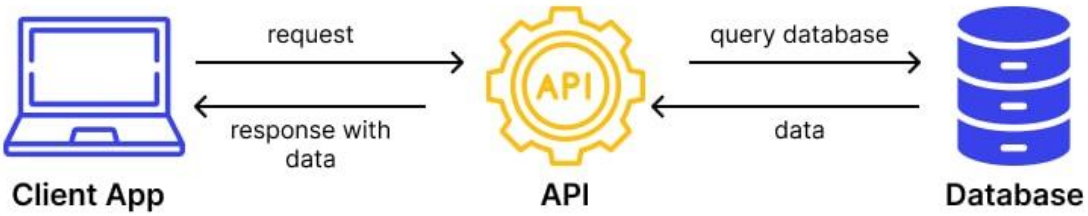
Dynamische Webseiten

Definition:

Dynamische Webseiten sind eine Art von Webseiten, die sich in Echtzeit anpassen und aktualisieren können, basierend auf Benutzerinteraktionen, Datenbankabfragen oder anderen Ereignissen. Im Gegensatz zu statischen Webseiten, die im Voraus erstellt und dann unverändert vom Server an den Benutzer gesendet werden, können dynamische Webseiten Inhalte generieren oder manipulieren, bevor sie an den Browser des Benutzers gesendet werden.

API-Requests:

Dynamische Webseiten setzten dabei auf Datenaustausch mittels API's um auch wirklich dynamisch realisierbar zu sein.



Definition:

Unter einer API Request versteht man eine Anfrage welche von einer Anwendung z.B: App, Web-Browser, abgesetzt wird um bei einem Server mittels Informationen, welche mit der Anfrage mitgesendet wird, spezifische Daten zurückliefert. Dabei sendet der Server, nachdem er die Request verarbeitet, hat eine Antwort mit den angefragten Daten.

Protokoll:

Der Großteil von API-Requests wird mittels HTTP-Standartmethoden durchgeführt. Das HTTP-Protokoll dient dabei als Client Server Kommunikationsprotokoll.

Aufruf:

Jede API verfügt über einen eindeutigen Endpunkt, über welchen sie erreicht werden kann. Dieser Endpunkt ist stehts in der Form einer eindeutigen URL (doppelte URL-Belegung nicht möglich) festgelegt.

Nutzen

API-Requests sind nützlich da sie es uns erlauben Daten auf einen externen Server zu speichern und nicht für jedes Gerät lokal zu hosten. Damit muss z.B. eine Datenbank welche Wetterdaten für ein ganzes Jahr enthält nicht für jede Applikation lokal gespeichert werden, sondern es kann mittels API-Requests, nur spezifische Wetterdaten für Tage, welche in der Applikation auch angezeigt werden, abgefragt werden und lokal gecached werden um die gespeicherten Daten so klein wie möglich zu halten.

Abfrage (Allgemeines Beispiel)

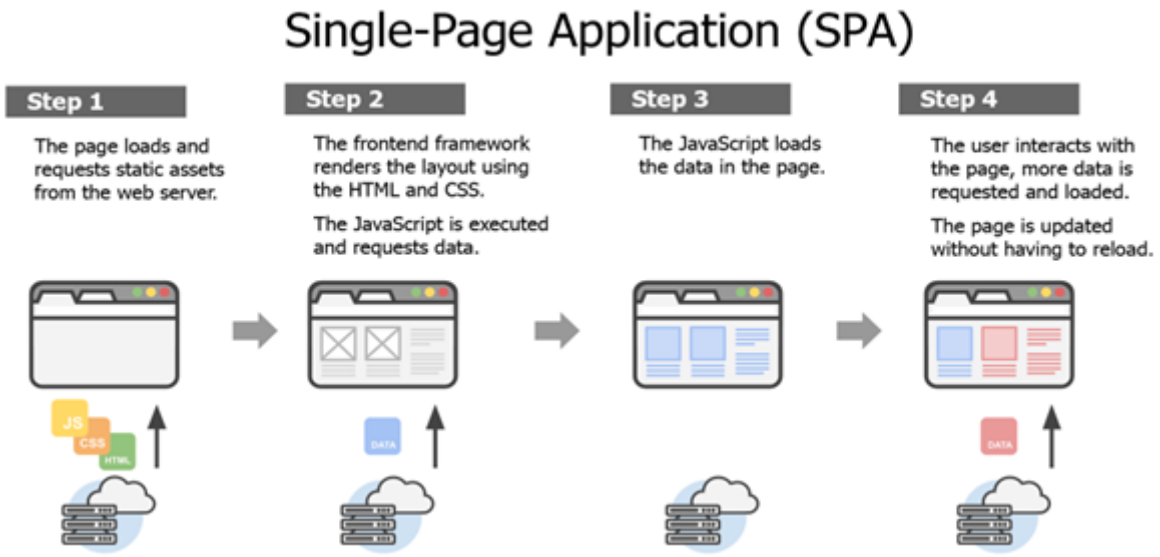
1. Die API-URL wird festgelegt
2. Ein Request Call wird basierend auf der URL durchgeführt
3. Ein Listener wird registriert welcher basierend auf der Antwort der API (In HTTP-Status Codes) ausgelöst werden
4. Die Rückmeldung, welcher aus einem Response Body besteht oft im JSON-Format wird dann zerlegt und die abgefragten Daten werden entnommen.

