**Discuss TCP/IP Protocol**

The TCP/IP (Transmission Control Protocol/Internet Protocol) suite is a set of communication protocols that form the foundation for the Internet and most local networks. Developed in the 1970s and 1980s, TCP/IP has become the standard protocol suite for networking, and it provides a reliable and efficient means of communication between devices across diverse networks. Here's an overview of the key components of the TCP/IP protocol suite:

TCP (Transmission Control Protocol):

Functionality: TCP provides reliable, connection-oriented communication between devices. It ensures that data is delivered in the correct order and without errors. To achieve this, TCP uses a combination of sequence numbers, acknowledgments, and retransmission mechanisms.

Connection Establishment and Termination: TCP follows a three-way handshake process to establish a connection and a four-way handshake to terminate it.

IP (Internet Protocol):

Functionality: IP is responsible for the addressing and routing of packets across a network. It provides an addressing scheme to uniquely identify devices in a network and enables the routers to forward packets from the source to the destination.

Versions: There are two main versions of IP in use today: IPv4 (Internet Protocol version 4) and IPv6 (Internet Protocol version 6). IPv4 is the older version and uses 32-bit addresses, while IPv6 uses 128-bit addresses and was introduced to address the exhaustion of IPv4 addresses.

**What is a form ? Explain form components with example.**

In the context of web development and user interfaces, a "form" refers to an HTML (Hypertext Markup Language) element that allows users to input data and submit it to a server for processing. Forms are a crucial part of web applications and websites, providing a means for users to interact with and input information.

A basic form structure in HTML looks like this:

```html

<form action="/submit" method="post">

<!-- Form components go here -->

<label for="username">Username:</label>

<input type="text" id="username" name="username" required>

<label for="password">Password:</label>

<input type="password" id="password" name="password" required>

<input type="submit" value="Submit">

</form>

```

Let's break down the components of this form:

1. \*\*`<form>` element:\*\* This is the container for all the form components. It has attributes like `action` (specifying where the form data should be sent) and `method` (specifying the HTTP method to be used for submitting the form, commonly "get" or "post").

2. \*\*`<label>` element:\*\* This is used to associate a text label with a form control, enhancing accessibility and user experience. The `for` attribute in the `<label>` tag should match the `id` attribute of the associated form control.

3. \*\*`<input>` element:\*\* This is one of the most versatile form elements. It is used for various types of user input, such as text, password, checkbox, radio button, etc. The `type` attribute determines the kind of input field, and the `id` and `name` attributes help identify and associate the input with labels.

- \*\*Text Input Example:\*\*

```html

<label for="email">Email:</label>

<input type="text" id="email" name="email" required>

```

- \*\*Password Input Example:\*\*

```html

<label for="password">Password:</label>

<input type="password" id="password" name="password" required>

```

4. \*\*`<textarea>` element:\*\* This is used for multiline text input. It's useful when users need to enter longer passages of text.

```html

<label for="message">Message:</label>

<textarea id="message" name="message" rows="4" cols="50"></textarea>

```

5. \*\*`<select>` and `<option>` elements:\*\* These are used for dropdown menus. Users can select an option from the list.

```html

<label for="cars">Choose a car:</label>

<select id="cars" name="cars">

<option value="volvo">Volvo</option>

<option value="saab">Saab</option>

<option value="mercedes">Mercedes</option>

<option value="audi">Audi</option>

</select>

```

6. \*\*`<button>` element:\*\* This is used to create a clickable button within the form. It can be used to trigger form submission or perform other actions.

```html

<button type="submit">Submit</button>

```

These are some of the basic form components in HTML. Depending on the requirements, you can use additional attributes and elements to enhance the functionality and appearance of forms.

**Explain rowspan and colspan with an example.**

Rowspan and colspan are attributes used in HTML tables to control the spanning of cells across multiple rows or columns.

Rowspan:

The rowspan attribute is used to specify how many rows a cell should span vertically. It allows a cell to occupy multiple rows in a table. For example:

```html

<table>

<tr>

<td rowspan="2">Cell 1</td>

<td>Cell 2</td>

<td>Cell 3</td>

</tr>

<tr>

<td>Cell 4</td>

<td>Cell 5</td>

</tr>

<tr>

<td>Cell 6</td>

<td>Cell 7</td>

<td>Cell 8</td>

</tr>

</table>

```

In this example, the first cell in the first row has a rowspan of 2, which means it spans across two rows. As a result, it occupies the space of the first cell in the second row as well. The table will be rendered with the first cell spanning two rows.

