ES6

Prevent Object Mutation → JavaScript provides a function `Object.freeze` to prevent data mutation.

Use Arrow Functions to Write Concise Anonymous Functions

```
const myFunc = () => {
  const myVar = "value";
  return myVar;
}

//if there is no function body
const myFunc = () => "value";

//Arrow Functions with Parameters
const doubler = (item) => item * 2;

//Set Default Parameters for Your Functions
const greeting = (name = "Anonymous") => "Hello " + name;
```

Use the Rest Parameter with Function Parameters

```
/*The rest parameter eliminates the need to check the args array and allows us to apply ma
p(), filter() and reduce() on the parameters array.*/
function howMany(...args) {
  return "You have passed " + args.length + " arguments.";
}
```

Use the Spread Operator to Evaluate Arrays In-Place

```
/*We had to use Math.max.apply(null, arr) because Math.max(arr) returns NaN. Math.max() ex
pects comma-separated arguments, but not an array. The spread operator makes this syntax m
uch better to read and maintain.*/
var arr = [6, 89, 3, 45];
var maximus = Math.max.apply(null, arr);

//...arr returns an unpacked array. In other words, it spreads the array.
const arr = [6, 89, 3, 45];
const maximus = Math.max(...arr);
const arrCopy = [...arr]
```

Destructuring assignment is special syntax introduced in ES6, for neatly assigning values taken directly from an object.

Use Destructuring Assignment:

1. **to Extract Values from Object**

```
const user = { name: 'John Doe', age: 34 };
const { name, age } = user;
/*the name and age variables will be created
and assigned the values of their respective values from the user object.*/
```

2. **to Assign Variables from Objects**

```
const { name: userName, age: userAge } = user;
```

3. **to Assign Variables from Nested Objects**

```
const user = {
  johnDoe: {
    age: 34,
    email: 'johnDoe@freeCodeCamp.com'
  }
};
const { johnDoe: { age, email }} = user;
const { johnDoe: { age: userAge, email: userEmail }} = user;
```

4. **to Assign Variables from Arrays**

```
const [a, b] = [1, 2, 3, 4, 5, 6];
console.log(a, b); //The console will display the values of a and b as 1, 2.

const [a, b,,, c] = [1, 2, 3, 4, 5, 6];
console.log(a, b, c); //The console will display the values of a, b, and c as 1, 2, 5.
```

5. **with the Rest Parameter to Reassign Array Elements**

```
const [a, b, ...arr] = [1, 2, 3, 4, 5, 7];
//The result is similar to Array.prototype.slice()
```

6. **to Pass an Object as a Function's Parameters**

```
const profileUpdate = ({ name, age, nationality, location }) => {
}
```

Create Strings using Template Literals → Template literals allow you to create multi-line strings and to use string interpolation features to create strings.

```
const person = {
  name: "Zodiac Hasbro",
  age: 56
};

const greeting = `Hello, my name is ${person.name}!
I am ${person.age} years old.`;

console.log(greeting);
```