# Unit-3

Ajax: Introduction, Advantages and Disadvantages, Purpose of Ajax, Ajax-based Web Application, Alternatives of Ajax JavaScript & Ajax: Introduction to array-operators, making statement, date & time, mathematics strings, Event Handling, Form Properties, AJAX-Introduction to jQuery and AngularJS.

by Pratishtha Gupta

# AJAX

AJAX = Asynchronous JavaScript And XML.

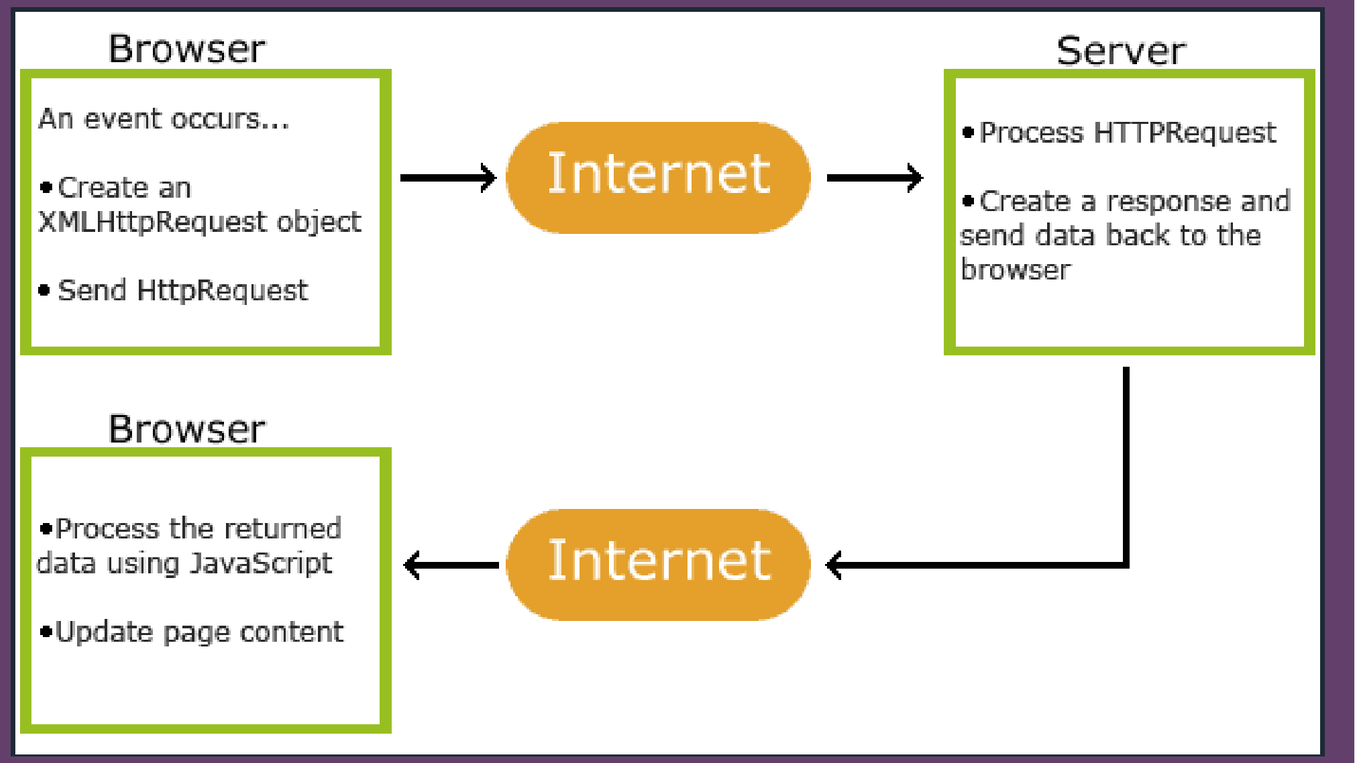
AJAX is not a programming language.

AJAX just uses a combination of:

- A browser built-in XMLHttpRequest object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data)
- AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

**Purpose of AJAX**

- Update a web page without reloading the page
- Request data from a server - after the page has loaded
- Receive data from a server - after the page has loaded
- Send data to a server - in the background




1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript

**Advantages of AJAX include:**

- **Improved user experience:** AJAX can improve user interaction by integrating content seamlessly.
- **Faster response times:** AJAX can reduce server traffic and increase speed.
- **Reduced bandwidth usage:** AJAX can reduce bandwidth usage, which can improve web performance and load speed.
- **Form validation:** AJAX can enable precise and immediate form validation.
- **Partial page updates:** AJAX can update parts of a web page without reloading the whole page.