Vorteile von API-Requests:

- Datenaustausch kann Betriebssystem unabhängig durchgeführt werden.
- Performance verbessert sich, weil nicht so viele Daten lokal gespeichert werden müssen.
- Webseiten-Daten können einfach geändert werden, ohne etwas am Code zu ändern durch Datenbankanpassung.
- Ermöglicht die Verwendung von CMS-Systemen anpassen von Content basierend auf Datenbankeinträgen
- Zahlungsdienste von Drittanbieter wie Paypal können mittels API's abgewickelt werden.
- Authentifizierung mittels Authentifikation Provider wie Google ist möglich durch API-Call.

Was sind Single-Page-Applications?

Single-Page-Applications (SPAs) sind Webanwendungen, die aus nur einer HTML-Seite bestehen. Im Gegensatz zu traditionellen Multi-Page-Applications, bei denen jede Aktion des Benutzers eine neue Seite vom Server geladen wird, lädt eine SPA nur einmal die HTML-, CSS- und JavaScript-Dateien. Die Inhalte werden dann dynamisch nachgeladen, wenn der Benutzer mit der Anwendung interagiert, ohne dass die Seite neu geladen wird.



Schlüsselkonzepte von SPAs:

1. Dynamische Inhaltsaktualisierung – SPAs nutzen JavaScript und AJAX, um Inhalte dynamisch nachzuladen, während der Benutzer durch die Anwendung navigiert. Dies ermöglicht eine flüssige Benutzererfahrung, ohne dass die gesamte Seite neu geladen werden muss.
2. Clientseitiges Routing – Beim clientseitigen Routing handelt es sich um einen Ansatz zur Navigation innerhalb einer Single-Page-Application (SPA), bei dem die Verwaltung der Routen oder URLs vollständig im Webbrowser des Benutzers erfolgt, ohne dass zusätzliche Serveranfragen erforderlich sind.
3. State-Management – SPAs verwalten den Anwendungszustand effizient, um die Konsistenz zwischen verschiedenen Ansichten sicherzustellen. State-Management-Lösungen wie Redux, Vuex und MobX ermöglichen eine zentrale Verwaltung des Anwendungszustands und erleichtern die Entwicklung komplexer SPAs.
4. Wiederverwendbare Komponenten – SPAs verwenden wiederverwendbare Komponenten, um die Benutzeroberfläche in kleinere, eigenständige Bausteine zu zerlegen. Diese Komponenten können unabhängig voneinander entwickelt und wiederverwendet werden, was die Wartbarkeit und Erweiterbarkeit der Anwendung verbessert. Grundsätzlich diese werden Komponenten mithilfe von Frameworks wie Angular oder React.js erstellt da dies Grundbausteine von Frameworks sind.

Warum SPAs verwenden?

- Verbesserte Benutzererfahrung: SPAs bieten eine reaktionsschnelle und flüssige Benutzeroberfläche, da Inhalte dynamisch nachgeladen werden, ohne die gesamte Seite neu zu laden.
- Schnellere Ladezeiten: Durch das clientseitige Rendern und das **Lazy Loading** von Ressourcen werden die Ladezeiten optimiert, was zu einer verbesserten Leistung führt.

