Introduction to Javascript

Prepared by-
Vishesh Shrivastava 0801IT211097
Cheshta Arora 0801EC211014
Saumya Dixit 0801EC211067

Under the Guidance of-

Mr. Mukul Shukla Sir
Mr. Upendra Singh Sir

Introduction-

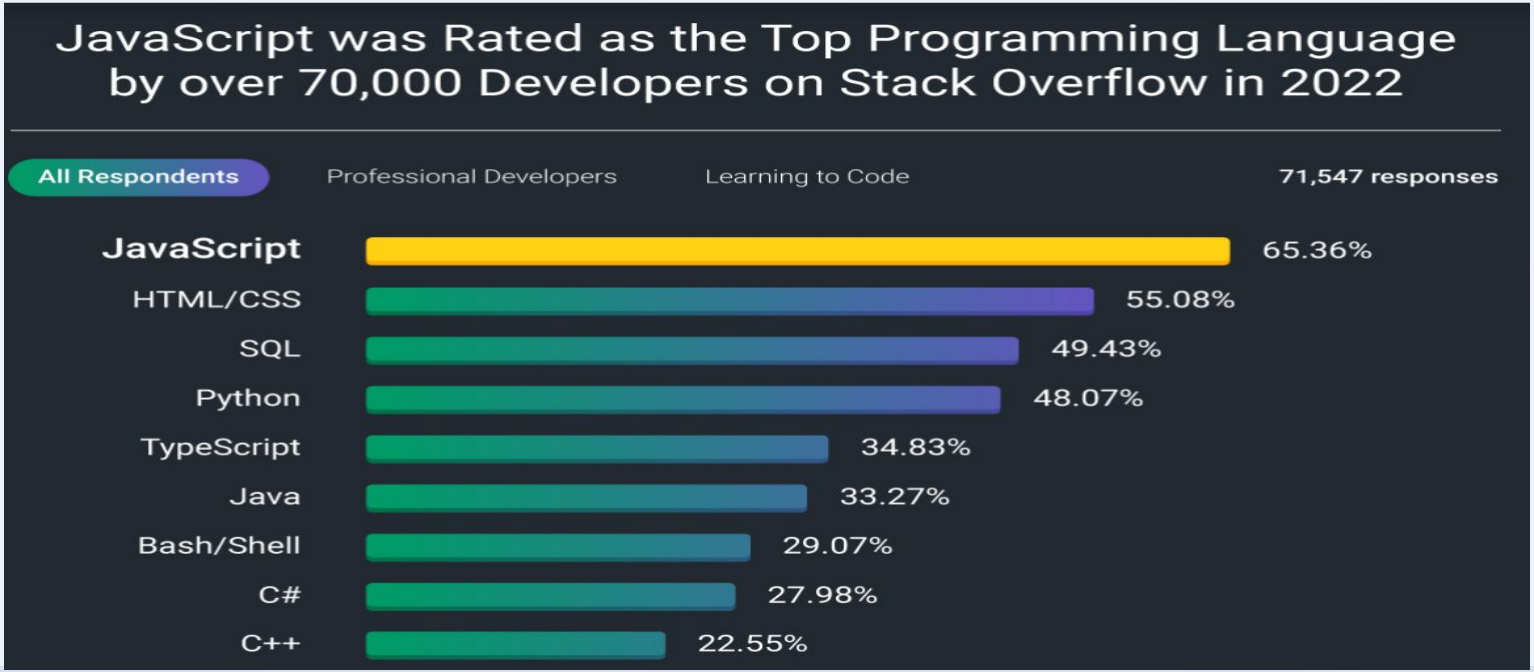
What is Javascript?

How does it work?

How/where to write JS? (console)/code editor

What is Javascript?

Js is a programming language , we use it to give instructions to the computer.

