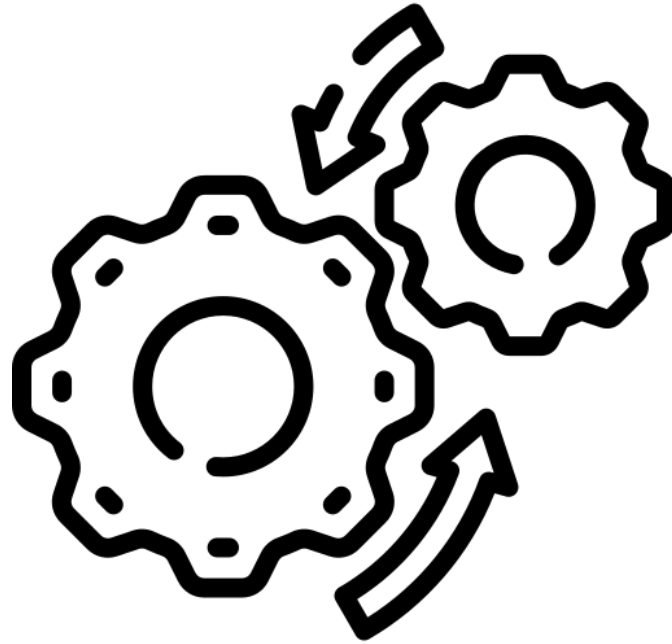
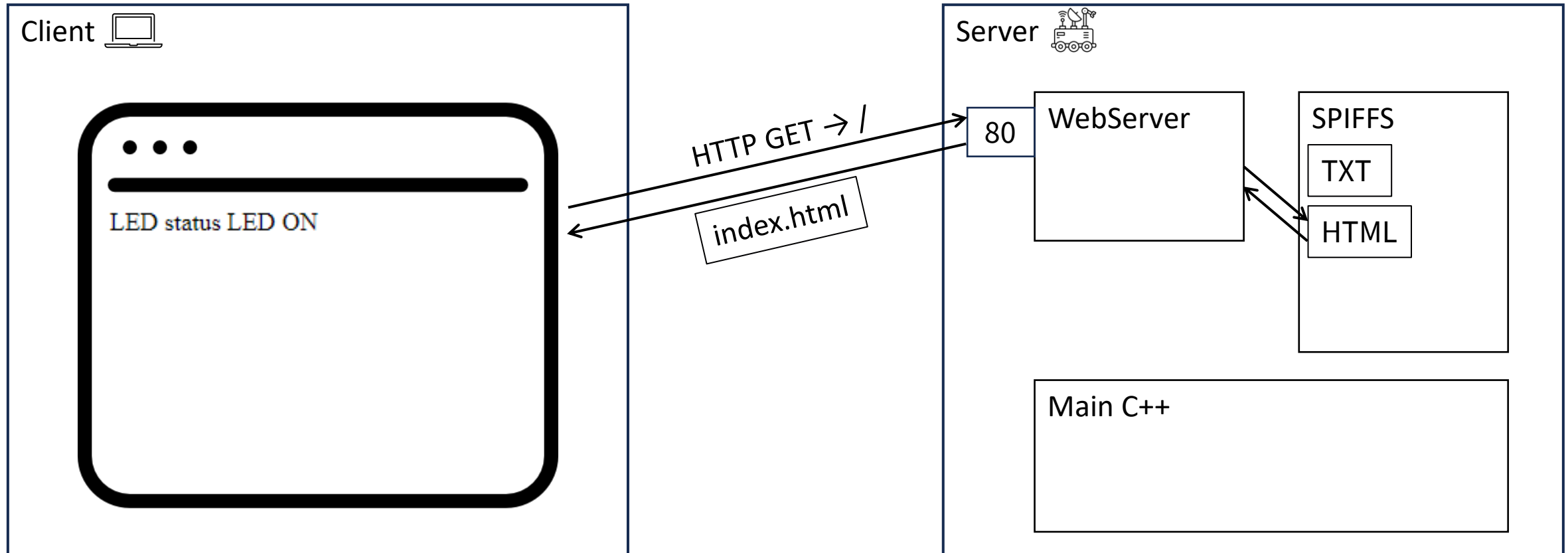


WebServer – Dynamic Content



Server Architektur



WebServer – server.on() (1)

- Setup → server.on();
 - URI → Eingabepfad der Verarbeitet werden soll
 - RequestMethod → HTTP Verb (z.B. HTTP_GET)
 - ArRequestHandlerFunction → Funktion zur Verarbeitung der Clientanfrage
- Beispiel:

```
01. server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {  
02.     request->send(SPIFFS, "/index.html", "text/html", false, templateHandler);  
03. });
```

WebServer – server.on() (2)

```
01. server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {  
02.     request->send(SPIFFS, "/index.html", "text/html", false, templateHandler);  
03. });
```

- "/" → jeder Aufruf des Webservers ohne weitere Pfadangabe (WebServer root)
- HTTP_GET → bei Verwendung eines HTTP GET Aufrufs
- Lambda Funktion zur Verarbeitung des Aufrufs (Request Objekt)

