Table of Contents

Ich erkläre dir nun die `index.js` Datei Schritt für Schritt, einfach und verständlich für Anfänger:

🟢 Schritt 1: Importieren von benötigten Modulen

```js
import express from 'express';
import path from 'path';
import { fileURLToPath } from 'url';
import PDFDocument from 'pdfkit';
```

Was passiert hier?

- `express` : Ein Web-Framework, das hilft, Server und Routen zu erstellen.

- `path` : Hilft dabei, Dateipfade (wie den Ort der HTML-Dateien) zu verwalten.
- `fileURLToPath` : Wandelt URLs von Dateien in Dateipfade um (notwendig, wenn du Module verwendest).
- `pdfkit` : Eine Bibliothek zum Erstellen von PDFs mit JavaScript.

🟢 Schritt 2: Vorbereitung von Variablen für Pfade

```js
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
```

Was passiert hier?

- `__filename` : Speichert den aktuellen Dateipfad.
- `__dirname` : Speichert den Ordnerpfad der aktuellen Datei.

🟢 Schritt 3: Webserver erstellen

```js
const app = express();
```

Was passiert hier?

- Eine neue Express-Anwendung (ein Server) wird erstellt und in der Variable `app` gespeichert.

🟢 Schritt 4: Middleware hinzufügen (Vorverarbeitung von Anfragen)

```js
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(express.static('public'));
```

Was passiert hier?

- `express.json()` : Ermöglicht es, JSON-Daten aus dem Client (z.B. Webseite) auszulesen.
- `express.urlencoded({ extended: true })` : Ermöglicht es, Formulardaten auszulesen.
- `express.static('public')` : Ermöglicht es dem Server, statische Dateien wie HTML, CSS oder Bilder aus dem Ordner `public` auszuliefern.

🟢 Schritt 5: Startseite bereitstellen (GET-Anfrage)

```js
app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'index.html'));
});
```

Was passiert hier?

- Wenn jemand deine Webseite besucht (`GET /`), liefert der Server die Datei `index.html` .

🟢 Schritt 6: PDF erzeugen (POST-Anfrage)

```js
app.post('/generate-pdf', (req, res) => { ... });
```

Was passiert hier?

- Sobald jemand auf den Button „PDF erstellen “ klickt, sendet das Formular eine POST-Anfrage an `/generate-pdf` .

Erklärung des Codes im Detail:

- Werte auslesen:

```js
const inputText = req.body.text;
const headerText = req.body.header || "PDF.TEXT - Ihr Dokument";
const fontSize = parseInt(req.body.fontSize) || 12;
```

Hier werden Text, Überschrift und Schriftgröße aus dem Formular gelesen.

- Eingabe validieren:

```js
if (!inputText || inputText.trim() === "") {
    res.status(400).send("Der eingegebene Text darf nicht leer sein.");
    return;
}
```

Wenn der Text leer ist, wird eine Fehlermeldung zurückgegeben.

- Neues PDF-Dokument erstellen:

```js
const doc = new PDFDocument({ margin: 50 });
res.setHeader('Content-Disposition', 'attachment; filename="download.pdf"');
res.setHeader('Content-Type', 'application/pdf');
doc.pipe(res);
```

Erstellt ein PDF-Dokument und sendet es an den Benutzer.

- Kopfzeile hinzufügen:

```js
doc.fontSize(18)
   .text(headerText, { align: "center" })
   .moveDown();
```

Erstellt eine Überschrift für das PDF.

- Linie unter Kopfzeile ziehen:

```js
doc.moveTo(50, 80)
   .lineTo(doc.page.width - 50, 80)
   .stroke();
```

- Haupttext hinzufügen:

```js
doc.moveDown().fontSize(fontSize).text(inputText, { align: "left" });
```

- Fußzeile (Seitenzahl) hinzufügen:

```js
const addFooter = (doc) => {
  doc.fontSize(10)
     .text("Seite " + doc.page.number, 50, doc.page.height - 50, {
       align: "center",
       width: doc.page.width - 100
     });
};
```

Diese Funktion fügt eine Seitenzahl unten auf jeder Seite hinzu.

- Fußzeile bei jeder neuen Seite einfügen:

```js
addFooter(doc);
doc.on('pageAdded', () => {
    addFooter(doc);
});
```

- PDF-Dokument abschließen:

```js
doc.end();
```

- Fehlerbehandlung (falls beim Erstellen Probleme auftreten):

```js
} catch (error) {
  console.error("Fehler bei der PDF-Erstellung:", error);
  res.status(500).send("Interner Serverfehler bei der PDF-Erstellung.");
}
```

🟢 Schritt 7: Server starten

```js
app.listen(3000, () => {
    console.log('Server läuft auf Port 3000');
```

```
});
```

Was passiert hier?

- Der Server startet und wartet auf Anfragen unter dem Port `3000`.

🔍 Zusammenfassung der Funktionsweise

1. Der Nutzer öffnet die Webseite und sieht das Formular (`index.html`).
2. Wenn das Formular abgeschickt wird, sendet es eine POST-Anfrage an den Server.
3. Der Server verarbeitet die Anfrage, erzeugt ein PDF mit den gewünschten Inhalten und gibt es zurück.
4. Der Browser bietet dem Nutzer den PDF-Download an.

📚 Wichtige Begriffe im Überblick

- **Express:** Ein Framework zur einfachen Erstellung eines Webservers.
- **Middleware:** Funktionen, die jede Anfrage durchlaufen, bevor die eigentliche Verarbeitung startet.
- **Routing:** Das Weiterleiten von Anfragen an bestimmte Funktionen (wie bei `/generate-pdf`).
- **PDFKit:** Eine JavaScript-Bibliothek zur dynamischen Erstellung von PDFs.

Ich hoffe, dass diese Erklärung dir hilft, den Code besser zu verstehen! Wenn du Fragen zu einzelnen Punkten hast, lass es mich gerne wissen.