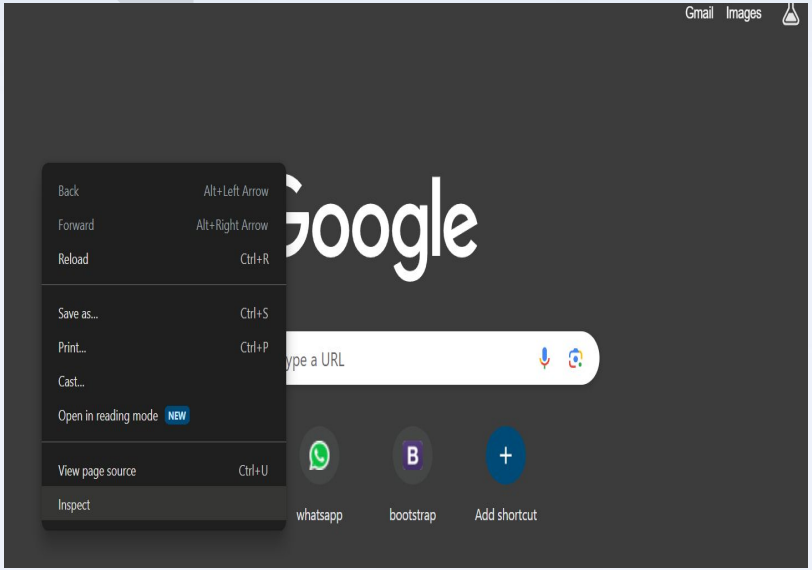


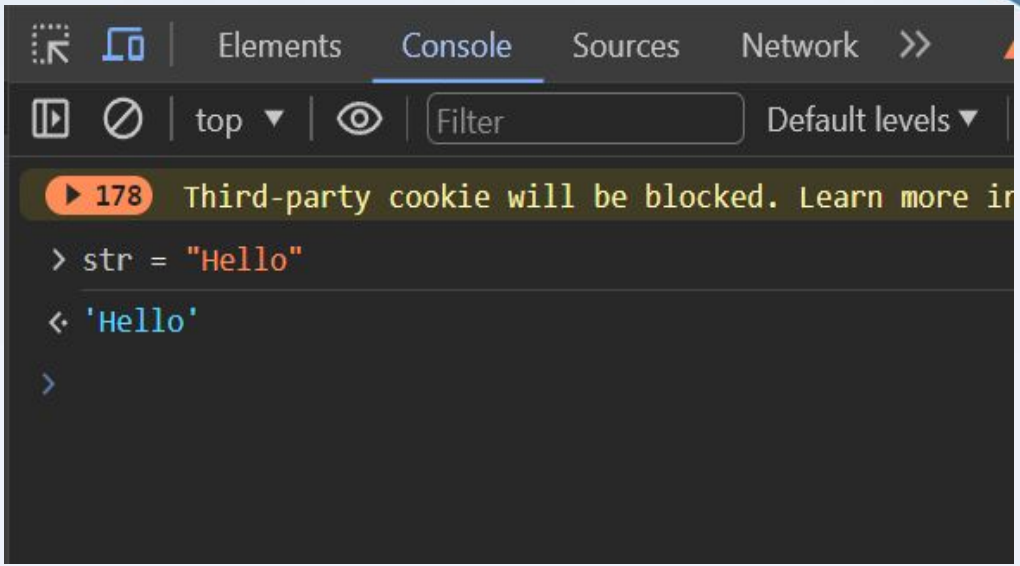
JavaScript is incredibly versatile and widely used across various domains, leading to the creation of numerous popular applications. Some of the most notable ones include:

- **Social media platforms** like Facebook, Twitter, and Instagram heavily utilize JavaScript for their **dynamic interfaces** and **real-time updates**.
- **Single-page applications (SPAs) l**ike Gmail, Google Maps, and GitHub leverage JavaScript frameworks (such as Angular, React, or Vue.js) to create seamless and **responsive user experiences** without the need for constant page reloads.
- Streaming giants like Netflix, Spotify, and YouTube utilize JavaScript to provide smooth, interactive media streaming experiences to all of their users.

What is console in web browser?

Writing js on console is a temporary way , on clicking on refresh , we will have to start again.





MAJORLY we write js on a code editor.
For eg- VS CODE </>
A free(no license) and popular code editor by microsoft.

How does it work?

Embedding in HTML: JavaScript code is typically embedded within HTML documents using <script> tags. You can include the script directly in the HTML file or link to an external JavaScript file.

```
<script>
document.getElementById("demo").innerHTML = "Hello JavaScript!";
</script>
```

html

```
<script src="yourScript.js"></script>
```

Variables in JS

Variables are containers in javascript for storing data.