Lambda Funktionen (1)

- Anonyme Funktion für kurzen Code, der nicht wiederverwendet wird

- Syntax:

```
[capture clause] (parameters) mutable -> return-type {  
    body  
}
```

- Verwendung seit C++ 11
- Dokumentation: <https://en.cppreference.com/w/cpp/language/lambda>

Lambda Funktionen (2)

- **Capture Clause (optional)** → beschreibt welche Variablen von außerhalb, in der Funktion verwendet werden können
 - [] Empty Capture → keine Variablen von außen sind in der Funktion verfügbar (Standardverhalten)
 - [=] Capture by Value → Variablen die innerhalb der Funktion verwendet werden, werden in den scope kopiert
 - [&] Capture by Reference → Variablen die innerhalb der Funktion verändert werden, ändern auch außerhalb der Funktion ihren Wert
 - Mixed Capture → Spezifische Variablen mit Angabe ob Capture by Value, oder Capture by Reference (z.B. [a, &b])

Lambda Funktionen (3)

- **Parameter (optional)** → Liste von Parametern die an die Lambda Funktion übergeben werden (siehe normale Funktionen)
- **Mutable (optional)** → Schlüsselwort bei Capture by Value, falls capture Variablen in der Funktion verändert werden sollen
- **Return type (optional)** → Datentyp des Rückgabewerts der Lambda Funktion (meistens automatisch bestimmt)
- **Body** → Code der in der Lambdafunktion ausgeführt wird

WebServer – server.on() (3)

```
01. server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {  
02.     request->send(SPIFFS, "/index.html", "text/html", false, templateHandler);  
03. });
```

- Lambda Funktion
 - [] → Empty Capture (keine Variablen von außen)
 - (AsyncWebServerRequest *request) → Funktion bekommt request Objekt als Parameter übergeben
 - Funktion schickt Antwort auf request mit send() Methode

WebServer – server.on() (4)

```
01. server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {  
02.     request->send(SPIFFS, "/index.html", "text/html", false, templateHandler);  
03. });
```

- Request→send()
 - SPIFFS → Filesystem von dem eine Datei geladen werden soll
 - "/index.html" → Datei die gesendet werden soll
 - "text/html" → MIME-type der zu sendenden Datei
 - false → Datei nicht als Download senden, sondern im Browser rendern
 - templateHandler → eigene Funktion zur Verarbeitung von Templates

WebServer – Template Engine



- ESPAsyncWebserver bietet eine einfache Form einer Template Engine
- Platzhalter können, vor senden eines Response, mit tatsächlichen Werten ersetzt werden
- Platzhalter der im Code mit % Zeichen markiert (z.B. %PLACEHOLDER%)
- Platzhalter werden aus Response extrahiert und an, vom Programmierer bereitgestellte, Funktion übergeben

WebServer – Template Engine

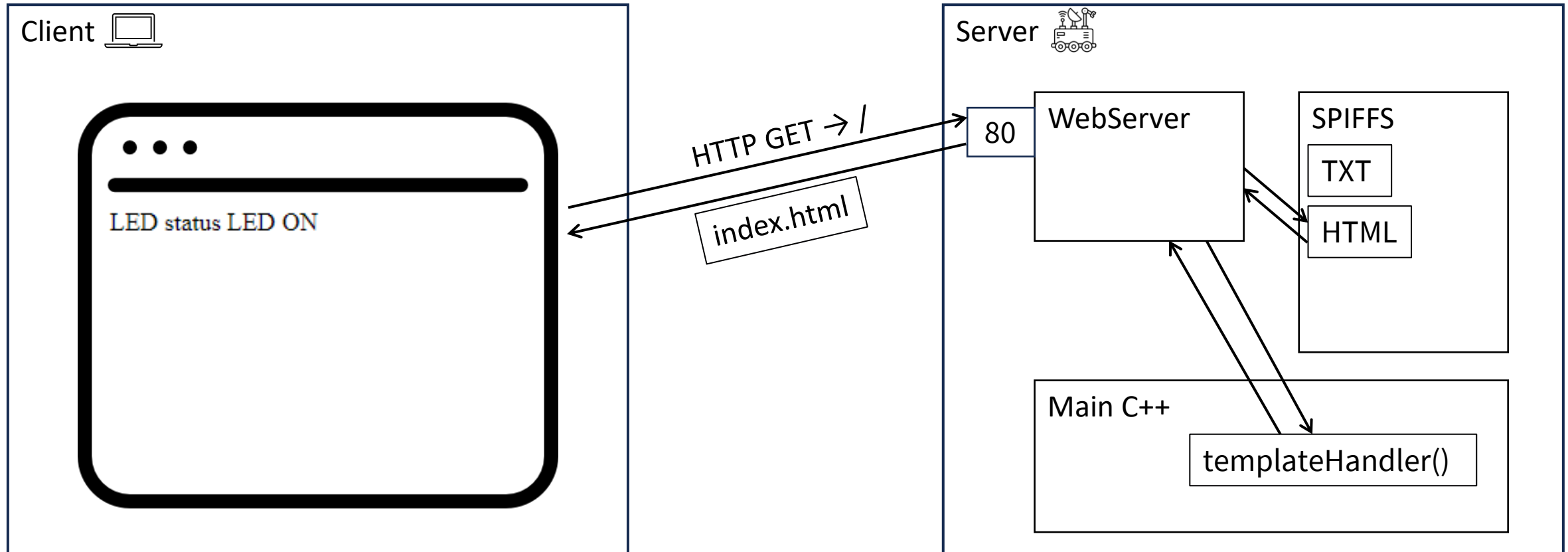
index.html

```
01. <!DOCTYPE html>
02. <html lang="en">
03.   <head>
04.     <title>ESP32 Main Page</title>
05.   </head>
06.   <body>
07.     <div>
08.       LED status %LED_STATE%
09.     </div>
10.   </body>
11. </html>
```

