Unit 7. JavaScript

- Overview of JavaScript
- Advantages of JavaScript
- Implementing JavaScript code to HTML page using SCRIPT tag
- Variables in JavaScript
- JavaScript Data Type-Variant subtypes
- JavaScript Functions
- Event Handling and JavaScript objects
- Document Object Model in JavaScript
- Browser Objects and Events
- Document Objects and Events
- Form Objects and Events
- Dialog Box supported by JavaScript
- Form validation

JavaScript

- JavaScript is one of the most popular programming languages in the world.
- It is a lightweight, Interpreted Object-oriented scripting language.
- Client-side scripting refers to scripts that run within a web browser.
- JavaScript is used to add interactivity and dynamic effects to the web pages.
- JavaScript was originally developed as LiveScript by Netscape in the mid-1990s. It was later renamed to JavaScript in 1995 and became an ECMA standard in 1997.
- JavaScript runs on almost every modern web browser like Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge etc.

Applications/Advantages of JavaScript Programming

- Client-side validation- JavaScript can be used to verify any user input before submitting it to the server.
- Manipulating HTML Pages- JavaScript helps in manipulating HTML pages on the fly. This helps in adding and deleting any HTML tag very easily using JavaScript.
- User Notifications- You can use JavaScript to raise dynamic pop-ups on the webpages to give different types of notifications to your website visitors.
- Back-end Data Loading- JavaScript provides Ajax library which helps in loading back-end data.
- Presentations- JavaScript also provides the facility of creating presentations which give the website look and feel.
- Server Applications- Node JS is built on Chrome's JavaScript runtime for building fast and scalable server-side applications.

Adding and Using JavaScript:

There are two ways we can use JavaScript in our project.

a. Internal JavaScript
b. External JavaScript

a. Internal JavaScript

- The internal JavaScript is written inside the HTML document using <script> </script> tag.
- The script tag can be written either inside body tag or head tag. **b. External JavaScript**
- In external JavaScript, the JavaScript file with the extension (.js) is created and linked to the HTML document.

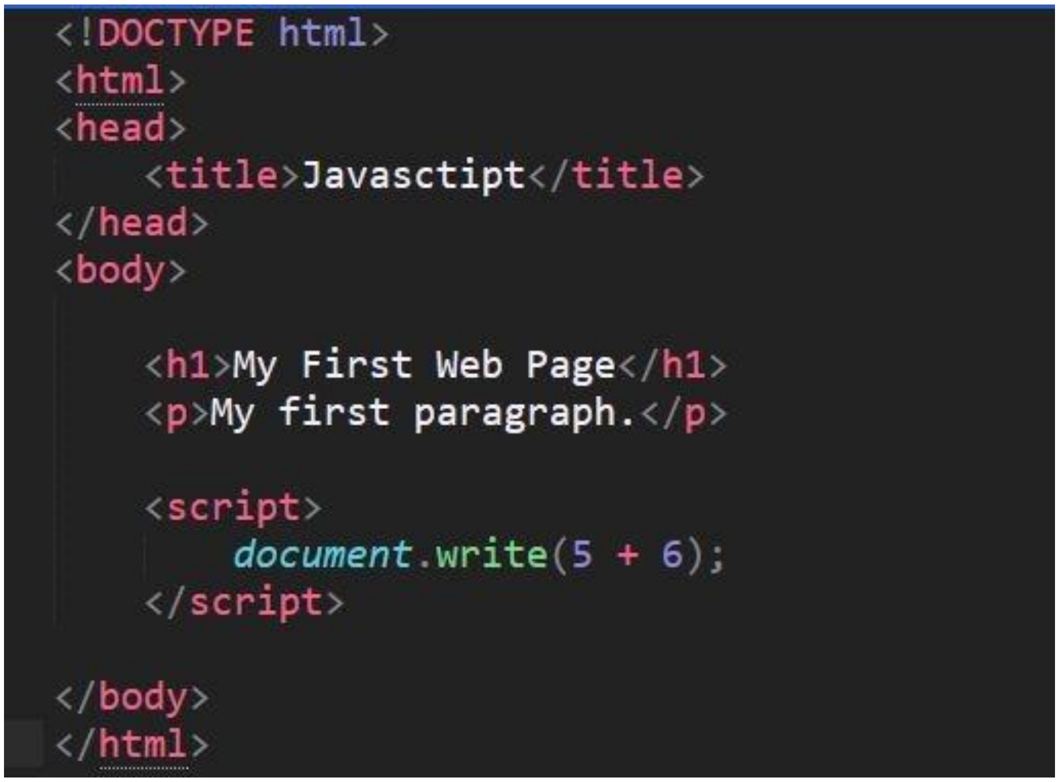
Example:

```
<script type = "text/javascript" src = "filename.js" ></script>
```

