2.– 5. September 2013
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Hallo JavaScript

Eine Einführung in JavaScript

Frank Goraus

MATHEMA Software GmbH

JavaScript - Steckbrief

- Scriptsprache
- dynamisch typisiert
- objektorientiert (Prototype)
- Sprachkern als EMCAScript spezifiziert
- vereint objektorientierte, prozedurale und funktionale Paradigmen

Historie (1/2)

- erschienen 1995 mit Netscape Navigator 2.0
- damals noch *LiveScript*
- entwickelt von Brendan Eich (Mozilla)

- Kooperation mit Sun Microsystems zur Interaktion mit Java-Applets
- Umbenennung in JavaScript

- Mai 1996 - JScript-fähige Internet Explorer 3 Beta von Microsoft

Historie (2/2)

- Juni 1997 – Standard ECMA-262 (ECMAScript) von European Computer Manufacturers Association veröffentlich (Zusammenarbeit mit Netscape)
  - April 1998 – ISO/IEC 16262:1998 Information technology – ECMAScript language specification
  - JavaScript 1.4 – erste voll ECMA-262 kompatible Version
- Oktober 1997 – IE4 mit ECMA-262 Support (Sprachumfang von JavaScript 1.1) und DOM-ähnlicher Syntax
- Juni 2002 – Firefox 1.0 mit Javascript 1.5 und DOM Level 1

  ...
- seit 2008 - aktuelle Version 1.8.5 (ECMAScript 5)

Einsatzgebiete

- clientseitig
  - Browser
- Server-side JavaScript
  - z.B.: node.js
- Interpreter
  - z.B. Rhino (Java VM, Mozilla)
  - oder V8 (Chrome+Standalone, Google)

Dateitypen (1/2)

- String
  - *„“, „Hallo Campus“, „1“*
- Number
  - *1, 1.00001*
- Boolean
  - *true, false*

Dateitypen (2/2)

- Function
  - *var func = function(...) {...}; var result = func(...);*
- Object
  - *{ id: 1, ... }*, *[1,2,...]*, */Hallo/g* (Regular Expression), *null*
  - vordefinierte Objekte: Array, Math, Date, RegEx (SelfHTML-Links)
- Undefined
  - *undefined*

typeof

- *typeof „Hallo“* //String
- *typeof new String(„Hallo“)* //Object ?
  - intern: *String {0: "H", 1: "a", 2: "l", 3: "l", 4: "o"}*

Operatoren

- +, -, *, /, %
  - *1 + 1* // 2
  - *"1" + "1"* // „11“
  - *"1" - "1"* // 0
  - *var a = "1"; var b = "1";*
    a + b // „11“
    (+a) + (+b) // 2

- ==, !=, <, >, >=, <=, ===
  - *"1"* == *1* // true
  - *"1"* === *1* // false

Kontrollstrukturen

- *if (...) { ... } else { ... }*

- *switch (...) {*

```
case ... :
    ...;
    break;
default:
    ...;
}
```

Schleifen

- *while (...) { ... }*
- *do { ... } while (...);*
- *for (var i = 0; i < 10; i++) { ... }*

//var array = [„Hallo“, „Campus“, „!“];

- *for (var i in array) { ... }* // i = 0, 1, 2
- *for each (var i in array) { ... }* // i = „Hallo“, „Campus“, „!“

Geltungsbereich von Variablen

- innerhalb der umschließenden Funktion (function scope)
  - *var func = function() {*
    var a = b = 1;
    if (a == b) { var i = 42; }
    alert(i);
    }
    func(); //42
- global an window (global scope)
  - *var a = 42;*
    function echoA() { alert(a); }
    echoA(); //42

Zugriff auf Objekte

- per Punkt-Notation
  - *object.id*
  - *object.machWas(...)*
- per Klammer-Notation
  - *object[„id“]*
  - *object[„machWas“](...)*
- „illegale“ (für Punkt-Notation) Eigenschaften oder Methoden:
  - *object[„a-b“]*
  - *object[„mach was“]()*

dynamische Objekte

- Erweiterung zur Laufzeit:
  - *object.name = „Campus“*
  - *object[„Die Antwort auf alle Fragen“] = function() { alert(42); };*
  - *for (var i=0; i < 10; i++) {*
    *object[„quadrat“+i] = i*i;*
    }
- Löschen zur Laufzeit
  - *delete object.name;*

Erstellung von Objekten

- als Literal:
  - *var meinObjekt = {*
    name : „Campus“,
    machWas : function() { alert(42); },
    id : 0, //Aufpassen mit Komma im IE!!!
    }
- per Konstruktor
  - *function MeinObjekt(parameter) { **this.**eigenschaft = paramter; }*
    var meinObjekt = new MeinObjekt(42);
    alert(meinObjekt.eigenschaft); //42

private Eigenschaften?

