

Rapid DOM Scripting and Ajax Development

What is it?

It's JavaScript.

(Shouldn't this just be called
Rapid JavaScript Development?)

JavaScript is Powerful

It's power is under-utilized

Functional Programming

Closures

Anonymous Functions

```
function loadImage() { }
```

```
function () { }
```

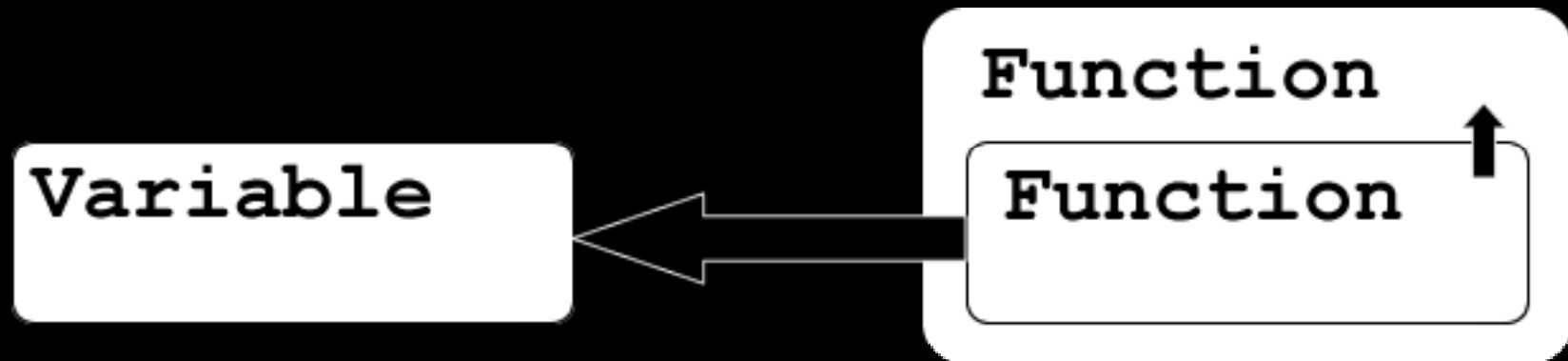
Functional Programming

Useful for Callbacks
Partial Application
Internal Iterators

<http://www.vivabit.com/bollocks/2006/06/13/cleaner-callbacks-with-partial-application>

Closures

Retain access to internal scope



Closures

Retain access to internal scope

```
function sayHello(name) {  
    var text = 'Hello ' + name; // local variable  
    var sayAlert = function() { alert(text); }  
    return sayAlert;  
}
```

http://blog.morrisjohns.com/javascript_closures_for_dummies

Keys to Rapid Development

1. Maintainability
2. Extensibility
3. Flexibility

1. Maintainability

Use a Coding Style

Be consistent

Examples: Use camelCase, Upper case for "Classes",
All caps for CONSTANTS

Use a Coding Style

```
var MYCONST = 5;
var ClassName = function(){
    this.initialValue = MYCONST;
}

ClassName.prototype.methodName = function(){ }
```

Use a Namespace

Self-contained

```
var SNOOK = {  
    prepareCommentForm:function()  
    {  
        /* code goes here */  
    }  
};
```

```
SNOOK.prepareCommentForm();
```

Self Documenting

Use underscores to indicate internal methods or
create private/privileged methods

```
function myObject() { }  
myObject.prototype._loopBack = function(){ }  
myObject.prototype.getList = function(){ }
```

<http://www.crockford.com/javascript/private.html>

<http://www.dustindiaz.com/javascript-private-public-privileged/>

Singleton

Either through Object Notation or
using new with an anonymous function

```
var myObject = {  
    loopBack:function(){ /*code goes here*/ },  
    getList:function(){ /*code goes here*/ }  
}
```

```
var myObject = new function(){ /*code goes here*/ }
```

<http://novemberborn.net/javascript/scopes-and-closures-funk>

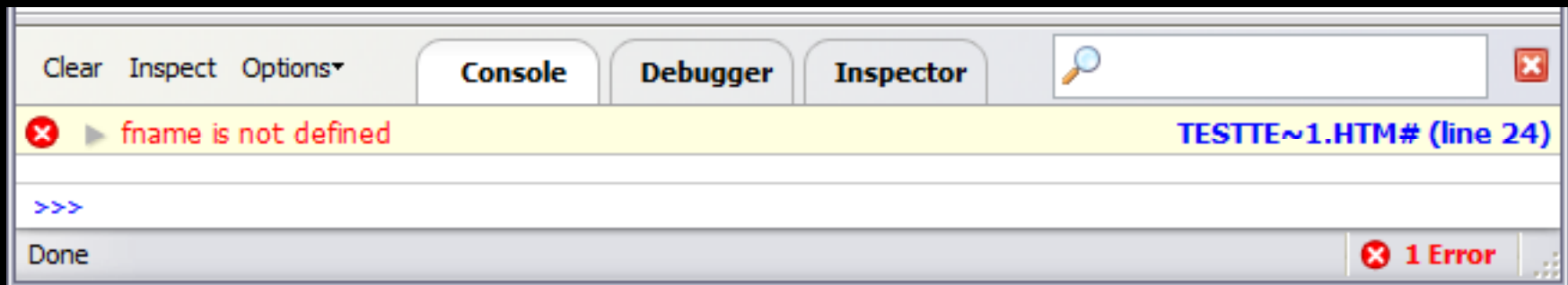
Object Notation Syntax

```
var myObject = {  
    propertyName:value,  
    propertyName:value  
}
```