Template processing Funktion in main.cpp

```
01. String templateHandler(const String& var) {
02.   String processedVar = "";
03.
04.   if (var == "LED_STATE") {
05.     if (ledState)
06.       processedVar = "LED ON";
07.     else
08.       processedVar = "LED OFF";
09.   }
10.
11.   return processedVar;
12. }
```

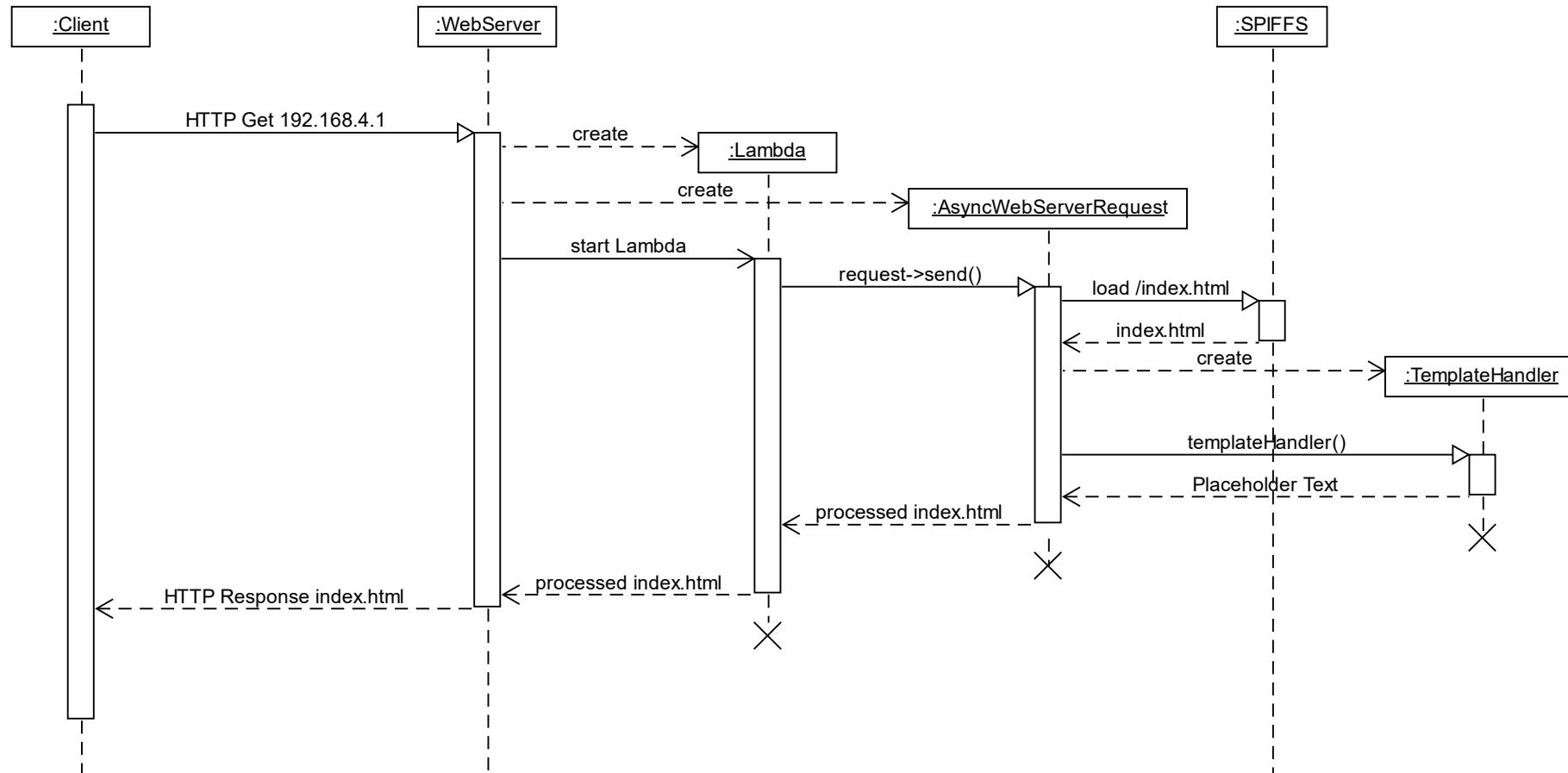
WebServer – Ablauf



WebServer – Ablauf

1. Client Aufruf von „http://192.168.4.1/“
2. WebServer.on() Methode wird aufgerufen
3. Lambda Funktion wird mit request Objekt aufgerufen
4. Request→send() lädt Datei von SPIFFS
5. TemplateHandler() Funktion wird aufgerufen
6. TemplateHandler() ersetzt Platzhalter im response
7. Response wird an Client gesendet
8. Client Browser rendert Webseite