**Disadvantages of AJAX include:**

- Asynchronous JavaScript and XML (AJAX) is a web development technique that can improve user experience, performance, and functionality. However, it also has some disadvantages, including:
- **Browser compatibility:** Not all browsers support JavaScript and XMLHttpRequest objects.
- **Security:** Security and privacy issues must be considered when developing AJAX applications.

- **Accessibility:** Not all browsers support JavaScript or XMLHttpRequest objects, so web applications must be accessible to all users.
- **Bookmarks and navigation:** Bookmarks and browser history may not behave correctly.
- **Search engine:** AJAX applications cannot be searched.
- **Implementation complexity:** AJAX can be more complex to implement.
- **Graceful degradation:** Users with JavaScript disabled may need graceful degradation.

**AJAX-based Web Applications**

- **Google Maps** – It is a great example of an AJAX application. It uses AJAX to dynamically update the maps and show only the requested data without reloading the whole page.
- **Facebook –** It is also a good example of an AJAX application. It uses AJAX to update the feeds, notifications, news, and other features. Ajax is also used for update the Facebook content of the web page according to the action of the user.
- **Gmail –** Gmail also uses AJAX to provide a seamless and interactive environment to the user. With the help of AJAX Gmail can update the inbox, delete emails, or mark emails as read without reloading the page.

- **Twitter –** Using AJAX provide a real-time environment to the user. Whenever a new tweet is posted it will add to the timeline without refreshing the whole page. The same goes for the notification.
- **Google –** uses AJAX for its auto-complete feature which provides real-time suggestions in the drop-down list without reloading the original web page.

**Alternatives to AJAX:**

- **Fetch API:** A modern JavaScript interface that simplifies network requests and is native to the browser.
- **Axios:** A popular JavaScript library for making HTTP requests with a modern, promise-based API.
- **GraphQL:** A query language for APIs that allows clients to request only the data they need.
- **Server-Sent Events (SSE):** An efficient way to push updates from the server to the client.
- **RESTful APIs:** A traditional approach that is still a viable option for many applications.
- **gRPC**: A high-performance RPC framework.
- **WebRTC:** A peer-to-peer communication method.

# Javascript Arrays

- An array is a special variable, which can hold more than one value:
- Eg: const cars = ["Saab", "Volvo", "BMW"];

## Creating an Array

- Using an array literal is the easiest way to create a JavaScript Array.
- Syntax: const array_name = [item1, item2, ...];
- You can also create an array, and then provide the
- elements:

**Note:** Array indexes start with 0.

## Using the JavaScript Keyword new

- The following example also creates an Array, and assigns values to it:
- Eg: const cars = new Array("Saab", "Volvo", "BMW");

## Accessing Array Elements

- You access an array element by referring to the index number:
- const cars = ["Saab", "Volvo", "BMW"];
- let car = cars[0];

```
const cars = [];
cars[0]= "Saab";
cars[1]= "Volvo";
cars[2]= "BMW";
```

## Changing an Array Element

- This statement changes the value of the first element in cars:
- cars[0] = "Opel";

**Access the Full Array**

- With JavaScript, the full array can be accessed by referring to the array name:
- Eg: const cars = ["Saab", "Volvo", "BMW"];
- document.getElementById("demo").innerHTML = cars;

**Array Properties and Methods**

- The real strength of JavaScript arrays are the built-in array properties and methods:
- cars.length  // Returns the number of elements
- cars.sort()  // Sorts the array

**The length Property**

- The length property of an array returns the length of an array (the number of array elements).
- Eg: const fruits = ["Banana", "Orange", "Apple", "Mango"];
  let length = fruits.length;

**Accessing First and Last Element of Array**

- const fruits = ["Banana", "Orange", "Apple", "Mango"];
- let fruit = fruits[0];
- let fruit = fruits[fruits.length - 1];

The `length` property is always one more than the highest array index.

## Looping Array Elements

- One way to loop through an array, is using a for loop:
- You can also use the Array.forEach() function:

Example

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
let fLen = fruits.length;

let text = "<ul>";
for (let i = 0; i < fLen; i++) {
  text += "<li>" + fruits[i] + "</li>";
}
text += "</ul>";
```

Example

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];

let text = "<ul>";
fruits.forEach(myFunction);
text += "</ul>";

function myFunction(value) {
  text += "<li>" + value + "</li>";
}
```

## Adding Array Elements

- The easiest way to add a new element to an array is using the push() method:
- Eg: const fruits = ["Banana", "Orange", "Apple"];
- fruits.push("Lemon");  // Adds a new element (Lemon) to fruits
- New element can also be added to an array using the length property:
- Eg: const fruits = ["Banana", "Orange", "Apple"];
- fruits[fruits.length] = "Lemon";  // Adds "Lemon" to fruits

## The Difference Between Arrays and Objects

- In JavaScript, arrays use numbered indexes.
- In JavaScript, objects use named indexes.
- Arrays are a special kind of objects, with numbered indexes.
- You should use objects when you want the element names to be strings (text).
- You should use arrays when you want the element names to be numbers.

