software studio

javascript in 3 minutes

Daniel Jackson

programming the browser

Java

› 1990: project at Sun to replace C++
› 1994: "Oak"retargeted to the web for "applets"
› Java takes off, first safe language in widespread use

Javascript

› 1995: "Mocha" project at Netscape
› Javascript takes off, included in Microsoft's IE
› 1996: submitted to Ecma as standard

today

› Java alive and well server-side
› but JS dominates client-side
› making inroads server-side too (eg, node.js)

on javascript, from its inventor

JS had to "look like Java" only less so, be Java's dumb kid brother or boy-hostage sidekick. Plus, I had to be done in ten days or something worse than JS would have happened.

—Brendan Eich on Javascript

the good parts

In Javascript, there is a beautiful, elegant, highly expressive language that is buried under a steaming pile of good intentions and blunders. The best nature of Javascript is so effectively hidden that for many years the prevailing opinion of Javascript was that it was an unsightly, incompetent toy. My intention here is to expose the goodness in Javascript, an outstanding dynamic programming language...

Deep down, Javascript has more in common with Lisp and Scheme than with Java. It is Lisp in C's clothing.

—Douglas Crockford in *Javascript: The Good Parts*

syntax

statements like Java

› while, for, if, switch, try/catch, return, break, throw

comments

› use //, avoid /**/

semicolons

› inserted if omitted (yikes!)

declarations

› function scoping with **var**

functions

› are expressions; closures (yippee!)

```
var MAX = 10;
var line = function (i, x) {
    var l = i + " times " + x
          + " is " + (i * x);
    return l;
}
var table = function (x) {
    for (var i = 1; i <= MAX; i += 1) {
        console.log(line(i, x));
    }
}
// display times table for 3
table(3);
```

```
1 times 3 is 3
2 times 3 is 6
3 times 3 is 9
4 times 3 is 12
5 times 3 is 15
6 times 3 is 18
7 times 3 is 21
8 times 3 is 24
9 times 3 is 27
10 times 3 is 30
undefined
>
```

basic types

primitive types

› strings, numbers, booleans
› operators autoconvert

arrays

› can grow, and have holes

funny values

› undefined: lookup non-existent thing
› null: special return value

equality

› use ===, !==

```
> 1 + 2
3
> 1 + '2'
"12"
> 1 * 2
2
> 1 * '2'
2
```

```
> a = []
[]
> a[2] = 'hello'
"hello"
> a.length
3
> a[1]
undefined
> a[2]
"hello"
> a[3]
undefined
```

```
> 1 === 1
true
> 1.0 === 1
true
>'hello' === 'hello'
true
> [] === []
false
```

objects

literals

› *o = {prop: val, ...}*

properties

› get: *x = o.p*
› set, add: *o.p = e*
› delete: *delete o.p*

```
> point = {x: 1, y: 2}
Object
 1. x: 1
 2. y: 2
 3. __proto__: Object
> point.x
1
> point.z
undefined
> point.z = 3
3
> point.z
3
> delete point.z
true
> point.z
undefined
```

prototypes

› lookup along chain

```
> var Point = function (x, y) {this.x = x; this.y = y;}
undefined
> Point.prototype.magnitude = function () {return
Math.sqrt(this.x * this.x + this.y * this.y);}
function () {return Math.sqrt(this.x * this.x + this.y *
this.y);}
> p = new Point(1,2)
Point
> p.x
1
> p.magnitude
function () {return Math.sqrt(this.x * this.x + this.y *
this.y);}
> p.magnitude()
2.23606797749979
```

good, bad & awful

good	*bad, can work around*	*awful, stuck with these*
uniform & simple	default variable scope	no access control
first class functions	==, NaN	no immutable lists
lexical closures		no standard libraries
properties		no packaging
prototypes		new & this

recommended reading

Douglas Crockford. *JavaScript: The Good Parts*.

Short, entertaining, well explained. In good taste.

Candid about bad and awful parts too.

Sometimes need more explanation (eg, constructors)

Stoyan Stefanov. *JavaScript Patterns*.

Also fairly short, and well explained.

Sophisticated collection of useful patterns.

6.170 Software Studio
Spring 2013