Radius = 14;

JavaScript Variables can be declared in 4 ways:

- Automatically
- Using `var`
- Using `let`
- Using `const`

Automatically-

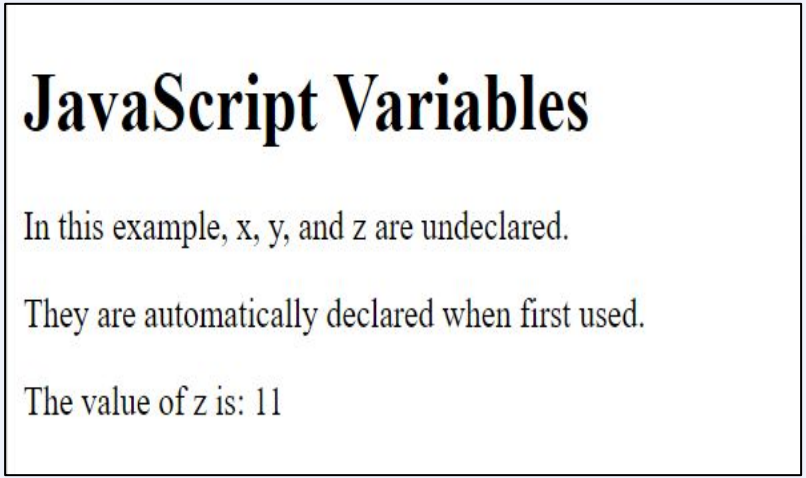
```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Variables</h1>

<p>In this example, x, y, and z are undeclared.</p>
<p>They are automatically declared when first used.</p>

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

<script>
x = 5;
y = 6;
z = x + y;
document.getElementById("demo").innerHTML =
"The value of z is: " + z;
</script>

</body>
</html>
```



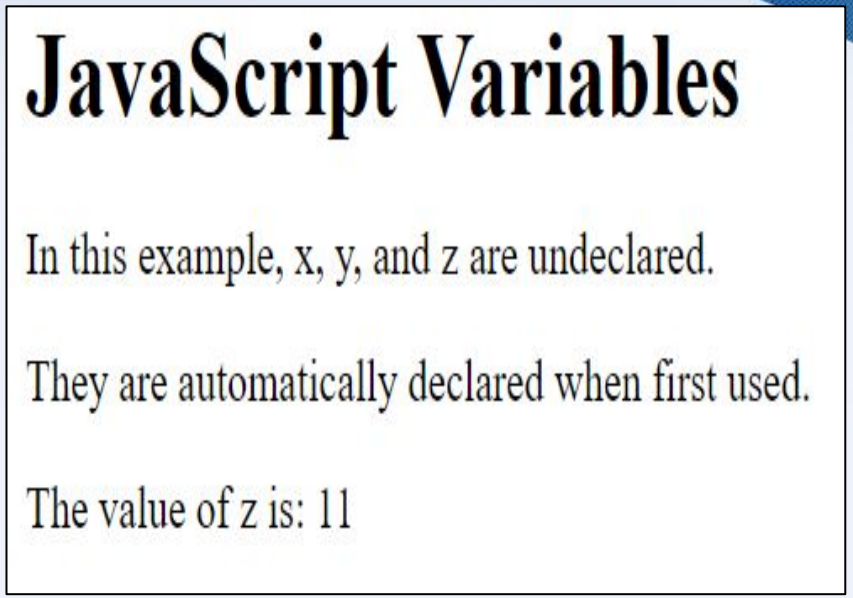
```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Variables</h1>

<p>In this example, x, y, and z are variables.</p>

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

<script>
let x = 5;
let y = 6;
let z = x + y;
document.getElementById("demo").innerHTML =
"The value of z is: " + z;
</script>

</body>
</html>
```



When to Use var, let, or const?

