Jubiläum: 20 Jahre JavaScript

Schubladendenken

Julia Schmidt

Zwar entwickelte Brendan Eich JavaScript schon im Mai 1995, ihr öffentliches Debüt machte die damals LiveScript genannte Sprache allerdings erst im September des Jahres, in einer Betaversion des Netscape Navigator 2.0. Ganz gleich wie man rechnet, 20 Jahre JavaScript sind es wert, mit einem Rückblick gefeiert zu werden. Besonders, wenn man sich den Stellenwert vor Augen führt, den die Sprache mittlerweile innehat.

Einen Vortrag zum 20. JavaScript-Geburtstag schloss Eich im Mai mit dem Rat: „Always bet on JavaScript." Tatsächlich wurde der Sprache sehr lang sehr wenig zugetraut. Mittlerweile ist sie allerdings in der Webentwicklung allgegenwärtig und es ist wenig bis überhaupt nicht sinnvoll, ohne aktiviertes JavaScript ins Netz zu gehen. Vor allem vor diesem Hintergrund fällt es schwer zu glauben, dass Entwickler noch vor wenigen Jahren ihre Ernsthaftigkeit bezweifeln lassen mussten, wenn sie JS unter ihren Programmiererfahrungen aufführten.

Der Grund dafür, dass sich das Vorurteil, JavaScript sei gar keine richtige Programmiersprache, sondern lediglich etwas für Amateurprogrammierer, über so viele Jahre gehalten hat, hängt mit Sicherheit auch mit ihren Wurzeln zusammen. 1995 suchte Netscape gerade jemanden für die Entwicklung einer Programmiersprache, die man in den hauseigenen Browser einbetten konnte.

Zunächst war Scheme als Vorlage gedacht, womit man Entwickler Brendan Eich für die Stelle gewinnen konnte. Kurze Zeit später befand sich das Unternehmen allerdings in Verhandlungen mit Sun Microsystems, das seine neue Sprache Java gern in Netscapes Navigator sehen wollte. Schnell war Scheme vergessen und die neue Marschroute gab vor, etwas Java-Ähnliches zu entwickeln – da die Programmiersprache neu war, ließ sich nicht auf bestehende Sprachen zurückgreifen. Kam die Frage auf, warum man nicht einfach komplett auf Java setze, dienten häufig die unterschiedlichen Zielgruppen als Argument. Die professionellen Komponentenschreiber würden weiterhin mit C++ arbeiten und eventuell nach und nach außerdem Java nutzen, während die Entwickler, die eher im Hobbybereich unterwegs waren, ihre eigene Sprache bräuchten, die sich in HTML einbetten ließe.

Machbarkeitsbeweis in T-10

Um zu demonstrieren, dass diese Idee tatsächlich sinnvoll war, brauchte es eine Demo, die Brendan Eich zunächst unter dem von Netscape-Communications-Mitbegründer Marc Andreessen gewählten Namen Mocha zusammenstellen sollte. Zeit hatte er dafür 10 Tage, was wiederum einige Unebenheiten in der Implementierung erklären mag. Innerhalb der zwei Arbeitswochen schrieb Eich einen lexikalischen Scanner und Parser, der an Java angelehnten Stack-Machine-Bytecode generierte, den ein Interpreter verstehen konnte. Außerdem entwickelte er einen Bytecode-Decompiler und eine Standardbibliothek, die aus Zeitmangel allerdings etwas rudimentär blieb. Zudem hoffte Eich, eine Möglichkeit zu finden, beide Sprachen kombiniert einsetzen zu können, sodass Entwickler die Java-Standard-Library nutzen konnten. Viele Probleme bereiteten unter anderem die Umsetzung von Arrays und des Date-Objekts, wobei hier ein Kollege zur Hilfe kam und den Code der Java-Implementierung portierte.

Durch die Orientierung an Java in diesen Fällen entstanden einige weniger ideale Konstruktionen, die Eich häufig angelastet werden (z. B. der y2k-Datums-Bug). Allerdings brachte er auch sein Wissen über Sprachen wie Scheme und Self ein, was Entwicklern heute noch Vorteile beschert. So hatten Programmierer durch das Funktionskonzept von Anfang an viele Freiheiten und sind nicht durch Konstrukte wie Klassen und Namensräume eingeschränkt. In Java sind Aspekte funktionaler Programmierung erst seit Version 8 zu finden. Interessant war für viele Entwickler die Mischung der Paradigmen in JavaScript, da die Sprache auch einen objektorientierten Programmierstil erlaubte. Im Gegensatz zu denen

in Java nehmen die Objekte in JavaScript allerdings Prototypen zur Grundlage, deren Konstruktor eine Funktion ist. Als größten Unterschied zwischen Java und JavaScript führt Eich an, dass Java typisiert ist.

