Sergio Pereira

WI.NET Users Group – July 14th 2009

Advanced JavaScript

BEYOND THE CURLY BRACES

WHOIS

About Sergio


sergio@sergiopereira.com


http://sergiopereira.com/blog


@sergiopereira

http://chicagoalt.net

AGENDA

JavaScript trivia

- What, when, who, where

Drive-by shooting

- Data types, functions, DOM, XMLHttpRequest

Gotchas for C# programmers

Idiomatic JavaScript

~~**Objects and inheritance**~~

Coding conventions and pitfalls

ADVANCED JAVASCRIPT

ADVANCED JAVASCRIPT

BUT FIRST THE BASICS

- JavaScript history (fear not, it's short)
- Data types
- The Browser DOM is not JavaScript
- XMLHttpRequest is not JavaScript
- Common mistakes for C# folk

ADVANCED JAVASCRIPT

Once upon a time…

- 1995: Mocha, LiveScript, JavaScript (Netscape)
- 1995: NS Navigator 2.0 (Java + JavaScript)
- 1995: Confusion started
- 1996: MS IE 3.0 calls it JScript (not quite the same)
- 1997: ECMA-262, ECMAScript

ADVANCED JAVASCRIPT

World's Most Misunderstood Language

- The Name
- Mispositioning
- Design Errors
- Bad Implementations
- The Browser
- Bad Books
- Substandard Standard
- JavaScript is a Functional Language

ADVANCED JAVASCRIPT

But The Truth Is

- JavaScript is not a toy
- Solves real problems as a real language
- Quite fancy sometimes
- Not freaking Java (or C#), OK?