JavaScript Output:

JavaScript can "display" data in different ways:

- Writing into the HTML output using **document.write("hello world");**
- Writing into an alert box, using **alert("hello world");**
- Writing into the browser console, using **console.log("hello world");**
- Example:

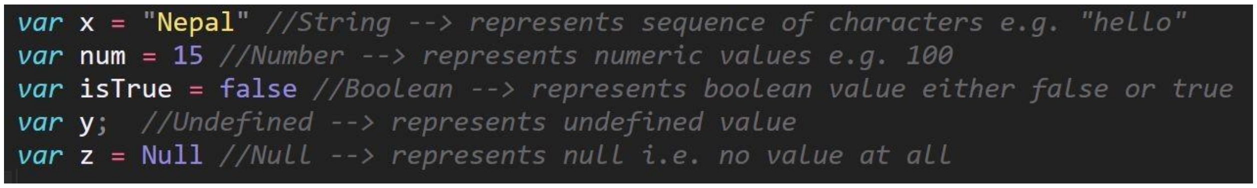


```
<!DOCTYPE html>
<html>
<head>
    <title>Javasctipt</title>
</head>
<body>

    <h1>My First Web Page</h1>
    <p>My first paragraph.</p>

    <script>
        document.write(5 + 6);
    </script>

</body>
</html>
```

JavaScript Variables and Datatypes:

- JavaScript variables are declared using <u>var</u> and <u>let</u> keyword.
- There are five primitive data types in JavaScript Number, String, Boolean, Null and Undefined.

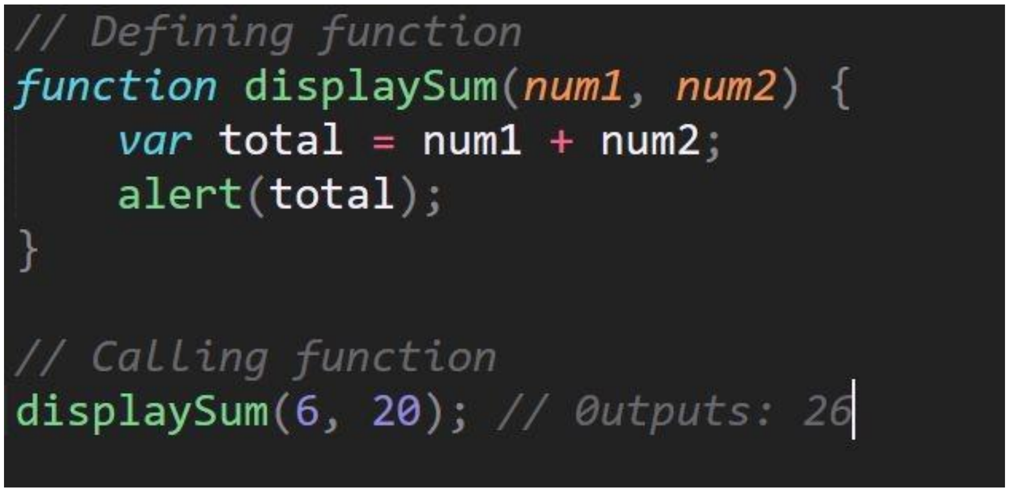
Example



```
var x = "Nepal" //String --> represents sequence of characters e.g. "hello"
var num = 15 //Number --> represents numeric values e.g. 100
var isTrue = false //Boolean --> represents boolean value either false or true
var y;  //Undefined --> represents undefined value
var z = Null //Null --> represents null i.e. no value at all
```