Colspan:

The colspan attribute is used to specify how many columns a cell should span horizontally. It allows a cell to occupy multiple columns in a table. For example:

```html

<table>

<tr>

<td>Cell 1</td>

<td colspan="2">Cell 2</td>

<td>Cell 3</td>

</tr>

<tr>

<td>Cell 4</td>

<td>Cell 5</td>

<td>Cell 6</td>

<td>Cell 7</td>

</tr>

</table>

```

In this example, the second cell in the first row has a colspan of 2, which means it spans across two columns. As a result, it occupies the space of the second and third cells in the first row. The table will be rendered with the second cell spanning two columns.

Rowspan and colspan are useful when you want to merge cells in a table to create more complex layouts or to group related data.

**a) Write a HTML code to generate following output**

**Maharashtra**

**o Pune**

**I. Dighi**

**II. Moshi**

**III. Shivajinagar**

**o Mumbai**

**I. Santakruiz**

**II. Vikroli**

**III. Mumbra**

<html>

<body>

<h2>Maharashtra</h2>

<li type="circle">Pune</li>

<li type="i">Dighi</li>

<li type="i">Moshi</li>

<li type="i">Shivajinagar</li>

<br>

<li type="circle">Mumbai</li>

<li type="i">Santakruiz</li>

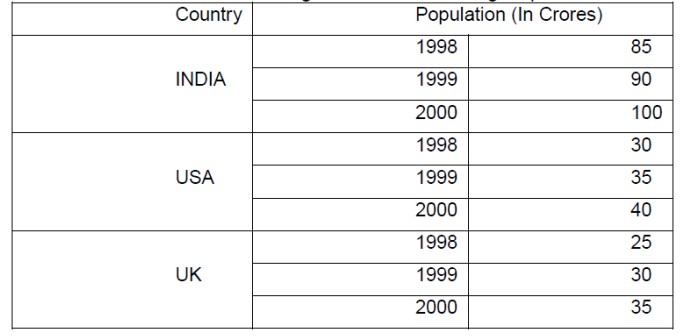
<li type="i">Vikroli</li>

<li type="i">Mumbra</li>

</body>

</html>

**Write HTML code to generate the following output**

****

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Population by Country</title>**

**</head>**

**<body>**

**<h2>Population by Country</h2>**

**<table border="1" style="border-collapse:collapse">**

**<tr>**

**<th>Country</th>**

**<th colspan="2">Population(In crores)</th>**

**</tr>**

**<tr>**

**<td rowspan="3">INDIA</td>**

**<td>1998</td>**

**<td>85</td>**

**</tr>**

**<tr>**

**<td>1999</td>**

**<td>90</td>**

**</tr>**

**<tr>**

**<td>2000</td>**

**<td>100</td>**

**</tr> <tr>**

**<td rowspan="3">USA</td>**

**<td>1998</td>**

**<td>30</td>**

**</tr>**

**<tr>**

**<td>1999</td>**

**<td>35</td>**

**</tr>**

**<tr>**

**<td>2000</td>**

**<td>40</td>**

**</tr> <tr>**

**<td rowspan="3">UK</td>**

**<td>1998</td>**

**<td>25</td>**

**</tr>**

**<tr>**

**<td>1999</td>**

**<td>30</td>**

**</tr>**

**<tr>**

**<td>2000</td>**

**<td>35</td>**

**</tr>**

**</table>**

**</body>**

**</html>**

**Explain the structure of the HTML webpage with an example.**

The structure of an HTML webpage typically includes several essential elements that define the document's layout, content, and other meta-information. Here's a basic example of the structure of an HTML webpage:

