then, you might have been thinking about what the need for this Quick Sort algorithm is. To understand this, first, we need what is sorting and what is the default sorting in JavaScript.

Quicksort follows the Divide-and-Conquer algorithm. It divides elements into smaller parts based on some conditions and performs the sort operations on those divided smaller parts. Hence, it works well for large datasets. So, here are the steps of how Quicksort works in simple words.

1. First, select an element that is to be called the pivot element.
2. Next, compare all array elements with the selected pivot element and arrange them so that elements less than the pivot element are left. Greater than pivot is to its right.
3. Finally, perform the same operations on the left and right side elements to the pivot element.

So, that is the basic outline of Quicksort. Here are the steps which need to be followed one by one to perform Quicksort.

94.How does QuickSort Work

Step 1) First, find the “pivot” element in the array.

Step 2) Start the left pointer at the first element of the array.

Step 3) Start the right pointer at the last element of the array.

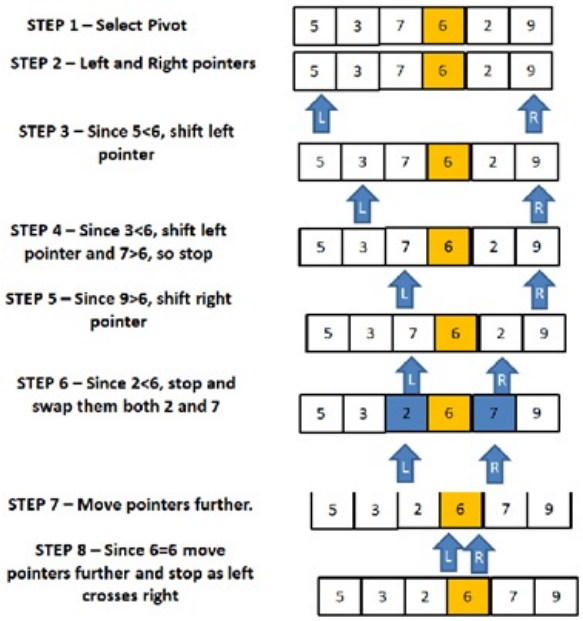
Step 4) Compare the element pointing with the left pointer, and if it is less than the pivot element, then move the left pointer to the right (add 1 to the left index). Continue this until the left side element is greater than or equal to the pivot element.

Step 5) Compare the element pointing with the right pointer. If it is greater than the pivot element, move the right pointer to the left (subtract 1 to the right index). Continue this until the right-side element is less than or equal to the pivot element.

Step 6) Check if the left pointer is less than or equal to a right pointer, then saw the elements in these pointers’ locations.

Step 7) Increment the left pointer and decrement the right pointer.

Step 8) If the left pointer index is still less than the right pointer’s index, repeat the process; else, return the left pointer’s index.



So, let us see these steps with an example. Let us consider an array of elements which we need to sort is [5,3,7,6,2,9].

Here are the steps to perform Quick sort that is being shown with an example [5,3,7,6,2,9].

STEP 1) Determine pivot as a middle element. So, 7 is the pivot element.

STEP 2) Start left and right pointers as first and last elements of the array, respectively. The left pointer points to 5 at index 0, and the right pointer points to 9 at index 5.

STEP 3) Compare the left pointer element with the pivot element, since 5 < 6 shift left pointer to the right to index 1.

STEP 4) Now, still 3 <6, so shift the left pointer to one more index to the right. Now 7 > 6 stops incrementing the left pointer, and now the left pointer is index 2.

STEP 5) Now, compare the value at the right pointer with the pivot element. Since 9 > 6, move the right pointer to the left. Now, as 2 < 6, stop moving the right pointer.

STEP 6) Swap both values present at left and right pointers with each other.

STEP 7) Move both pointers one more step.

STEP 8) Since 6 = 6, move pointers to one more step and stop as the left pointer crosses the right pointer and returns the left pointer's index.

Here, based on the above approach, we need to write code for swapping elements and partitioning the array as mentioned in the above steps.

