Javascript Syllabus

1 Introduction to JavaScript

- What is JavaScript?
- Relationship between HTML, CSS, and JavaScript
- How to use JavaScript in a browser (Console, External JavaScript files)
- Understanding ECMAScript (ES) versions.

2 Basic Syntax and Structure

- **Variables (using var, let, and const)**
- Data Types: String, Number, Boolean, **Undefined**, Null
- **Comparison Operators**: ==, ===, !=, !==, <, >, <=, >=
- Scope
    - Local
    - Global
    - Functional
- Conditional Statements (if, else)
- Loops (for, while)
    - Loop Control: break, continue
- Template Literals
- Spread and Rest Operators (...)

3. Functions

- What is a function?
- Function Declaration and Execution
- Parameters and Arguments
- Return Values
- Higher-Order Functions
- First class function
- Calling Return function by parent function call. Eg. div()()
- **Arrow Functions** vs **Regular Functions**
- **Hoisting**
- **Execution context**
- **TDZ**

- rest parameter, default parameter
- **callback function**

4. Object

- Introduction to objects
- Creating objects with properties and methods
- Accessing Object Properties (Dot Notation, Bracket Notation, fo-in loop)
- modifying object properties
- **Object cloning (reference and primitive variable)**
- Nested objects
- Arrays inside objects
- Function inside objects
- Factory function
- Destructuring (Objects to variable)
- Constructuring (Variable to Object)
- Window object (global object)
    - Global window scope

5. Arrays

- What are arrays?
- Accessing array elements (indexing)
- Common array methods: push(), pop(), shift(), forEach(), map(), filter(), reduce(),reverse(), sort(), some(), includes(), find(), etc.
- Printing array by loop(for of, for)
- Nested Arrays and Multidimensional Arrays
- Array Destructuring
- Array Constructuring
- **Array of object,**
- Searching Objects in Arrays using Callback Functions.

6. DOM Manipulation (Document Object Model)

- What is the DOM?
- DOM tree
    - Understanding Nodes and Elements
    - HTML tags as “elements” in the DOM
    - Parent, child, and sibling relationships

1. Accessing and Selecting DOM Elements

- getElementById()
- getElementsByClassName()
- getElementsByTagName()
- querySelector()
- querySelectorAll()

2. Modifying DOM Elements

a) **textContent :** Functionality: Gets or sets the text content of an element.
element.textContent = "New Text";

b) **innerHTML :** Functionality: Gets or sets the HTML content inside an element.
element.innerHTML = "Bold Text";

3. Creating and Removing Elements

a) createElement() : Functionality: Creates a new element in the DOM.
Let newElement = document.createElement("div");

b) appendChild() : Append the element to an existing element in the DOM
element.appendChild(newParagraph);

c) insertAdjacentHTML: method to add an HTML element to an existing HTML tag in different position

"beforebegin": Before the element itself.
"afterbegin": Just inside the element, before its first child.
"beforeend": Just inside the element, after its last child.
"afterend": After the element itself.

Eg. myDiv.insertAdjacentHTML("beforeend", " World! ");

d) insertAdjacentElement : method to add an dom element to an existing HTML tag in different position.
Eg. exitingElement.insertAdjacentElement('afterbegin', createdElement);

e) removeChild(): : Removes a child node from the DOM.
Eg. element.removeChild(childElement);

f) Element.remove() : Directly removes the element from the DOM without needing to refer to its parent.

4. Adding style to HTML content by DOM

1. element.style.property = "value";
```
element.style.backgroundColor = "blue";
element.style.color = "white";
element.style.width = "200px";
element.style.height = "100px";
```

Common Properties for style:
backgroundColor, color, fontSize, width, Height, padding, margin, border, display, position, top, right, bottom, left

2. Using style.cssText to Apply Multiple Styles

element.style.cssText = "background-color: green; color: white; padding: 20px;";

5. Adding Class to html element

element.className Assigns or replaces the class of an element.
element.classList.add('class') Adds a class to the element.
element.classList.remove('class') Removes a class from the element.
element.classList.toggle('class') Toggles the presence of a class.
Element.classList.contains("class") check class is present or not

6. Event Basics:

- What are Events?
- What are listeners? (action)
- Event target:
- Event Types (click, mouseover, keydown, keyup.)

addEventListener() Method
Eg. Button.addEventListener("event", function)

removeEventListener() Method
Eg. Button.removeEventListener("event", function)

- event.preventDefault() : it use to prevent default event of <a>, <form>, <button> etc.

7. Working with json data :

JSON vs Object

Step 1: Parse JSON data into a JavaScript object
```
let productData = JSON.parse(jsonData);
```

Step 2: Update the data
```
productData.key = 700;
```

Step 3: Convert the updated object back to a JSON string
```
jsonData = JSON.stringify(productData);
```

8. APIs and Its working

- Fetch API
- Working with JSON
- Handling API Responses
- Async Programming with APIs

9. Working with fetch() function for api calling :

1. Read (GET)

```
const response = await fetch('/api/products');
```

2.Create (POST)

```
const response = await fetch('/api/products', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
```

```
        body: JSON.stringify(data)
    });
```

NOTE: we can also use FormData() to create object from input tags and need not to stringify it

3.update (PUT)

```
    const response = await fetch('/api/products/101', {
      method: 'PUT',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify(data)
    })
```

4.Delete (DELETE)

```
const response = await fetch( '/api/products/101', {
      method: 'DELETE'
    }

);
```

10. Making web page by JSON data

- **Make JSON data related to website for making web page :**
- **Request Object**:
  - Primarily used to configure and make HTTP requests.
  - Often used with the fetch() function to make network requests and handle HTTP
- **Normal JavaScript Object**:

11.Making setInterval, setTimeout, clearInterval in DOM

```
1. setTimeout(() => {

        }, timeout);

2. setInterval(() => {

        }, interval);

3.clearTimeout(id)

4.clearInterval(id)
```

12. Asynchronous JavaScript

- Introduction to Asynchronous Programming
- **Event loop**
- Event Queue
- Callback Hell
- Promise
- Promise Chaining
- async and await
- Try and catch for error handling

13.Web Storage API

- Local Storage,
- Session Storage
- Cookies

14.Making web page by class and object (OOPS concept)

```
class TodoList{
                        addItem(){
                                //          Define function for add item
                         }
}

let todoObj  = new TodoList()
//accessing class method by object with (.) operator

 <button onclick="todoObj.addItem()">Add Item</button>
```

15.Clousers

- nesting function
- Lexical scoping
- Outer scoping variable
- Function returning
- Clousers scoping