**WARNING !**

Adding elements with high indexes can create undefined "holes" in an array:

```
const fruits = ["Banana", "Orange", "Apple"];
fruits[6] = "Lemon";  // Creates undefined "holes" in fruits
```

**JavaScript new Array()**

- JavaScript also has a built-in array constructor new Array().
- These two different statements both create a new empty array named points:
- const points = new Array();
- const points = [];
- These two different statements both create a new array containing 6 numbers:
- const points = new Array(40, 100, 1, 5, 25, 10);
- const points = [40, 100, 1, 5, 25, 10];
- The new keyword can produce some unexpected results: Therefore, we should avoid using new keyword

**Typeof Operator**

- The **typeof** operator returns object because a JavaScript array is an object.
- To solve this problem ECMAScript 5 (JavaScript 2009) defined a new method

**Array.isArray():**

- Array.isArray() returns ans in true or false. Eg: Array.isArray(fruits);

**instanceof operator**

- returns true if an object is created by a given constructor:
- const fruits = ["Banana", "Orange", "Apple"];
- (fruits instanceof Array);

# JavaScript Date and Time

- By default, JavaScript will use the browser's time zone and display a date as a full text string:
- Sat Oct 19 2024 16:23:07 GMT+0530 (India Standard Time)

## Creating Date Objects

- Date objects are created with the new Date() constructor. Eg: const d = new Date();
- There are 9 ways to create a new date object:
- new Date(date string) creates a date object from a date string:
- new Date(year, month, ...) creates a date object with a specified date and time.
- 7 numbers specify year, month, day, hour, minute, second, and millisecond (in that order):
- Eg: const d = new Date(2018, 11, 24, 10, 33, 30, 0);

```
new Date()
new Date(date string)

new Date(year,month)
new Date(year,month,day)
new Date(year,month,day,hours)
new Date(year,month,day,hours,minutes)
new Date(year,month,day,hours,minutes,seconds)
new Date(year,month,day,hours,minutes,seconds,ms)

new Date(milliseconds)
```

JavaScript counts months from 0 to 11: January = 0, December = 11.

Specifying a month higher than 11, will not result in an error but add the overflow to the next year: const d = new Date(2018, 15, 24, 10, 33, 30);

- 15-11=4 so, the month will be april and year will become 2019

Specifying a day higher than max, will not result in an error but add the overflow to the next month:

- const d = new Date(2018, 5, 35, 10, 33, 30)& const d = new Date(2018, 6, 5, 10, 33, 30); are same

**Using 6, 5, 4, 3, or 2 Numbers**

- 6 numbers specify year, month, day, hour, minute, second:
- 5 numbers specify year, month, day, hour, and minute and so on...