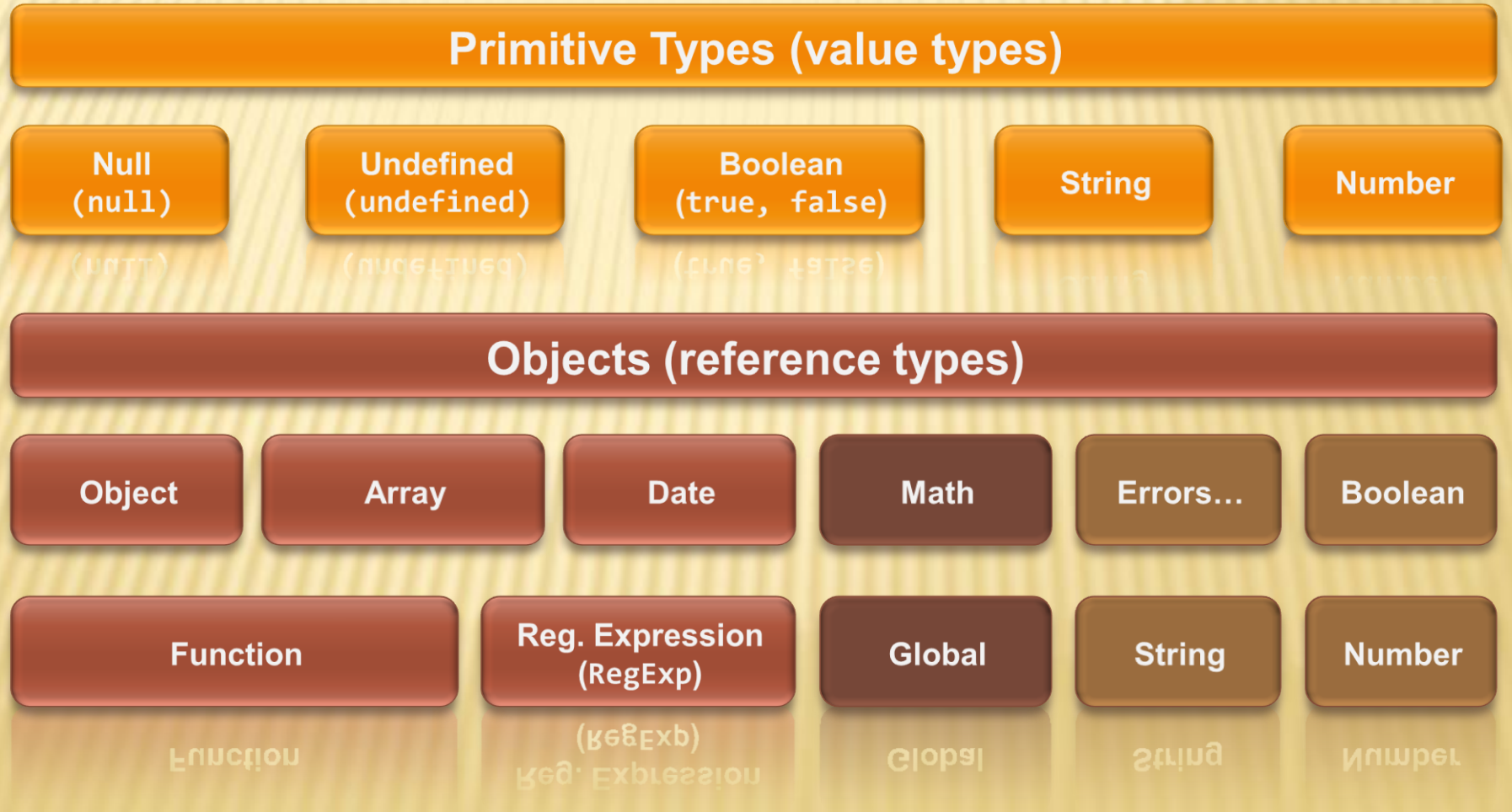
ADVANCED JAVASCRIPT

JavaScript in a Snapshot

- Dynamic
- Objects: glorified containers
- Prototypal inheritance
- Textual delivery
- Lambdas
- Global variables tie it all

ADVANCED JAVASCRIPT

Data Types



ADVANCED JAVASCRIPT

Null (null)

Undefined (undefined)

Default value for variables, parameters, and missing object properties

```
var name = "Homer";
var hairColor = null;
```

```
var city = "Springfield";
var state;
```

ADVANCED JAVASCRIPT

Boolean
(true, false)

```
var firstTime = true;
var found = false;
```

String

```
var firstName = "Selma";
var lastName = 'Bouvier';
```

Number
(IEEE-754, 64-bit, Double)

```
//integer literals
var age = 26;
var temperature = -8;
var permission = 0775; //=509
var flags = 0x1c; //=28

//Double prec. literals
var height = 6.15;
var factor = 5.73e-1; //=.573
//NaN
var what = "x"/3; //= NaN
```

ADVANCED JAVASCRIPT

Array

```
var names = new Array();
names[0] = 'Steve';
names[1] = 'Chuck';
names[names.length] = 'Jeff';
names.push('Jason');
var names = [ ];
//repeat as above…

//or, preferred
var names = ['Steve', 'Chuck',
   'Jeff', 'Jason'];
```

ADVANCED JAVASCRIPT

Date

```
var today = new Date();

//milliseconds since 1/1/1970
var d = new Date(1234);

//explicit
var xmas = new Date(2008, 11, 25);

//parsing, locale-dependent, avoid
var xmas = new Date(Date.parse('12/25/2008'));
```

ADVANCED JAVASCRIPT

RegExp

```
var patt1 = new RegExp('\\d{2}$');
var patt2 = /\d{2}$/;
var patt3 = new RegExp('\\d{2}$', 'gi');
var patt4 = /\d{2}$/gi;

alert(patt1.test('abc78')); // → true

'12a78'.replace(/\d{2}$/, 'XYZ'); //→ '12aXYZ'
'a1b2c7d8e'.split(/\d/); //→ ['a','b','c','d','e']
```

ADVANCED JAVASCRIPT

Function

```
//normal declaration
function play(chord) {
    alert('playing ' + chord);
}

//normal invocation
play('Cm'); // → 'playing Cm'
```