  Lazy Loading ist ein Konzept, bei dem Ressourcen (wie JavaScript-Dateien oder Bilder) erst dann geladen werden, wenn sie benötigt werden.

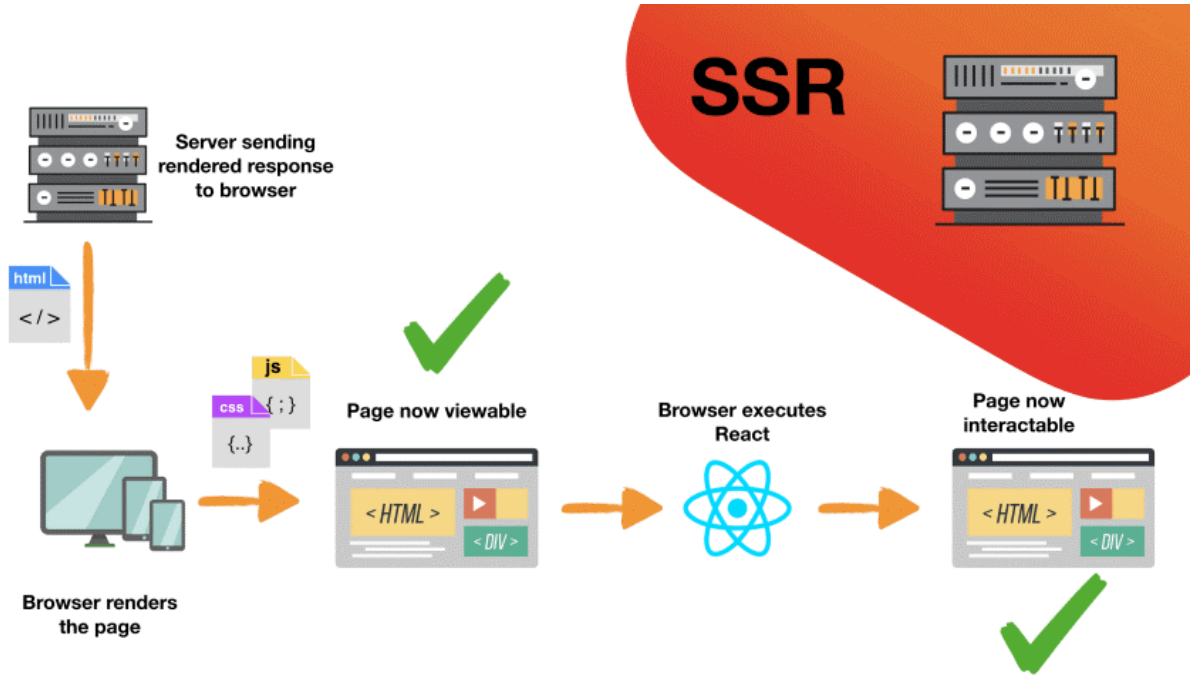
- Bessere Skalierbarkeit: Die klare Trennung von Frontend und Backend erleichtert die Skalierung der Anwendung und die Zusammenarbeit im Entwicklerteam.

Fazit:

Single-Page-Applications sind eine leistungsfähige und moderne Art, Webanwendungen zu entwickeln. Durch ihre Fähigkeit, Inhalte dynamisch nachzuladen und eine reaktionsschnelle Benutzererfahrung zu bieten, haben sie die Art und Weise, wie wir im Web entwickeln und interagieren, revolutioniert.

Server-Side-Rendering

Server-Side-Rendering (SSR) ist ein Prozess in der Webentwicklung, bei dem HTML-Seiten auf dem Server gerendert und diese fertig gerenderten Seiten an den Browser gesendet werden. Im Gegensatz zu Client-Side-Rendering bietet SSR eine schnellere initiale Ladezeit und verbesserte Suchmaschinenoptimierung (SEO). Dadurch wird eine bessere Benutzererfahrung ermöglicht, da Benutzer sofort Inhalte sehen können, bevor JavaScript und andere Ressourcen geladen sind. SSR wird häufig in Kombination mit Single-Page-Applications (SPAs) eingesetzt.



Session-Management

Session-Management ermöglicht die Verwaltung des Benutzerzustands und Interaktionen über mehrere Seitenaufrufe hinweg. Es bezieht sich darauf, wie Informationen über eine Sitzung zwischen Client und Server verwaltet werden.

