JavaScript Useful Tips

JS

Variable Declaration

Use the const and let keywords instead of var to declare variables. This helps prevent accidental variable reassignment and enforces block scoping.

index.js

```
// Bad:
var x = 10;
x = 20;

// Good:
let x = 10;
x = 20;

const y = 30; // y cannot be reassigned
```

JS

Spread Syntax

Use the spread syntax (...) to clone arrays and objects, merge arrays, and pass multiple arguments to functions. **Do not** use spread syntax inside a loop as it will create O(n2) time complexity.

index.js

```
// Cloning an array
const original = [1, 2, 3];
const clone = [ ... original];

// Merging arrays
const arr1 = [1, 2, 3];
const arr2 = [4, 5, 6];
const merged = [ ... arr1,  ... arr2];

// Passing multiple arguments
const numbers = [1, 2, 3, 4, 5];
console.log(Math.max( ... numbers)); // Output:
5
```

JS

Array Methods

Take advantage of array methods like map(), filter(), and reduce() for cleaner and more expressive code when working with arrays.

index.js

```
const numbers = [1, 2, 3, 4, 5];

// Using map()
const doubled = numbers.map(num => num * 2);
console.log(doubled); // Output: [2, 4, 6, 8, 10]

// Using filter()
const evenNumbers = numbers.filter(num => num % 2 === 0);
console.log(evenNumbers); // Output: [2, 4]

// Using reduce()
const sum = numbers.reduce((acc, num) => acc + num, 0);
console.log(sum); // Output: 15
```

JS

Arrow functions

Use arrow functions (=>) for concise function definitions, especially for callbacks and anonymous functions.

index.js

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

// Arrow function
const add = (a, b) => a + b;

// Usage
console.log(add(2, 3)); // Output: 5
```

JS

Destructuring

Take advantage of object destructuring to extract values from objects and arrays easily.

index.js

```
// Object destructuring
const person = { name: 'John', age: 25 };
const { name, age } = person;

console.log(name); // Output: John
console.log(age); // Output: 25

// Array destructuring
const numbers = [1, 2, 3];
const [first, second, third] = numbers;

console.log(first); // Output: 1
console.log(second); // Output: 2
console.log(third); // Output: 3
```

FOLLOW