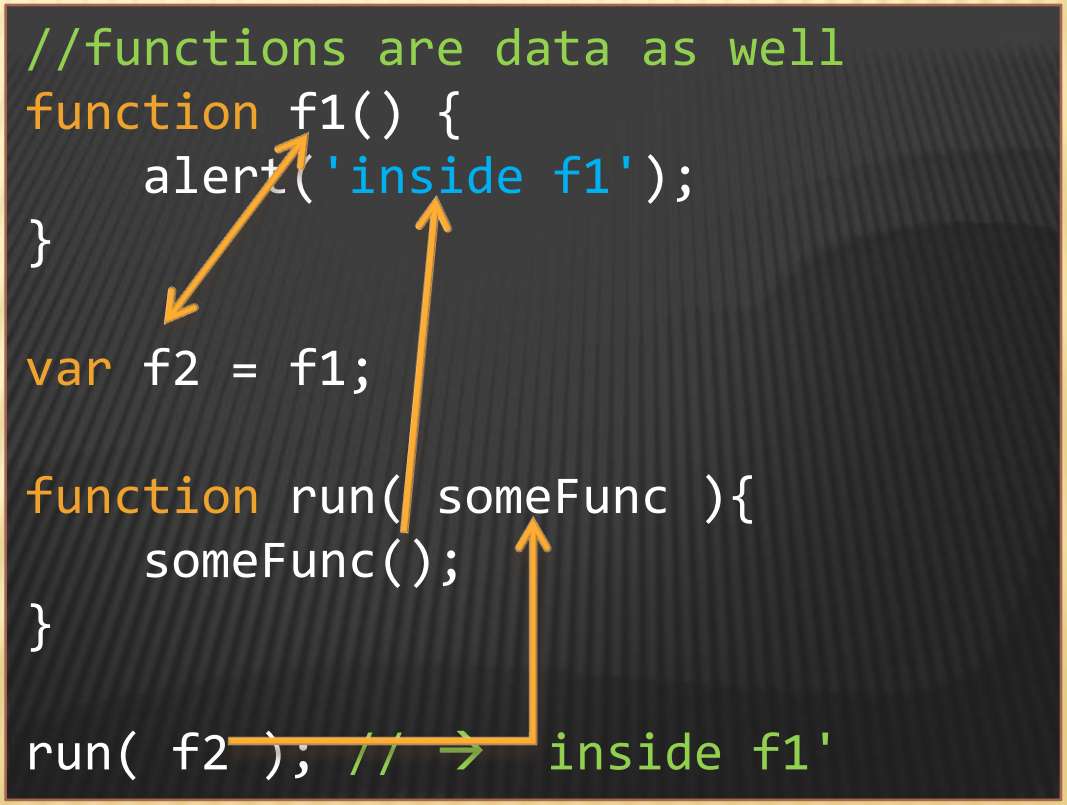
ADVANCED JAVASCRIPT

Function

```
//functions are data as well
function f1() {
    alert('inside f1');
}

var f2 = f1;

function run( someFunc ){
    someFunc();
}

run( f2 ); // → 'inside f1'
```

ADVANCED JAVASCRIPT

Function



ADVANCED JAVASCRIPT

Function

```
//functions don't need a name
function run( someFunc ){
    someFunc();
}

run(function () {
  alert('inside function');
}); // → 'inside function'
```

ADVANCED JAVASCRIPT

Function

```
//functions don't need a name
function run( someFunc ){
    someFunc();
}

run(    function () {
            alert('inside function');
        }
    ); // → 'inside function'
```

ADVANCED JAVASCRIPT

Function

```
//parameter lists are optional
function concat(){
    var res = '';
    for (var i=0; i<arguments.length; i++) {
        res += arguments[i];
    }
    return res;
}

alert( concat(123, 'abc', [7, 8, 9]) );
// → '123abc7,8,9'
```

ADVANCED JAVASCRIPT

Function

```
//can be nested
function prettyPrint(value, currency, rate){
    function formatCurr(num) {
        return num + ' ' + currency;
    }
    function convert() {
        return rate * value;
    }
    var converted = convert(value);
    var text = formatCurr(converted);
    alert(text);
}
prettyPrint(10.5, 'EUR', 0.797); // → 8.3685 EUR
```

ADVANCED JAVASCRIPT

Function

```
//single-use, pinned functions
(function (someParameter){
    //do something
    // …
})(someArgument);
```

Invocation!

ADVANCED JAVASCRIPT

Invocation and the problem with `this`

```
function func(arg1, arg2){
    return [this, arg1, arg2];
}
func(123, 456); // → [window, 123, 456]

var car = new Object();
car.start = func;
car.start('now', 9); // → [car, 'now', 9]

func.apply(car, ['p1', 'p2']); // → [car, 'p1', 'p2']
func.call(car, 'p1', 'p2'); // → [car, 'p1', 'p2']
```

ADVANCED JAVASCRIPT

Inner functions and this

```
function calculateTotal(){
    function getTax(){
        return this.price * this.tax;
    }
    return this.price + getTax() +
           this.shippingCost;
}
var order = new Object();
order.price = 10.25;
order.tax = 0.07;
order.shippingCost = 5;
order.getTotal = calculateTotal;
order.getTotal();
```