JavaScript functions:

- JavaScript function is a group of reusable code which can be called anywhere in the program.
- A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ().



```
// Defining function
function displaySum(num1, num2) {
    var total = num1 + num2;
    alert(total);
}

// Calling function
displaySum(6, 20); // Outputs: 26
```

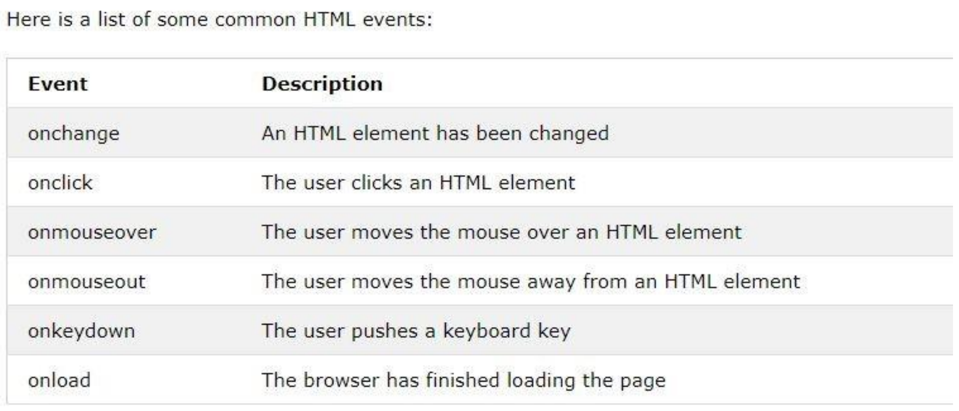
JavaScript Events

HTML events are "things" that happen to HTML elements.

When JavaScript is used in HTML pages, JavaScript can "react" to these events.

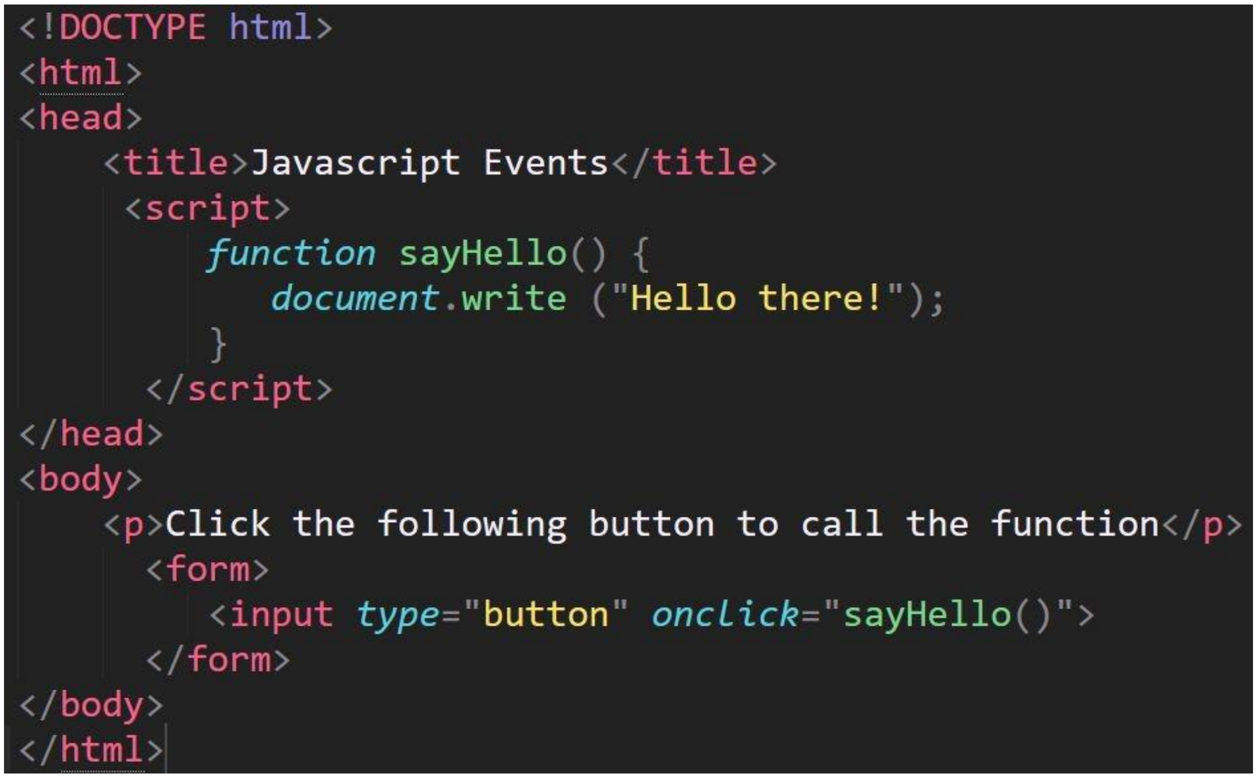
Here are some examples of HTML events:

- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked



Here is a list of some common HTML events:

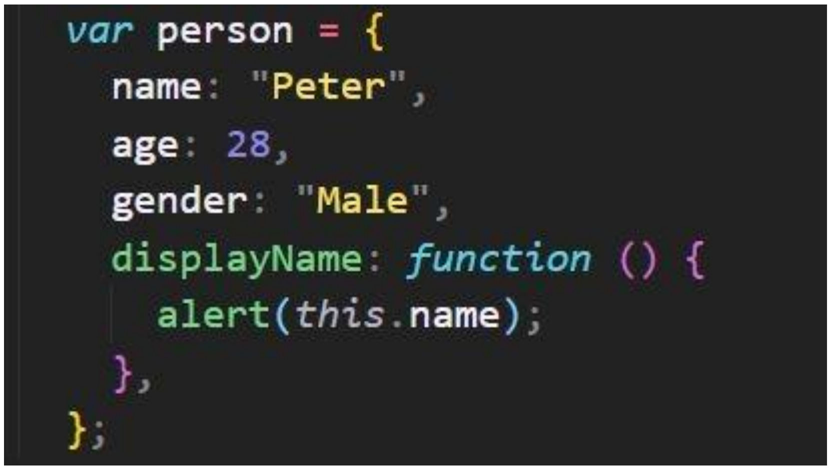
Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page


```html
<!DOCTYPE html>
<html>
<head>
    <title>Javascript Events</title>
     <script>
         function sayHello() {
            document.write ("Hello there!");
         }
      </script>
</head>
<body>
    <p>Click the following button to call the function</p>
      <form>
         <input type="button" onclick="sayHello()">
      </form>
</body>
</html>
```

JavaScript objects:

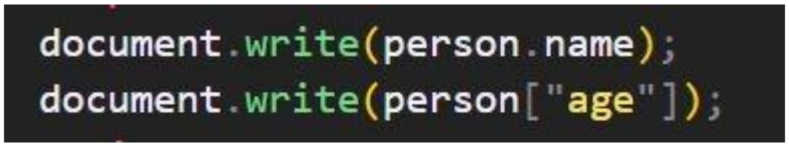
- A JavaScript object is an entity having state and behavior (properties and method).
- For example: car, pen, bike, chair, glass, keyboard, monitor etc.
- JavaScript is an object-based language. Everything is an object in JavaScript.
- An object can be created with curly brackets {} with an optional list of properties.
- A property is a "key: value" pair, where the key which is always a string, and value (or property value).