Session Creation

Die Sitzungserstellung bezieht sich auf den Prozess, bei dem der Server eine neue Sitzung für einen Benutzer initialisiert, normalerweise nachdem der Benutzer die Website besucht hat oder sich angemeldet hat. Dies beinhaltet oft das Zuweisen einer eindeutigen Sitzungs-ID und das Speichern von Sitzungsinformationen.

Session Tracking

Die Sitzungsverfolgung bezieht sich auf den Prozess, bei dem der Server den Verlauf der Interaktionen eines Benutzers während einer Sitzung über mehrere Seitenaufrufe hinweg verfolgt. Dies wird typischerweise durch die Verwendung von Cookies oder durch das Weiterleiten der Sitzungs-ID in URLs erreicht.

Session Timeout

Die Sitzungszeitüberschreitung tritt ein, wenn eine Sitzung abläuft, weil der Benutzer für einen bestimmten Zeitraum keine Aktionen auf der Website

durchgeführt hat. Dies ist eine wichtige Sicherheitsmaßnahme, um zu verhindern, dass Sitzungen unbegrenzt aktiv bleiben und potenzielle Sicherheitsrisiken bergen.

Session Termination

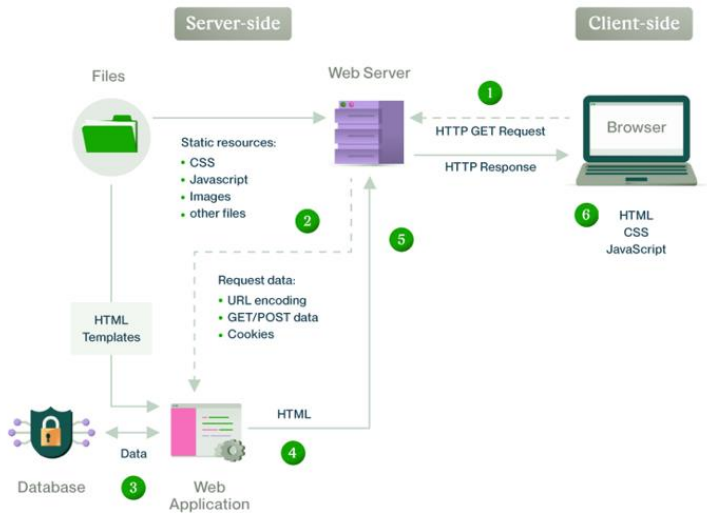
Die Sitzungsbeendigung ist der Prozess, bei dem eine aktive Sitzung bewusst beendet wird. Dies kann durch den Benutzer selbst erfolgen, indem er sich abmeldet oder die Website verlässt, oder durch den Server als Reaktion auf eine Zeitüberschreitung oder andere Ereignisse.

Session Security

Die Sicherheit von Sitzungen ist von entscheidender Bedeutung, um sicherzustellen, dass sensible Benutzerdaten und Interaktionen während einer Sitzung geschützt sind. Dazu gehören Maßnahmen wie die Verschlüsselung von Sitzungsdaten, die Verhinderung von Session Hijacking und Cross-Site-Scripting (XSS)-Angriffen sowie die Implementierung von Richtlinien zur sicheren Sitzungsführung.

Server-Side-Scripting

Server-Side Scripts sind Programme, die auf einem Webserver ausgeführt werden, um dynamische Webseiten zu generieren und so für jeden Benutzer ein einzigartiges Erlebnis zu schaffen. Diese Skripte bilden das Grundgerüst für die Back-End-Webentwicklung. Allerdings arbeiten zahlreiche weitere Komponenten mit den Skripten zusammen, um ein funktionsfähiges Backend-System zu schaffen. Dazu gehören Web-Apps, die in verschiedenen Programmiersprachen geschrieben wurden, Datenbanken, APIs, Cookies und mehr.



Jede Website gliedert sich in zwei Teile: das Frontend und das Backend. Das Frontend oder die clientseitige Komponente enthält alle visuellen Elemente und Schaltflächen, die Ihnen bei der Interaktion mit der Website helfen. Das Back-End steuert, wie jedes Visual und jede Schaltfläche reagiert.

