**HTML Event –**

* HTML events are actions that happen in the browser and can be detected by HTML element.
* These html event are used to trigger javascript function or code which is interact with the browser
* In Simple word, html events are used to attached javascript function or code with html element

Some Imp events –

* + - * onclick
      * onchange
      * onsubmit
      * onmouseover
      * onmouseout
      * onkeypress / onkeydown

 **Mouse Events**

* click → when an element is clicked
* dblclick → double click
* mouseover → cursor moves over element
* mouseout → cursor leaves element
* mousedown / mouseup → mouse button pressed / released
* mousemove → cursor moves

 **Keyboard Events**

* keydown → key is pressed
* keyup → key is released
* keypress → key is pressed and produces a character (deprecated in modern JS)

 **Form Events**

* submit → form submitted
* change → input value changed
* focus → element gets focus
* blur → element loses focus
* input → value changes in real time

### What is addEventListener ?

1. addEventListener is a method in JavaScript used to attach an event handler to an HTML element. It allows you to run a function when a specified event occurs (like a click, mouseover, keypress, etc.).

Syntax -

htmtElement.addEventListener(event, function)

event: The name of the event (without the on prefix, e.g., "click" instead of "onclick").

function: The callback function to run when the event occurs.

**Why Use addEventListener?**

* You can attach **multiple event listeners** to the same element and event type.
* It provides **cleaner separation** of HTML and JavaScript.
* It's more flexible than assigning events directly (e.g., element.onclick = ...).

## **What is RegEx (Regular Expression)?**

A **Regular Expression (RegEx)** is a sequence of characters that defines a search pattern used for matching, searching, and manipulating strings. It is widely used in programming languages for tasks like input validation, search-and-replace operations, or data extraction.

## **Basic Structure of RegEx**

A RegEx pattern consists of **literal characters** (exact matches) and **metacharacters** (symbols with special meanings). RegEx engines evaluate the pattern and apply it to strings for matching.

**Key Components of RegEx**

1. **Literal Characters**: Exact characters that must be matched.
   * Example: /hello/ matches the word "hello" in a string.
2. **Metacharacters**: Characters with special meanings in RegEx.
   * Example: . (dot) matches any character except a newline.

**RegEx Metacharacters and Their Meanings**

|  |  |  |
| --- | --- | --- |
| **Metacharacter** | **Description** | **Example** |
| . | Matches any character except newline | /a.c/ matches "abc", "axc" |
| ^ | Matches the start of a string | /^a/ matches "apple" but not "banana" |
| $ | Matches the end of a string | /e$/ matches "apple" but not "edge" |
| \* | Matches 0 or more of the preceding character | /ab\*c/ matches "ac", "abc", "abbc" |
| + | Matches 1 or more of the preceding character | /ab+c/ matches "abc", "abbc" |
| ? | Matches 0 or 1 of the preceding character | /colou?r/ matches "color" or "colour" |
| {n} | Matches exactly n occurrences | /a{3}/ matches "aaa" |
| {n,} | Matches n or more occurrences | /a{2,}/ matches "aa", "aaa" |
| {n,m} | Matches between n and m occurrences | /a{2,4}/ matches "aa", "aaa", "aaaa" |
| [] | Matches any character inside the brackets | /[aeiou]/ matches any vowel |
| [^] | Matches any character **not** inside the brackets | /[^aeiou]/ matches any non-vowel |
| () | Groups characters for capturing or applying quantifiers | /(abc)+/ matches "abcabc" |
| ` | ` | Acts as an OR operator |
| \ | Escapes special characters | /\$/ matches "$" |

**Character Classes (Predefined Sets)**

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | **Example** |
| \d | Matches any digit (0-9) | /\d/ matches "123" |
| \D | Matches any non-digit | /\D/ matches "abc", "@" |
| \w | Matches any word character (alphanumeric + \_) | /\w/ matches "abc123\_" |
| \W | Matches any non-word character | /\W/ matches "@" or "!" |
| \s | Matches any whitespace (space, tab, newline) | /\s/ matches " " or "\t" |
| \S | Matches any non-whitespace | /\S/ matches "abc", "1" |

**Anchors**

|  |  |  |
| --- | --- | --- |
| **Anchor** | **Description** | **Example** |
| ^ | Matches the beginning of a string | /^hello/ matches "hello world" |
| $ | Matches the end of a string | /world$/ matches "hello world" |
| \b | Matches a word boundary | /\bword\b/ matches "word" |
| \B | Matches non-word boundary | /\Bword/ matches "password" |

## Examples of regex

1. Validate Name (only alphabets)
   1. /^[A-Za-z\s]+$/; - / Uppercase, lowercase, space /
2. Validate Username must start with an uppercase letter, include one special character, one number, and be at least 6 characters long.
   1. /^[A-Z][A-Za-z\d!@#$%^&\*()\_+~\-=`{}|:";'<>,.?/]{6,}$/;
   2. /^[A-Z][A-Za-z\d@]{6,}$/ - @ only
   3. Username - Virat@123, ViraT@123
3. Validate Email (required, proper format)
   1. /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
4. Validate Password must be at least 8 characters, include one uppercase letter, and one special character
   1. /^(?=.\*[A-Z])(?=.\*[!@#$%^&\*])[A-Za-z\d!@#$%^&\*]{8,}$/;
      1. ^ -
         1. Indicates the start of the string.
      2. (?=.\*[A-Z])
         1. Ensures that the string contains at least one uppercase letter (A-Z).
      3. (?=.\*[!@#$%^&\*])
         1. Ensures that the string contains at least one special character from the set [!@#$%^&\*]
      4. [A-Za-z\d!@#$%^&\*]{8,}
         1. Matches a string of **at least 8 characters**.
         2. The allowed characters are:
            * Uppercase letters (A-Z)
            * Lowercase letters (a-z)
            * Digits (0-9)
            * Special characters (!@#$%^&\*).
5. Validate only numbers -
   1. /^\d+$/;
6. **Password must contain at least one uppercase, one lowercase, one special character, and one number and at least 8 character long**

/^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{8,}$/