Swift On Linux/Haupteil/Vergleich mit anderen Sprachen/

Aus SwiftWiki

Inhaltsverzeichnis

JavaScript als Reverenz

JavaScript und Node.js

Ursprünglich wurde JavaScript als Teil vom Webbrowsern implementiert, um clientseitig Skripte auszuführen. Dabei soll dem Anwender eine Schnittstelle für interaktives Web geboten werden, asynchrone Anfragen gestllt und der Dokumenteninhalt verändert werden können. Die Sprache war in der Vergangenheit sehr negativ besetzt und hat auch heute noch bei vielen einen negativen Beigeschmack. Leider wird oft die Entwicklung von JavaScript in den letzten Jahren außer Acht gelassen, wo sich JavaScript immer mehr Richtung "Unverzichtbarkeit" gearbeitet hat. Derzeit ist JavaScript beinahe auf jeder WebSite und Webanwendung vertreten. Früher als Spielzeug bezeichnet, ist es heute ein beliebtes Werkzeug, um dynamische WebSites zu erstellen und es Usern zu ermöglichen, interaktiv im Browser und im Internet unterwegs zu sein. Auch große Firmen wie Mozilla, Google, Microsoft und Apple setzen mittlerweile auf JavaScript[1]. Mittlerweile ist die Sprache prototypenbasiert, dynamisch und typensicher. Grundlage für JavaScript bildet die Skriptsprache C sowie sehr viele Begriffsstandards von Java[2]. JavaScript am Server auszuführen, ist seit Entstehung der Sprache eine laufende Idee von JavaScript Entwicklern und wurde bereits ein Jahr nach Entwicklung 1996 umgesetzt. Der Druchbruch gelang aber erst am 8. November 2009 als Ryan Dahl Node.js vorstellte, dass bis heute immer beliebter wird. Mittlerweile gibt es sehr viele Tutorials, die es ermöglichen, JavaScript und Node.js zu lernen und auch sehr viele Beispiele um diese Sprache und das Framework zu erlernen.

Pros and Cons

Pros of JavaScript	Cons of JavaScript	Pros of Node.js	Cons of Node.js
Benutzer freundlichere	JavaScript kann clientseitig deaktiviert	Open source server Framework[3]	Unübersichtlichkeit der Community und der Entwicklung, durch sehr

Websites	werden		schnelles Wachstum[4]
Dynamischer Inhalt von Websites	Wenn deaktiviert, oft negative Auswirkungen	lauffähig auf Windows, Linux, Unix, Mac OS X, ...[3]	Modulsytem verbraucht viel Speicher[4]
objekt-basiert, prototyping für Vererbung	ursprünglich keine Vererbung	Verwendung von serverseitigen JavaScript[4]	nur ein Prozess mit einem Speicher von 1,7 GB [4]
Objekt-Referenzen werden erst zur Laufzeit geprüft[5]	schwach typisierte Variablen[5]	Asynchron, event basiert[4]	benötigt clustering um große Mengen abzuarbeiten[4]
einfache prozeduale Sprache[5]	kein natives Modulsystem[4]	einbinden von C und C++ libraries[4]	nicht alle libraries von C und C++ verfügbar[4]

ECMAScript

JavaScript wurde in der ECMA (European association for standardizing information and communication systems) spezifiziert. Der Name "ECMAScript" wurde deshalb gewählt, weil Netscape und Microsoft sich nicht auf einen gemeinsamen Namen für die eigenständigen Sprachen "JScript und JavaScript" einigen konnten. Jedoch wurde dieser Name nie weitläufig zur Gewohnheit, weil Brendan Eich, Erfinder von JavaScript meinte, dass dies "wie eine Hautkrankheit" klinge[6]. Entwickler können die offene Sprache zum Entwickeln von Programmen nutzen und dabei sicher gehen, dass der Code unterstützt wird, wenn dieser den Standard einhält.