1. Always declare variables

2. Always use const if the value should not be changed

3. Always use const if the type should not be changed (Arrays and Objects)

4.Only use let if you can't use const.(**Block level scope)**

5. Only use var if you MUST support old browsers.(function level scope)

Data Types-

1. **String**
2. **Number**
3. **Bigint**
4. **Boolean**
5. **Undefined**
6. **Null**
7. **Symbol**
8. **Object**

The Object Data Type

The object data type can contain

1. **An object**
2. **An array**
3. **A date**

```
// Numbers:
let length = 16;
let weight = 7.5;

// Strings:
let color = "Yellow";
let lastName = "Johnson";

// Booleans
let x = true;
let y = false;

// Object:
const person = {firstName:"John", lastName:"Doe"};

// Array object:
const cars = ["Saab", "Volvo", "BMW"];

// Date object:
const date = new Date("2022-03-25");
```

Properties:-

1)When adding a number and a string, JavaScript will treat the number as a string

What will be the output ?

```
let x = 16 + "Volvo";
```

x = ?

Output = 16Volvo

2. JavaScript Evaluates Expression from Left to Right.

Different Sequence can produce different results.

```
let x = 16+4+"Volvo";
```

Output?

```
let x = "Volvo"+16+4;
```

Output?

In the second example , since the first operand is a string , all operands will be treated as a string.

Output1 = 20Volvo **Output2 = Volvo164**

Marks = [22,33,44];
Marks[0] = 11;
New Marks array becomes = [11,33,44];

Str = “Hello”;
Str[0] = “y”;

Str will still remain same i.e. Hello
Strings are immutable in JS.

Functions in JS

Block of code that perform a specific task,can be invoked whenever needed.
Eg. console.log("HELLO") , "abc".toUpperCase()

Parenthesis is to invoke a function.

1)Arrow Functions - Compact way of writing a function

```
functionName = (param1,para,2...) =>{
  // do some work
}
```

2)Callback Functions - a function passed as an argument to another function

3)Higher Order Functions/Methods - that either take a function as parameter or return a function.
Eg. forEach arr.forEach(CallBack Function)

JS: Document Object Model (DOM)

DOM stands for Document Object Model, and allows programmers generic access to:

- adding, deleting, and manipulating - of all styles, attributes, and elements in a document.
- It can be accessed via any language available in the browser, including Java, JavaScript/ECMAScript/JScript

Every tag, attribute, style, and piece of text is available to be accessed and manipulated via the DOM

-- the possibilities are endless:

- adding and removing tags, attributes and styles,
- animating existing elements
- hiding/ showing elements on a page.

Before we get started, you need to know a few terms that we will use:

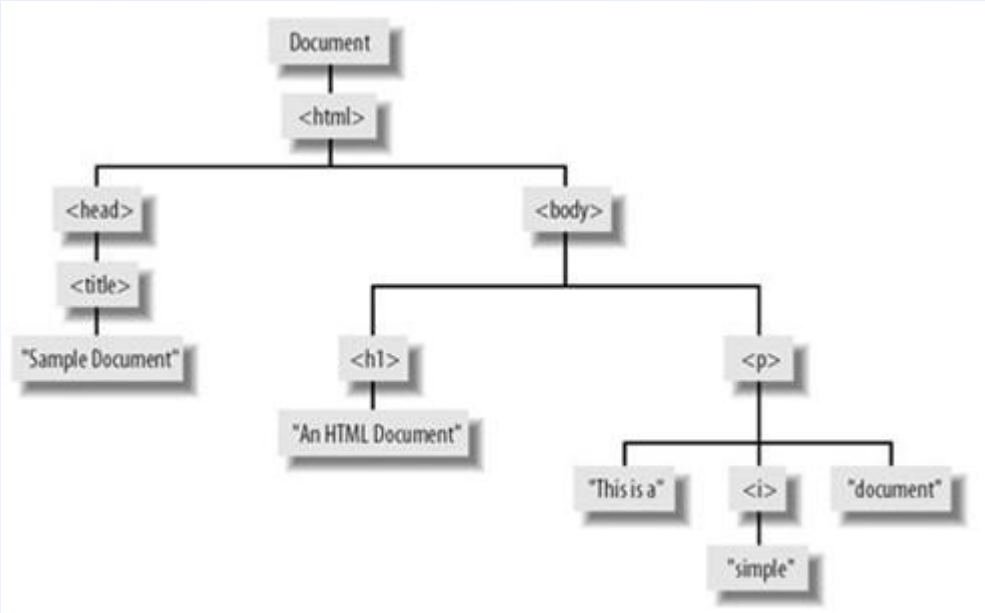
- Node: A reference to an element, its attributes, or text from the document.
- Element: A representation of a <TAG>.
- Attribute: A property of an element. for example,HREF is an attribute of <A>,
- The DOM represent an HTML documents as a tree of objects.
- The tree representation of an HTML document contains nodes representing HTML tags or elements, such as <body> and <p>, and nodes representing strings of text.
- An HTML document may also contain nodes representing HTML comments.

