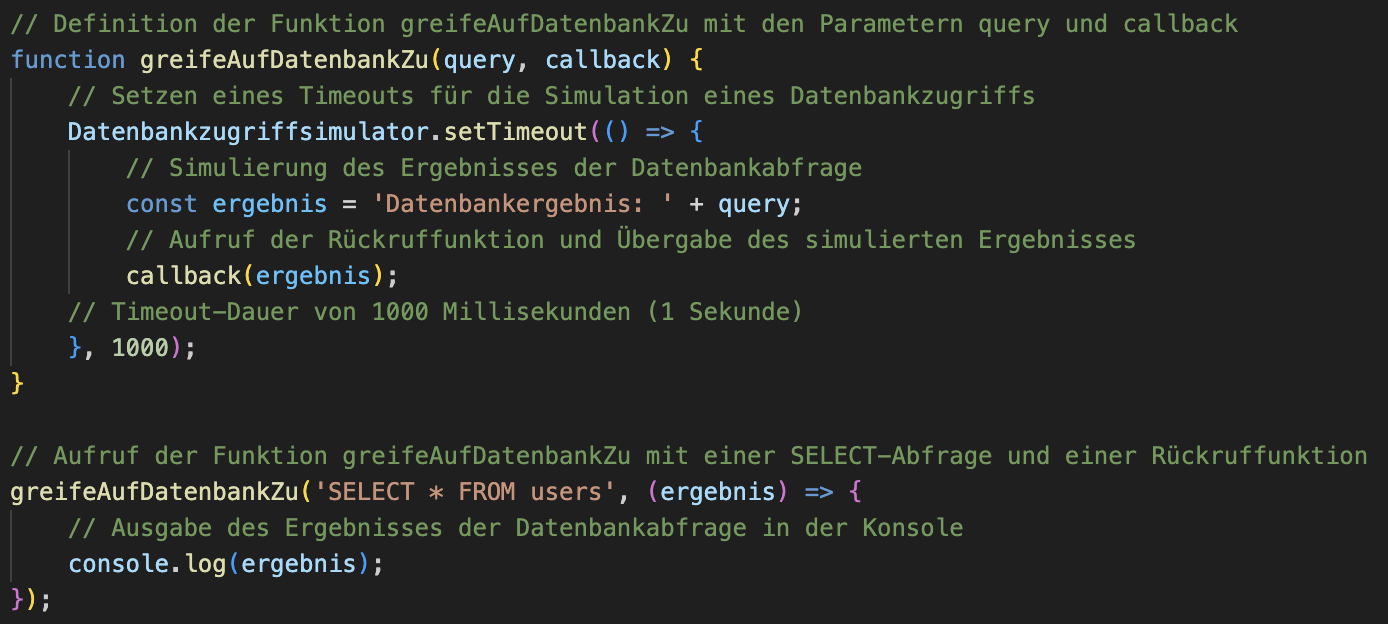
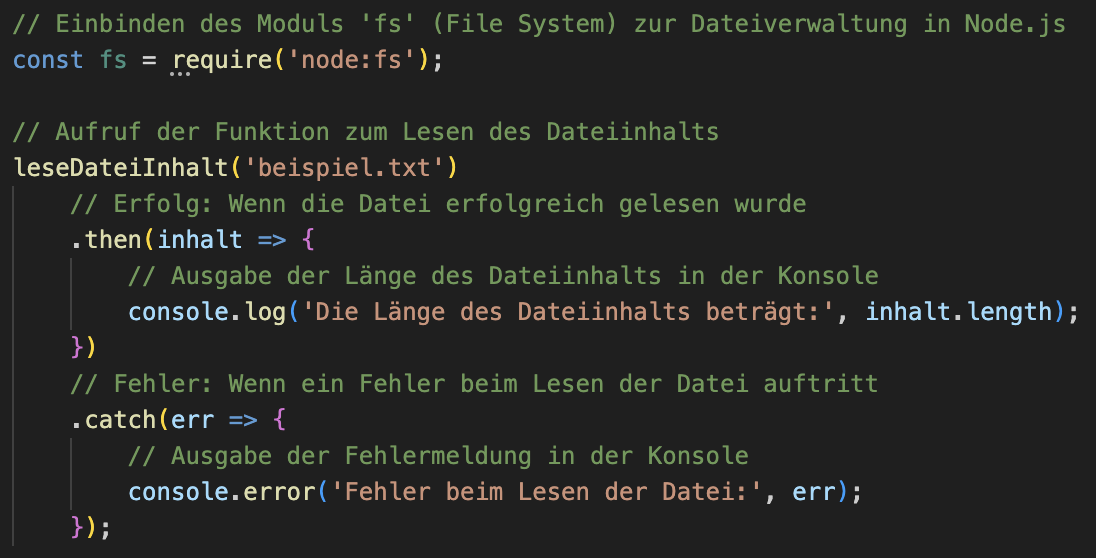
