Swift On Linux/Haupteil/Vergleich mit anderen Sprachen/

Aus SwiftWiki

JavaScript als Reverenz

JavaScript und Node.js

Ursprünglich wurde JavaScript als Teil vom Webbrowsern implementiert um cleint-seitig Skripte auszuführen. Dabei soll dem Anwender eine Schnittstelle für interaktives Web geboten werden, asynchrone Anfragen und der Dokumenteninhalt verändert werden können. Die Sprache war in der Vergangenheit sehr negativ besetz und hat auch heute noch bei vielen einen negativen beigeschmack. Leider wird oft die Entwicklung von JavaScript in den lezten Jahren außer acht gelassen, wo sich JavaScript immer mehr Richtung "Unverzichtbarkeit" gearbeitet hat. Derzeit ist JavaScript beinahe auf jeder WebSite und Webanwendung vertretten. 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. Acuh große Firmen wie Mozilla, Google, Microsoft und Apple setzen mittlerweile auf JavaScript. Mittlerweile ist die Sprache prototypenbasierte, dynamisch und typensicher ist. Grundlage für JavaScript bildet die Skriptsprache C sowie sehr viele Begriffsstandards von Java. Quelle:https://www.seo-analyse.com/seo-lexikon/j/javascript/ 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 Beispiel um sich 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 Websites	JavaScript kann Clientseitig deaktiviert werden	Open source server Framework	Unübersichtlichkeit der Community und Entwicklung durch sehr schnelles Wachstum
Dynamischer Inhalt von Websites	Wenn deaktiviert, oft schlechte Auswirkungen auf die Benutzerfreundlichkeit	lauffähig auf Windows, Linux, Unix, Mac OS X, ...	Modulsytem verbraucht viel Speicher
objekt-basiert, prototyping für Vererbung	ursprünglich keine Vererbung	Verwendung von serverseitigen JavaScript	nur ein Prozess mit einem Speicher von 1,7 GB
Objekt-Referenzen werden erst zu Laufzeit geprüft	schwach typisierte Variablen	Asyncron, event bassiert	benötigt cluster um große Mengen abzuarbeiten
einfache prozeduale Sprache	kein natives Modulsystem	einbinden von C und C++ Libories	nicht alle Libories von C unc C++ verfügbar

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. 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=== (Quellen: https://www.elektronik-kompendium.de/sites/com/1705231.htm, Schrödinger lern HTML, CSS und JavaScript

Ziel der Entwicklung

Einen großen Unterschied bei den beiden Sprachen liegt im Ziel der Entwicklung der Sprachen. JavaScript wurde entwickelt mit dem Ziel, statisches und langweiliges HTML und CSS im Web dynamischer zu gestallten und dem Client ein interaktives Web zu bietet. JavaScript ist beinahe auf jeder Website zu finden und auch nicht mehr wegzudenken. 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 großen Wert gelegt:

- Sicherheit: Dabei wird davon ausgegengen das 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 wurde mit Jahrzenten von Programmiererfahrung entwicklet und wird auch immer weiter entwicklt werden.

Compiling

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

'**Interpretiert Sprachen**' In dem Moment wo das Programm gestartet wird, wird der Code in Insturktionen übersetzt und auch ausgeführt. Diese Übersetzung übernimmt ein zusätliches 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 Porzessor 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 ineffizinter als kompilierte Sprachen, da Kontrollflusssteuernte Funktionen (Schleifen, Funktionen) immer wieder übersetzt werden müssen.

'**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 Porgramm bzw. die Anweisungen im Programm werden direkt vom Prozessor ausgeführt. Jedesmal wenn sich im Programmcode etwas ändert, muss der gesamte Code neue 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.

Warum JavaScript mit Node.js als Reverenz

JavaScript hat mit dem immer aufsteigenden Interesse und der schnellen Weiterentwicklung einen sehr interessante Geschichte. Auch wird es zusehens in der Webentwicklung serverseitig immer öfters eingesetz. Damit kann man neben PHP davon ausgehen das JavaScript mit Frameworks wie Node.js eine solide Basis für einen Vergleich ermöglicht. 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
```

Zurück zur Startseite des Buches