Example:


```javascript
var person = {
  name: "Peter",
  age: 28,
  gender: "Male",
  displayName: function () {
    alert(this.name);
  },
};
```

Accessing Object's Properties

To access or get the value of a property, you can use the dot (.), or square bracket ([]) notation, as demonstrated in the following example:



```
document.write(person.name);
document.write(person["age"]);
```

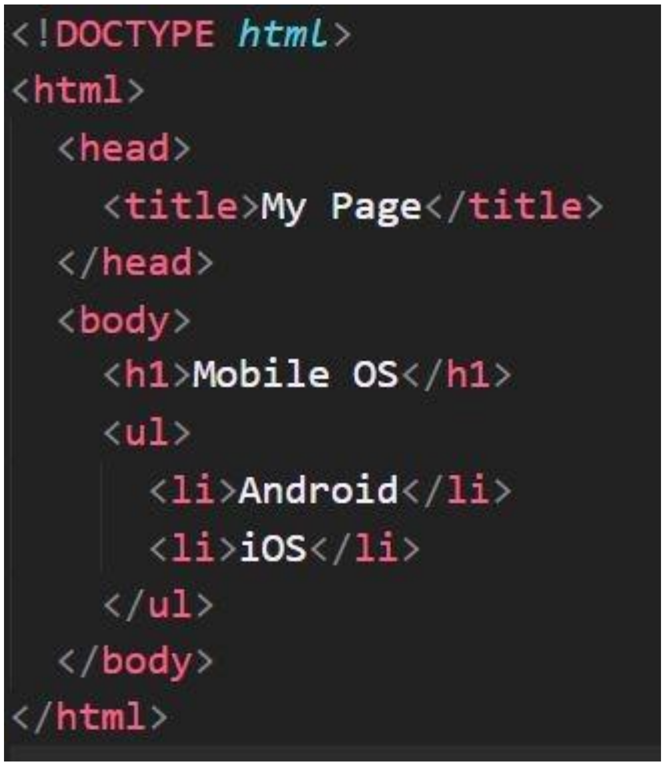
DOM (Document Object Model)

- When an HTML document is loaded into a web browser, it becomes a document object.
- It represents the structure of an HTML document as a tree-like structure.
- With the HTML DOM, JavaScript can access and change all the elements of an HTML document.
- It has properties and methods. By the help of document objects, we can add dynamic content to our web page.
- If we want to access any element in an HTML page, we always start with accessing the document object.

Here are some of the things we can do with the DOM:

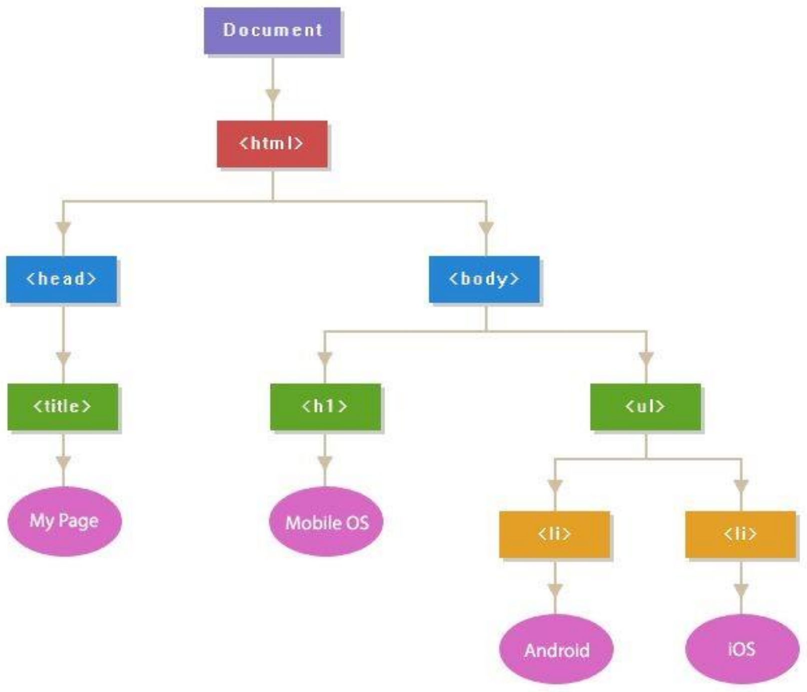
- **Access and manipulate the structure of a document.** You can use the DOM to get the elements of a document, their attributes, and their children. You can also use the DOM to add, remove, and move elements around in a document.
- **Change the style of a document**. You can use the DOM to get the style properties of an element, and you can also use the DOM to change the style properties of an element.
- **Change the content of a document.** You can use the DOM to get the text content of an element, and you can also use the DOM to change the text content of an element.
- **Respond to user events.** You can use the DOM to respond to user events, such as mouse clicks and key presses.
- **Create dynamic web pages.** You can use the DOM to create dynamic web pages that change their content or appearance based on user input.

Let's consider the following example:



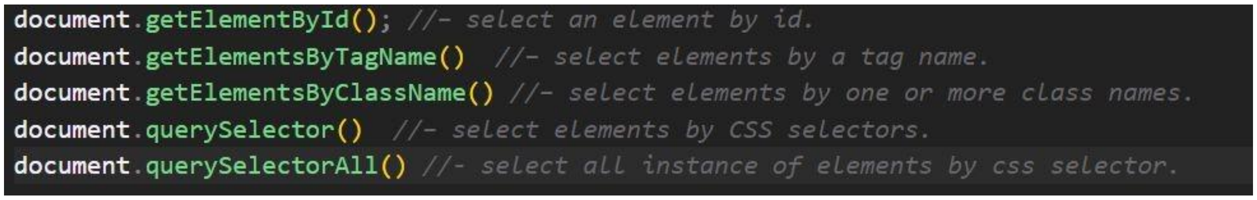
```
<!DOCTYPE html>
<html>
  <head>
    <title>My Page</title>
  </head>
  <body>
    <h1>Mobile OS</h1>
    <ul>
      <li>Android</li>
      <li>iOS</li>
    </ul>
  </body>
</html>
```

The above HTML document can be represented by the following DOM tree:



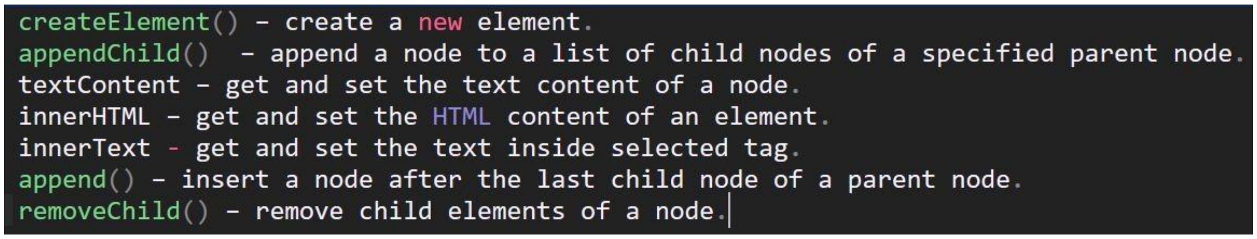
- The above diagram demonstrates the parent/child relationships between the nodes. The topmost node i.e. the Document node is the root node of the DOM tree, which has one child, the <html> element.
- Whereas, the <head> and <body> elements are the child nodes of the <html> parent node.
- The <head> and <body> elements are also siblings since they are at the same level.