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>My First Webpage</title>

<!-- Additional meta tags, stylesheets, or scripts can be included here -->

</head>

<body>

<header>

<h1>Welcome to My Website</h1>

<!-- Navigation menu or other header content goes here -->

</header>

<main>

<section>

<h2>About Me</h2>

<p>This is a brief introduction about myself.</p>

</section>

<section>

<h2>My Interests</h2>

<ul>

<li>Web Development</li>

<li>Programming</li>

<li>Reading</li>

</ul>

</section>

</main>

<footer>

<p>&copy; 2023 My First Webpage. All rights reserved.</p>

<!-- Additional footer content goes here -->

</footer>

</body>

</html>

```

Let's break down the structure:

1. `<!DOCTYPE html>`: Declares the HTML version being used (HTML5 in this case).

2. `<html lang="en">`: The root element that wraps the entire HTML content. The `lang` attribute specifies the language of the document.

3. `<head>`: Contains meta-information about the HTML document, such as character set, viewport settings, and the document's title.

4. `<meta charset="UTF-8">`: Specifies the character encoding for the document.

5. `<meta name="viewport" content="width=device-width, initial-scale=1.0">`: Sets the viewport configuration for responsive design.

6. `<title>`: Defines the title of the HTML document, which appears in the browser tab.

7. `<body>`: Contains the main content of the HTML document.

8. `<header>`: Typically contains the heading or introductory content of the webpage, as well as navigation elements.

9. `<main>`: Contains the main content of the webpage, often organized into sections.

10. `<section>`: Represents a thematic grouping of content, such as different sections of a webpage.

11. `<h1>`, `<h2>`: Heading elements to define the hierarchy of the content.

12. `<p>`: Paragraph element for text content.

13. `<ul>`, `<li>`: Unordered list and list item elements for creating lists.

14. `<footer>`: Contains footer content, such as copyright information or links to related pages.

**Write a Javascript code which checks the contents entered in a forms text element. If the text entered is in the lower cases convert to upper case.**

<!DOCTYPE html>

<html>

<head>

<script>

function convertToUpperCase() {

var inputText = document.getElementById("myInput").value;

var convertedText = inputText.toUpperCase();

document.getElementById("myInput").value = convertedText;

}

</script>

</head>

<body>

<form>

<label for="myInput">Enter Text:</label>

<input type="text" id="myInput" name="myInput">

<button type="button" onclick="convertToUpperCase()">Convert to Uppercase</button>

</form>

</body>

</html>

**Explain various datatypes used in Javascript.**

JavaScript has several built-in data types that are used to store different kinds of values. Here are the main data types in JavaScript:

1. Number: Represents numeric values, including integers and floating-point numbers. For example: `let age = 25;` or `let price = 9.99;`

2. String: Represents a sequence of characters enclosed in single quotes ('') or double quotes (""). For example: `let name = "John";` or `let message = 'Hello, world!';`

3. Boolean: Represents a logical value, either `true` or `false`. It is often used in conditional statements and comparisons. For example: `let isLogged = true;` or `let isFound = false;`

4. Null: Represents the intentional absence of any object value. It is a special value that indicates the absence of a value. For example: `let myVariable = null;`

5. Undefined: Represents a variable that has been declared but has not been assigned a value. It is the default value of variables that have not been initialized. For example: `let myVariable;` or `let myVariable = undefined;`

6. Object: Represents a collection of key-value pairs, where each key is a string (or symbol) and each value can be of any data type. Objects are used to store and organize complex data structures. For example:

```javascript

let person = {

name: "John",

age: 25,

isStudent: true

};

```

7. Array: Represents an ordered list of values, enclosed in square brackets ([]). Arrays can store multiple values of any data type, and each value is assigned an index starting from 0. For example:

```javascript

let fruits = ["apple", "banana", "orange"];

```

8. Symbol: Represents a unique identifier. Symbols are often used as keys in objects to avoid naming conflicts. They are created using the `Symbol()` function. For example:

```javascript

let id = Symbol("uniqueId");

```

These are the main data types in JavaScript. Understanding and working with these data types is fundamental to programming in JavaScript.

**Write a Javascript to find factorial of a number.**

<!DOCTYPE html>

<html>

<head>

<script>

function calculateFactorial() {

var number = parseInt(document.getElementById("numberInput").value);

var factorial = 1;

for (var i = 1; i <= number; i++) {

factorial \*= i;

}

document.getElementById("result").innerHTML = "The factorial of " + number + " is: " + factorial;

}

</script>

</head>

<body>

<label for="numberInput">Enter a number:</label>

<input type="number" id="numberInput">

<button onclick="calculateFactorial()">Calculate Factorial</button>

<p id="result"></p>

</body>

</html>

**Explain pattern matching. What is the different meta characters used in pattern matching ?**

Pattern matching, also known as regular expression matching or regex matching, is a technique used to search for specific patterns within strings of text. It involves defining a pattern using a combination of literal characters and special meta characters, and then searching for matches of that pattern within a given text.

Meta characters are special characters that have a specific meaning in regular expressions. They allow you to define more complex patterns and perform advanced matching operations. Here are some commonly used meta characters in pattern matching:

1. `.` (dot): Matches any single character except a newline character.

2. `\*` (asterisk): Matches zero or more occurrences of the preceding character or group.

3. `+` (plus): Matches one or more occurrences of the preceding character or group.

4. `?` (question mark): Matches zero or one occurrence of the preceding character or group.

5. `|` (pipe): Acts as an OR operator, allowing you to match either the pattern before or after the pipe.

6. `[]` (square brackets): Defines a character class, allowing you to match any single character within the brackets.

7. `()` (parentheses): Groups multiple characters or subpatterns together, allowing you to apply quantifiers or other operations to them.

8. `\` (backslash): Escapes special characters, allowing you to match them as literal characters. For example, `\.` matches a literal dot character.

These are just a few examples of meta characters used in pattern matching. Regular expressions provide a rich set of meta characters and syntax that allow for powerful pattern matching operations, including matching specific characters, ranges of characters, word boundaries, and more.

By using meta characters and combining them with literal characters, you can create complex patterns to match and manipulate text in a flexible and efficient manner.

**Create a form for student information. Write Java Script code to find total, average, result and grade.**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<script>**

**function calculateResult() {**

**var marks1 = parseInt(document.getElementById("marks1").value);**

**var marks2 = parseInt(document.getElementById("marks2").value);**

**var marks3 = parseInt(document.getElementById("marks3").value);**

**var total = marks1 + marks2 + marks3;**

**var average = total / 3;**

**document.getElementById("total").innerHTML = "Total Marks: " + total;**

**document.getElementById("average").innerHTML = "Average Marks: " + average.toFixed(2);**

**if (average >= 50) {**

**document.getElementById("result").innerHTML = "Result: Pass";**

**document.getElementById("grade").innerHTML = "Grade: A";**

**} else {**

**document.getElementById("result").innerHTML = "Result: Fail";**

**document.getElementById("grade").innerHTML = "Grade: B";**

**}**

**}**

**</script>**

**</head>**

**<body>**

**<h2>Marks Obtained by Student</h2>**

**<form>**

**<label for="marks1">Marks of 1<sup>st</sup> subject:</label>**

**<input type="number" id="marks1" name="marks1"><br>**

**<label for="marks2">Marks of 2<sup>nd</sup> subject:</label>**

**<input type="number" id="marks2" name="marks2"><br>**

**<label for="marks3">Marks of 3<sup>rd</sup> subject:</label>**

**<input type="number" id="marks3" name="marks3"><br>**

**<button type="button" onclick="calculateResult()">Calculate</button>**

**</form>**

**<p id="total"></p>**

**<p id="average"></p>**

**<p id="result"></p>**

**<p id="grade"></p>**

**</body>**

**</html>**

**What is an array object ? How to create an array object ? Explain any six array methods.**

An array object in JavaScript is a built-in object that allows you to store multiple values in a single variable. It is a container that holds a fixed number of elements, which can be of any data type. Arrays are commonly used to store and manipulate collections of related data.

To create an array object in JavaScript, you can use the array literal syntax, which is represented by square brackets ([]). Here are a few examples:

```javascript

// Empty array

var myArray = [];

// Array with initial values

var numbers = [1, 2, 3, 4, 5];

var fruits = ["apple", "banana", "orange"];

```

In the above examples, we create an empty array `myArray` and two arrays `numbers` and `fruits` with initial values.

Now, let's discuss six commonly used array methods in JavaScript:

1. `push()`: Adds one or more elements to the end of an array and returns the new length of the array.