**Date Methods**

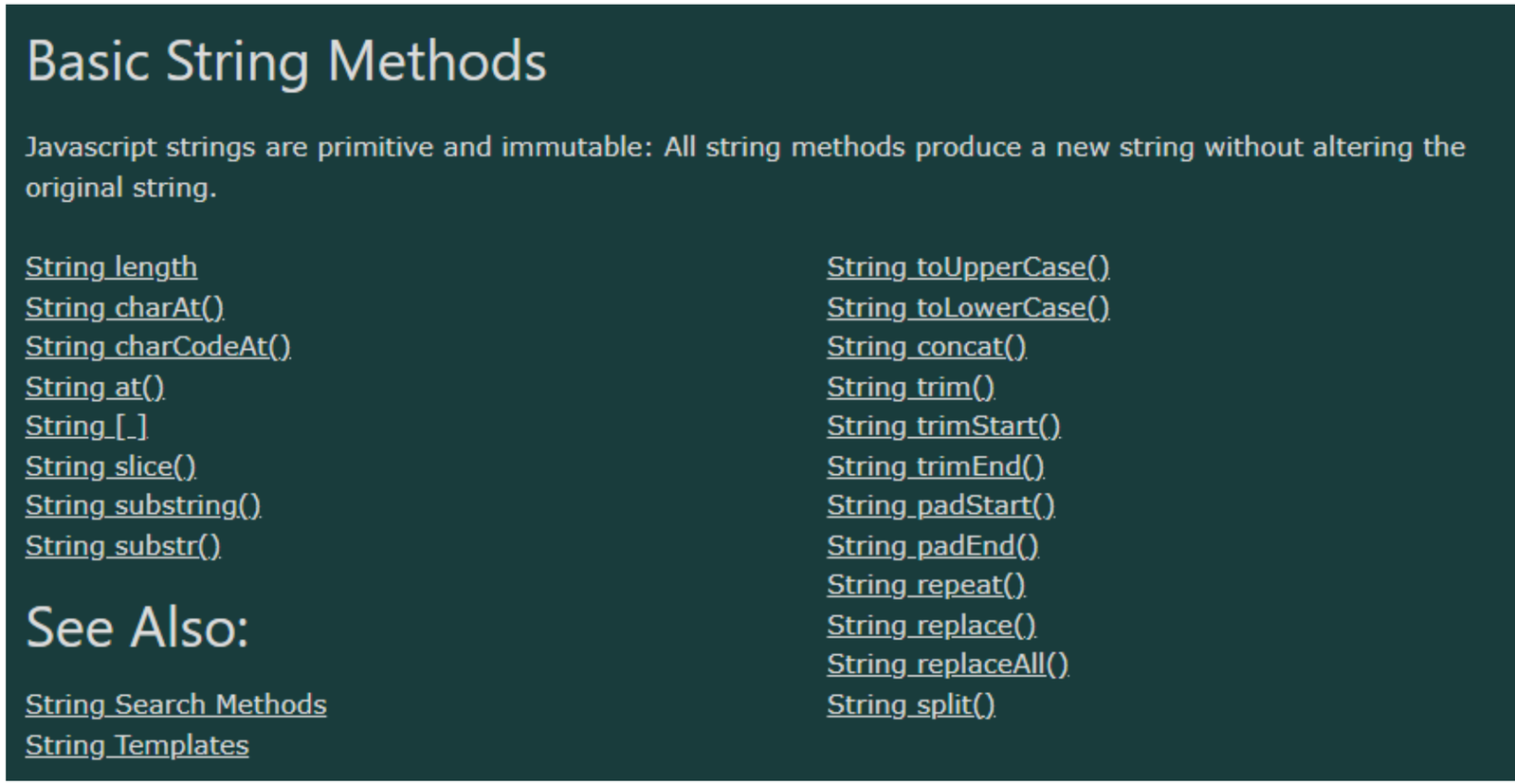
- When a date object is created, a number of methods allow you to operate on it.
- Date methods allow you to get and set the year, month, day, hour, minute, second, and millisecond of date objects, using either local time or UTC.

**Displaying Dates**

- JavaScript will (by default) output dates using the toString() method. This is a string representation of the date, including the time zone. The format is specified in the ECMAScript specification:
- Eg: Sat Oct 19 2024 16:23:07 GMT+0530 (India Standard Time)
- When you display a date object in HTML, it is automatically converted to a string, with the toString() method.
- Eg: const d = new Date();
  d.toString();
- The toDateString() method converts a date to a more readable format:
- Eg: const d = new Date();
  d.toDateString();
- The toUTCString() method converts a date to a string using the UTC standard:
- Eg: const d = new Date();
  d.toUTCString();

# Javascript Strings

- Strings are for storing text
- Strings are written with quotes
- The length property returns the length of a string:
- Eg: let text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
- document.getElementById("demo").innerHTML = text.length;
- Output: 26



**Extracting String Characters- There are various ways of doing so**

1. **JavaScript String charAt()**

- The charAt() method returns the character at a specified index (position) in a string:
- Eg: var text = "HELLO WORLD";
- document.getElementById("demo").innerHTML = text.charAt(0);// Output: H

2. **JavaScript String At()**

- Eg: const name = "W3Schools";
- let letter = name.at(2);// Output: S

**Extracting String Parts: There are 3 methods for extracting a part of a string:**

1. **JavaScript String slice()**

- slice() extracts a part of a string and returns the extracted part in a new string.
- The method takes 2 parameters: start position, and end position (end not included).
- Eg: let text = "Apple, Banana, Kiwi";
- let part = text.slice(7,13);
- document.getElementById("demo").innerHTML = part; // Output: Banana

- Eg: let text = "Apple, Banana, Kiwi";
- let part = text.slice(-12);
- document.getElementById("demo").innerHTML = part;// Output:: Banana, Kiwi
- let text = "Apple, Banana, Kiwi";
- let part = text.slice(-12,-6) // Output: Banana