Selecting Elements:



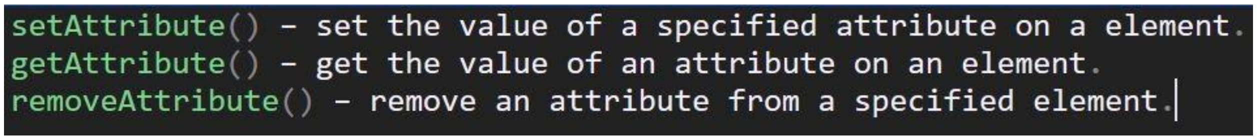
```
document.getElementById(); //- select an element by id.
document.getElementsByTagName()  //- select elements by a tag name.
document.getElementsByClassName() //- select elements by one or more class names.
document.querySelector()  //- select elements by CSS selectors.
document.querySelectorAll() //- select all instance of elements by css selector.
```

Manipulating Elements:



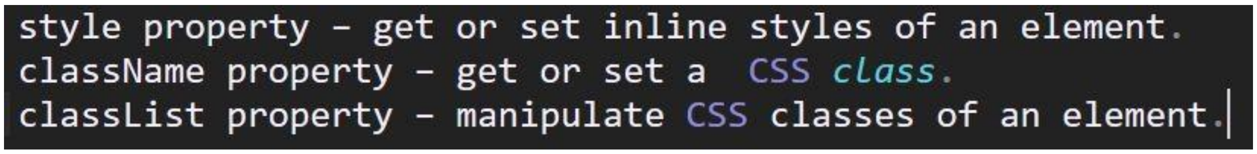
```
createElement() - create a new element.
appendChild()  - append a node to a list of child nodes of a specified parent node.
textContent - get and set the text content of a node.
innerHTML - get and set the HTML content of an element.
innerText - get and set the text inside selected tag.
append() - insert a node after the last child node of a parent node.
removeChild() - remove child elements of a node.
```

Working with attributes



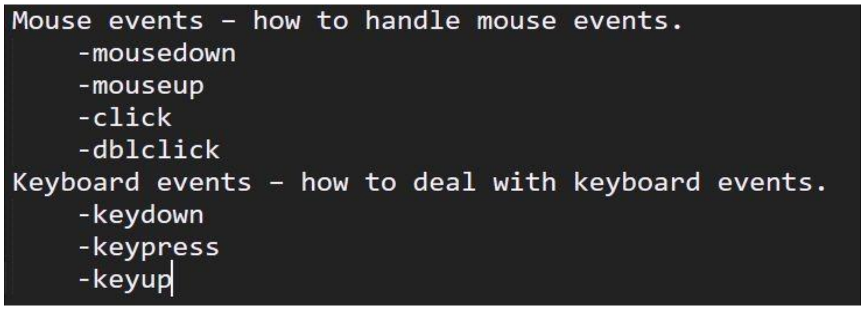
```
setAttribute() - set the value of a specified attribute on a element.
getAttribute() - get the value of an attribute on an element.
removeAttribute() - remove an attribute from a specified element.
```

Manipulating styline



```
style property - get or set inline styles of an element.
className property - get or set a  CSS class.
classList property - manipulate CSS classes of an element.
```

Working with Events



```
Mouse events - how to handle mouse events.
    -mousedown
    -mouseup
    -click
    -dblclick
Keyboard events - how to deal with keyboard events.
    -keydown
    -keypress
    -keyup
```

Browser objects and Events

Browser objects and events are two important concepts in web development. Browser objects are the elements that make up a web page, such as the window, the document, and the elements within the document. Events are the things that happen in a web page, such as a user clicking on a button or scrolling down the page.

Here are some examples of browser objects:

- Window: The window object represents the browser window itself. It contains properties and methods for accessing and manipulating the window, such as the title, the size, and the location of the window.
- Document: The document object represents the HTML document that is being displayed in the browser window. It contains properties and methods for accessing and manipulating the document, such as the elements in the document, the style of the document, and the content of the document.
- Element: An element is a part of the document, such as a <div>, <p>, or <a> tag. Elements have properties and methods for accessing and manipulating their content, style, and behavior.

