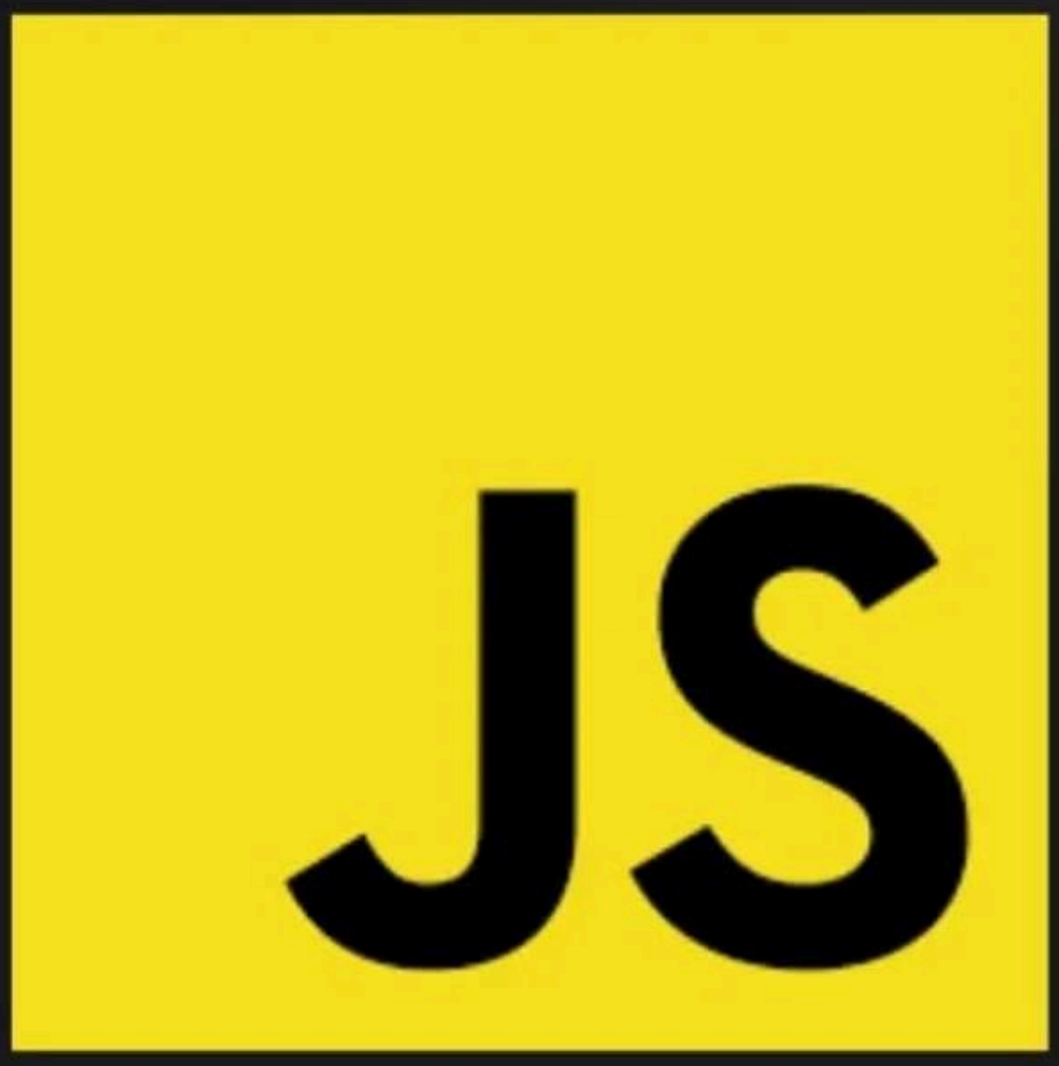
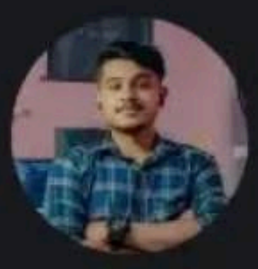
Swipe

JavaScript Noob to Pro





Nikhil | FullStack Dev
@codewithnikhill

Instead of using **var** for variable declarations

```
var name = "John";
```

Use let or const

```
const name = "John";  // for constants
let age = 30;  // for variables that can change
```

Instead of using == for comparison

```
if (age == "30") {
  // true even if one is a string
}
```

Use === for strict comparison

```
if (age === 30) {
  // true only if both value and type match
}
```

Instead of not handling **async/await** properly

```
async function fetchData() {
  const data = fetch("https://api.example.com");
  console.log(data);  // logs promise, not the data
}
```

Always **await** asynchronous operations

```
async function fetchData() {
  const data = await fetch("https://api.example.com");
  console.log(data);  // logs the actual data
}
```

Instead of using **for loops** unnecessarily

```
const numbers = [1, 2, 3];
for (let i = 0; i < numbers.length; i++) {
  console.log(numbers[i]);
}
```

Use **higher-order** array methods like **forEach**, **map**, etc.

```
numbers.forEach(number => console.log(number));
```

Instead of modifying the **original array** with **splice** or other mutable methods

```
let arr = [1, 2, 3];
arr.splice(1, 1); // modifies the original array
```

Use immutable methods like **filter**

```
let newArr = arr.filter(item => item !== 2);
// returns a new array
```

Instead of using **function declarations** for everything

```
function add(a, b) {
  return a + b;
}
```

Use **arrow functions** for cleaner, more concise code

```
const add = (a, b) => a + b;
```

Instead of **chaining** too many **promises**

```
fetchData()
  .then(result => processResult(result))
  .then(processed => display(processed))
  .catch(error => console.log(error));
```

Use **async/await** for cleaner async code

```
async function handleData() {
  try {
    const result = await fetchData();
    const processed = await
processResult(result);
    display(processed);
  } catch (error) {
    console.log(error);
  }
}
```

Instead of using **innerHTML** when possible XSS risks exist

```
document.getElementById("content").innerHTML = userInput;
```

Use **textContent** to avoid injecting untrusted content

```
document.getElementById("content").textContent = userInput;
```