JavaScript: the Good, the Bad, and the Ugly

Nathaniel T. Schutta
Who am I?

- Nathaniel T. Schutta
- Foundations of Ajax & Pro Ajax and Java Frameworks
- UI guy
- Author, speaker, teacher
- More than a couple of web apps
The Plan

- History
- Basics
- Browser issues
- Objects
- Prototype inheritance
- Tools
- JavaScript 2
Not a toy!
JavaScript has been around for a while.
1995.
Began as LiveScript.
Marketing!
Suffered from some issues...
Lack of tools.
Browsers!
“Object not found.”
Many dismissed it as “toy for designers.”
But...
Developers care again!
Ajax.
Suffers from the “EJB issue.”
Powerful language.
“Revenge of the Nerds”

http://www.paulgraham.com/icad.html
Paul Graham - 9 ideals.
1. Conditionals.
2. Functions as data.
3. Recursion.
4. Dynamic typing.
5. Garbage collection.
6. Programs as expressions.
7. Symbol type.
8. Code as tree of symbols.
9. Read/Run/Compile.
“The Next Big Language”

Is JavaScript the NBL?
Garbage collected.
Not particularly pretty...
C-like syntax (for better or worse).
Dynamically typed - duh.
ECMAScript 4 supports optional static typing.
Performance - Tamarin.

http://www.mozilla.org/projects/tamarin/
Tools - Firebug, plugins.
Most of the “kitchen sink.”
Runs on lots of platforms - including the JVM.
Could be the NBL...
Ruby like?
“Rhino on Rails”

Orto - JVM written in JavaScript.

http://ejohn.org/blog/running-java-in-javascript/
JS-909.

http://www.themaninblue.com/experiment/JS-909/
JS books, 07 market share went up > 3%.

Java went down...
JavaScript is cool again!
Basics.
JavaScript has most of what you expect...
//Comments
var foo = "variables";
Notice what’s missing?
JS is “loosely” typed.
BTW, var is key.
If you forget it...global!
Easy to overwrite something...
Reserved words.
Objects!
Inheritance even - enter the prototype.
More on this later.
Datatypes
Strings.
Numbers.
Booleans.
Arrays.
Functions!
That’s right - functions as a data type.
var square = function(x) { return x*x; }
Used repeatedly in Prototype, Dojo, YUI, etc.
How do you check types?
function $(element) {
    if (arguments.length > 1) {
        for (var i = 0, elements = [], length = arguments.length;
            i < length; i++)
            elements.push($($(arguments[i])));
        return elements;
    }
    if (typeof element == 'string')
        element = document.getElementById(element);
    return Element.extend(element);
}
Flow control.
Loops.
try/catch/finally.
Keep in mind:
Designed with low barrier of entry in mind.
Supposed to be forgiving.
Might have gone a bit far.
Basically everything we’ve come to expect.
But, JavaScript is a little...different.
Browser issues.
Though *mostly* consistent, still some headaches.
Browser: {
    IE:   !!window.attachEvent &&
          navigator.userAgent.indexOf('Opera') === -1),
    Opera: navigator.userAgent.indexOf('Opera') > -1,
    WebKit: navigator.userAgent.indexOf('AppleWebKit/') > -1,
    Gecko: navigator.userAgent.indexOf('Gecko') > -1 &&
           navigator.userAgent.indexOf('KHTML') === -1,
    MobileSafari: !!navigator.userAgent.match(/Apple.*Mobile.*Safari/)}

http://www.webaim.org/blog/user-agent-string-history/
Simple boolean checks.
Clearer code.
Different versions of JavaScript.
FF 3 supports 1.8.
IE...not so much.
May get better with 8!
Five years?
Still quirks.
What to do?
Develop to a library.
Several great libraries!
Let them handle the incompatibilities.
“VM” as it were...
Worst case - sniff.
Browser sniffing.

http://andrewdupont.net/2007/04/04/browser-sniffing/
Objects.
Objects - collection of attributes.
Basically - a hash.
Object literal notation.
var foobar = {foo: "foo", bar: "bar"};
Very useful.
new Ajax.Request('/DesigningForAjax/validate', {
  asynchronous: true,
  method: "get",
  parameters: {zip: $F('zip'), city: $F('city'), state: $F('state')},
  onComplete: function(request) {
    showResults(request.responseText);
  }
});
How do you know if an object has an attribute?
Could use “in”.
if ("foo" in bar) { //do something}
Or just ask.
If not, returns undefined.
We get properties with the . notation.
Could use ["prop"].
Prototype inheritance.
JavaScript doesn’t have “classical” inheritance.
Each object has a prototype.
Takes a little...getting used to.
Prototypal Inheritance in JavaScript

http://javascript.crockford.com/prototypal.html
Hidden link.
“Walk the tree.”
Not technically inheritance.
Can add members to the prototype.
Confused?
Libraries help.
Make JS more “classical.”
Class.create === constructor function.
Object.extend ==

property copier.
Ajax.Request = Class.create();
Object.extend(Array.prototype, Enumerable);
Speaking of classes...
JS has open classes.
Feel like adding something to String?
Go ahead!
Prototype does it quite a bit actually.
JSON support.
strip, stripTags, stripScripts, extractScripts, truncate, escapeHTML, camelize, capitalize, toJSON...
<cough>Ruby</cough>
Tools.
First rule of JavaScript:
Use Firefox.
Second rule of JavaScript:
Use Firefox.
JavaScript console.
Web developer tools.
Firebug!
```javascript
var browserId = "w3c";
if (document.all)
    browserId = "ie";

document.write('<script language="javascript" src="/toolkit/dom_+browserI

///////////////////////////////////////////////////////////////////////////
// DOM Iteration

function $(id)
{
    return document.getElementById(id);
}

function findNode(node, criteria, container)
{
    while (node)
    {
        if (criteria(node))
            return node;
```
JavaScript debugging!
Console - real errors.
Log to the console.
CSS/HTML inspector.
Modify CSS.
DOM explorer.
Works in FF but not IE?
Firebug Lite.

http://getfirebug.com/lite.html
YSlow.
http://www.getfirebug.com/
http://developer.yahoo.com/yslow/
JSLint.
VERY strict.
Will it run on IE/FF/Opera?
Cross Check.
Doesn’t require you to install all the browsers!
http://www.thefrontside.net/crosscheck
IDE support is getting better.
JavaScript 2.
ECMAScript 4.
Oops - Harmony.
4 vs 3.1.
How much to implement?
3. I group wanted things that would impact 4.
Bit of infighting.
The Oslo meeting.
ECMAScript Harmony.
ES 3.1 + “best of” ES 4.
4 was getting big...
Refocus on simplicity.
Death to packages, namespaces and early binding.
Some notion of classes.
Some defacto standards.
Getters/setters.
Lots of work left.
Better chance this time.
http://ejohn.org/blog/ecmascript-harmony/
http://alex.dojotoolkit.org/2008/08/harmony-fallout/
http://yuiblog.com/blog/2008/08/14/premature-standardization/
https://mail.mozilla.org/pipermail/es-discuss/2008-August/003400.html
Keep in mind...
ES3 - 1999.
1999?
When will we see Harmony?
Don’t hold your breath.
Five years? More?
JavaScript is a real language.
Very powerful.
Widely used.
It’s not a toy!
Want more?
Number of new books.
Questions???
Thanks!

Please complete your surveys.