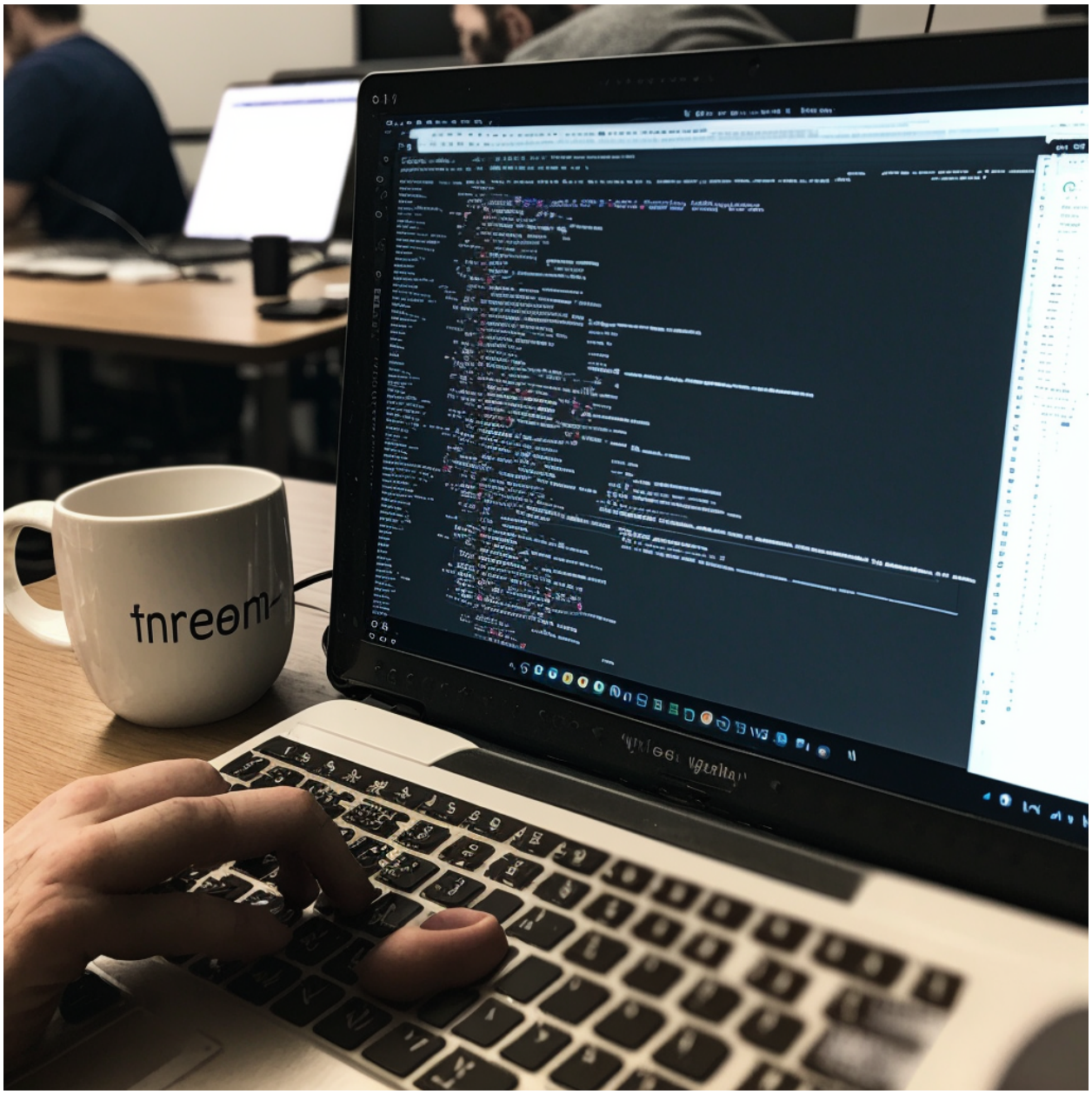
Code JavaScript Examples



Use strict mode: Always use strict mode to enforce better coding practices and catch errors early on. You can enable strict mode by adding "use strict" at the beginning of your code. 2

Avoid using global variables: Using global variables can lead to unexpected behavior and make it harder to debug code. Instead, use variables with limited scope. 3

Use strict mode: Always use strict mode to enforce better coding practices and catch errors early on. You can enable strict mode by adding "use strict" at the beginning of your code.

Example:

```
"use strict";

function myFunction() {
  x = 10; // This will throw an error in strict mode
}
```

Avoid using global variables: Using global variables can lead to unexpected behavior and make it harder to debug code. Instead, use variables with limited scope.

Example:

```
function myFunction() {
  var x = 10; // Declare variables with var, let or
const
  console.log(x);
}
```

Use template literals for string concatenation: Template literals make it easier to concatenate strings and variables.

Example:

```
const name = "John";
console.log(`My name is ${name}`);
```

Use arrow functions: Arrow functions provide a more concise syntax and lexical scoping of the "this" keyword.

Example:

```
const numbers = [1, 2, 3];
const doubledNumbers = numbers.map(num => num * 2);
```

```
console.log(doubledNumbers);
```

Use destructuring: Destructuring allows you to extract values from arrays and objects more easily.

Example:

```
const person = {
  name: "John",
  age: 30,
  country: "USA"
};

const { name, age } = person;
console.log(name, age);
```

Use const and let instead of var: Using const and let instead of var provides better scoping and can prevent redeclaration.

Example:

```
const PI = 3.14; // Declare constants with const
let x = 10; // Declare variables with let
```

Use the spread operator: The spread operator allows you to spread elements of an array or object into another array or object.

Example:

```
const arr1 = [1, 2, 3];
const arr2 = [4, 5, 6];
const combinedArray = [...arr1, ...arr2];
console.log(combinedArray);
```

Use promises: Promises provide a better way to handle asynchronous code and prevent callback hell.

Example:

```
function fetchData() {
  return new Promise((resolve, reject) => {
    // Perform async operation
    if (data) {
      resolve(data);
    } else {
      reject("Error fetching data");
    }
  });
}

fetchData().then(data => console.log(data)).catch(error
=> console.log(error));
```

Use let and const for loops: Using let and const for loop variables prevents unexpected behavior due to variable hoisting.

Example:

```
for (let i = 0; i < 10; i++) {
  console.log(i);
}

for (const element of array) {
  console.log(element);
}
```

Use comments: Comments can help explain code and make it easier to maintain.

Example:

```
// This is a comment

/* This is a
multiline comment */
```

Summary of points:

1. Use strict mode
2. Avoid using global variables
3. Use template literals for string concatenation
4. Use arrow functions
5. Use destructuring
6. Use const and let instead of var

7. Use the spread operator
8. Use promises
9. Use let and const for loops
10. Use comments