Consider the following simple HTML document:

```
<html>
    <head>
        <title>Sample Document</title>
    </head>
    <body>
        <h1> A HTML document</h1>
        <p> This is a <i>simple</i> document


</html>
```

The tree representation of the HTML document:



The node relationships of the previous DOM tree:

- HTML is an ancestor for head, h1, i, … etc
- head and body are sibling
- h1 and p are children of body
- head is parent of title
- “this is a” is the first children of p
- head is the firstChild of html

The DOM says that:

The entire document is a document node

Every HTML tag is an element node

The texts contained in the HTML elements are text nodes

Every HTML attribute is an attribute node

Comments are comment nodes

Relationship Properties

- firstChild is the first child node for the current node.
- lastChild is the last child object for the current node.
- childNodes is an array of all the child nodes under a node.
- previousSibling is the sibling before the current node.
- nextSibling is the sibling after the current node.
- parentNode is the object that contains the current node.

Node Methods

- appendChild(node) adds a new child node to the node after all its existing children.
- insertBefore(node, oldnode) inserts a new node before the specified existing child node.
- replaceChild(node, oldnode) replaces the specified old child node with a new node.
- removeChild(node) removes an existing child node.
- hasChildNodes() returns a Boolean value of true if the node has one or more children, or false if it has none.
- cloneNode() returns a copy of the current node.

Adding or modifying HTML element properties

- Create the element - var linkElement = document.createElement("a");
- Set the element content linkElement.innerHTML = "Click me to go to google!";
- Set element property linkElement.setAttribute("href", "https://www.google.com”)
- To change attribute value:
  Get the element
  Use setAttribute with a new value

Adding or modifying inline CSS

- Using the STYLE properties
- Only useful for setting style on one specific element.
- However, it is not useful for learning about the element's style in general,
- since it represents only the CSS declarations set in the element's inline style attribute,
- not those that come from style rules elsewhere, such as
  style rules in the <head> section, or
  external style sheets.
- Get the element, eg:
  var div = document.getElementById("div1");
- Set the style
  div.style.marginTop = ".25in";

FORM VALIDATION in JavaScript

It is important to validate the form submitted by the user because it can have inappropriate values. So, validation is must to authenticate user.

JavaScript provides facility to validate the form on the client-side so data processing will be faster than server-side validation. Most of the web developers prefer JavaScript form validation.

Through JavaScript, we can validate name, password, email, date, mobile numbers and more fields.

Example of Form validation

In this example, we are going to validate the name and password. The name can’t be empty and password can’t be less than 6 characters long.

Here, we are validating the form on form submit. The user will not be forwarded to the next page until given values are correct.

HTML CODE

```
<html>
<head>
   <title>Dice Simulator 2015</title>
   <link rel="stylesheet" href="style.css">
   </head>
<body>
<form name="myform" method="post" action="abc.jsp" onsubmit="return
validateform()" >
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
</form>
</html>
```

JAVASCRIPT CODE

```
<script>
function validateform(){
var name=document.myform.name.value;
var password=document.myform.password.value;

if (name==null || name==""){
  alert("Name can't be blank");
  return false;
}else if(password.length<6){
  alert("Password must be at least 6 characters long.");
  return false;
  }
}
</script>
```

https://codepen.io/Pyremell/pen/eZGGXX?editors=1000

JavaScript Simple Dice Roller

Jacob Schaefer + Follow

Settings | Sign Up | Log In

HTML

```
  return false;
}else if(password.length<6){
  alert("Password must be at least 6 characters long.");
  return false;
  }
}
</script>
<body>
<form name="myform" method="post" action="abc.jsp" onsubmit="return validateform()" >
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
</form>
</html>
```

