JavaScript im Unternehmen
Indiana Nimphius und der Tempel des Todes

Frank Nimphius
Oracle

Schlüsselworte
JavaScript, JET, Angular, Ionic, React, Mobile, Cloud, Sweet Dreams

Einleitung

Folgendes Gedankenspiel: Sie als Entwickler sind auserwählt für die Geschichtsbücher der Zukunft eine Momentaufnahme heutiger Trends in der IT zu erstellen. Sofort fallen Ihnen Mobile und Cloud ein. Sie schreiben über REST als den präferierten Webservice und JSON als das universelle Format für Service-Inhalte. "Mobile First" ist auch in aller Munde, also schreiben Sie dazu noch ein paar Worte. Was aber schreiben Sie über Entwicklungssprachen? Würden Sie über Java schreiben, oder PL/SQL, oder doch vielleicht JavaScript? Nicht nur dieser Artikel, sondern die Präsentation die dazu entstanden ist, ist aus diesem Gedankenspiel geboren.

Wenn Sie mit meinem Namen etwas anfangen können, dann wissen Sie mich im Oracle Forms (PL/SQL), Oracle ADF und vielleicht auch Oracle MAF Umfeld einzuordnen. JavaScript kommt in meiner Vita nur am Rande vor, nämlich dann wenn ich Erweiterungen für Oracle ADF, Oracle MAF und Forms geschrieben habe die JavaScript bedurften. Aber ich bin nicht blind oder taub und sehe, dass JavaScript heftig an die Pforten unternehmensweiter Anwendungsentwicklung klopft. Grund genug mit dem JavaScript Tempel des Todes mal auseinander zu setzen um für mich die Frage zu beantworten warum JavaScript und warum jetzt. Wenn Sie Lust haben, lesen Sie weiter und begleiten Sie mich auf meine Reise. Oder Sie hören jetzt auf zu lesen und machen sich auf die eigene Reise.

JavaScript oder ECMA Script

Zu Beginn meiner Reise gibt es gleich den ersten Stolperstein bei der Zuordnung von JavaScript zu ECMA Script und TypeScript zu überwinden. Was verwende ich denn nun wenn ich JavaScript Anwendungen schreiben möchte. Es stellt sich heraus, daß alle drei Begriffe miteinander verbunden sind und sogar ähnliches meinen. JavaScript ist eine Trademark von Oracle. Oracle hat mit der Übernahme von Sun das Namensrecht an JavaScript erworben, aber bisher keinen Gebrauch davon gemacht. ECMA ist der Standard hinter dem was wir üblicherweise als JavaScript bezeichnen. Zur Zeit ist ECMA in der Version 6, auch ECMA 2015 genannt, verfügbar. ECMA beschreibt neue Funktionen der Script Sprache, die dann von den Laufzeitherstellen, zum Beispiel Browserherstellern, implementiert werden. Mitunter zieht es sich die Implementierung etwas und auch ECMA Script ist nicht perfekt. Microsoft hat seine eigene Version von JavaScript erstellt und unter dem Namen TypeScript als Open Source veröffentlicht. Da kaum ein Browser TypeScript versteht und somit eine Verwendung nicht möglich ist, gibt es sogenannte Transpiler, die TypeScript in klassisches ECMA Script wandeln. Während der Entwicklung genießt der Entwickler also die Vorteiler von TypeScript, zum Beispiel Klassen und statische Variablen Typen, während zur Laufzeit der Anwendungscode ECMA konform im Browser abläuft.

Um eindeutig zu sein verwende ich ab hier den Begriff JavaScript für Script Sprache und den ECMA Script Standard.

Ein bißchen Spaß muß sein!

Wenn Sie, so wie ich, nicht mit JavaScript groß geworden sind und Schwimmen in der IT mit anderen Programmiersprachen wie PL/SQL, Java, C oder 4GL gelernt haben, dann werden Sie dazu geneigt sein graue Haare und Haarausfall auf JavaScript zu schieben. Und das nicht ganz zu unrecht. Auch wenn nicht alles was in JavaScript komisch erscheint ein Fehler in der Sprache ist, sondern ernstgemeintes Design, so erscheint die Sprache dennoch hier und da wenig konsequent. Im Englischen gibt es die Bezeichnung "a lot of rope to hang yourself", was gut zu JavaScript zu passen scheint.

Die wohl prominenteste Anfängerfalle ist "this" als Referenz auf den auszuführenden Kontext. "this" zeigt nicht auf den Kontext in dem eine Funktion definiert ist, wie zum Beispiel in Java, sondern in dem die Funktion ausgeführt wird. Den genauen Verhalt von "this" zu erläutern ist leider vielschichtig, zu vielschichtig um es in diesem Artikel kurz und prägnant zu erläutern. Allerdings lade ich Sie ein einmal auf YouTube nach JavaScript "scope" und "this" zu suchen. Sie werden bestimmt etwas zum Schmunzeln finden (Beispiele: Informational[1], Funny[2], Educational[3],[4]).

