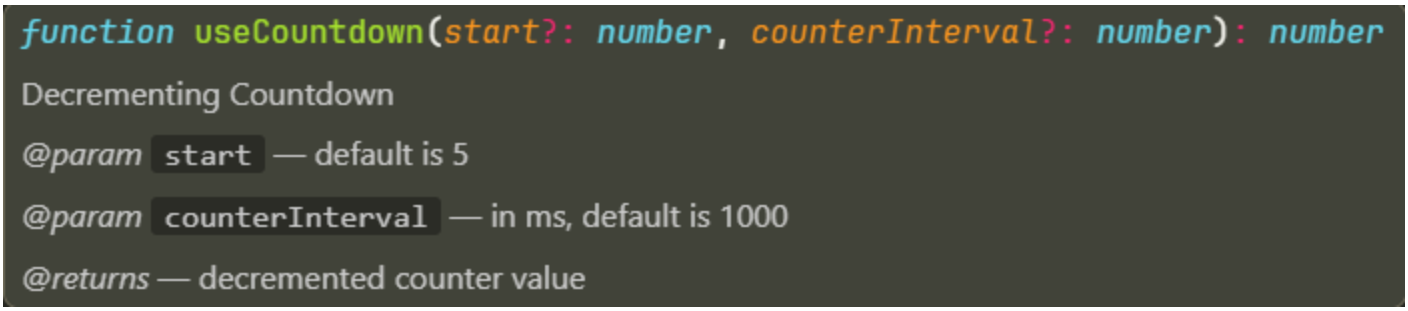
JavaScript mit Typsicherheit - JSDoc

Vielen Entwicklern ist TypeScript zu strikt oder schlichtweg zu akademisch gedacht. Die simple Verwendung von JSDoc in Kombination mit JavaScript bietet hier eine einfache Alternaive, die ohne zusätzliches Setup in modernen Anwendungen genutzt werden kann.

Warum sollte man JSDoc nutzen ?

JSDoc bietet die Möglichkeit, ältere Codebasen zu modernisieren, ohne das Diese einen kompletten Rewrite unterzogen werden müssen durch TypeScript. Darüber hinaus bricht JSDoc den natürlichen dynamischen Codefluss von JavaScript nicht, sondern "ergänzt" diesen in Kombination mit `//@ts-check` durch simples Type-Checking.

Die Regeln der Anwendung sind denkbar einfach und JavaScript Codebasen profitieren bei der Verwendung von JSDoc in Form besserer Autovervollständigung und der Qualität aufkommender Fehlermeldungen in der IDE.
Zudem beschreibt JSDoc den genauen Aufbau und die Verwendung einer Funktion, wodurch diese besser verständlich und wesentlich einfacher nutzbar wird.



Der entsprechende JSDoc-Kommentar dazu sieht wie folgt aus:

```
/**
 * Decrementing Countdown
 *
 * @param {number} start default is 5
 * @param {number} counterInterval in ms, default is 1000
 * @returns decremented counter value
 */
```

Die Funktion selbst bleibt dabei in gewöhnlichem JavaScript geschrieben:

```
export function useCountdown(start = 5, counterInterval = 1000) {
 const [count, setCount] = useState(start);

 useEffect(() => {
   // create timer
```

```
    const timer1 = setInterval(() => {
      setCount((c) => c - 1);
    }, counterInterval);

    // clear timer, otherwhise multiple timers are running
    return () => clearInterval(timer1);
  }, []);

  return count;
}
```

Was genau ist JSDoc ?

Bei JSDoc handelt es sich im Kern um ein reines Dokumentationsformat für JavaScript.
Das bedeutet, nur in Kombination mit dem TypeScript-Typechecker steht die Funktionalität der Typüberprüfung zur Verfügung.

Nun stellt sich dem Einen oder Anderen die Frage, warum man denn nicht direkt TypeScript verwenden sollte, wenn man doch sowieso nur typsicher mit dem TypeScript-Typechecker arbeiten kann und die Frage ist durchaus angebracht, aber leicht zu beantworten. TypeScript ist komplex, erfordert einiges an Übung im Umgang und der Anwendung und ist äußerst strikt. Diese Striktheit macht für enorm umfangreiche Enterpriseanwendungen durchaus Sinn, hat jedoch ihre Schattenseiten ! Ganze Foren und Social Media Threads sind voll mit Entwicklern, die TypeScript selbst als "unintuitiv" oder gar als "abstoßend" beschreiben und ihre Ansichten mit allerlei Beispielen untermauern. Diese Kritik ist absolut berechtigt und kommt keinesfalls von ungefähr.

Genau für diese Entwickler bietet die Kombination aus JSDoc zur Beschreibung des darunter befindlichen JavaScript-Codes und der Verwendung des TypeScript-Typechecker durch die Angabe von `//@ts-check` am Anfang der entsprechenden Datei einen Sweetspot zwischen Sicherheit und Einfachheit.
Es gilt jedoch zu beachten, das es sich weiterhin um reguläres JavaScript handelt und im Gegensatz zur Verwendung von TypeScript es keinen expliziten Compiler gibt der bei einem Typfehler die Ausführung des Codes verhindert !