Testung und Automatisierung

Build-Tools

In der Webentwicklung sind Build-Tools Softwarewerkzeuge, die den Prozess der Entwicklung, des Testens und des Bereitstellens von Webanwendungen automatisieren. Sie helfen dabei, Aufgaben wie das Kompilieren von Code, die Optimierung von Assets und das Verwalten von Abhängigkeiten zu vereinfachen. Ein Build kann viele Schritte beinhalten zB:

- Module bundling (mit zB Webpack oddedr Rollup analysiert den Code und seine Abhängigkeiten, einschließlich aller importierten Module, und erstellt dann ein oder mehrere Bündel, die den gesamten Code und alle Abhängigkeiten enthalten)
- Code Compilation (Transpilieren Quellcode, der in Sprachen wie TypeScript oder neueren Versionen von JavaScript (ES6+) geschrieben wurde, mit Tools wie Babel in browserfähiges JavaScript.)
- Linting (zB: EsLint Analysiert code auf Fehler oder schlechte Patterns oder best practices)
- Documentation Gen
- Testing (ausführen von automatisierten Tests um zu überprüfen ob gebuildeter code auch richtig funktioniert)
- Webpack Webpack ist ein leistungsstarkes Build-Tool, das hauptsächlich für das Bündeln von JavaScript-Modulen verwendet wird. Es ermöglicht auch das Bündeln von CSS, Bildern und anderen Ressourcen sowie erweiterte Funktionen wie das Optimieren von Assets.

Installation

```
npm install webpack webpack-cli --save-dev
```

Konfiguration

Eine Webpack-Konfigurationsdatei webpack.config.js muss im Projektverzeichnis erstellt werden. Hier ist ein einfaches Beispiel:

```
const path = require('path');
module.exports = {
entry: './src/index.js',
       output: {
        path: path.resolve(__dirname, 'dist'),
            filename: 'bundle.js',
       },
};
```

Verwendung

Führen Sie den folgenden Befehl aus, um Webpack zu verwenden und Ihr Projekt zu bündeln:

```
npx webpack
```

Website Testing

Theorie

Übersicht

Web-Tests umfassen die Bewertung verschiedener Aspekte einer Webanwendung, einschließlich ihrer Funktionalität, Benutzerfreundlichkeit, Sicherheit, Kompatibilität und Leistung. Das Ziel besteht darin, Probleme zu identifizieren und zu lösen, um sicherzustellen, dass die Anwendung effektiv auf verschiedenen Browsern, Geräten und Netzwerken arbeitet und eine nahtlose Benutzererfahrung bietet.

In der heutigen vernetzten Welt sind Websites allgegenwärtig und erfüllen verschiedene Zwecke, von der Rechnungszahlung bis zum Online-Shopping. Daher wird die Gewährleistung der Zuverlässigkeit und Effektivität von Webanwendungen für Entwickler und Tester entscheidend. Effiziente Web-Tests erfordern die Anwendung geeigneter Methoden und Techniken, um potenzielle Probleme anzugehen.

Was ist Web-Testing?

Web-Testing kann als der Prozess der Prüfung einer Webanwendung zusammengefasst werden. Es umfasst die Validierung von Funktionalitäten, Benutzeroberfläche (UI), Benutzererfahrung (UX) und Reaktionsfähigkeit. Web-Tests gehen jedoch über diese Aspekte hinaus und umfassen Sicherheit, API-Funktionalität, Leistungsmetriken und die Kompatibilität mit verschiedenen Browsern.

Warum ist Web-Testing wichtig?

Web-Testing ist entscheidend, um eine positive Benutzererfahrung sicherzustellen und Risiken wie Sicherheitsverletzungen zu minimieren. Die Vernachlässigung von Web-Tests kann schwerwiegende Folgen haben, einschließlich finanzieller Verluste und Schäden am Ruf eines Unternehmens. Mehrere Beispiele zeigen die bedeutende Auswirkung von Web-Tests auf die Benutzerzufriedenheit und die Umsatzgenerierung für Unternehmen.

Web-Test-Lebenszyklus

Der Web-Test-Lebenszyklus spiegelt den Lebenszyklus des Software-Tests wider und umfasst die Vorbereitung von Checklisten, das Sammeln von Anforderungen, die Planung und Gestaltung von Tests, Testausführungstechniken, Testausführung und Berichterstattung.

Arten von Webanwendungen und deren Testpunkte

Webanwendungen können in statische, dynamische, E-Commerce- und mobile Anwendungen unterteilt werden. Jeder Typ erfordert spezifische Testmethoden, die auf seine Eigenschaften zugeschnitten sind, wie z. B. Benutzeroberflächentests, Funktionalitätstests, Sicherheitstests und Leistungstests.

Elemente des Web-Testens

Wichtige Elemente des Web-Testens sind Browser, Geräte, Testwerkzeuge und die Erfahrung der Teammitglieder. Diese Elemente spielen eine entscheidende Rolle bei der Sicherstellung einer umfassenden Testabdeckung und einer effektiven Fehlerbehebung.