UML Sequenzdiagramm



HTML



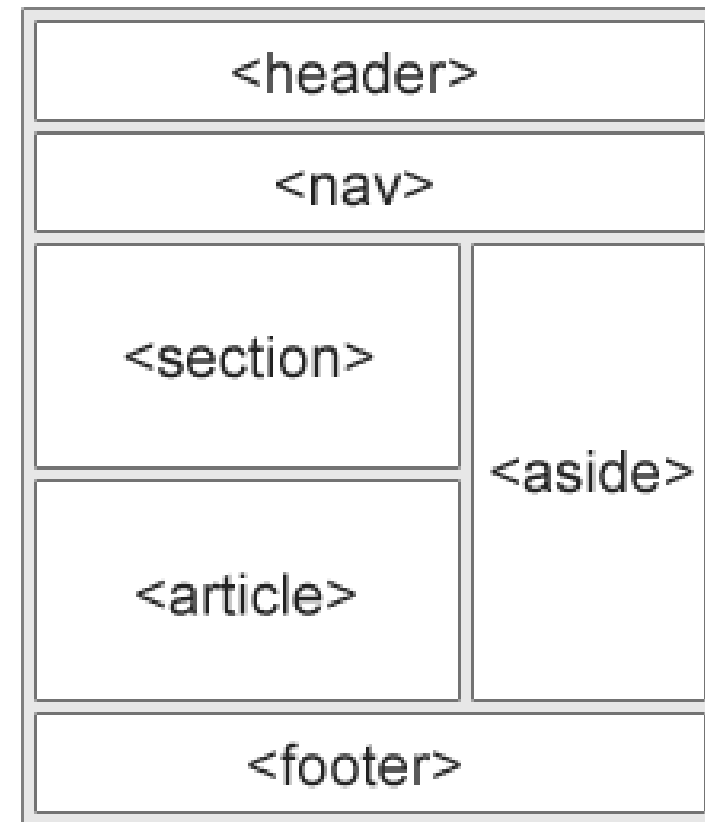
HTL ST. PÖLTEN

HTML



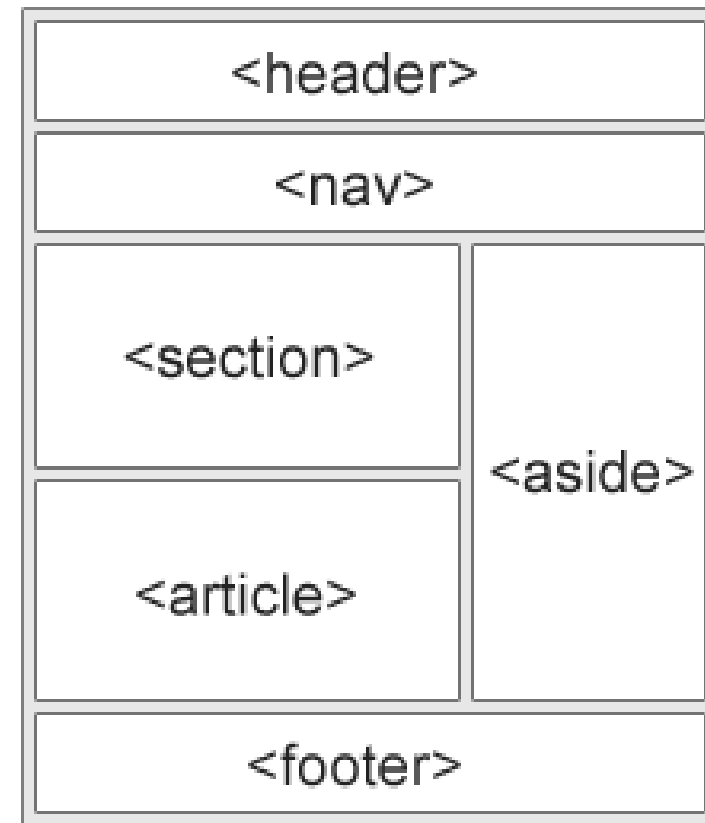
Semantik Web – HTML (1)

- Elemente mit Bedeutung, die von Systemen verarbeitet werden können (z.B. Screenreader)
- **header** → Container für Allgemeine Seiteninformationen (Logo, Titel, ...)
- **nav** → Navigationsleite mit Links zu Subseiten



Semantik Web – HTML (2)

- **section/article** → Bereich zur Gliederung von Seiteninhalten (Hauptcontent)
- **aside** → Inhalt der neben dem Hauptinhalt dargestellt werden soll (Sidebar)
- **footer** → Fußzeile der Seite (meißt Impressum, Kontakt, ...)