"Hoisting" drückt die Unbekümmertheit von JavaScript aus in Sachen wann eine Variable oder Funktion verwendet werden kann und wo sie definiert wird. So ist zum Beispiel folgende Reihenfolge der Verwendung der Variable *x* und deren Definition gültig.

```
x = 5;
// .. Do other work
var x;
```

Auch das nächste Beispiel funktioniert ohne Probleme.

```
test();
function test (){ x = 10; }
var x;
```

Sieht komisch aus, kann man sich aber dran gewöhnen. Was ist aber hiermit?

```
a(); var a =
function test (){ x = 10; }
var x;
```

Dieser Code kann so nicht ausgeführt werden und verursacht einen *TypeError*. Natürlich kann der erfahrene JavaScript Entwickler erklären, warum das so ist. Der Anfänger allerdings versteht die Welt nicht mehr.

"Destructuring" ist ein sinnvolles Feature in ECMA Script 6 und erlaubt Inhalte eines Arrays einfacher Variablen zuzuordnen.

---

[1] https://www.youtube.com/watch?v=2pL28CcEijU
[2] https://www.youtube.com/watch?v=PV_cFx29Xz0
[3] https://www.youtube.com/watch?v=PV_cFx29Xz0
[3] https://www.youtube.com/watch?v=lTUyNW9oTLo
[4] https://www.youtube.com/watch?v=Bv_5Zv5c-Ts
[4] https://www.youtube.com/watch?v=Bv_5Zv5c-Ts

```
var a, b, rest;
[a, b, ...rest] = [1, 2, 3, 4, 5];
console.log(rest); // => [3, 4, 5]
```

Tolle Sache! Der Variablen "rest", die bis dahin selbst nicht wußte, daß sie vom Typ Array ist, werden die Werte 3,4,5 als Array zugeordnet. Das geht übrigens auch umgekehrt:

```
var a = [1, 2, ...rest];
console.log(rest); // => [1,2,3, 4, 5]
```

Sicher, diese Eigenschaft ist gewöhnungsbedürftig. Aber wer hat sich das mit den drei Punkten nur einfallen lassen? Und wird es irgendwann mal vier Punkte für negierte Array Werte geben, oder fünf für die umgekehrte Reihenfolge der Elemente? Oder 6, für das Umkehren der Reihenfolge und dem gleichzeitigen negieren der Werte.

Mehr solcher kleinen Zirkusstückchen können Sie im Internet nachsehen oder auf YouTube nachschauen. Wer hat nochmal gesagt, daß Oracle ADF eine steile Lernkurve hat?

Darwin ist Schuld!

JavaScript wurde 1995 als *LiveScript* von Netscape veröffentlicht und kurze Zeit später, im Rahmen einer Kooperation mit Sun Microsystems, in JavaScript umbenannt. Warum aber ist es gerade jetzt und heute, daß JavaScript an Bedeutung gewinnt und auf bestem Weg ist den Titel "Populärste Programmiersprache der Welt" von Java zu übernehmen? Das Argument, daß es hauptsächlich Studenten sind die JavaScript in die Unternehmen bringen, will ich nicht gelten lassen, da es auch zwischen 1995 und 2016 Universitäten und Studenten gegeben hat. Für mich gibt es zwei Gründe, warum JavaScript dominiert. Am Anfang hat sich ECMA Script als Standard für JavaScript sehr schwer getan und kam mit ECMA Script 4 zunächst ins Stocken und dann zum Stillstand. Netcape, Sun und Microsoft hatten ein unterschiedliches Verständnis von dem was Scripting im Browser können sollte. Mit ECMA Script 5, und jetzt 6, wurde JavaScript dann um Funktionalität aufgewertet, die einen sinnvollen Einsatz außerhalb des Browser DOM ermöglichen.

Der entscheidende Grund aber ist so alt wie die Menschheit selbst und wurde von Herbert Spencer, nicht Charles Darwin, mit "Survival oft the Fittest" bezeichnet. Basierend auf Darwin's Erkenntnisse erkannte Spencer, daß sich die Spezies am besten entwickelt, die sich einer bestimmten Umgebung am besten anpassen kann. Oftmals findet man "Survival oft the Fittest" als "Überleben des Stärkeren" interpretiert, was aber Humbug ist, da zum Beispiel JavaScript nicht besser als Java ist.