- kein explizites Sprachfeature
- per Closure „nachrüstbar“:

```
var privateTest = function() {
  var antwort = 42,
      dasLeben = function() { return antwort; };
  return { dasUniversum : function() { alert(antwort); },
           derGanzeRest : function() { return dasLeben(); } };
}
privateTest().dasLeben(); //TypeError: … has no method ...
privateTest().dasUniversum(); //42
var antwortAufDieFrage = privateTest().derGanzeRest();
```

Fehlerbehandlung

- ähnlich wie in Java

```
var gehtNicht = function() { throw "kaputt"; }
var gehtAuchNicht = function() {
                        throw new Error("ganz kaputt"); }
try {
  gehtNicht();
} catch (exception) {
  alert(exception);
  gehtAuchNicht();
} finally {
  alert("Ich hab's versucht!");
}
```

Document Object Model

- Zugriff auf HTML-Gerüst im Browser
  - vordefinierte Variable document
  - von da aus Zugriff auf den HTML-Baum

  → demo1.html

  - document.forms
  - document.getElementById
  - document.getElementByName

Bibliotheken

- Vielzahl von Erweiterungen:
  - jQuery (& jQuery UI)
  - Prototype (& script.aculo.us)
  - Dojo Toolkit
  - MooTools (& Moo.fx)
  - ExtJS
  - YUI Library (von Yahoo!)
  - ...

Einbinung von JS

- direkt am Element
  - *<input type="text" ... onChange="alert(this.value)" />*
- besser: Script-Block in der Seite
  - *<script type="text/javascript">*
    alert(„Hallo Campus");
    </script>
- optimal: externe JS-Resource
  - bietet Chance auf Wiederverwendbarkeit statt Copy-and-Paste
  - *<script src="script.js">* **</script> (/> ist invalide)**
- Beispiele: → demo2.html

JSON

- JavaScript Object Notation
  - Darstellung von Objekten und Arrays in Textform
  - Serialisierungsmöglichkeit

  - von Douglas Crockford („Javascript – The Good Parts“)
  - spezifiziert als RFC 4627
  - mime type - „application/json“

JSON - Beispiele

- JSON-Objekt:

```
{ "id" : "Campus", "jahrInFolge" : 10,
  "von" : "19.04.2013", "bis" : "20.04.2013" }
```

- JSON-Array

```
[ "1", "2", "3"]
```

- Nutzung in JS:

```
var json = '{ "id" : "Campus", "jahrInFolge" : 10, "von" :
"19.04.2013", "bis" : "20.04.2013" }';
var object = eval("("+json+")");        // „eval is evil“
object = JSON.parse(json);
```

Requests per JavaScript - AJAX

- Asynchronous JavaScript and XML
  - Basis bildet das XMLHTTPRequest-Objekt
- dynamisches Nachladen von Ressourcen
  - HTTP(S)-Anfragen werden aus JS heraus gestartet
  - Browser verweilt auf der selben Seite (kein Page Refresh)
  - Antwort muss vom JS verarbeitet werden, z.B. DOM Update
  - Vorteil: asynchron, kleinere Requests
  - Nachteil: keine Historie

AJAX - Beispiele

- „normales“ Javascript:
  - http://de.wikipedia.org/wiki/XMLHttpRequest
- jQuery:
  - http://api.jquery.com/jQuery.ajax/

→ demo3.html

Barrierefreiheit

- JS-freie Bedienung muss gewährleistet sein
- JS nur für „Convenience“ und „Eyecandy“ verwenden

- Anti-Beispiele:
  - JS-Dialoge, welche zum Fortschritt in der Anwendung benötigt werden
  - Forms, die nur per JS abgeschickt werden (und sonst keinerlei Submit-Button besitzen)
  - Bilder, Divs, ... mit elementaren JS-Funktionen darauf

Cross-Site-Scripting (XSS)

- Einschleußen schadhaften Codes
- „ungeprüftes“ Einbinden von Input
  - Ausgabe von Suchanfragen
  - Übernahme von Parametern

Vielen Dank!

Frank Goraus

MATHEMA Software GmbH