HTML Styleguide



- Doctype am Beginn des Dokuments
- Lowercase Tag-Namen und Attribute
- Tag immer schließen
- Einrückung wenn Tags verschachtelt sind
- Viewport angeben
- ...
- Dokumentation:
https://www.w3schools.com/html/html5_syntax.asp

```
<!DOCTYPE html>
<html lang="de-at">
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <title>Basis HTML</title>

    <link rel="stylesheet" href="style.css"/>
    <script src="script.js"></script>
  </head>
  <body>
    <header>
      
      <h1>Styleguide für HTML</h1>
    </header>

    <nav>
      <div>
        <ul>
          <li>
            <a href="#">Home</a>
          </li>
          <li>
            <a href="#">Unterseite 1</a>
          </li>
          <li>
            <a href="#">Unterseite 2</a>
          </li>
        </ul>
      </div>
    </nav>

    <article>
      <h2>Hauptinhalt</h2>
      <p>
        Hier kommt der Hauptinhalt der Seite
      </p>
    </article>

    <footer>
      &copy; HTL St. Pölten 2024
    </footer>
  </body>
</html>
```

Basis HTML Seite



HTL ST. PÖLTEN

```
<!DOCTYPE html>
<html lang="de-at">
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <title>Basis HTML</title>

    <link rel="stylesheet" href="style.css"/>
    <script src="script.js"></script>
  </head>
  <body>
    <header>
      
      <h1>Styleguide für HTML</h1>
    </header>

    <nav>
      <div>
        <ul>
          <li>
            <a href="#">Home</a>
          </li>
          <li>
            <a href="#">Unterseite 1</a>
          </li>
          <li>
            <a href="#">Unterseite 2</a>
          </li>
        </ul>
      </div>
    </nav>

    <article>
      <h2>Hauptinhalt</h2>
      <p>
        Hier kommt der Hauptinhalt der Seite
      </p>
    </article>

    <footer>
      &copy; HTL St. Pölten 2024
    </footer>
  </body>
</html>
```



Styleguide für HTML

- [Home](#)
- [Unterseite 1](#)
- [Unterseite 2](#)

Hauptinhalt

Hier kommt der Hauptinhalt der Seite

© HTL St. Pölten 2024

CSS



HTL ST. PÖLTEN

CSS



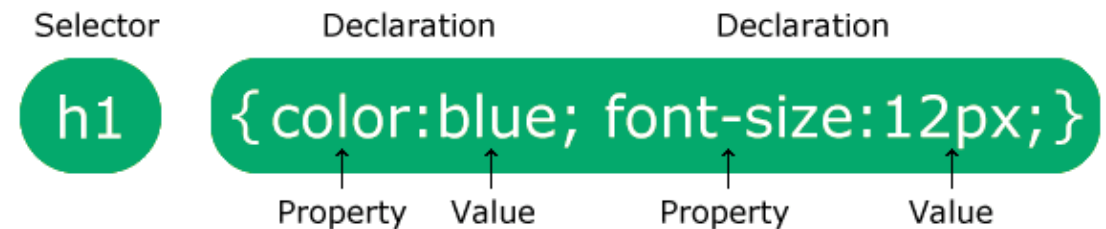
Cascading Style Sheets



- CSS ist eine Formatierungssprache, mit der man den Stil von HTML Dokumenten beschreiben kann.
- CSS beschreibt, wie HTML Elemente angezeigt werden sollen.
- „Sinn von CSS“ besteht in der Trennung von Inhalt und Design → [Demo](#)

CSS Syntax

- Aufbau aus Selektor und Deklarationen
- Selektor wählt betroffenes HTML Element
- Deklaration beschreibt Aussehen
- Deklaration aus property – value Paar mit ; getrennt



CSS Selektoren

- **Element Selektor** → Name des HTML Tag
- **ID Selektor** → ID des HTML Tag
- **Class Selektor** → Klasse des HTML Tag
- **Universal Selektor** → Anwendung auf alle HTML Tags
- **Group Selektor** → Zusammenfassung mehrerer Tags

```
01. p { color: green; }
02.
03. #id { color: green; }
04.
05. .class { color: green; }
06.
07. * { color: green; }
08.
09. p, h1, div { color: green; }
```

