**4 JavaScript**

**4.1 Basics of JavaScript**

**4.1.1 JavaScript Introduction**

* JavaScript was initially created to “make web pages alive”.
* The programs in this language are called *scripts*. They can be written right in a web page’s HTML and run automatically as the page loads.
* Scripts are provided and executed as plain text. They don’t need special preparation or compilation to run.
* JavaScript can execute not only in the browser, but also on the server, or actually on any device that has a special program called [the JavaScript engine](https://en.wikipedia.org/wiki/JavaScript_engine).
* The browser has an embedded engine sometimes called a “JavaScript virtual machine”.

**4.1.2 Use of JavaScript**

* **Creating Interactive User Interfaces**: Adds interactivity to web pages, enabling elements like sliders, pop-ups, and form validations.
* **Manipulating HTML and CSS**: Dynamically changes the content, layout, and styling of web pages without reloading, ensuring a more responsive user experience.
* **Single-Page Applications (SPAs)**: Helps build SPAs using frameworks like React, Angular, and Vue, allowing seamless page navigation without full reloads.
* **Handling Asynchronous Operations**: Manages data fetching from servers using AJAX or Fetch API, enabling real-time content updates without refreshing the page.
* **Animations and Effects**: Creates animations, transitions, and visual effects to enhance user engagement.
* **Form Handling and Validation**: Validates user inputs directly in the browser, improving user experience by providing instant feedback.
* **Browser Storage**: Utilizes local storage, session storage, and cookies for saving data on the client side, allowing persistent user sessions and preferences.

**4.1.3 Way to include javascript**

1. **Inline JavaScript**: Placed directly within an HTML element using the onclick, onmouseover, or other event attributes.

Ex : <button onclick="alert('Inline World!')">Click Me</button>

1. **Internal JavaScript**: Written inside a <script> tag within the <head> or <body> section of an HTML document.

Ex :

<script>

function showMessage() {

alert('Internal JavaScript!');

}

</script>

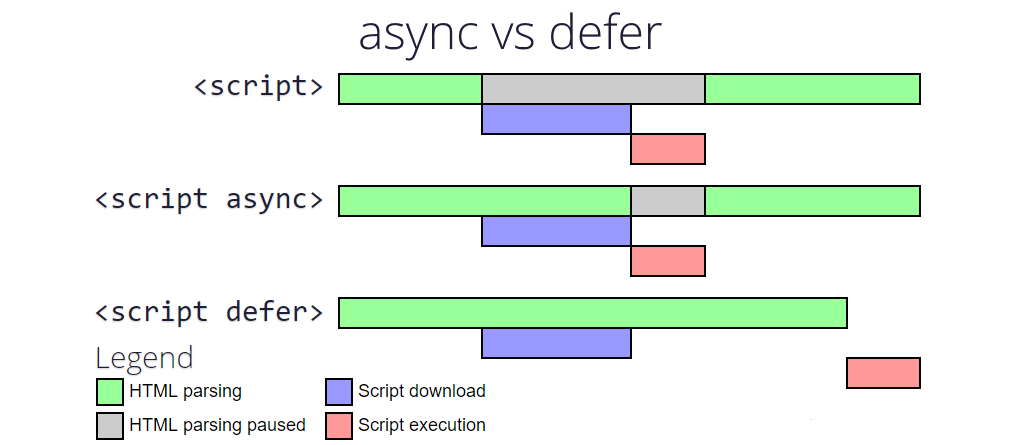
1. **External JavaScript**: Stored in a separate .js file and linked to the HTML document using a <script src="script.js"></script> tag. This is the most organized and reusable method.

Ex :

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

Note : Internal and External can be placed in body or head. Preferred is at last of body.

**Loading strategies :**



**4.1.4 Syntax of JavaScript**

**1. Case Sensitivity**

* All JavaScript keywords, variables, function names, and identifiers are case-sensitive.
* Example: myName is different from myname .

**2. Statements and Semicolons**

* **Statements:** Instructions executed by JavaScript. You can have multiple statements on a single line, separated by semicolons.
* **Semicolons (;):** Although optional, it's good practice to end statements with a semicolon to avoid errors, especially in complex code.

**3. Comments**

* **Single-line:** // This is a single-line comment
* **Multi-line:**

/\*

This is

a multi-line comment

\*/

**4. Variables**

* **Declaration Keywords:**
  + **var**: Function-scoped, can be redeclared and updated.
  + **let:** Block-scoped, can be updated but not redeclared within the same scope.
  + **const**: Block-scoped, must be initialized at declaration and cannot be reassigned.

**Ex** :

var name = “keyur”; // function-scoped

let age = 20; // block-scoped

const birthYear = 2004; // cannot be changed

**5. Data Types**

* **Primitive Types**: String, Number, Boolean, Undefined, Null
* **Composite Types**: Object (includes Arrays, Functions, and other Objects)

Ex :

let name = "keyur"; // String

let age = 20; // Number

let bigInt = 1234567890123456789012345678901234567890n

// BigInt (for number that outside range of ±(253-1))

let like = true; // Boolean

let data = null; // Null

let info; // Undefined

let obj = {name: "keyur"}; // Object

let arr = [1, 2, 3]; // Array

**6. Operators**

* **Arithmetic Operators:** + (addition), - (subtraction), \* (multiplication), / (division), % (modulus), ++ (increment), -- (decrement)
* **Assignment Operators:** =, +=, -=, \*=, /=
* **Comparison Operators:** == (equal to), != (not equal), >, <, >=, <=
* **Logical Operators:** && (AND), || (OR), ! (NOT)
* **typeof Operator :** returns the type of the operand.