```javascript

var fruits = ["apple", "banana"];

fruits.push("orange");

// fruits is now ["apple", "banana", "orange"]

```

2. `pop()`: Removes the last element from an array and returns that element.

```javascript

var fruits = ["apple", "banana", "orange"];

var removedFruit = fruits.pop();

// removedFruit is "orange", fruits is now ["apple", "banana"]

```

3. `concat()`: Combines two or more arrays and returns a new array.

```javascript

var fruits = ["apple", "banana"];

var moreFruits = ["orange", "mango"];

var allFruits = fruits.concat(moreFruits);

// allFruits is ["apple", "banana", "orange", "mango"]

```

4. `indexOf()`: Returns the first index at which a specified element is found in an array, or -1 if not found.

```javascript

var fruits = ["apple", "banana", "orange"];

var index = fruits.indexOf("banana");

// index is 1

```

5. `slice()`: Returns a shallow copy of a portion of an array into a new array.

```javascript

var fruits = ["apple", "banana", "orange", "mango"];

var slicedFruits = fruits.slice(1, 3);

// slicedFruits is ["banana", "orange"]

```

6. `forEach()`: Executes a provided function once for each array element.

```javascript

var numbers = [1, 2, 3, 4, 5];

numbers.forEach(function(number) {

console.log(number);

});