Das Ergebnis konnte trotz der kurzen Implementierungszeit überzeugen und vier Monate später, bei der Veröffentlichung der ersten Betaversion von Navigator 2.0, war aus Mocha LiveScript geworden – in Anlehnung an Netscapes damaliges Server-Produkt LiveWire. Zwar war zu dem Zeitpunkt auch JavaScript als Name für den kleinen Java-Bruder und Werbemaßnahme für die Sprache im Gespräch, allerdings war noch nicht sicher, ob Sun seine Zustimmung geben würde, da die Verhandlungen noch liefen. Die Umbenennung erfolgte aber vergleichsweise schnell, im Zuge der Veröffentlichung einer der nächsten Betaversionen.

Standardisierung im Eiltempo

Um für eine weitere Verbreitung seiner Erfindung zu sorgen, gab Netscape im November 1996 bekannt, JavaScript zur Standardisierung durch Ecma International angemeldet zu haben. Bereits im Juni des darauffolgenden Jahres fand die Veröffentlichung des ersten offiziellen ECMAScript-Standards ECMA-262 statt. Die Spezifikation erhielt in der Folge 1998 und 1999 Updates, Version 4 scheiterte an politischen Differenzen und wurde nie veröffentlicht. Mittlerweile ist ECMAScript 6 beziehungsweise 2015 aktuell, wobei das zuständige Komitee laut dem momentanen Redakteur des Standards, Allen Wirfs-Brook, nun versucht, wieder jährliche Aktualisierungen herauszugeben. Beim Update auf Version 6 im Sommer 2015 fanden unter anderem Lambda-Ausdrücke und Symbole ihren Weg in die Sprache. Was die Zukunft angeht, so soll die nächste ES-Version unter anderem die Arbeit mit asynchronem Code durch entsprechende Schlüsselwörter erleichtern.

Implementierungen des Standards finden sich in aktuellen JavaScript-Engines wie V8. Die letzte offizielle JavaScript-Version erschien mit JS 1.8.5 im Juli 2010. Da ECMAScript als Name allerdings unangenehme Assoziationen mit Hautkrankheiten wecken kann, ist häufig noch immer von JavaScript die Rede, wenn ES gemeint ist. Zwischenzeitlich haben einige Unternehmen wie Microsoft und Google versucht, alternative eigene Sprachen auf dem Markt zu platzieren. Während CoffeeScript gerade wieder von der Bildfläche verschwindet und Google die Bemühungen um sein Dart zurückzuschrauben scheint, geben die Entwickler um Microsofts TypeScript immer wieder Impulse für die ECMAScript-Standardisierung.

Ökosystem macht JS fit für die Zukunft

In den letzten Jahren konnte JavaScript unter anderem durch Projekte wie Node.js von sich reden machen. Das Framework für den serverseitigen Einsatz der Sprache findet in vielen großen Projekten Verwendung und konnte sich mit seiner aktiven Anhängerschaft einen Platz in der Unternehmenswelt sichern. Aber auch andere Tools wie das Webanwendungsframework AngularJS oder Ember.js lassen das Interesse an JavaScript nicht abebben. Zuletzt brachte asm.js beinahe ungeahnte Beschleunigung, wobei die Potenziale noch nicht ausgeschöpft scheinen und der Sprache eine interessante Zukunft verheißen – und das nicht nur für Hobbyentwickler. Dafür sprechen zudem die hohen Platzierungen in diversen Programmiersprachenindizes und die Vielzahl von Projekten, die sie in aussichtsreichen Aufgabenbereichen wie dem Internet der Dinge positionieren.

Eich scheint sich also nicht zu weit aus dem Fenster zu lehnen, wenn er meint, dass man JavaScript nicht unterschätzen sollte. Viele der anfänglichen Kanten haben sich über die Jahre abgeschliffen und das Ökosystem weist genug experimentierfreudige Entwickler auf, die dafür sorgen werden, dass auch in Zukunft mit der Sprache zu rechnen ist. (jul)

Literatur

[1] Stefan Mintert; JavaScript; Klassentreffen; ECMAScript in Version 6 verabschiedet; *iX* 8/2015, S. 123

[2] Brendan Eich; Popularity; https://brendaneich.com/2008/04/popularity/

[3] Brendan Eich; JavaScript at 20 Years; https://brendaneich.github.io/Modern-Web.tw-2015/

[4] Stefan Mintert; WWW-Programmierung; Annäherungsversuch; JavaScript: neue Netscape-Möglichkeiten; *iX* 2/1996, S. 134

Alle Links: www.ix.de/ix1510100