Top 10 Coding Examples and Tips for JavaScript Code

1. Use strict mode to enforce modern JavaScript syntax and catch errors early.
2. Always declare variables with const or let, rather than var.
3. Use arrow functions instead of function for cleaner and concise code.
4. Make use of destructuring to extract values from arrays and objects into variables.
5. Use template literals for string concatenation and embedding expressions.
6. Prefer forEach over for loop for simple iterations.
7. Make use of higher-order functions like map, filter, and reduce to process arrays.
8. Avoid using global variables and always use const or let to scope variables.

9. Use modules to organize your code and avoid naming collisions.
10. Always initialize variables with default values to avoid undefined values.

strict mode example

Use strict mode to enforce modern JavaScript syntax and catch errors early:

```
'use strict';
```

Use const and let

Always declare variables with const or let, rather than var:

```
// Use let
let name = 'John Doe';

// Use const
const PI = 3.14;
```

Use Arrows functions

Use arrow functions instead of function for cleaner and concise code:

```
// Function expression
const multiply = (a, b) => a * b;
```

```
// Implicit return
const square = x => x * x;
```

Use Destructuring to get values from arrays

Make use of destructuring to extract values from arrays and objects into variables:

```
// Destructuring arrays
const colors = ['red', 'green', 'blue'];
const [first, second, third] = colors;

// Destructuring objects
const person = {
  name: 'John Doe',
  age: 30,
  job: 'Software Engineer'
};
const { name, age, job } = person;
```

Use template literals

Use template literals for string concatenation and embedding expressions:

```
const name = 'John Doe';
const message = `Hello, ${name}!`;
```

Use forEach over for loop

Prefer forEach over for loop for simple iterations:

```
const numbers = [1, 2, 3, 4, 5];
numbers.forEach(number => console.log(number));
```

use of higher-order functions

Make use of higher-order functions like map, filter, and reduce to process arrays:

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

// Use map
const double = numbers.map(number => number * 2);

// Use filter
const even = numbers.filter(number => number % 2 ===
0);

// Use reduce
const sum = numbers.reduce((acc, number) => acc +
number, 0);
```

Avoid Global Variables

Avoid using global variables and always use const or let to scope variables:

```
// Global variable (not recommended)
```

```
let name = 'John Doe';

// Scoped variable (recommended)
function sayHello() {
  const name = 'John Doe';
  console.log(`Hello, ${name}!`);
}
```

Avoid Naming Collisions

Use modules to organize your code and avoid naming collisions:

```
// math.js
export const PI = 3.14;
export const add = (a, b) => a + b;

// main.js
import { PI, add } from './math.js';
console.log(PI); // 3.14
console.log(add(1, 2)); // 3
```

Initialize variables with default values

Always initialize variables with default values to avoid undefined values:

```
// Default value
let name = 'John Doe';

// Default value with destructuring
const person = {
```

```
  name = 'John Doe',
  age: 30
};
const { name, age = 0 } = person;
```