// Output: 1 2 3 4 5

```

These are just a few examples of array methods in JavaScript. Arrays provide a wide range of methods to manipulate and work with the elements they contain, making them a powerful tool for handling collections of data.

**Explain GET and POST request methods.**

GET and POST are two commonly used HTTP request methods used to send data from a client (such as a web browser) to a server.

1. GET Method:

- GET is the default method used by web browsers when you enter a URL or click on a link.

- It is used to retrieve data from a server.

- The data is appended to the URL as query parameters.

- GET requests are visible in the browser's address bar and can be bookmarked and cached.

- GET requests are limited in the amount of data they can send (typically around 2048 characters).

- GET requests are considered "safe" and "idempotent" as they should not have any side effects on the server.

2. POST Method:

- POST is used to send data to the server to create or update a resource.

- The data is sent in the body of the request, not visible in the URL.

- POST requests are not cached and cannot be bookmarked.

- POST requests can send larger amounts of data compared to GET requests.

- POST requests are not considered "safe" or "idempotent" as they can have side effects on the server, such as creating a new resource or updating an existing one.

In summary, GET is used to retrieve data from a server, while POST is used to send data to a server. GET requests are visible in the URL and have limitations on data size, while POST requests are not visible in the URL and can send larger amounts of data.

**Design A JavaScript to display whether given number is prime or not.**

<html>

<head>

<script>

function checkPrime() {

var number = parseInt(document.getElementById("numberInput").value);

var isPrime = true;

if (number === 1) {

isPrime = false;

} else if (number > 1) {

for (var i = 2; i <= Math.sqrt(number); i++) {

if (number % i === 0) {

isPrime = false;

break;

}

}

}

if (isPrime) {

document.getElementById("result").innerHTML = number + " is a prime number.";

} else {

document.getElementById("result").innerHTML = number + " is not a prime number.";

}

}

</script>

</head>

<body>

<label for="numberInput">Enter a number:</label>

<input type="number" id="numberInput">

<button onclick="checkPrime()">Check Prime</button>

<p id="result"></p>

</body>

</html>

**. Explain briefly about Built in Java script Objects?**

Built-in JavaScript objects are pre-defined objects that provide useful functionality and methods to perform various operations in JavaScript. These objects are available globally and can be accessed and used without the need for explicit instantiation.

Here are some commonly used built-in JavaScript objects:

1. Math: Provides mathematical operations and constants.

- Example methods: `Math.random()`, `Math.floor()`, `Math.max()`, `Math.min()`

2. Date: Represents dates and times.

- Example methods: `Date.now()`, `Date.getFullYear()`, `Date.getMonth()`, `Date.getDate()`

3. String: Represents a sequence of characters.

- Example methods: `String.length`, `String.charAt()`, `String.indexOf()`, `String.toUpperCase()`

4. Array: Represents an ordered collection of elements.

- Example methods: `Array.length`, `Array.push()`, `Array.pop()`, `Array.join()`

5. Object: Represents a collection of key-value pairs.

- Example methods: `Object.keys()`, `Object.values()`, `Object.assign()`

6. JSON: Provides methods for working with JSON (JavaScript Object Notation) data.

- Example methods: `JSON.parse()`, `JSON.stringify()`

7. RegExp: Represents regular expressions for pattern matching and manipulation.

- Example methods: `RegExp.test()`, `RegExp.exec()`, `RegExp.match()`

These built-in objects offer a wide range of functionality and methods to perform common tasks in JavaScript. By utilizing these objects, you can enhance your code and leverage their capabilities to simplify and optimize your JavaScript programs.

Unit III

**Explain the CSS selectors .**

Answer on this link https://www.tutorialspoint.com/jquery/jquery-selectors.htm

1 Element Selectors

Element selectors select all instances of a given HTML element. You can select all

elements by using the universal element selector, which is the \* (asterisk) character.

You can select a group of elements by separating the different element names with

commas. This is a sensible way to reduce the size and complexity of your CSS files,

by combining multiple identical rules into a single rule.

2 Class Selectors

A class selector allows you to simultaneously target different HTML elements

regardless of their position in the document tree. If a series of HTML elements have

been labeled with the same class attribute value, then you can target them for styling

by using a class selector, which takes the form: period (.) followed by the class name.

3 Id Selectors

An id selector allows you to target a specific element by its id attribute regardless of

its type or position. If an HTML element has been labeled with an id attribute, then

you can target it for styling by using an id selector, which takes the form: pound/hash

(#) followed by the id name.

4 Attribute Selectors

An attribute selector provides a way to select HTML elements either by the

presence of an element attribute or by the value of an attribute. This can be a very

powerful technique, but because of uneven support by some of the browsers, not all

web authors have used them. Attribute selectors can be a very helpful technique in

the styling of hyperlinks and images. For instance, perhaps we want to make it more

obvious to the user when a pop-up tooltip is available for a link or image. We can do

this by using the following attribute selector: [title] { … } This will match any

element in the document that has a title attribute.

5 Pseudo-Element and Pseudo-Class Selectors

A pseudo-element selector is a way to select something that does not exist explicitly

as an element in the HTML document tree but which is still a recognizable selectable

object. For instance, you can select the first line or first letter of any HTML element

using a pseudo-element selector. A pseudo-class selector does apply to an

HTML element, but targets either a particular state or, in CSS3, a variety of family

relationships. The most common use of this type of selectors is for targeting link

states. By default, the browser displays link text blue and visited text links purple.

Do be aware that this state does not occur on touch screen devices. Note the syntax of

pseudo-class selectors: the colon (:) followed by the pseudo-class selector name. Do

