Emoji Support Test 🎉

Headings with Emoji 🚀

Mixed text and emoji ✨

Hello 🎉🔥 world! This paragraph mixes regular text with emoji characters.

Here is a **bold section with** 💪 **emoji** and *italic with* 🌟 *stars*.

- 📋 First item with emoji
- 🎯 Second item with emoji
- Plain item with trailing emoji 🏆

1. 🥇 Gold
2. 🥈 Silver
3. 🥉 Bronze

Feature	Status
Emoji in headings	✅
Emoji in paragraphs	✅
Emoji in tables	✅
Emoji in lists	✅

> *A blockquote with emoji:* 💬 *Wise words here.*

Some flags: 🇺🇸 🇬🇧 🇯🇵

ZWJ sequences: 👨‍👩‍👧‍👦 👩‍💻 🏳️‍🌈

Skin tones: 👋🏻 👋🏼 👋🏽 👋🏾 👋🏿

Code Blocks

Here is some inline code: `const x = 42;` and `console.log("hello")`.

TypeScript

```typescript
 1  import { readFile } from 'fs/promises';
 2
 3  interface Config {
 4    name: string;
 5    port: number;
 6    debug: boolean;
 7  }
 8
 9  async function loadConfig(path: string): Promise<Config> {
10    const raw = await readFile(path, 'utf-8');
11    const config: Config = JSON.parse(raw);
12    // Validate required fields
13    if (!config.name || config.port <= 0) {
14      throw new Error(`Invalid config: ${config.name}`);
15    }
16    return config;
17  }
18
19  const DEFAULT_PORT = 3000;
20  export { loadConfig, DEFAULT_PORT };
```

JavaScript

```javascript
1  class EventEmitter {
2    constructor() {
3      this.listeners = new Map();
4    }
5
6    on(event, callback) {
7      if (!this.listeners.has(event)) {
8        this.listeners.set(event, []);
9      }
10     this.listeners.get(event).push(callback);
11     return this;
12   }
13
14   emit(event, ...args) {
15     const handlers = this.listeners.get(event) || [];
16     for (const handler of handlers) {
17       handler(...args);
18     }
19     return handlers.length > 0;
20   }
21 }
22
23 // Usage
24 const emitter = new EventEmitter();
25 emitter.on('data', (msg) => console.log(`Received: ${msg}`));
26 emitter.emit('data', 'Hello World!');
```

Python

```python
from dataclasses import dataclass
from typing import Optional

@dataclass
class User:
    name: str
    email: str
    age: Optional[int] = None

    def greet(self) -> str:
        """Return a greeting message."""
        return f"Hello, {self.name}!"

    @property
    def is_adult(self) -> bool:
        return self.age is not None and self.age >= 18

# Create and use
users = [User("Alice", "alice@example.com", 30), User("Bob", "bob@example.com")]
active = [u for u in users if u.is_adult]
print(f"Found {len(active)} adult users")
```