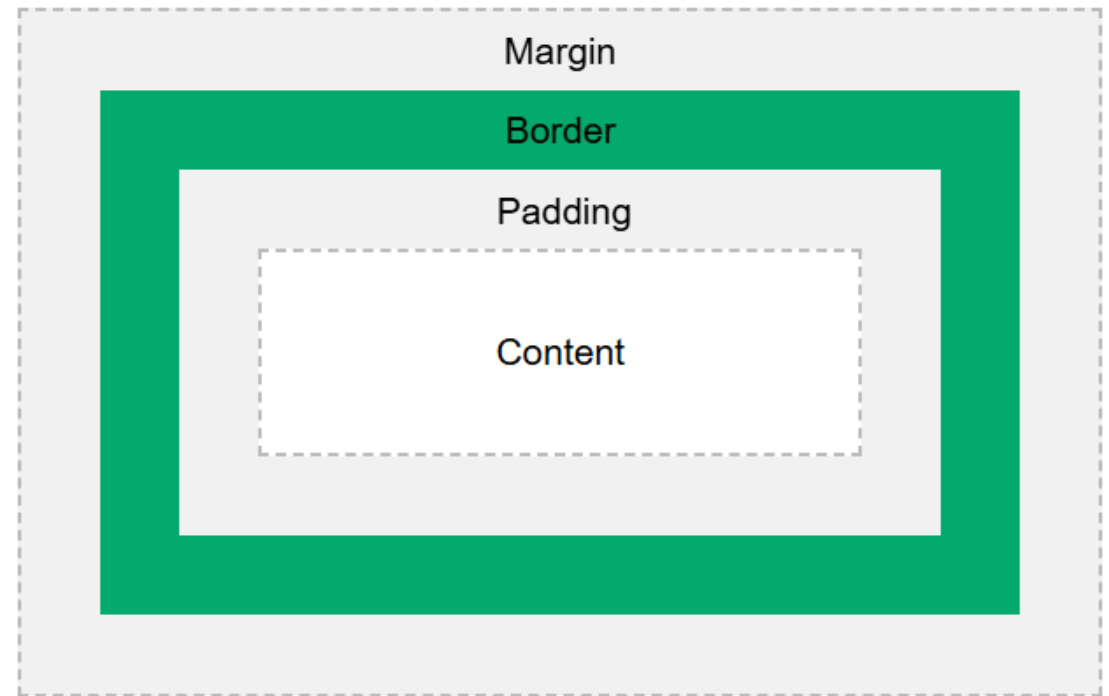
CSS Einbinden

- **Inline CSS** → im style Attribut des HTML Tags
- **Internal CSS** → im style Tag des head einer HTML Datei
- **External CSS** → externe *.css Datei mit Definitionen im link Tag des head eingebunden

```
01. <p style="color:green;">
02.     Hier kommt der Hauptinhalt der Seite
03. </p>
04.
05. <head>
06.     <style>
07.         p { color: green;}
08.     </style>
09. </head>
10.
11.
12. <head>
13.     <link rel="stylesheet" href="style.css"/>
14. </head>
```


CSS Box Model

- Jedes HTML Element ist mit den gleichen Boxen umgeben
- **Content** → Inhalt des Elements, der angezeigt wird
- **Padding** → Transparenter Bereich um den Inhalt herum
- **Border** → Rahmen um das Padding herum
- **Margin** → Transparenter Bereich um den Rahmen herum



Bootstrap



Bootstrap



- Freies Frontend Framework für schnelle und einfache Webentwicklung
- Enthält HTML und CSS Gestaltungsvorlagen und optionale JavaScript Erweiterungen

My First Bootstrap Page

Resize this responsive page to see the effect!

Column 1

Lorem ipsum dolor sit amet, consectetur adipisicing elit...

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris...

Column 2

Lorem ipsum dolor sit amet, consectetur adipisicing elit...

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris...

Column 3

Lorem ipsum dolor sit amet, consectetur adipisicing elit...

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris...

Responsive Design



Bootstrap Versionen

A screenshot of the Bootstrap website's version page. The page has a purple header with navigation links (Docs, Examples, Icons, Themes, Blog), a search bar, and social media icons. The main content area is divided into five columns, each representing a different Bootstrap version. Each column includes a title (v5.x, v4.x, v3.x, v2.x, v1.x), a brief description of the release type, and a list of available versions. The v5.x column has a 'Latest' badge next to version 5.3. The v2.x and v1.x columns have a 'Stable' badge next to their first listed version (2.3.2 and 1.4.0 respectively).