be aware that a space is not allowed after the colon. Believe it or not, the order of

these pseudo-class elements is important. The :link and :visited pseudo-classes should

appear before the others. Some developers use a mnemonic to help them remember

the order. My favorite is “Lord Vader, Former Handle Anakin” for Link, Visited,

Focus, Hover, Active.

6 Contextual Selectors

A contextual selector (in CSS3 also called combinators) allows you to select

elements based on their ancestors, descendants, or siblings. That is, it selects

elements based on their context or their relation to other elements in the document

tree. While some of these contextual selectors are used relatively infrequently, almost

all

web authors find themselves using descendant selectors.A descendant selector

matches all elements that are contained within another element. The character used to

indicate descendant selection is the space character.

Selector Matches Example

Descendant A specified element that is contained somewhere within another

specified element. div p Selects a <p> element that is contained somewhere within a

<div> element. That is, the <p> can be any descendant, not just a child. Child A

specified element that is a direct child of the specified element.

div>h2

Selects an <h2> element that is a child of a <div> element.

Adjacent sibling A specified element that is the next sibling (i.e., comes directly

after) of the specified element.

h3+p

Selects the first <p> after any <h3>. General sibling A specified element that shares

the same parent as the specified element.

h3~p

Selects all the <p> elements that share the same parent as the <h3>

What is JQuery? Explain JQuery Selectors?

What is a box model? And what are the different elements of a box model?

Difference between JavaScript and JQuery

Answer to the above question

Both of these are useful in programming, but there is a primary difference between JavaScript and JQuery. JavaScript, on the one hand, is a scripting language used in programming. It is dynamic and helps in web development. But it is more complex since the programmer has to write the complete JavaScript code in a program. JQuery, on the other hand, is a JavaScript library. It is comparatively less complex- since a programmer only has to write the important JQuery code.

In this article, we will dive deeper into the difference between JavaScript and JQuery. Read ahead to know more.

**What is JavaScript?**

It is a scripting language used by programmers to make their websites more interactive and responsive. Along with CSS and HTML, JavaScript (commonly abbreviated as JS) is one of the most important resources used for creating web pages. CSS and HTML are used to design and decorate web pages. But JS helps us create a more dynamic web page. We can say that it breathes life into the pages.

JS is a major client-side language, but it isn’t confined to web development alone. We can also use JS in various server and desktop programs (the most common example here is Node.JS). A few databases also make use of JS, such as CouchDB and MongoDB.

Whenever any browser parses the web pages, then its primary responsibility is the creation of a tree-structure presentation in the memory.

**What is JQuery?**

It is a JS framework that is developed from JavaScript. JQuery is a very popular JS library. It is free, fast, rich-featured, concise, open-source, and has cross-browser compatibility. John Resign invented it, and it was released at BarCamp NYC back in January 2006.

The reason why JQuery was developed was to make the programmers’ lives easier. It acts as a feature-rich library that provides better client-side services. This way, JavaScript can be used effectively for creating browser-based apps and websites.

Here are a few features that JQuery offers:

* Manipulation of DOM: We can easily traverse and modify the DOM elements.
* Built-in features for animations in an app or a website.
* Event handling in HTML and manipulation.
* CSS manipulation.
* It is only 19kb in size. Very lightweight to operate on and work with.
* It consists of a very high-level widget library for UI.

**Difference between JavaScript and JQuery**

Let us talk about the differences between JavaScript and JQuery.

|  |  |  |
| --- | --- | --- |
| **Parameters** | **JavaScript** | **JQuery** |
| Basics | JS is a programming language. It is a scripting language that is dynamic and helps in web development. | It is a JS library. |
| Complexity | It is more complex since the programmer has to write the complete JavaScript code in a program. | It is comparatively less complex. A programmer only has to write the important JQuery code. |
| Consumption of Time | Since one has to write the whole script, it consumes much more time. | It consumes comparatively much less time since it makes programming easy and fast. |
| Multi-Browser Compatibility | To handle multi-browser compatibility, a developer has to develop code of their own. | One doesn’t need to focus on their app’s issue with compatibility on multiple browsers. This feature comes as a prerequisite. |
| Browser Support | Every browser supports JavaScript. We don’t need to include any additional plugins to make JS work. | We have to include the JQuery library’s URL in the page’s header so that JQuery works. |
| Inter-Dependency | JQuery forms a part of JS. The JS code does not necessarily depend on JQuery. | JQuery always depends on JS- since it is a JavaScript library. |
| Lines of Code | The code can be long and complicated. It takes time to program using JavaScript. | It consists of very few lines of code; it is easy to operate and work with. |
| Uses | It is a very crucial programming language used for web designing and programming desktop programs and servers. | It optimises the working of the JavaScript language. We can make all apps and websites more interactive, fast, and efficient while still decreasing the complexity of development. |
| Type of Approach | It is a very weakly typed approach used in programming. | It is a fast, simple, and easy approach used in programming. |
| DOM | It is very slow for the creation of DOM. | It is capable of creating DOM much faster. |

Define AJAX. Explain working of Ajax

Answer to the above question in the following link

https://www.tutorialspoint.com/ajax/what\_is\_ajax.htm

Explain AJAX request

Discuss the Jquery Dom manipulation methods

Anwer on this link <https://www.tutorialspoint.com/jquery/jquery-dom.htm>

Explain Jquery Dom traversing

Answer to the above question in the link given below

https://www.tutorialspoint.com/jquery/jquery-traversing.htm

Unit 5

List and Explain different super global arrays