



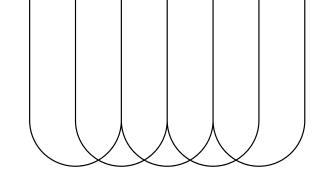
Form handling and validation in JavaScript is a fundamental concept used to collect and validate user input in web applications

Basic Form Handling in JavaScript

```
document.getElementById("myForm").addEventListener("submit", function (e) {
    e.preventDefault(); // prevent default form submission
    let name = document.getElementById("name").value.trim();
    let email = document.getElementById("email").value.trim();
    let message = document.getElementById("message");
    message.innerHTML = `Submitted Name: ${name}, Email: ${email}`;
});
```

Tips for Better Validation

- Use trim() to avoid spaces.
- Show error messages near input fields for better UX.
- Use HTML5 attributes like required, minlength, maxlength, and type for basic validation.
- Consider using libraries like <u>Validator.js</u> for complex forms





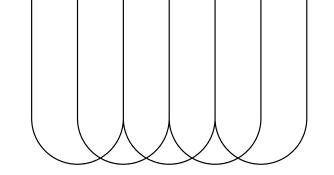
Form Validation

Common Types of Validation

- Required fields
- Email format
- Minimum/maximum length
- Custom rules

Required fields validation

```
if (name === "" || email === "") {
  message.innerHTML = "All fields are required.";
  message.style.color = "red";
  return;
}
```



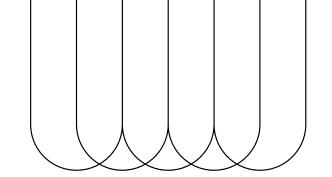


Email fields validation

```
// Simple email validation using RegEx
let emailPattern = /^[^]+@[^]+\.[a-z]{2,3}$/;
if (!emailPattern.test(email)) {
  message.innerHTML = "Please enter a valid email.";
  message.style.color = "red";
  return;
}
```

Minimum/Maximum Length validation

```
if (!(password.length >= 8)) {
  message.innerHTML = "password must be at least 8 character long";
  message.style.color = "red";
  return;
}
```





Debounce

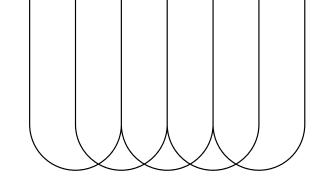
Debouncing means "wait for a pause".

It delays the function call until a certain amount of time has passed since the last time the event was triggered.

Use Case

- Search box auto-suggestions
- Mousemove event
- Input validation
- Filtering or Sorting in UI

```
let searchBar = document.querySelector('input[name="search"]');
searchBar.addEventListener('input', debounce(searchQuery, 300));
function debounce(func, delay){
  let timer;
  return function (...args){
    clearTimeout(timer)
    timer = setTimeout(()=>{
       func.apply(this, args);
    }, delay)
function searchQuery(){
  let val = this.value.trim();
  if(val.length){ console.log(val); }
```





Throttling

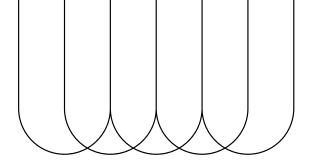
Throttling means "limit the rate".

It ensures the function is called at most once every specified interval.

Use Case

- Scroll event (e.g., infinite scrolling)
- Resizing the window
- Game Controls
- Auto-Sync or Polling

```
document.addEventListener('scroll', throttle(updateScroll, 300));
function throttle(func, delay){
  let lastCall = 0; let timer;
  return function(...args){
    let now = Date.now();
    if(now-lastCall > delay){
       lastCall = now; func.apply(this, args)
    }else{
       clearTimeout(timer);
       timer = setTimeout(()=>{
         func.apply(this, args)
       }, delay)
function updateScroll(e){ console.log(window.scrollY )}
```





THANK YOU

PHONE NUMBER

(+91) 778 899 2897

WEBSITE

www.indixpert.com