CSS JS

Name:

Password:

register

Console | Assets | Comments | Shortcuts | Fork | Embed | Export | Share

Other ways of Form Validation

1. Retype password Validation //p1=p2
2. JavaScript Number Validation //Nan() fn
3. JavaScript validation with image
4. JavaScript email validation //

API Calls

API(Application Programming Interface) is a set of protocols, rules, and tools that allow different software applications to access allowed functionalities, and data, and interact with each other. API is a service created for user applications that request data or some functionality from servers.

To give specific access to our system to other applications that may be useful to them, developers create APIs and give them endpoints to interact and access the server data. While working with JavaScript it is common to interact with APIs to fetch data or send requests to the server.

Ways to make API Calls

- API Call in JavaScript Using XMLHttpRequest
- API Call in JavaScript Using the fetch() method
- API call in JavaScript using Axios
- API call in JavaScript Using the jQuery AJAX

API Call in JavaScript Using XMLHttpRequest

XMLHttpRequest is an object used to make API calls in JavaScript. Before the release of ES6 which came with Fetch and libraries like Axios in 2015, it is the only way to call API.

XMLHttpRequests are still used in multiple places because all new and old browsers support this.

- To implement advanced features like request cancellation and progress tracking, XMLHttpRequest is used.
- To maintain old codebases which are written before the announcement of ES6, we use XMLHttpRequest.

Example:-

```
const xhttpr = new XMLHttpRequest();
 xhttpr.open('GET', 'Api_address', true);
```

//**Step A:** Create an instance of XMLHttpRequest object.
//**Step B:** Make an open connection with the API endpoint by providing the necessary HTTP method and URL. Here '*true*' represents that the request is asynchronous.

```
xhttpr.send();
```

//**Step C:** Send the API request

```
    xhttpr.onload = ()=> {

if (xhttpr.status === 200) {
        const response = JSON.parse(xhttpr.response);
        // Process the response data here
} else {
        // Handle error
}
};
```

//**Step D:** Create a function onload when the request is completely received or loaded:

API Call in JavaScript Using the fetch() method

fetch is a method to call an API in JavaScript. It is used to fetch resources from a server. All modern browsers support the fetch method. It is much easy and simple to use as compared to XMLHttpRequest.

It returns a promise, which contains a single value, either response data or an error. fetch() method fetches resources in an asynchronous manner.

Example:-

```
fetch('Api_address')
.then(response => {
        if (response.ok) {
        return response.json();
        } else {
        throw new Error('API request failed');
        }
})
.then(data => {
        // Process the response data here
        console.log(data); // Example: Logging the data to the console
})
.catch(error => {
        // Handle any errors here
        console.error(error); // Example: Logging the error to the console
});
```

//**Step A:** Make an API request to the URL endpoint. Pass the API URL to the fetch() method to the request API which will return the Promise.

//Step B: Handle the response and parse the data. Use the .then() method to handle the response. Since the response object has multiple properties and methods to access data, use the appropriate method to parse the data. If the API return JSON data, then use .then() method.

//Step C: Handle the parsed data. Make another .then() method to handle the parsed data. Inside that, you can use that data according to your need.

//Step D: Handle the Error. To handle errors, use .catch() method at the end of the change.

API Call in JavaScript Using Axios

Axios is an open-source library for making HTTP requests to servers. It is a promise-based approach. It supports all modern browsers and is used in real-time applications. It is easy to install using the *npm* package manager.

It has better error handling than the fetch() method. Axios also does automatic transformation and returns the data in JSON format.

Example:-

```
import axios from 'axios';

axios.get('APIURL')
        .then(response => {
                // Access the response data
                const responseData = response.data;
                // Process the response data here
        })
        .catch(error => {
                // Handle any errors
        });
```

Explanation: While sending the HTTP request it will respond to us with the error or the data which is already parsed to JSON format(which is the property of Axios). We can handle data with the .then() method and at the end we will use .catch() method to handle the error.

Thank you!