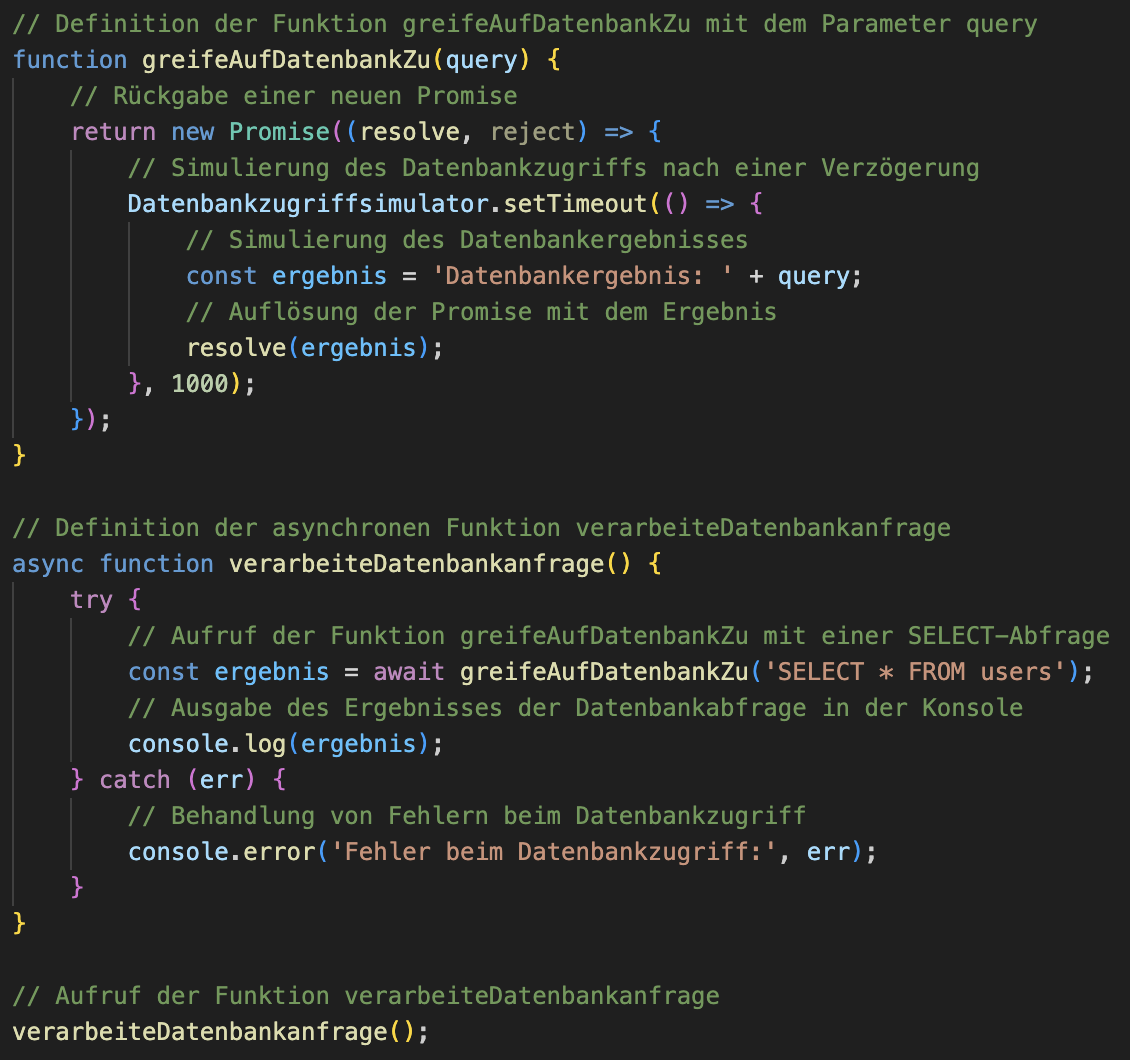
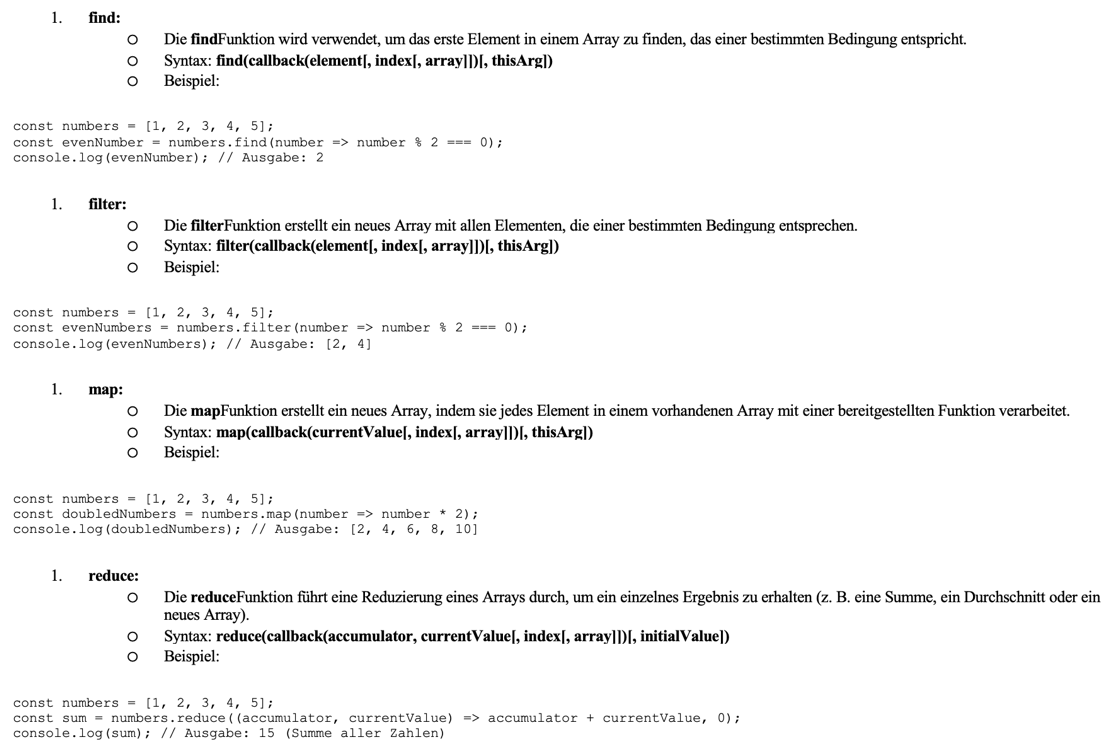