Neben der Schaffung einer personalisierten Benutzererfahrung verlässt sich der Server auf den Back-End-Code, um Anmeldedaten zu speichern, den Quellcode zu verwalten, relevante Analysen zum Benutzerverhalten durchzuführen und zu steuern, wer Zugriff auf Informationen in der Datenbank hat.

Die Grundprinzipien von serverseitigem Scripting sind:

1. **Serverseitige Verarbeitung**: Im Gegensatz zur clientseitigen Verarbeitung, bei der der Code im Browser des Benutzers ausgeführt wird, wird serverseitiger Code auf dem Webserver ausgeführt, bevor die Ergebnisse an den Browser gesendet werden.

2. **Dynamische Generierung von Inhalten**: Serverseitiges Scripting ermöglicht es, Inhalte dynamisch zu generieren und an den Benutzer zu senden, basierend auf verschiedenen Bedingungen wie Benutzerinteraktionen, Datenbankabfragen oder Systemzuständen.
3. **Interaktion mit Datenbanken und externen Diensten**: Serverseitiges Scripting ermöglicht den Zugriff auf Datenbanken und externe Dienste, um Daten abzurufen, zu bearbeiten und zu speichern. Dies ermöglicht die Entwicklung von dynamischen und interaktiven Webanwendungen.
4. **Sicherheit**: Serverseitiges Scripting ermöglicht eine bessere Kontrolle über die Sicherheit der Anwendung, da sensible Operationen und Daten auf dem Server ausgeführt und verwaltet werden können. Dies hilft, Sicherheitsrisiken wie XSS (Cross-Site Scripting) und CSRF (Cross-Site Request Forgery) zu minimieren.
5. **Skalierbarkeit**: Serverseitiges Scripting ermöglicht die Skalierung von Webanwendungen, da die Verarbeitung auf dem Server erfolgt und nicht auf den Endgeräten der Benutzer. Dies erleichtert die Verwaltung von Lastspitzen und die Bereitstellung von Ressourcen für eine wachsende Benutzerbasis.
6. **Plattformunabhängigkeit**: Viele serverseitige Scripting-Sprachen und Frameworks sind plattformunabhängig, was bedeutet, dass sie auf verschiedenen Serverumgebungen und Betriebssystemen laufen können, was die Flexibilität und Portabilität von Webanwendungen erhöht.

   Folgende Sprachen werden bei Server-Side-Scripting eingesetzt:

   PHP: Eine sehr beliebte serverseitige Skriptsprache, die speziell für die Webentwicklung entwickelt wurde. PHP ist weit verbreitet und wird für eine Vielzahl von Webanwendungen verwendet. (z.B: WordPress, Slack)

   Python: Python ist eine vielseitige Programmiersprache, die auch für serverseitiges Scripting verwendet werden kann. Beliebte Frameworks wie Django und Flask erleichtern die Entwicklung von Webanwendungen in Python. (z.B: Google, Reddit)

   JavaScript (Node.js): Obwohl JavaScript ursprünglich eine clientseitige Skriptsprache war, wird sie mit Node.js auch für serverseitige Entwicklung verwendet. Node.js ermöglicht es Entwicklern, JavaScript auf dem Server auszuführen, was zu einer einheitlichen Codebasis führen kann. (z.B: Netflix, PayPal)

   Ruby: Ruby ist eine dynamische, objektorientierte Programmiersprache, die oft mit dem Ruby on Rails Framework für die Entwicklung von Webanwendungen verwendet wird. (z.B: Twitter, airbnb, GitHub)

Java: Java wird häufig für die Entwicklung von Unternehmensanwendungen und Webanwendungen eingesetzt. Es gibt verschiedene Frameworks wie Spring und JavaServer Faces (JSF), die die Entwicklung von Webanwendungen in Java erleichtern. (z.B: Amazon, Spotify)

ASP.NET (C#): ASP.NET ist ein von Microsoft entwickeltes Framework für die Entwicklung von Webanwendungen. Es wird häufig mit der Programmiersprache C# verwendet und bietet umfangreiche Funktionen und Werkzeuge für die serverseitige Entwicklung. (z.B: Microsoft Sotre)

Perl: Perl ist eine flexible und leistungsstarke Skriptsprache, die für eine Vielzahl von Anwendungen eingesetzt werden kann, einschließlich serverseitiger Webentwicklung. (z.B: Apache-Webserver)