Here are some examples of events:

- Click: The click event occurs when the user clicks on an element.
- Scroll: The scroll event occurs when the user scrolls the page.
- Change: The change event occurs when the user changes the value of an input element.

Example

```
<script>
      // Get the button element.
      var button = document.getElementById("myButton");

      // Add an event listener to the button.
      button.addEventListener("click", function () {
        // Do something when the button is clicked.
      });
```

```
</script>
```

Form objects and events

Form objects are the elements that make up a form, such as the input fields, the buttons, and the labels. Events are the things that happen in a form, such as a user clicking on a button or typing in an input field.

Here are some examples of form objects:

- Input: An input is an element that allows the user to enter text, numbers, or other data.
- Button: A button is an element that allows the user to submit the form or perform an action.
- Label: A label is an element that provides a description for an input field.

Here are some examples of events:

- Submit: The submit event occurs when the user submits the form.
- Change: The change event occurs when the user changes the value of an input field.
- Focus: The focus event occurs when the user focuses on an input field.

Example:

```
 // Get the form element.
     var form = document.getElementById("myForm");

     // Add an event listener to the form.
     form.addEventListener("submit", function () {
       // Do something when the form is submitted.
     });
   </script>
```

Dialog Box supported by JavaScript

JavaScript provides three types of dialog boxes: **alert()**, **confirm()**, and **prompt()**.

These dialog boxes allow us to interact with users by displaying messages, getting user confirmation, or obtaining user input. Here's an explanation of each dialog box with a simple example for each:

1. alert() dialog box:
    - The **alert()** dialog box displays a message to the user with an OK button.
    - It is commonly used to show information or notify the user about something.
    - Example:

```
    <script>
     alert("Hello, there! Welcome to our website.");
    </script>
```

2. confirm() dialog box:
    - The **confirm()** dialog box displays a message to the user with OK and Cancel buttons.
    - It is used to get user confirmation for an action, where the user can choose between two options (OK or Cancel).
    - It returns **true** if the user clicks OK, and **false** if the user clicks Cancel.
    - Example:

```
 <script>
      var result = confirm("Are you sure you want to delete
this item?");
</script>
```

3. prompt() dialog box:
    - The **prompt()** dialog box displays a message to the user along with an input field and OK and Cancel buttons.
    - It is used to get user input by prompting the user to enter some information.
    - It returns the value entered by the user if they click OK, and **null** if they click Cancel or leave the input field empty.
    - Example:

```
<script>
      var name = prompt("What is your name?");
      document.write("Hello " + name);
</script>
```

JavaScript form validation

- JavaScript form validation is a way to check the data entered into a form before it is submitted to a server.
- This can help to ensure that the data is valid and that the form is filled out correctly.

There are two main types of JavaScript form validation:

- Client-side validation: This is done on the user's browser before the form is submitted to the server.
- Server-side validation: This is done on the server after the form is submitted. Here is a simple example of JavaScript client-side form validation:

Example:

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

    <script>
      function validateform() {
        var name = document.myform.name.value;
        var password1 = document.myform.password1.value;
        var password2 = document.myform.password2.value;

        if (name == null || name == "") {
          alert("Name can't be blank");
          return false;
        } else if (password1.length < 6) {
          alert("Password must be at least 6 characters long.");
          return false;
        } else if (password2 !== password1) {
          alert("Password must be the same");
          return false;
        }
      }
    </script>
  </head>
  <body>
    <form name="myform" method="post" action="#" onsubmit="return
validateform()">
      Name: <input type="text" name="name" /> <br />
      Password: <input type="password" name="password1" /> <br />
      Confirm Password: <input type="password" name="password2" /> <br
/>
      <input type="submit" value="register" />
    </form>
  </body>
</html>
```