ADVANCED JAVASCRIPT

Inner functions and `this` (the fix)

```
function calculateTotal(){
    var that = this; // ← very common
    function getTax(){
        return that.price * that.tax;
    }
    return this.price + getTax() +
           this.shippingCost;
}
var order = new Object();
order.price = 10.25;
order.tax = 0.07;
order.shippingCost = 5;
order.getTotal = calculateTotal;
order.getTotal();
```

ADVANCED JAVASCRIPT

Closures

```
function createFuncArray() {
    var funcs = [ ];

    for (var n=0; n<10; n++) {
        funcs[n] = function () {
            alert('The number is ' + n);
        };
    }
    return funcs;
}

var allFunctions = createFuncArray();
var show = allFunctions[6];
show();// → Can you guess?
```

ADVANCED JAVASCRIPT

Other Function-related gotchas

- Variables are scoped by function, not block.
- No overloading, last one wins. Use default values.
- `arguments` and `this` not passed to inner functions.

ADVANCED JAVASCRIPT

Object

```
//Explicit assembly
var guitar = new Object();
guitar.stringCount = 6;
guitar.make = 'Gibson';
guitar.model = 'Les Paul';
alert(guitar.make); // → 'Gibson'
```

ADVANCED JAVASCRIPT

Object

```
//factory function
function createGuitar(strings, make, model){
    var guitar = new Object();
    guitar.stringCount = strings;
    guitar.make = make;
    guitar.model = model;
    return guitar;
}
var g = createGuitar(6, 'Gibson', 'Les Paul');
alert(g.make); // → 'Gibson'
```

ADVANCED JAVASCRIPT

Object

```
//Object literal notation
var emptyObject = {};

var guitar = {
    stringCount: 6,
    make: 'Gibson',
    model: 'Les Paul'
}; // ← don't forget the ';'
//or
var guitar = {
    'stringCount': 6,
    'make': 'Gibson',
    'model': 'Les Paul'
};
```

ADVANCED JAVASCRIPT

Object

```
//notation for methods
var guitar = {
    stringCount: 6,
    make: 'Gibson',
    model: 'Les Paul',
    play: function (chord) {
        alert(chord + ' from a ' + this.model);
    }
};
guitar.play('C#'); // → 'C# from a Les Paul'
```

ADVANCED JAVASCRIPT

Object

Wait, was that JSON? Not exactly.

```
//JSON (see http://www.json.org/ )
var guitarJson = "{ " +
    " 'stringCount': 6, " +
    " 'make': 'Gibson', " +
    " 'colors': [ 'black', 'white', 'red'], " +
    " 'model': 'Les Paul' " +
    "} ";
```

JSON is basically a collection of `name:value` pairs.

- `name` is a `string` literal
- `value` can be: a `string`, a `number`, `null`, `true`, `false`, an `array`, or another JSON `object` literal.

ADVANCED JAVASCRIPT

Object

```
//constructor function
function Guitar(strings, make, model){
    this.stringCount = strings;
    this.make = make;
    this.model = model;
    this.play = function (chord) {
        alert(chord + ' from a ' + this.model);
    };
}
var g = new Guitar(6, 'Gibson', 'Les Paul');
g.play('C#'); // → 'C# from a Les Paul'
```

ADVANCED JAVASCRIPT

Member access notations

```
//good 'ole dot notation
guitar.model = 'Gibson';
var p = guitar.price;

//indexer notation
guitar['model'] = 'Gibson';
guitar['play']('C#');
```

ADVANCED JAVASCRIPT

What about `document`, `window`, etc?

- Not JavaScript.
- Part of the DOM.
- Provided by the hosting environment.
- Like the DTE in VS macros and other objects in Office automation.

ADVANCED JAVASCRIPT

What about `XMLHttpRequest (xhr)` ?

- Not JavaScript.
- Not even part of the DOM.
- Provided by the hosting environment.