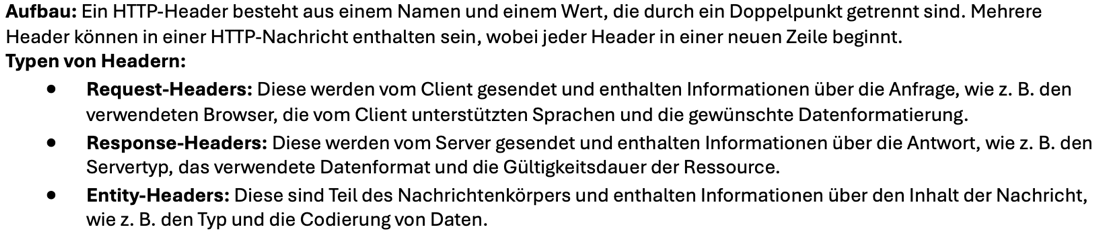
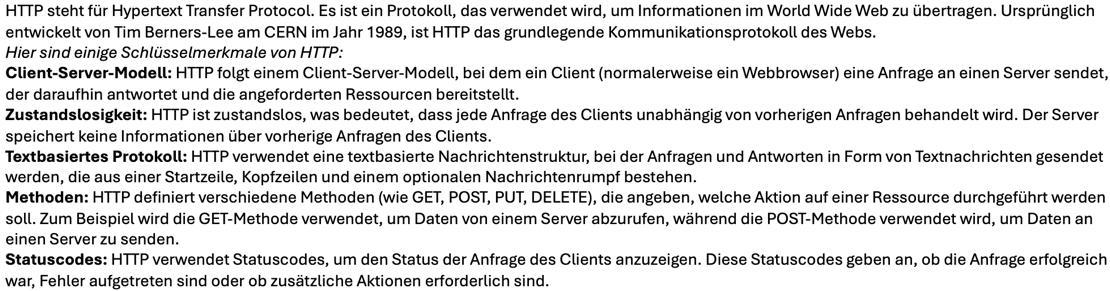
Callbacks   