Example:

```
var items = [5,3,7,6,2,9];
function swap(items, leftIndex, rightIndex){
    var temp = items[leftIndex];
    items[leftIndex] = items[rightIndex];
    items[rightIndex] = temp;
}
function: partition(items, left, right) {
    var pivot   = items[Math.floor((right + left) / 2)], //middle element
```

```
        i       = left, //left pointer
        j       = right; //right pointer
    while (i <= j) {
        while (items[i] < pivot) {
            i++;
        }
        while (items[j] > pivot) {
            j--;
        }
        if (i <= j) {
            swap(items, i, j); //sawpping two elements
            i++;
            j--;
        }
    }
    return i;
}

function quickSort(items, left, right) {
    var index;
    if (items.length > 1) {
        index = partition(items, left, right); //index returned from partition
        if (left < index - 1) { //more elements on the left side of the pivot
            quickSort(items, left index - 1);
        }
        if (index < right) { //more elements on the right side of the pivot
            quickSort(items, index, right);
        }
    }
    return items;
}
// first call to quick sort
var sortedArray = quickSort(items, 0, items.length - 1);
console.log(sortedArray); //prints [2,3,5,6,7,9]
```

---

95. What is DOM in JavaScript?

JavaScript can access all the elements in a web page using the Document Object Model (DOM). The web browser creates a DOM of the webpage when the page is loaded.

---

96. 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>
```

---

97. 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);
```

---

98. 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.

---

99. 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.

```
document.cookie = "cookiename=cookievalue"
```

You can even add an expiry date to your Cookie to remove the particular Cookie from the computer on the specified date. The expiry date should be set in the UTC/GMT format. If you do not set the expiry date, the cookie will be removed when the user closes the browser.

```
document.cookie = "cookiename=cookievalue; expires= Thu, 21 Aug 2014 20:00:00 UTC"
```

You can also set the domain and path to specify which domain and to which directories in the specific domain the Cookie belongs to. By default, a cookie belongs to the page that sets the Cookie.

```
document.cookie = "cookiename=cookievalue; expires= Thu, 21 Aug 2014 20:00:00 UTC; path=/
```

//create a cookie with a domain to the current page and a path to the entire domain.

JavaScript get Cookie

You can access the Cookie like this, which will return all the cookies saved for the current domain.

```
var x =  document.cookie
```

JavaScript Delete Cookie

To delete a cookie, you just need to set the cookie's value to empty and set the value of expires to a passed date.

Example:

```
<html>
<head>
    <title>Cookie!!!</title>
    <script type="text/javascript">
        function createCookie(cookieName,cookieValue,daysToExpire)
        {
          var date = new Date();
          date.setTime(date.getTime()+(daysToExpire*24*60*60*1000));
          document.cookie = cookieName + "=" + cookieValue + "; expires=" + date.toGMTString();
        }
        function accessCookie(cookieName)
        {
          var name = cookieName + "=";
          var allCookieArray = document.cookie.split(';');
          for(var i=0; i<allCookieArray.length; i++)
          {
            var temp = allCookieArray[i].trim();
            if (temp.indexOf(name)==0)
            return temp.substring(name.length,temp.length);
             }
            return "";
        }
        function checkCookie()
        {
          var user = accessCookie("testCookie");
          if (user!="")
            alert("Welcome Back " + user + "!!!");
          else
          {
            user = prompt("Please enter your name");
            num = prompt("How many days you want to store your name on your computer?");
            It (user!="" && user!=null)
            {
            createCookie("testCookie", user, num);
            }
          }
        }
    </script>
</head>
<body onload="checkCookie()"></body>
</html>
```

100. Give an example of JavaScript Multiplication Table

Here, are example of simple multiplication table asking the user the number of rows and columns he wants.

Example:

```
<html>
<head>
  <title>Multiplication Table</title>
  <script type="text/javascript">
    var rows = prompt("How many rows for your multiplication table?");
    var cols = prompt("How many columns for your multiplication table?");
    if(rows == "" || rows == null)
            rows = 10;
    if(cols== "" || cols== null)
            cols = 10;
    createTable(rows, cols);
    function createTable(rows, cols)
    {
      var j=1;
      var output = "<table border='1' width='500' cellspacing='0'cellpadding='5'>";
      for(i=1;i<=rows;i++)
      {
        output = output + "<tr>";
        while(j<=cols)
        {
            output = output + "<td>" + i*j + "</td>";
             j = j+1;
           }
            output = output + "</tr>";
            j = 1;
    }
    output = output + "</table>";
    document.write(output);
    }
  </script>
</head>
<body>
</body>
</html>
```