ADVANCED JAVASCRIPT

Identifiers

- Start with a letter, _ or $ followed by more of these and numbers.
- Convention: start variables and functions with lower case.
- Convention: start constructors with upper case.
- Avoid starting with _ or $

ADVANCED JAVASCRIPT

Reserved Words

abstract boolean **break** byte **case catch** char class const **continue** debugger **default delete do** double **else** enum export extends **false** final **finally** float **for function** goto **if** implements import **in instanceof** int interface long native **new null** package private protected public **return** short static super **switch** synchronized **this throw** throws transient **true try typeof var** volatile **void while with**

```
// Bug in the language
driversLicense.class = 'B'; // → invalid!
driversLicense['class'] = 'B'; // → huzzah!
```

ADVANCED JAVASCRIPT

JavaScript Idioms: Truthy & Falsy

```
//Falsy values
var f[1] = false; // D'oh!
var f[2] = null;
var f[3] = undefined;
var f[4] = 0;
var f[5] = "";
var f[6] = NaN;

if( f[X] ){
    // never gets here
}
```

```
//Truthy values
var t[1] = true;
var t[2] = new Object();
var t[3] = "1";
var t[4] = "0";     // ← !!
var t[5] = "false";// ← !!

if( t[X] ){
    // gets here always
}
```

ADVANCED JAVASCRIPT

JavaScript Idioms: || Default operator

```
//returns the first Truthy or last operand.
//e.g.: overriding missing arguments
var sortOrder;
if (arg) {
    sortOrder = arg;
} else {
    sortOrder = 'Name ASC';
}

//same as
var sortOrder = arg || 'Name ASC';
```

ADVANCED JAVASCRIPT

JavaScript Idioms: && Guard operator

```
//returns the first Falsy or last operand.
// e.g.: avoiding null reference errors
var obj2;
if (obj1) {
    obj2 = obj1.prop;
} else {
    obj2 = obj1;
}

//same as
var obj2 = obj1 && obj1.prop;
```

ADVANCED JAVASCRIPT

JavaScript Idioms: === and !==

```
if (0 == "") {
  alert('Hey, I did not expect to see this.'); //displayed
}
if (0 === "") {
  alert('This will not be displayed.'); //not displayed
}

if (1 != true) {
  alert('Hey, I expected to see this.'); //not displayed
}
if (1 !== true) {
  alert('This will be displayed.'); //displayed
}
```

ADVANCED JAVASCRIPT

JavaScript Idioms: Namespaces

```
var ACME = {
    version: '7.1',
    formatCurrency: function(num){ /* … */ },
    //…
};

ACME.Ajax = {
    ajaxifyForm: function(formId){ /* … */ },
    showError: function(){ /* … */ },
    //…
};

ACME.Ajax.ajaxifyForm('myForm');
```

ADVANCED JAVASCRIPT

Coding convention: Always use curly braces

```
if (something)
    doSomething();
    doSomethingElse();

if (something){
    doSomething();
}
    doSomethingElse();
```

ADVANCED JAVASCRIPT

Coding convention: Always end lines with punctuation

```
return
    tax + shipping + price;
```

Parsed as

```
return;
    tax + shipping + price;
```

```
return tax +
    shipping + price;
```

```
var amount = tax
       + shipping
       + price;

var amount = tax +
       shipping +
       price;
```

Coding convention: Globals

- Avoid them.
- Tend to clobber each other.
- Namespaces are OK.
- Name them with ALL_CAPS

ADVANCED JAVASCRIPT

Coding convention: `eval()` is evil

```
var p = eval('obj.' + propertyName);
var p = obj[ propertyName ];

//hidden evals
setTimeout("alert('Time is up!');", 9000);
setTimeout( function () {alert('Time is up');}, 9000);
var func = new Function('p1', 'p2', 'return p1+p2;');
```

ADVANCED JAVASCRIPT

Coding convention: Declare variables at the top of function

```
function f1(){
    var a, b, c;
    a = getA();
    if (a) {
        b = calcB(a);
    } else {
        c = getC();
        b = calcB(c);
    }
}
```

ADVANCED JAVASCRIPT

Coding convention:
Do not use assignments as expressions

```
if ( flag = getFlag() ) {
    alert(flag);
}

//prefer:
var flag = getFlag();
if (flag) {
    alert(flag);
}
```

QUESTIONS ?

REFERENCE

- JavaScript: The Good Parts by *Douglas Crockford*.
- JavaScript: The Definitive Guide by *David Flanagan.*
- Videos: Search for *Douglas Crockford*.
- Shameless plug: my blog, JavaScript category.


sergio@sergiopereira.com


http://sergiopereira.com/blog


@sergiopereira


http://chicagoalt.net