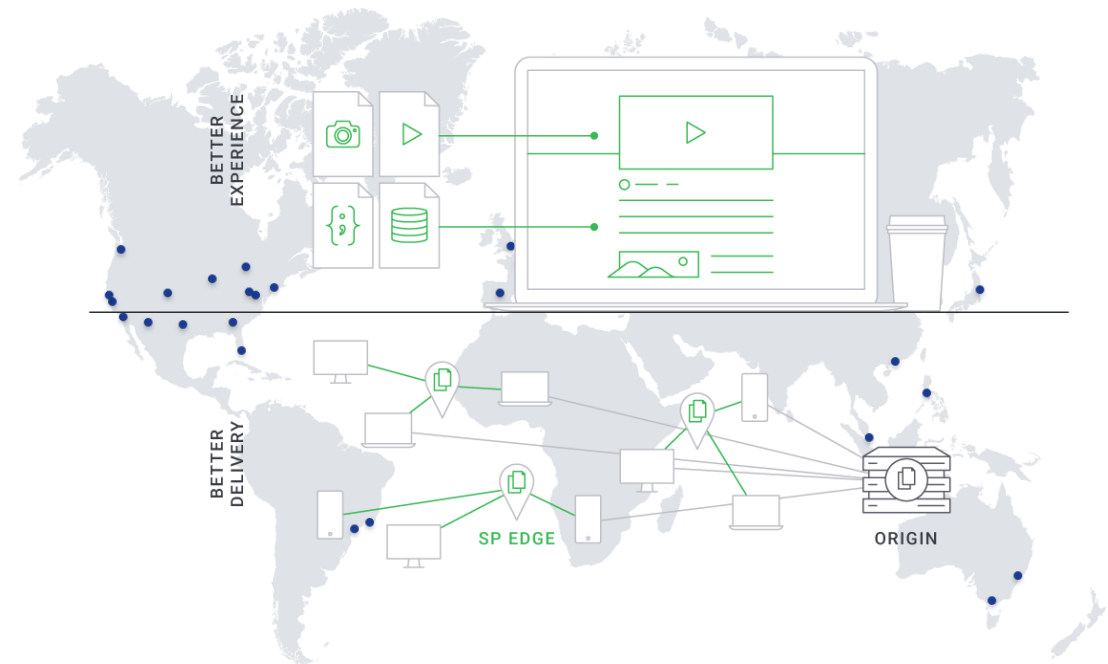
Version	Description	Available Versions
v5.x	Current major release. Last update was v5.3.2.	5.3 (Latest), 5.2, 5.1, 5.0
v4.x	Our previous major release with its minor releases. Last update was v4.6.2.	4.6, 4.5, 4.4, 4.3, 4.2, 4.1, 4.0
v3.x	Every minor release from v3 is listed below. Last update was v3.4.1.	3.4, 3.3
v2.x	Every minor and patch release from v2 is listed below.	2.3.2 (Stable), 2.3.1, 2.3.0, 2.2.2, 2.2.1, 2.2.0, 2.1.1, 2.1.0, 2.0.4, 2.0.3, 2.0.2, 2.0.1, 2.0.0
v1.x	Every minor and patch release from v1 is listed below.	1.4.0 (Stable), 1.3.0, 1.2.0, 1.1.1, 1.1.0, 1.0.0

Browserverbreitung



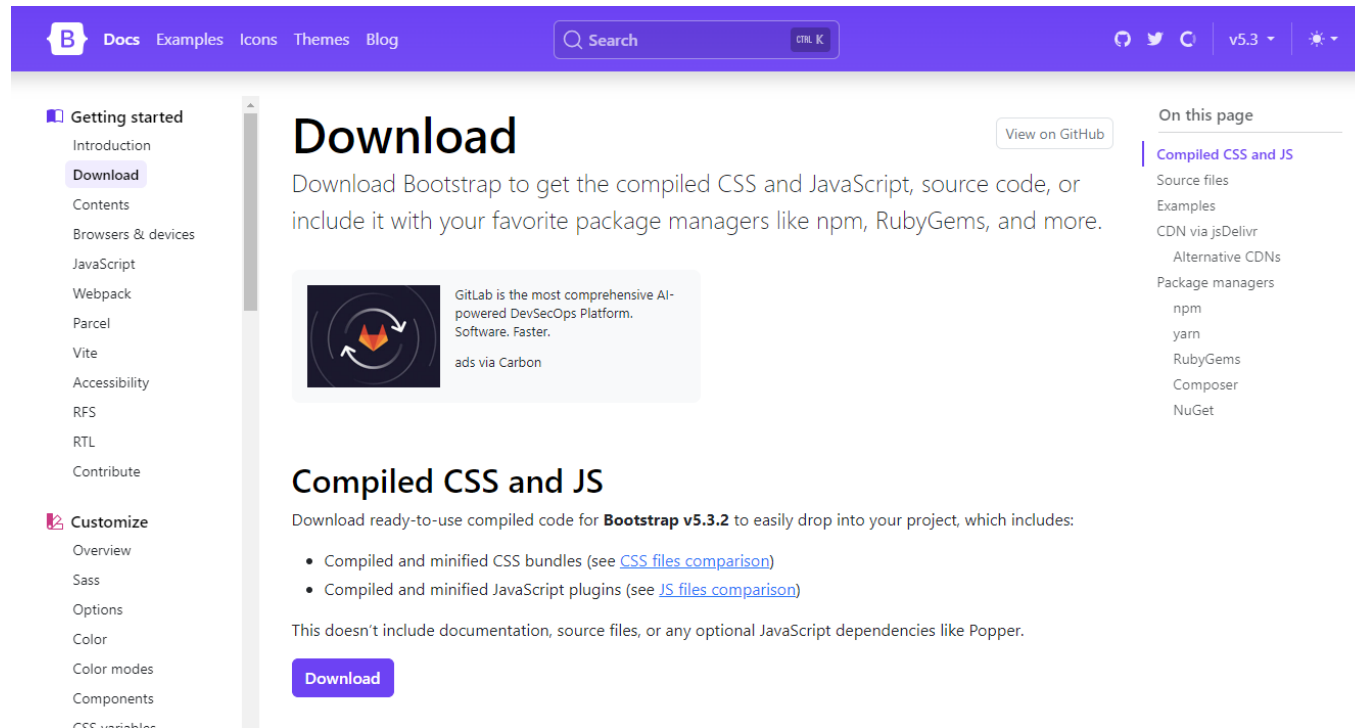
Bootstrap Einbinden (1)