2. **JavaScript String substring()**
   - substring() is similar to slice().
   - The difference is that start and end values less than 0 are treated as 0 in substring().
   - Eg: let str = "Apple, Banana, Kiwi";
   - let part = str.substring(7, 13);  // Output: Banana
   - If you omit the second parameter, substring() will slice out the rest of the string.
3. **JavaScript String substr()**
   - substr() is similar to slice().
   - The difference is that the second parameter specifies the length of the extracted part.
   - Eg: let str = "Apple, Banana, Kiwi";
   - let part = str.substr(7, 6);  // Output: Banana

- If the first parameter is negative, the position counts from the end of the string.
- Example
- let str = "Apple, Banana, Kiwi";
- let part = str.substr(-4); // Output: Kiwi

**Converting to Upper and Lower Case**

- let text1 = "Hello World!";
- let text2 = text1.toUpperCase();// Output: HELLO WORLD!
- let text1 = "Hello World!";  // String
- let text2 = text1.toLowerCase();  // Output: Hello World!
- JavaScript String concat()

**concat() joins two or more strings:**

- Eg: let text1 = "Hello";
- let text2 = "World";
- let text3 = text1.concat(" ", text2);// // Output: Hello World!

**JavaScript String trim()**

- The trim() method removes whitespace from both sides of a string:
- Eg: let text1 = "     Hello World!     ";
- let text2 = text1.trim(); // Length text1 = 22, Length text2 = 12

- JavaScript String trimStart() and trimEnd()
- The trimStart() method works like trim(), but removes whitespace only from the start of a string.
- The trimEnd() method works like trim(), but removes whitespace only from the end of a string.
- Eg: let text1 = "  Hello World!  ";
- let text2 = text1.trimStart(); OR let text2 = text1.trimEnd();

**JavaScript String padStart() and padEnd()**

- The padStart() method pads a string from the start.
- It pads a string with another string (multiple times) until it reaches a given length.
- Eg: Pad a string with "0" until it reaches the length 4:
- let text = "5";
- let padded = text.padStart(4,"0"); // Output : 0005
- The padEnd() method pads a string from the end.
- It pads a string with another string (multiple times) until it reaches a given length.
- Eg: let text = "5";
- let padded = text.padEnd(4,"0");// // Output : 5000

**JavaScript String repeat()**

- The repeat() method returns a string with a number of copies of a string.
- The repeat() method returns a new string.
- The repeat() method does not change the original string.
- Eg: Create copies of a text:
- let text = "Hello world!";
- let result = text.repeat(2);// Output: Hello world!Hello world!

**Replacing String Content**

- The replace() method replaces a specified value with another value in a string:
- Eg: let text = "Please visit Microsoft!";
- let newText = text.replace("Microsoft", "W3Schools");// Please visit W3Schools
- By default, the replace() method replaces only the first match:
- By default, the replace() method is case sensitive. Writing MICROSOFT (with upper-case) will not work:
- To replace case insensitive, use a regular expression with an /i flag (insensitive):
- Eg: let newText = text.replace(/MICROSOFT/i, "W3Schools");
- To replace all matches, use a regular expression with a /g flag (global match):
- Eg: let newText = text.replace(/Microsoft/g, "W3Schools");

**JavaScript String ReplaceAll()**

- Eg: let text = "I love cats. Cats are very easy to love. Cats are very popular."
- text = text.replaceAll("Cats","Dogs");
- text = text.replaceAll("cats","dogs");// Output:  I love dogs. Dogs are very easy to love. Dogs are very popular.

**JavaScript String split()**

- A string can be converted to an array with the split() method:
- Example
- text.split(",")  // Split on commas
- text.split(" ")  // Split on spaces
- text.split("|")  // Split on pipe
- Eg: let text = "a,b,c,d,e,f";
- const myArray = text.split(",");
- document.getElementById("demo").innerHTML = myArray[0];// Output: a

**Introduction to jQuery**

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, and animation much simpler with an easy-to-use API.

The purpose of jQuery is to make it much easier to use JavaScript on your website. jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code. jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

## Introduction to AngularJS

AngularJS is a popular open-source framework that simplifies web development by creating interactive single-page applications (SPAs). Unlike traditional websites that load new pages for each click, SPAs offer a smoother user experience by updating content on the same page.

AngularJS makes this possible by transforming static HTML into dynamic content that adapts to user interactions. Features like data binding and dependency injection streamline development, saving you time and effort. With regular updates and a large community, AngularJS ensures your web applications stay modern and efficient.