Swift vs. JavaScript

Ziel der Entwicklung

Ein großr Unterschied bei den beiden Sprachen liegt im Ziel der Entwicklung der Sprachen. JavaScript wurde entwickelt mit dem Ziel, statisches HTML und CSS im Web, dynamischer zu gestalten und dem Client ein interaktives Web zu bieten. JavaScript ist beinahe auf jeder Website zu finden und auch nicht mehr wegzudenken[7]. Hingegen zu JavaScript ist Swift als eine Allzwecksprache erfunden worden, mit der Absicht die beste, einheitliche Sprache für die Systemprogrammierung, zu mobilen Applikationen und Desktop Anwendungen bis hin zum Cloud Service zu bieten. Dabei wurde noch auf drei Punkte sehr großer Wert gelegt[8]:

- Sicherheit: Dabei wird davon ausgegangen, dass nicht definiertes Verhalten der Grund für unsichere Programmierung ist. So muss in Swift jedes mögliche Ende bedacht und definiert werden da sonnst eine Kompiliertungsfehler auftritt.
- Performance: Swift soll alle C-basierten Sprachen ersetzen. Dazu muss Swift vergleichbar in der Abarbeitung verschiedener Aufgaben sein wie diese Sprachen und diese genau so schnell umsetzen können und gleichzeitig die Ressourcen schonen.
- Ausdruck: Die Syntax[9] wurde mit Jahrzehnten von Programmiererfahrung entwickelt und wird auch immer weiter entwickelt werden.

Compiling

JavaScript iste eine dynamische typisierte, prototypenbasierte, interpretierte Programmiersprache, wobei die Betonung auf "interpretierte" liegt. Der Unterschied liegt darin, dass Swift zu den kompilierten Sprachen zählt.

'**Interpretierte Sprachen:**'

In dem Moment wo das Programm gestartet wird, wird der Code in Instruktionen übersetzt und auch ausgeführt. Diese Übersetzung übernimmt ein zusätzliches Tool, der Interpreter. Dieser läuft wie eine virtuelle Maschine auf dem PC und nimmt Eingaben sowie den Quellcode und wandelt diesen in einen hardwareunabhängigen Bytecode. Dies passiert während der Laufzeit, wobei pro Prozessor ein Interpreter benötigt wird.
Vorteil: Leichter bei der Entwicklung, da bereits während der Entwicklung getestet werden kann. JavaScript wird im Plaintext an den Client übertragen und kann von anderen Entwicklern gelesen und gelernt werden.
Nachteil: Langsamer und ineffizienter als kompilierte Sprachen, da kontrollflusssteuernte Funktionen (Schleifen, Funktionen) immer wieder übersetzt werden müssen. [10]

'**Kompilierte Sprachen:**' Der Compiler übersetz den Quellcode in ein maschinenlesbares Programm, sodass es vom Menschen nicht mehr gelesen werden kann, jedoch sofort ausgeführt werden kann. Dieses Programm bzw. die Anweisungen im Programm werden direkt vom Prozessor ausgeführt. Jedesmal wenn sich im Programmcode etwas ändert, muss der gesamte Code neu kompiliert werden.
Vorteil: Der Code wird durch den Compiler optimiert. Kompilierte Programme sind sehr schnell in der Ausführung.
Nachteil: Der Aufwand bei der Entwicklung ist ist zeitraubender, da das Programm vor einem Testlauf jedesmal neu kompiliert werden muss. [10]

Warum JavaScript mit Node.js als Reverenz

JavaScript hat mit der immer größer werdenden Nachfrage und der schnellen Weiterentwicklung eine sehr interessanten Geschichte. Auch wird es zusehends in der Webentwicklung serverseitig immer öfters eingesetz. Damit kann man neben PHP davon ausgehen, dass JavaScript mit Frameworks wie Node.js eine solide Basis für einen Vergleich bietet. Da Swift auch unter anderem die beste Sprache für die Webentwicklung werden möchte, kann somit ein Fazit gezogen werden, in wie weit Swift dieses Ziel bereits erreicht hat und in welchen Bereichen noch Nachholbedarf besteht.