- Bootstrap 5 über ein CDN einbinden → <https://www.bootstrapcdn.com/>
- Wiederverwendung von bereits gecachten Inhalten → schnellere Ladezeiten
- Bereitstellung von nächstem Server



Bootstrap Einbinden (2)

- Bootstrap 5 auf getbootstrap.com downloaden
- <https://getbootstrap.com/docs/5.3/getting-started/download/>



The screenshot shows the Bootstrap 5.3.2 Download page. The header is purple with navigation links: Docs, Examples, Icons, Themes, Blog. A search bar and version selector (v5.3) are also present. The left sidebar lists sections: Getting started (Introduction, Download, Contents, Browsers & devices, JavaScript, Webpack, Parcel, Vite, Accessibility, RFS, RTL, Contribute) and Customize (Overview, Sass, Options, Color, Color modes, Components, CSS variables). The main content area has a 'Download' heading and a description: 'Download Bootstrap to get the compiled CSS and JavaScript, source code, or include it with your favorite package managers like npm, RubyGems, and more.' Below this is a GitLab advertisement. The 'Compiled CSS and JS' section describes downloading ready-to-use compiled code for Bootstrap v5.3.2, which includes compiled and minified CSS bundles and JavaScript plugins. A 'Download' button is at the bottom. The right sidebar, 'On this page', lists links: Compiled CSS and JS, Source files, Examples, CDN via jsDelivr, Alternative CDNs, Package managers (npm, yarn, RubyGems, Composer, NuGet).

Bootstrap am µC

- CSS:
<https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css>
- JS:
<https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js>
- Inhalt lokal speichern am ESP wenn AP Mode

```
> .pio
> .vscode
v data
  JS bootstrap.bundle.min.js
  # bootstrap.min.css
  JS client.js
  ≡ config.txt
  <> index.html
  # style.css
> include
> lib
v src
  G main.cpp
> test
  .gitignore
  platformio.ini
```

CSS mit Bootstrap

```
<!DOCTYPE html>
<html lang="de-at">
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <title>Basis HTML</title>
    <!-- Bootstrap -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
          rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMhY6hW+ALEwIH"
          crossorigin="anonymous">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
           integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHh"
           crossorigin="anonymous"></script>
    <!-- eigener Style und JS -->
    <link rel="stylesheet" href="style.css"/>
    <script src="script.js"></script>
  </head>
  <body class="p-3 m-0 border-0">
    <header class="pb-3 mb-4 border-bottom d-flex align-items-center">
      
      <h1>Styleguide für HTML</h1>
    </header>

    <nav class="navbar navbar-expand-lg bg-body-tertiary rounded-2">
      <div class="container-fluid">
        <a class="navbar-brand" href="#">Navbar</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
              data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false"
              aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
          <ul class="navbar-nav">
            <li class="nav-item">
              <a class="nav-link active" aria-current="page" href="#">Home</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">Unterseite 1</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="#">Unterseite 2</a>
            </li>
          </ul>
        </div>
      </div>
    </nav>

    <article class="mt-4 p-2 border border-1 rounded-2">
      <h2>Hauptinhalt</h2>
      <p>
        Hier kommt der Hauptinhalt der Seite
      </p>
    </article>

    <footer class="pt-3 mt-4 text-body-secondary border-top">
      &copy; HTL St. Pölten 2024
    </footer>
  </body>
</html>
```

Styleguide für HTML

Navbar



Hauptinhalt

Hier kommt der Hauptinhalt der Seite

© HTL St. Pölten 2024

Bootstrap Dokumentation

- Abgesehen davon gibt es noch viel mehr...
 - <https://www.w3schools.com/bootstrap5/index.php>
 - <https://getbootstrap.com/docs/5.3/getting-started/introduction/>
- Und auch Übungen dazu z.B:
 - https://www.w3schools.com/bootstrap5/bootstrap_exercises.php

ESP32 Statusseite mit Bootstrap



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <link rel="stylesheet" href="bootstrap.min.css">
    <script src="bootstrap.bundle.min.js"></script>

    <title>ESP32 Main Page</title>
  </head>
  <body class="p-3 m-0 border-0">
    <header class="pb-3 mb-4 border-bottom d-flex align-items-center">
      
      <h1>ESP 32 Status site</h1>
    </header>
    <nav class="navbar navbar-expand-sm bg-body-tertiary mb-4">
      <div class="container-fluid">
        <ul class="navbar-nav me-auto mb-2 mb-lg-0">
          <li class="nav-item">
            <a class="nav-link active" aria-current="page" href="index.html">Home</a>
          </li>
        </ul>
      </div>
    </nav>
    <main>
      <div class="d-flex flex-wrap gap-3">
        <div class="card" id="LEDState">
          <h5 class="card-header">LED Status</h5>
          <div class="card-body">
            <p class="card-text">
              LED status %LED_STATE%
            </p>
          </div>
        </div>
      </div>
    </main>
    <footer class="pt-3 mt-4 text-body-secondary border-top">
      &copy; HTL St. P&ouml;lten 2024
    </footer>
  </body>
</html>
```



ESP 32 Status site

Home

LED Status

LED status ON

© HTL St. Pölten 2024