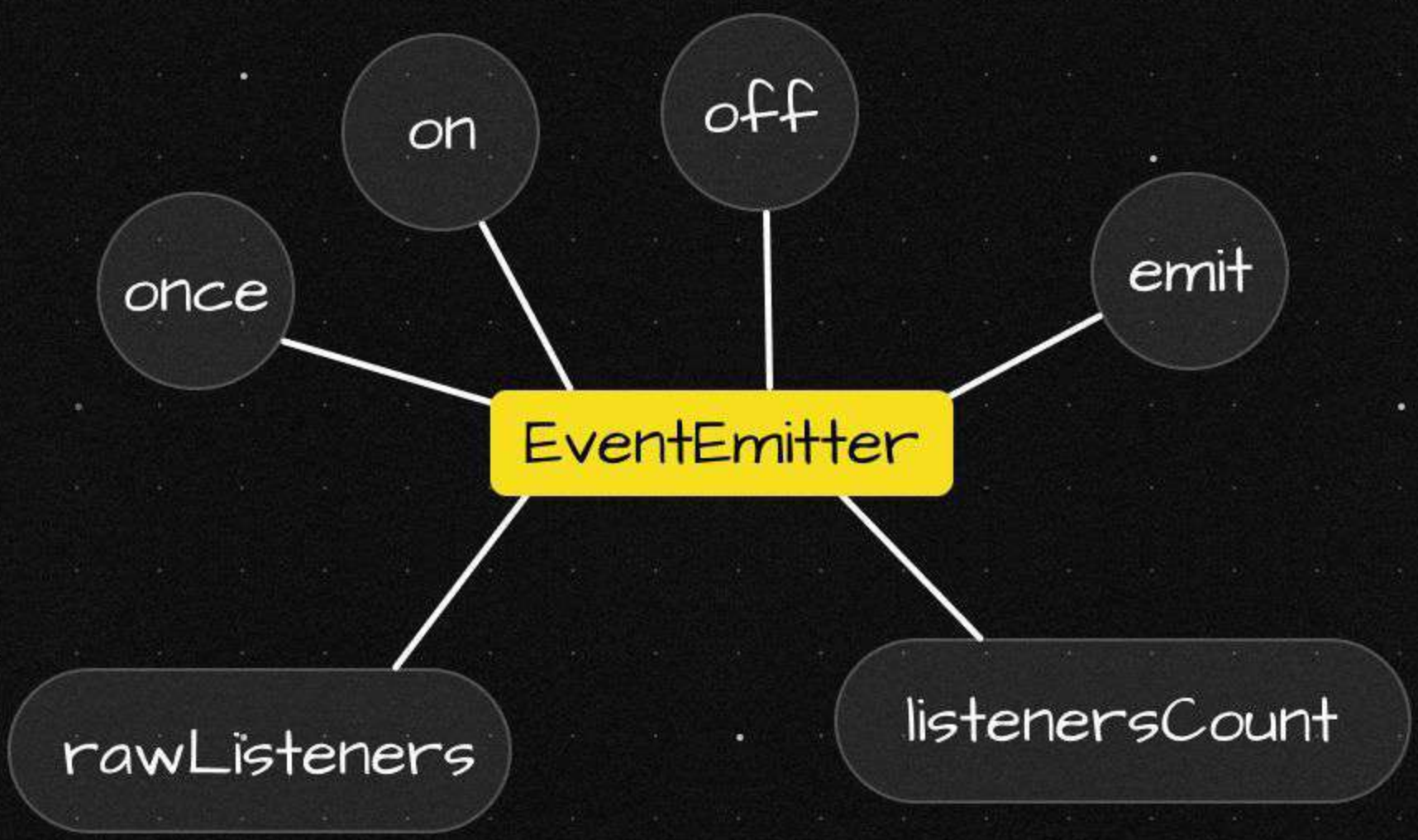
JS

Event Emitters

in JavaScript



patreon.com/CodewithSloba

Swipe →

Common Use cases

- **Tight Coupling**
  Without Emitters, direct function calls create rigid dependencies.

```
fetchData(() => updateUI());   // Tight coupling

dataEmitter.on('fetch', updateUI);
fetchData();
dataEmitter.emit('fetch');     // Decoupled logic
```

- **Async Complexity**: Simplify handling of asynchronous events (e.g., data arrival, errors).
- **Memory Leaks**: Explicit listener removal prevents leaks in long-running apps.
- **Unhandled Errors**: Centralized error listeners avoid silent failures.

Swipe →

Create a Custom Event Emitter

Implement a basic event emitter class for core functionality.

```
class EventEmitter {
  constructor() {
    this.events = {};
  }

  on(event, listener) {
    if (!this.events[event]) this.events[event] = [];
    this.events[event].push(listener);
  }

  emit(event, ...args) {
    if (this.events[event]) {
      this.events[event].forEach(listener => listener(...args));
    }
  }

  off(event, listener) {
    if (this.events[event]) {
      this.events[event] = this.events[event].filter(l => l !== listener);
    }
  }
}
```

Swipe →

Subscribe and Emit Events

Subscribe and Emit Events

```
const emitter = new EventEmitter();

// Subscribe
emitter.on('data', (chunk) => console.log(`Received: ${chunk}`));

// Emit
emitter.emit('data', 'Sample Data'); // Logs "Received: Sample Data"
```

Handle Errors Gracefully

Always include error listeners to avoid crashes.

```
emitter.on('error', (err) => console.error('Error:', err.message));

// Emit error
emitter.emit('error', new Error('Failed to fetch data'));
```

Swipe →

Remove Listeners to Prevent Leaks

Avoid memory leaks by cleaning up unused listeners.

```
const logData = (data) => console.log(data);

emitter.on('data', logData);
emitter.off('data', logData); // Remove specific listener
```

Prevent Memory Leaks

Monitor listener counts and enforce limits.

```
// Node.js example
myEmitter.setMaxListeners(10); // Warn if >10 listeners
console.log(myEmitter.getMaxListeners()); // Check limit
```

Swipe →

Use the once() Method

Execute a listener only once.

```
class EventEmitter {
  // ...
  once(event, listener) {
    const wrapper = (...args) => {
      listener(...args);
      this.off(event, wrapper);
    };
    this.on(event, wrapper);
  }
}

emitter.once('connect', () => console.log('Connected!'));
emitter.emit('connect'); // Logs "Connected!"
emitter.emit('connect'); // No output
```

Swipe →

Leverage Node.js Built-in EventEmitter

Use the native module for production-ready code.

```
const EventEmitter = require('events');
class MyEmitter extends EventEmitter {}

const myEmitter = new MyEmitter();
myEmitter.on('event', () => console.log('Event fired!'));
myEmitter.emit('event');
```

Swipe →