entwickler.press
shortcuts

JavaScript für .NET-Entwickler

Matthias Fischer, Dr. Holger Schwichtenberg
und Martin Möllenbeck

Matthias Fischer, Dr. Holger Schwichtenberg, Martin Möllenbeck

JavaScript für .NET-Entwickler

ISBN: 978-3-86802-529-3

Ein Imprint der Software & Support Media GmbH

1 Einführung in die Grundlagen der Programmiersprache JavaScript

Die Programmiersprache JavaScript wurde ursprünglich von der Firma Netscape unter dem Namen LiveScript entwickelt. Sie war vor allem darauf ausgelegt, einfache Interaktivitäten durch Manipulation des DOMs auf der Webseite hineinzubringen. Seitdem hat sich an den Grundlagen der Sprache selbst relativ wenig verändert, abgesehen von kleineren Erweiterungen mit neuen Versionen. Was sich jedoch seit der Einführung der Sprache grundlegend gewandelt hat, sind die verschiedenen Einsatzgebiete von JavaScript.

Neben Bibliotheken und Frameworks, die den Entwickler auf der Clientseite bei der Entwicklung moderner Oberflächen unterstützen, haben sich auch Technologien und damit Bibliotheken herausgebildet, die einen serverseitigen Einsatz von JavaScript ermöglichen. Dazu zählt etwa Node.js [1]. Projekte wie node-webkit [2] ermöglichen darüber hinaus sogar die Entwicklung plattformübergreifender Desktopanwendungen in JavaScript. Eines haben alle diese Anwendungen jedoch gemeinsam: die Programmiersprache JavaScript.

Auf den ersten Blick betrachtet scheint JavaScript anderen „Klammersprachen“ wie C#, C++ oder auch Java sehr ähnlich. Bei genauerer Betrachtung ergeben sich jedoch größere Unterschiede, die besonders aufgrund der syntaktischen Ähnlichkeit immer wieder zu Verwirrung führen.

JavaScript ist eine dynamisch typisierte, objektorientierte, aber klassenlose Programmiersprache. Trotzdem lassen sich Objekthierarchien und Vererbung mithilfe der so genannten prototypischen Vererbung realisieren. Dazu kommen weitere

Besonderheiten wie der Funktions-Scope, die automatische Konvertierung von Typen sowie das teilweise automatische Einfügen von Semikolons.

Das Typsystem: Dynamische Typisierung

Einer der größten Kritikpunkte an JavaScript ist immer wieder, dass diese Sprache keine strenge Typisierung verwendet, wie wir sie von den .NET-Sprachen, C/C++ oder auch Java kennen. In JavaScript nimmt eine Variable nach jeder Zuweisung eines Werts jeweils den Typ des zugewiesenen Werts an. Dazu kommt, dass Objekte zu jedem Zeitpunkt erweitert und verändert werden können. Damit ist zum Zeitpunkt des Erstellens keine statische Typprüfung des Programmcodes, wie sie beispielsweise ein C#-Compiler durchführt, möglich, da erst zum Zeitpunkt des Zugriffs bekannt ist, über welche Eigenschaften der Typ dann verfügt. Abhilfe schaffen hier Sprachen, die nach JavaScript kompiliert werden können, wie beispielsweise CoffeeScript [3], oder Erweiterungen der Programmiersprache JavaScript, wie sie TypeScript [4] ermöglicht. In diesem Kapitel geht es jedoch erst einmal um die Sprache JavaScript an sich.

Welche Typen gibt es eigentlich in JavaScript? JavaScript kennt insgesamt nur vier bzw. fünf Typen (*Number*, *String*, *Boolean*, *Object* und *Undefined*), wenn man „Undefiniert“ mitzählt. Dabei wird nicht explizit in ganzzahlige und Gleitkommawerte unterschieden. Alle Zahlen sind ausschließlich vom Typ *Number*, der intern als 64-Bit-Gleitkommawert nach dem IEEE-754-(double)-Standard gilt. Um auszudrücken, dass ein Wert keine gültige Zahl ist, wird die Konstante *Number.NaN* (Not a Number) verwendet. Hierbei ist zu beachten, dass diese Konstante nicht auf Gleichheit geprüft werden kann. *NaA* ist nicht gleich *NaN*. Um zu überprüfen, ob eine Variable einen ungültigen Wert enthält, muss die Funktion *Number.isNaN(variable)* verwendet werden. Andere Typen

lassen sich mithilfe der Funktion *Number(variable)* in eine Zahl umwandeln.

Zeichenketten sind in JavaScript 16-Bit-Zeichen nach der UCS-2-Kodierung (nicht UTF-16). Es gibt keinen separaten Typ, um ein einzelnes Zeichen darzustellen, ein Zeichen ist eine Zeichenkette mit der Länge 1. Alle Zeichenketten sind (wie in vielen Programmiersprachen) unveränderlich. Die Gleichheit von zwei Zeichenketten kann mithilfe des Gleichheitsoperators (*==*) geprüft werden. Andere Typen lassen sich mithilfe der Funktion *String(variable)* in eine Zeichenkette umwandeln.

Wahrheitswerte werden mithilfe des Typs *Boolean* abgebildet. In JavaScript ist dieser Typ nicht von einem anderen Typen (z. B. Ganzzahl *(int)*) abgeleitet. Es gibt nur zwei Werte, die dieser Typ annehmen kann, nämlich *true* und *false*. Alle Typen können auch unter Verwendung der Funktion *Boolean(variable)* in einen Wahrheitswert umgewandelt werden. Bei der Umwandlung nach Boolean werden folgende Werte immer nach *false* gewandelt:

```
false, null, undefined, "" (leere Zeichenkette), 0, NaN
```

Alle anderen Werte einschließlich *"0"* oder *"false"* werden nach *true* evaluiert.

Da die Typzuweisung in JavaScript dynamisch erfolgt, muss es einen Typ für Variablen geben, den diese haben, bevor Ihnen der erste Wert zugewiesen wurde. Dieser Typ heißt *undefined*. Alle verbleibenden Elemente sind vom Type *Object*, einschließlich Arrays, Regular Expressions und Funktionen.

JavaScript verfügt über eine automatische Typkonvertierung, die immer dann zum Einsatz kommt, wenn ein Ausdruck mit den aktuellen Typen nicht ausgewertet werden kann. Deshalb ist es wichtig zu wissen, wie sich diese Programmiersprache verhält. Da viele der Operatoren mehrfache Bedeutungen haben, die jeweils von den Typen auf der linken und rechten