Anwendungsentwicklung findet immer in einem Kontext statt, der sich durch Technologie, Pattern, Trends und Architektur bestimmt. Mit jeder Änderung des Kontext, und ich meine nicht die beliebte "Framework der Woche" Diskussion sondern den sogenannten Paradigmenwechsel, muß Software neu evaluiert werden. Nicht jede Änderungen des Kontextes führen zwangsweise zu einer Abkehr von einer gegenwärtig verwendeten Programmiersprache und –umgebung. Allerdings gibt es immer Sprachen die sich besser und andere, die sich schlechter an neue Begebenheiten anpassen. Im Moment sind "Mobile" in der Sparte Technologie und "Cloud" in der Kategorie Architektur die dominierenden Einflüsse. Beides zusammen schafft eine Umgebung zu der JavaScript sehr gut paßt. Einen dritten Grund werde ich dann später in diesem Artikel aufgreifen.

Wo wird JavaScript eingesetzt?

Für jeden Trend und Gegentrend gibt es Statistiken auf dieser Welt. Somit besteht die Kunst darin die Statistik für sich zu finden der man am ehesten glauben mag. Für mich ist die Stack Overflow[5] Webseite eine Institution im Bereich der Software Entwicklung. Einmal im Jahr werden Mitglieder dieser Seite ausgehend befragt. Der sogenannte *Developer Survey* für 2016 besagt, daß JavaScript sowohl im Full Stack, als auch im Front-End und Backend Bereich dominiert. Auch wenn 50.000 befragte Entwickler nicht unbedingt die Meinung der Weltpopulation an Entwicklern widergibt, so halte ich diese Statistik für repräsentativ. Noch mal zur Erinnerung: Ich bin kein JavaScript Evangelist, sondern gehöre zur Fraktion der Java Entwickler. Für mich ist professionelles Entwickeln mit JavaScript eher Neuland!
Im server-seitigen Einsatz von JavaScript sticht Node.js heraus. In einfachen Worten ausgedrückt ist Node eine server-seitige Laufzeitumgebung für Scripting die auf Google's V8 Interpreter basiert und von vielen Cloud Produkten als Primär- oder Sekundärsprache unterstützt wird. Ich würde nicht so weit gehen und Node als "Esperanto des Cloud Computing" bezeichnen, aber zumindest würde ich Node eine Omnipräsenz in diesem Bereich bescheinigen. Node wird, nach eigenen Angaben[6], von 95% der 500 führendsten Unternehmen eingesetzt und verfügte 2015 bereits über mehr als 3 Millionen aktive Anwender. Node wird monatlich um die zwei Millionen male heruntergeladen und Verzeichnet im selben Zeitraum bis zu 2 Milliarden Downloads einzelner Module.

JavaScript Frameworks, die Siegertypen!

Der Erfolg von Node führt mich zu meinem dritten Argument auf die Frage warum JavaScript jetzt und heute. Die Verfügbarkeit von ausgereiften JavaScript Frameworks hat die bis dato funktionale und imperative JavaScript Programmierumgebung in eine Plattform für modulare, komponenten-basierte und deklarative Anwendungsentwicklung gewandelt.

Moderne JavaScrip Anwendungsentwicklung ist Template-basiert, Komponenten orientiert, Modular und hat wenig mit der monotonen DOM Manipulation vergangener Zeiten gemein.

Moderne JavaScript Frameworks unterstützen die modulare Anwendungsentwicklung sogenannter Single Page Anwendungen (SPA). Je nach Framework wird das Data Binding durch das Framework selbst oder einem integrierten Framework oder Library übernommen. Alle Frameworks erzeugen client-seitige HTML Anwendungen und viele können auch zur Entwicklung hybrider Mobilanwendungen eingesetzt werden.

Aus dem Nebel[7] der vielen Frameworks stechen Angular, React, Ionic, Bootstrap und Ember hervor. Andere Frameworks, wie JQuery, Sass, Grunt, Bower und Yeoman, gehen dabei in die großen JavaScript Frameworks mit ein. JavaScript Frameworks stellen dem Entwickler Tooling und Infrastruktur zu Verfügung und helfen Programm Code zu organisieren.

Für JavaScript Entwickler hat Oracle das Oracle JavaScript Extension Toolkit (JET) im Portfolio. Oracle JET ist kein Framework, sondern ein Toolkit, wie der Name schon sagt. Wenn wir uns allerdings nicht mit Spitzfindigkeiten in der Namensgebung aufhalten, so muß man Oracle JET zu den

---

[5] http://stackoverflow.com/

[6] http://www.slideshare.net/NodejsFoundation/node-foundation-membership-overview-20150505

[7] https://hackernoon.com/how-it-feels-to-learn-javascript-in-2016-d3a717dd577f#.baa6hxvpe

JavaScript Frameworks hinzuzählen. Ich bin mir durchaus bewußt, daß es Unterschiede zwischen Frameworks und Toolkits gibt, bin aber von Natur aus nicht auf Haarspalterei aus.