Aspekte des Web-Testens

Unterschiedliche Testmethoden wie Funktionalitätstests, Schnittstellentests, Benutzerfreundlichkeitstests, nicht-funktionale Tests (einschließlich Lasttests, Leistungstests, Stresstests, Cross-Browser-Tests, Browserkompatibilitätstests) und Sicherheitstests zielen auf verschiedene Aspekte einer Webanwendung ab, um deren Zuverlässigkeit und Sicherheit zu gewährleisten.

Zusammenfassend ist Web-Testing entscheidend, um die Funktionalität, Benutzerfreundlichkeit, Sicherheit und Leistung von Webanwendungen zu garantieren und damit die Benutzererfahrung zu verbessern und potenzielle Risiken zu minimieren.

Ziele des Webseite-Testings:

Das Hauptziel des Webseite-Testings besteht darin, sicherzustellen, dass eine Webseite fehlerfrei funktioniert und eine positive Benutzererfahrung bietet. Zu den spezifischen Zielen gehören:

1. **Funktionalität**: Überprüfen, ob alle Funktionen der Webseite wie erwartet arbeiten. Dazu gehören Navigation, Formulare, Links, Suchfunktionen usw.
2. **Kompatibilität**: Testen der Kompatibilität der Webseite mit verschiedenen Browsern (Chrome, Firefox, Safari, etc.), Betriebssystemen (Windows, MacOS, Linux, etc.) und Gerätetypen (Desktop, Tablet, Smartphone).
3. **Responsivität**: Sicherstellen, dass die Webseite auf verschiedenen Bildschirmgrößen und Auflösungen richtig angezeigt wird und ein reaktionsschnelles Design bietet.
4. **Leistung**: Überprüfen der Ladezeit der Webseite, um sicherzustellen, dass sie schnell und effizient geladen wird.
5. **Sicherheit**: Testen der Webseite auf potenzielle Sicherheitslücken, wie z. B. Cross-Site-Scripting (XSS), SQL-Injection, unsichere Datenübertragung usw.

Methoden des Webseite-Testings:

Es gibt verschiedene Methoden, um Webseiten zu testen. Einige der häufigsten sind:

1. **Manuelles Testing**: Tester navigieren durch die Webseite und überprüfen alle Funktionen, Links und Inhalte manuell.
2. **Automatisiertes Testing**: Verwendung von Tools und Skripten, um wiederholbare Tests automatisch durchzuführen. Dies ermöglicht eine schnellere und effizientere Testdurchführung.
3. **Usability-Testing**: Reale Benutzer werden eingeladen, die Webseite zu verwenden, während ihre Interaktionen und Feedbacks beobachtet und aufgezeichnet werden.
4. **Kompatibilitätstests**: Testen der Webseite auf verschiedenen Browsern, Betriebssystemen und Geräten, um sicherzustellen, dass sie überall konsistent funktioniert.
5. **Performance-Testing**: Messung der Ladezeit und der Leistung der Webseite unter verschiedenen Bedingungen, um Engpässe zu identifizieren und zu optimieren.

Best Practices für Webseite-Testing:

- Definieren Sie klare Testziele und -anforderungen, bevor Sie mit dem Testing beginnen.
- Testen Sie die Webseite in verschiedenen Entwicklungsphasen, von der Entwicklung bis zur Bereitstellung.
- Führen Sie regelmäßige Tests durch, um sicherzustellen, dass die Webseite auch nach Aktualisierungen und Änderungen einwandfrei funktioniert.
- Verwenden Sie eine Kombination aus manuellen und automatisierten Tests, um eine umfassende Abdeckung zu gewährleisten.
- Berücksichtigen Sie die Feedbacks und Erfahrungen der Benutzer, um kontinuierliche Verbesserungen vorzunehmen.
- Halten Sie sich über die neuesten Entwicklungen und Best Practices im Bereich des Webseite-Testings auf dem Laufenden.

Was teste ich in einer statischen Webseite?

Da statische Webanwendungen unkompliziert sind, erfordern sie keine komplexen Testmechanismen. Die Schwerpunkte liegen im Bereich User Interface Testing und Responsive Testing.

Was teste ich in einer dynamischen Webseite?

Eine dynamische Webanwendung kann so einfach sein wie die regelmäßige Aktualisierung von Cricket-Kommentaren oder so komplex wie Netflix mit Hunderten von Microservices und maschinellem Lernen mit KI-Integration. Sie können auch das Anmeldemodul mit spezifischen Seiten für jeden Benutzer enthalten. All dies erfordert ein tiefes Verständnis von Webtests und komplexen Techniken. Zum Beispiel Funktionstests und Usability-Tests, um nur einige zu nennen. Auch hier können Sicherheits- und API-Tests erforderlich sein, da es sich um personenbezogene Daten handelt.

Was ist in einer mobilen Webanwendung zu testen?