**7. Control Structures**

* **Conditional Statements:**
  + if: Executes a block if the condition is true.
  + else if: Checks additional conditions if the previous ones are false.
  + else: Executes if none of the conditions are true.
  + switch: Evaluates an expression against multiple cases.
* **Loops:**
  + for: Repeats a block a specified number of times.
  + while: Repeats a block as long as a condition is true.
  + do...while: Executes the block at least once before checking the condition.

**8. Functions**

* Functions are reusable blocks of code that perform specific tasks.
* Can be declared using function keyword.

Ex :

function greet(name=”keyur”) {

  return "Hello, " + name;

}

console.log(greet("keyur"));

**4.1.5 Basic event of javascript**

**1.** **onclick :**  Triggered when an element is clicked.

Ex :

<button onclick="alert('Button clicked!')">Click Me</button>

**2.** **onmouseover** **:** Triggered when the mouse pointer hovers over an element.

Ex :

<p onmouseover="this.style.color='red'">Hover over me!</p>

**3.** **onmouseout** **:** Triggered when the mouse pointer leaves an element.

Ex :

<p onmouseout="this.style.color='black'">Move your mouse away from me!</p>

**4.** **onkeydown** **:** Triggered when a key is pressed on the keyboard.

Ex :

<input type="text" onkeydown="alert('Key pressed!')" placeholder="Press any key">

**5.** **onload** **:** Triggered when the web page or an element is completely loaded.

Ex :

<body onload="alert('Page loaded!')">

**6.** **onchange** **:** Triggered when the value of an input field changes.

Ex :

<input type="text" onchange="alert('Text changed!')" placeholder="Type something">

**7.** **onsubmit** **:** Triggered when a form is submitted.

Ex :

<form onsubmit="alert('Form submitted!'); return false;">

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

</form>

**8.** **onfocus** **:** Triggered when an element gains focus, typically an input field.

Ex :

<input type="text" onfocus="this.style.backgroundColor='yellow'" placeholder="Focus on me">

**9.** **onblur** **:** Triggered when an element loses focus.

Ex :

<input type="text" onblur="alert('Input lost focus!')" placeholder="Click and then click away">

**10.** **ondblclick** **:** Triggered when an element is double-clicked.

Ex :

<button ondblclick="alert('Button double-clicked!')">Double Click Me</button>

**4.1.6 Basic Validation with javascript**

**Code :**

function validateForm(){

let valid = true;

// Clear previous error messages

document.getElementById('usernameError').textContent = '';

document.getElementById('emailError').textContent = '';

document.getElementById('passwordError').textContent = '';

// Username validation

const username = document.getElementById('username').value.trim();

if (username.length < 3) {

document.getElementById('usernameError').textContent = 'Username must be at least 3 characters';

valid = false;

}

// Email validation

const email = document.getElementById('email').value.trim();

const emailPattern = /^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-z]{2,}$/;

if (!emailPattern.test(email)) {

document.getElementById('emailError').textContent = 'Invalid email format';

valid = false;

}

// Password validation

const password = document.getElementById('password').value.trim();

if (password.length < 6) {

document.getElementById('passwordError').textContent = 'Password must be at least 6 characters';

valid = false;

}

returnvalid;

};

**Interaction: alert, prompt, confirm**

**Alert :** shows a message in modal window.

Ex : alert("Hello");

**Prompt** : shows a modal window with a text message, an input field for the visitor, and the buttons OK/Cancel.

result = prompt(title, [default]);

Ex : let age = prompt('How old are you?', 100);

**Confirm** : shows a modal window with a question and two buttons: OK and Cancel. The result is true if OK is pressed and false otherwise.

Ex : let isBoss = confirm("Are you the boss?");

All these methods are modal: they pause script execution and don’t allow the visitor to interact with the rest of the page until the window has been dismissed.

https://www.google.com/search?q=.on%28%29+ans+.off%28%29+with+event+in+js&sca\_esv=52b6d338ebf5aab8&sxsrf=ADLYWIJmDWQVmij4nTfvSF8WAyGfw5YGUg%3A1729678901087&ei=Nc4YZ8j1BMimseMPytnf8AU&ved=0ahUKEwiI5OWDpKSJAxVIU2wGHcrsF14Q4dUDCA8&uact=5&oq=.on%28%29+ans+.off%28%29+with+event+in+js&gs\_lp=Egxnd3Mtd2l6LXNlcnAiIS5vbigpIGFucyAub2ZmKCkgd2l0aCBldmVudCBpbiBqczIGEAAYFhgeMgsQABiABBiGAxiKBTILEAAYgAQYhgMYigUyCxAAGIAEGIYDGIoFMggQABiiBBiJBTIIEAAYgAQYogRIjilQyANYoCdwAXgBkAEAmAHVAqABxSmqAQgwLjEuMTcuMrgBA8gBAPgBAZgCEaACtSLCAgoQABiwAxjWBBhHwgIGEAAYDRgewgIIEAAYFhgeGA\_CAgcQIRigARgKmAMA4gMFEgExIECIBgGQBgiSBwgxLjAuMTQuMqAH2ocB&sclient=gws-wiz-serp