Bei der Wahl eines JavaScript Frameworks zur unternehmensweiten Anwendungsentwicklung gibt es kein klares richtig oder falsch. Wie bei anderen Programmiersprachen und Frameworks gibt es auch bei JavaScript flammende Diskussionen verschiedener Meinungen. Das was für Unternehmen wichtiger sein sollte als das Framework der Stunde ist der Aufgabe die mit JavaScript programmiert werden soll, die Verwaltbarkeit der Anwendung durch das Framework, sowie die Unterstützung des Frameworks selbst im Bezug auf Komponenten, Visualisierungswerkzeuge, Datei Handling und Laden von Abhängigen Bibliotheken, Security, Responsive Layouts und Design, Verwendbarkeit für Web und Mobile, sowie die Dokumentation.

Ich bin als Oracle Mitarbeiter natürlich nicht objektiv was die Empfehlung eines Frameworks angeht. Als JavaScript Laie habe ich mir Angular JS, Ionic, React und Oracle JET angeschaut und bin für mich zu dem Ergebnis gekommen, daß Ionic und Oracle JET mir persönlich näher liegen als React und Angular. Ionic und Oracle JET bieten an und sind einfach zu lernen. Ionic basiert auf Angular JS (v1) und wird im Laufe der Zeit auf Angular JS (v2) umgestellt, was nicht ohne Migrationsprobleme sein wird, da Angular JS v1 und v2 selbst nicht kompatibel zueinander sind. Das sollte Sie aber nicht davon abhalten sich mal mit Ionic zu befassen. Am Ende werden Sie mir dann hoffentlich Recht geben, daß Sie mit Oracle JET auf ein gutes Pferd setzen und sich der wachsenden Oracle JET Gemeinde anschließen. Oracle JET is bereits jetzt in zahlreichen Oracle Cloud Produkten in Einsatz da Oracle, wie auch in der Vergangenheit, nicht anbietet was im Unternehmen nicht auch zum Einsatz kommt.

JavaScript im Unternehmen

Auch Oracle programmiert nach wie vor in Java und PL/SQL. Denken Sie in Richtung "Werkzeugkoffer". Mit modernen JavaScript Frameworks und Oracle JET gibt es ein zusätzliches Werkzeug, daß Sie immer dann verwenden sollten wenn es paßt. Ich habe in diesem Artikel 'Survival oft he Fittest' erwähnt und bleibe dabei, daß zunehmend die Einsatzmöglichkeiten von JavaScript über denen von Java und PL/SQL liegen werden. Allerdings ist es nur eine Momentaufnahme und das Pendel kann genau so gut wieder zurück schwingen.

Für Programmiersprachen ist mein Schnappschuss für die Geschichtsbücher der Zukunft **Java**Script, und zwar in genau dieser Formatierung.

Zuviel Arbeit, Zuwenig Skills

Die Welt besteht nicht nur aus Studenten! Und das hat sie auch nie. Um JavaScript im Unternehmen einzusetzen braucht es Ressourcen die mitunter nicht in der geforderten Zahl vorhanden sind. Um diesen Engpass, der nicht nur bei Ihnen vorliegen dürfte, zu überbrücken gibt es im JavaScript Umfeld Lösungen zum deklarativen Entwickeln, die sogenannten Citizen Developer Tools, oder auch Low Code Plattformen. Das Angebot in diesem Bereich ist riesig, was Ihnen eine einfache Google Suche bestätigen wird. Oracle bietet Oracle Mobile Accelerator (MAX) und den Application Builder Cloud Service (ABCS) für Mobile und Web Citizen Development. In einem Artikel[8] auf LinkedIn, aus dem sich die Verwendung von Citizen Development Tools als Low Code Plattform Tools ableiten läßt, habe ich die Zusammenarbeit von Core und Citizen Entwicklern beschrieben.

---

[8] https://www.linkedin.com/pulse/best-methodology-onboarding-citizen-development-frank-nimphius?articleId=6179596878414442496#comments-6179596878414442496&trk=prof-post

Kontaktadresse:
Frank Nimphius
Oracle Corporation
Bleiwäsche, 11
D-02058 Wülfrath

Telefon:	+49 (0) 12-345 6789
Fax:	+49 (0) 12-345 6788
E-Mail	Ihre@adresse.de
Internet:	www.adresse.de