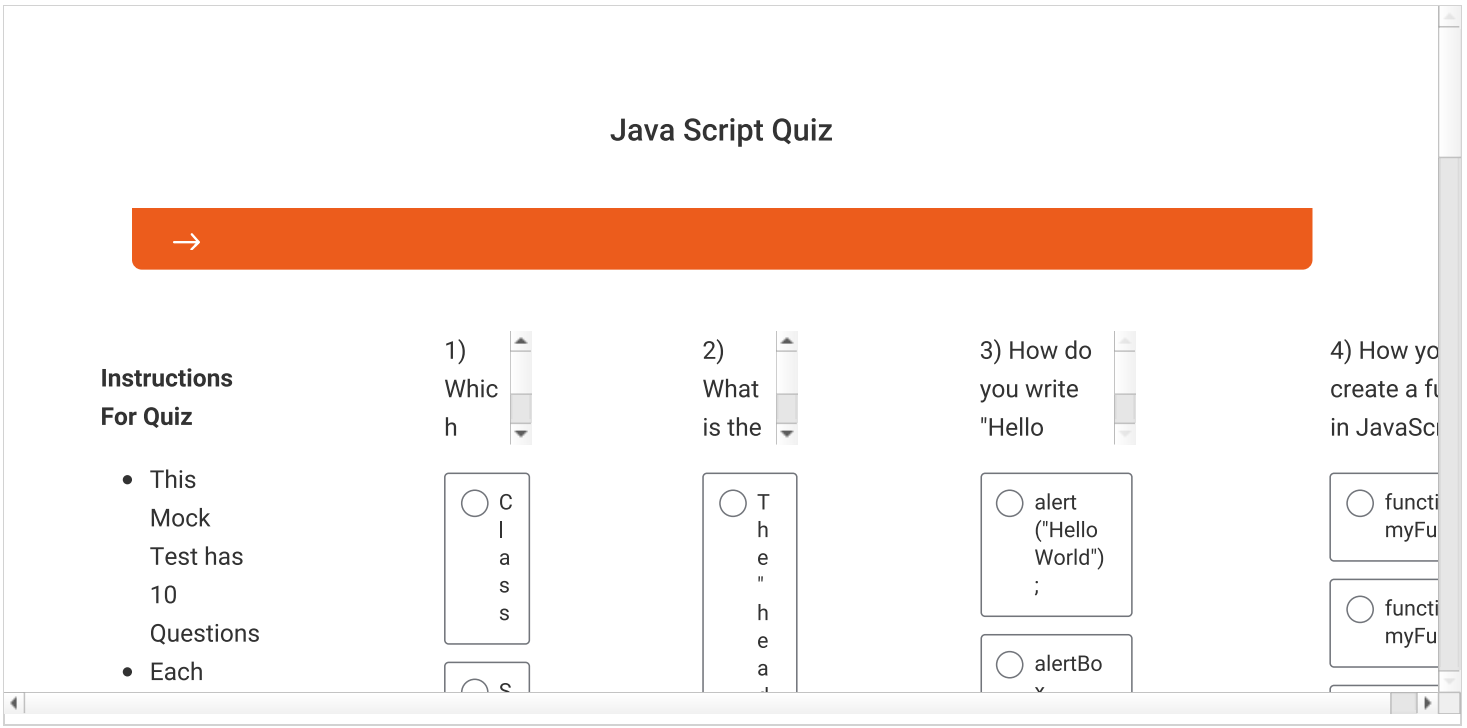
101. Explain Popup Message using event with example

Display a simple message “Welcome!!!” on your demo webpage and when the user hovers over the message, a popup should be displayed with a message “Welcome to my WebPage!!!”.

Example:

```
<html>
    <head>

    <title>Event!!!</title>
```

```
<script type="text/javascript">

function trigger()

{


document.getElementById("hover").addEventListener("mouseover", popup);

function popup()

{

alert("Welcome to my WebPage!!!");

}

}

</script>

<style>

p{
     font-size:50px;
     position: fixed;
     left: 550px;
     top: 300px;
}
</style>
</head>
<body  onload="trigger();">
<p id="hover">Welcome!!!</p>
</body>
</html>
```



These interview questions will also help in your viva(orals)

Don't Miss:

- Execute JavaScript Online
- QuickSort Algorithm in JavaScript
- 15 Best FREE JavaScript Certification Courses Online (2024)

← Prev

Report a Bug

Next →

Guru99's Headquarters

4023 Kennett Pike #50286,
Wilmington, Delaware,
United States