Installation des Node.js Servers

Der Server ist wie bereits bei der Installation des Swift-Servers von Github auszuchecken oder als .zip-Datei herunterzuladen. Nach dem Auschecken oder Entpacken des Projekts, muss mit einem Terminal ins Verzeichnis "NodeServer" gewechselt werden und der Server mit npm initalisiert werden. Dabei werden alle notwendigen Packages heruntergeladen und der Server kann mit node gestartet werden.

```
cd NodeServer
npm init
node index.js
```

Die Server im Vergleich

Beide Server sind im MVC (Model View Controller) Pattern gehalten und haben folgenden Ablauf: Der Request wird empfangen und an einen der jeweiligen Handler weitergegeben. Die Handler geben entweder den Request direkt an eine View-Klasse/Methode oder an eine Model-Klasse/Methode weiter. Die Model Dateien speichern User-Daten wie Usernamen, Passwort, Vorname, Nachname und E-Mail in einem JSON File. Das Passwort wird bevor es gespeichert wird verschlüsselt. Die Profile der User werden in beide Servern als eigenen Objekt angelegt und können als JSON serialisiert werden. Als Sammelverzeichnis wird der Ordner "profiles" angelegt in dem wiederum ein Verzeichnis mit dem Usernamen erstellt. Erst in diesem wird das JSON File gespeichert. Diese Verzeichnisse sind notwenig sollten Bilder oder ähnlichen zu den Usern gespeichert werden. Als zweites Aufgabengebiet sind in den Model Dateien Methoden und

Funktionen enthalten, die diese Daten aus den JSON Files wieder ausliest, und das Passwort z.B. für den Login überprüfen. Spätestens nachdem die Model-Funktionen fertig sind wird der Request an die View übergeben. Hier werde die HTMLs zusammengebaut und an den Client zurückgeschickt. Die Funktionalität ist für Studenten der FH Joanneum gedacht. Die Studenten können ihre tätsächlichen Noten mit dem Usernamen und Passwort der FH-Domain abrufen.

Die Server unterscheiden sich ausschließlich durch Sprach- und Framework spezifischen Vorgaben. Weiter haben wir versucht ähnliche Packages zu verwenden und so wenig Workarounds zu implementieren, die Packages ersetzen.

1. Gelsendörfer, Felix (2010): Wie Node.js JavaScript auf dem Server revolutioniert: Schubrakete für JavaScript, [online] http://t3n.de/magazin/nodejs-javascript-server-revolutioniert-schubrakete-226177/ [27.07.2017].
2. Fuchs Media Solutions (o.J.): JavaScript Begriffserklärung und Definition, [online] https://www.seo-analyse.com/seo-lexikon/j/javascript/ [27.07.2017].
3. Refnes Data (2017b): Node.js Introduction, [online] https://www.w3schools.com/nodejs /nodejs_intro.asp [27.07.2017].
4. Elke, Gregor (2014): Ist Node.js ein Superheld?, [online] https://blog.codecentric.de/2014/06/ist-node-js-ein-superheld/ [27.07.2017].
5. o.V. (o.J.): Gefahren und Anwendungsmöglichkeiten druch JavaScript, [online], http://www.mathematik.uni-ulm.de/sai/ws01/portalsem/wiede/ [27.07.2017].
6. Vorlage:Cite book Günster, Kai (2013): *Schrödinger lernt HTML5, CSS3 & JavaScript: das etwas andere Fachbuch*, Bonn: Galileo Press.
7. Mozilla Developer Network and individual contributors (2017): JavaScript Guide - Introduction, [online] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction [27.07.2017].
8. Apple Inc (2017a): About Swift, [online] https://swift.org/about/ [27.07.2017].
9. Apple Inc (2017b): Swift - Language Guide, [online] https://developer.apple.com/library/content /documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html#//apple_ref /doc/uid/TP40014097-CH5-ID309 [27.07.2017].
10. Schnabel, Patrick (2012): Compiler und Interpreter, [online] https://www.elektronik-kompendium.de /sites/com/1705231.htm [27.07.2017].