ES6 Code Notes for Frontend

chat.openai.com/chat

write ES6 notes with code used in frontend development

Here are some of the essential features of ECMAScript 6 (ES6) used in front-end development:

1. `let` and `const`: `let` is used to declare variables that can be reassigned, while `const` is used to declare variables that can't be reassigned.

javascript

```
let name = "John";
name = "Jane";
console.log(name); // outputs: Jane

const PI = 3.14;
PI = 3.14159; // TypeError: Assignment to constant variable.
```

2. Arrow Functions: Arrow functions are a shorthand for writing anonymous functions and have a shorter syntax compared to regular functions.

javascript

```
const square = (x) => x * x;
console.log(square(5)); // outputs: 25
```

3. Template Literals: Template literals are string literals that allow for expressions to be embedded within a string. They are denoted using backticks (`) instead of quotes (' or ").

```javascript
const name = "John";
console.log(`Hello, ${name}!`); // outputs: Hello, John!
```

4. Destructuring: Destructuring allows you to extract values from arrays or objects and assign them to variables.

```javascript
const person = { name: "John", age: 30 };
const { name, age } = person;
console.log(name, age); // outputs: John 30
```

5. Spread Operator: The spread operator allows you to spread the values of an array or object into a new array or object.

```javascript
const numbers = [1, 2, 3];
const newNumbers = [...numbers, 4, 5];
console.log(newNumbers); // outputs: [1, 2, 3, 4, 5]
```

6. Modules: ES6 introduced the concept of modules, which allows you to export and import values from one module to another.

```javascript
// moduleA.js
export const message = "Hello, World!";

// moduleB.js
import { message } from "./moduleA";
console.log(message); // outputs: Hello, World!
```

These are just a few of the features of ES6, and there is much more to explore!

give more

Sure, here are some additional features of ES6:

7. `class` : The `class` syntax provides a simpler way to create objects and handle inheritance in JavaScript.

javascript

```javascript
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }
  greet() {
    console.log(`Hello, I'm ${this.name}.`);
  }
}

const person = new Person("John", 30);
person.greet(); // outputs: Hello, I'm John.
```

8. Promises: Promises are objects representing the eventual completion or failure of an asynchronous operation.

```javascript
const fetchData = () => {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve({ data: "This is some data." });
    }, 1000);
  });
};

fetchData().then((data) => console.log(data)); // outputs: { data: "This is some data." }
```

9. Generators: Generators are functions that can be paused and resumed, allowing you to create iterators that generate a sequence of values.

```javascript
function* getNumbers() {
  yield 1;
  yield 2;
  yield 3;
}

const numbers = getNumbers();
console.log(numbers.next().value); // outputs: 1
console.log(numbers.next().value); // outputs: 2
console.log(numbers.next().value); // outputs: 3
```

10. `async` / `await` : The `async` / `await` syntax provides a way to write asynchronous code that looks and behaves like synchronous code.

```
javascript


const fetchData = () => {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve({ data: "This is some data." });
    }, 1000);
  });
};

const getData = async () => {
  const data = await fetchData();
  console.log(data); // outputs: { data: "This is some data." }
};

getData();
```

These are just a few more of the powerful features of ES6, and there's even more to discover!