Promises



Async/Await





HTTP-Schnittstellen:

**Address:** Die eindeutige URL (Uniform Resource Locator), die eine Ressource oder einen Endpunkt im Web identifiziert, zu dem die HTTP-Anfrage gesendet wird (z.B. <https://example.com/api/users>).

**Headers:** Schlüssel-Wert-Paare, die zusätzliche Informationen über den HTTP-Anforderer und die Anfrage selbst enthalten (z.B. Content-Type, Authorization).

**Methods:** HTTP-Verben, die angeben, welche Aktion auf der angeforderten Ressource durchgeführt werden soll (z.B. GET, POST, PUT, DELETE).

**Body:** Daten, die bei bestimmten HTTP-Methoden (z.B. POST, PUT) an den Server gesendet werden, um Ressourcen zu erstellen oder zu aktualisieren.

**Status Codes:** Dreistellige Zahlen, die das Ergebnis der HTTP-Anfrage anzeigen (z.B. 200 OK, 404 Not Found, 500 Internal Server Error).

**Callbacks:**

Callbacks sind Funktionen, die als Argumente an andere Funktionen übergeben werden und später ausgeführt werden, sobald eine asynchrone Operation abgeschlossen ist.

Sie wurden häufig verwendet, um asynchrone Operationen in JavaScript zu verarbeiten, können jedoch zu sogenanntem "Callback-Hell" führen, wenn sie stark verschachtelt sind.

**Promises:**

Promises sind ein Konzept, das es ermöglicht, asynchrone Operationen in JavaScript effektiver zu verwalten.

Sie repräsentieren den Abschluss oder den Fehlschlag einer asynchronen Operation und ermöglichen es, darauf zu warten und dann darauf zu reagieren, wenn die Operation abgeschlossen ist.

**async/await:**

async/await ist eine syntaktische Verbesserung in JavaScript, die auf Promises basiert und die Arbeit mit asynchronem Code erleichtert.

Die **async**-Kennzeichnung vor einer Funktion gibt an, dass sie asynchronen Code enthält und eine Promise zurückgibt.

Das **await**-Schlüsselwort kann innerhalb einer als **async** markierten Funktion verwendet werden, um auf das Ergebnis einer Promise zu warten, ohne die Ausführung des Codes zu blockieren.

Node.js ist eine Open-Source-Plattform für die serverseitige Entwicklung von JavaScript-Anwendungen. Hier sind einige wichtige Punkte zu Node.js:

1. **JavaScript auf dem Server:** Node.js ermöglicht es Entwicklern, serverseitige Anwendungen mit JavaScript zu erstellen, einer Sprache, die normalerweise im Browser für die clientseitige Entwicklung verwendet wird.
2. **Eventgetriebene Architektur:** Node.js verwendet eine ereignisgesteuerte, nicht blockierende Eingabe-/Ausgabe-Architektur, die es ermöglicht, mehrere gleichzeitige Verbindungen effizient zu handhaben. Dadurch eignet es sich besonders gut für Echtzeitanwendungen wie Websockets und Chat-Anwendungen.
3. **V8 JavaScript Engine:** Node.js basiert auf der V8-JavaScript-Engine von Google, die auch in Chrome verwendet wird. Diese Engine ermöglicht eine schnelle Ausführung von JavaScript-Code.
4. **NPM (Node Package Manager):** Node.js wird mit einem integrierten Paketverwaltungssystem namens npm geliefert, das es Entwicklern ermöglicht, Bibliotheken und Pakete von Drittanbietern einfach zu installieren, zu verwalten und in ihren Projekten zu verwenden.
5. **Modularität:** Node.js fördert die Modularität und Wiederverwendbarkeit von Code durch das Konzept der Module. Entwickler können Funktionen und Features in separate Module aufteilen und diese in ihren Anwendungen verwenden.
6. **Skalierbarkeit:** Durch die nicht blockierende Eingabe-/Ausgabe-Architektur und die effiziente Verarbeitung von Ereignissen ermöglicht Node.js die Entwicklung skalierbarer Anwendungen, die mit einem hohen Datenaufkommen umgehen können.