An dieser Stelle erleichtern reaktionsfähige Tests die Arbeit, indem sie automatisch entsprechend dem Ansichtsfenster arbeiten und sicherstellen, dass das Erlebnis unabhängig vom Gerät eines Benutzers konsistent bleibt wie bei einem Desktop-Benutzer.

Tools

Task Runner

Task Runner ist ein Programm, das einen oder mehrere Computerprozesse automatisiert. Vorallem nützlich bei Tasks die oft wiederholt werden müssen.

Gulp

Ist ein Task-Runner der dazu verwendet wird, Tests zu automatisieren

- Flexible Code over Configuration: Wenig Konfigurieren, stattdessen die Standardkonventionen durch Code definiert.

```
export function html() {
            return gulp
            .src("src/index.html")
```

```
            .pipe(htmltidy())
            .pipe(gulp.dest("build"));
}
```

- Composable Taks für bestimmte Aufgaben schreiben, was Geschwindigkeit und Genauigkeit erhöht.
- Efficient Änderungen von Dateien werden in den Arbeitsspeicher geschrieben, was den Build-Prozess beschleunigt.

Vite

Vite ist ein modernes Build-Tool für die Webentwicklung, das auf Geschwindigkeit und Einfachheit ausgelegt ist. Es wurde von Evan You, dem Schöpfer von Vue.js, entwickelt und bietet eine schnellere Startzeit im Vergleich zu herkömmlichen Tools wie Webpack.

Hauptmerkmale

- Schnell: Vite bietet eine schnellere Startzeit dank des "Instant-Start"-Konzepts, das nur die geänderten Teile der Anwendung neu lädt.
- ES-Modul-Unterstützung: Vite unterstützt native JavaScript-Module, was zu schlankerem und effizienterem Code führt.
- Linting
- Prettier Code Formatting
- Framework-Integration: Es bietet native Unterstützung für eine Vielzahl von Frameworks wie Vue.js, React und Preact.

Um ein Projekt zu erstellen:

```
npm create vite@latest app-name
```

Vitest

Gehört zu Vite.

```
npm install -D vitest
```

```
// sum.js
export function sum(a, b) {
    return a + b
}
```

```
// sum.test.js
import { expect, test } from 'vitest'
```

```
import { sum } from './sum'

test('adds 1 + 2 to equal 3', () => {
        expect(sum(1, 2)).toBe(3)
    })
```

Selenium

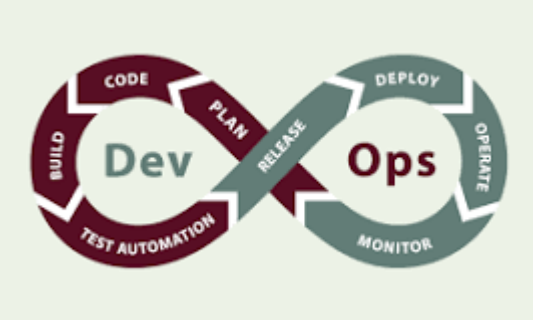
Selenium ist ein opensource Framework zum Testen von Ui-Webanwendungen. Es ermöglicht das automatisierte Ausführen von Aktionen auf einer Webseite, wie das Klicken auf Buttons, das Ausfüllen von Formularen oder das Auslesen von Text. Selenium kann in verschiedenen Programmiersprachen verwendet werden.

Bestandteile

1. **Selenium Core**: Das Core-Modul enthält die komplette Basisfunktionalität von Selenium, also die Testbefehl-API und den TestRunner
2. **Selenium WebDriver**: Selenium WebDriver ist der Nachfolger von Selenium Remote Control. Er akzeptiert Befehle (in Selene oder über die Client-API) und leitet sie an einen Headless-Browser (Browser ohne grafische Oberfläche) weiter.
3. **Selenium Grid**: Ermöglicht das gleichzeitige Testen auf verschiedenen Browsern und Betriebssystemen.
4. **Selenium IDE**: Selenium IDE ist ein Add-on für Google Chrome, Microsoft Edge und Mozilla Firefox, mit dem es möglich ist, direkt im Browser durch die Interaktion mit einer Webanwendung Testfälle aufzunehmen und diese im Browser wieder abzuspielen.

DevOps

Unter DevOps versteht man diverse Praktiken, Tools und eine Kulturphilosophie, die die Prozesse zwischen Development und Operations automatisieren und integrieren. Im Vordergrund steht dabei teamübergreifende Kommunikation und Zusammenarbeit sowie Technologieautomatisierung.



CI

Continuous Integration (CI) ist ein DevOps-Verfahren in der Software-Entwicklung, bei der Entwickler alle Codeänderungen regelmäßig in einem zentralen Repository zusammenführen. Diese Änderungen werden automatisch erkannt (z.B. durch git-pushes) und vordefinierte Tests angestoßen. Man spricht auch von CI-Pipelines.

CD

Continuous Deployment (CD) ist ein DevOps-Verfahren, bei dem Softwareänderungen automatisiert auf die Produktionssysteme deployed wird.