Value can be string, number, function,
or other object literals

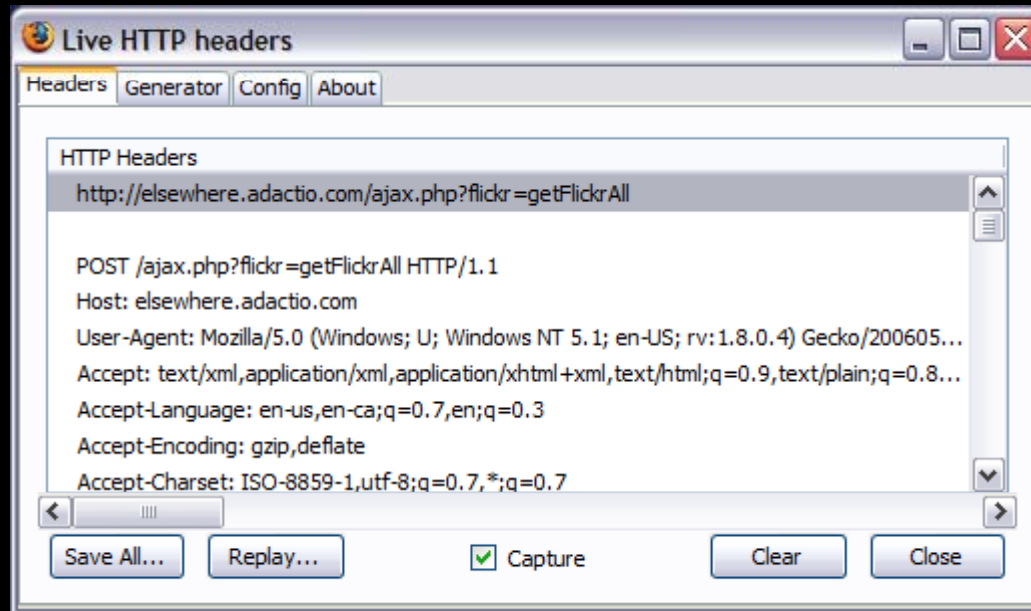
Use Debug Tools

E.g. Firebug Extension for Firefox



Trace Http Calls

Live HTTP Headers Firefox Extension



2. Extensibility

Callbacks

Call a function when you're done something

Callbacks

```
var myObject = {  
  _callbacks:[],  
  register:function(func){this._callbacks.push(func);},  
  getList:function(){  
    var len = this._callbacks.length;  
    for(var i=0; i<len;i++) this._callbacks[i]();  
  }  
}
```

```
function listcalled(){alert('list called!')}
```

```
myObject.register(listcalled);  
myObject.getList();
```

Override

Do your own thing first

```
// this already exists: s5.go(step)

_old = s5.go;

s5.go = function(step){
    /* perform multi-frame iteration here */
    _old(step);
}

s5.go(1); // move forward one frame
```

3. Flexibility

Don't be specific

don't use predefined class names or element names

```
function makeSpecial(){
    var links = document.getElementsByTagName('a');
    for(var i=0;i<links.length;i++)
    {
        if(links[i].className == 'special'){
            links[i].style.color = 'pink';
        }
    }
}
```

Don't be specific

don't use predefined class names or element names

```
function makeSpecial(class, color){
  var links = document.getElementsByTagName('a');
  for(var i=0;i<links.length;i++)
  {
    if(links[i].className == class){
      links[i].style.color = color;
    }
  }
}
```

Avoid Library Calls

make the calls earlier rather than later

```
function makeBlack(elementname){  
    $(elementname).style.color = '#000';  
}
```

```
function makeBlack(element){  
    element.style.color = '#000';  
}
```

Change Class Names

do not change styles directly

Separation of behavior from presentation from content

```
element.style.color='#000';  
element.style.backgroundColor='#F00';  
element.style.fontSize='12px';  
  
element.className = 'error';
```

Attach to Elements

Easily accessed during events

```
var el = document.getElementById( 'open' );  
el.myX = 5;  
el.myY = 6;  
  
el.onclick = function(){  
    alert(this.myX + this.myY);  
    return false;  
}
```

Keys to Rapid Development

1. Maintainability
2. Extensibility
3. Flexibility



script.aculo.us



Rico



Libraries



(aka don't reinvent the wheel)



dōjō

Libraries generally...

1. Reduce the mundane
2. Create effects

Reduce the Mundane

Prototype, Jquery, Behavior

```
document.getElementById('id')
```

VS

```
$('#id')
```

Reduce the Mundane

Prototype, Jquery, Behavior

```
var els = document.getElementsByTagName('*');  
var list=[];  
for(var i=0;i<els.length;i++){  
    if(els[i].className == 'question'){  
        list.push(els[i]);  
    }  
}}
```

VS

```
$$('.question')
```

Reduce the Mundane

Modifying Class Names

Prototype

```
$( 'element' ).addClassName( 'error' );  
$( 'element' ).removeClassName( 'error' );
```

jQuery

```
$( 'p#element' ).addClass( "error" );  
$( 'p#element' ).removeClass( "error" );
```

Internal Iterators

Be able to loop on itself

Internal Iterators

```
$$('.question').each(function(item){  
    item.fx = new fx.Height(item.nextSibling);  
    item.fx.hide();  
    item.onclick = function(){this.fx.toggle();}  
});
```

```
<div class="question">Question 1</div><div>Answer</div>  
<div class="question">Question 2</div><div>Answer</div>  
<div class="question">Question 3</div><div>Answer</div>
```

Chainable Methods

Primarily found in JQuery

```
$( "p" ).addClass( "test" ).show().html( "foo" );
```

Create Effects

Moo.fx, Script.aculo.us, JQuery

```
item.fx = new fx.Height(element);  
item.fx = new fx.Opacity(element);
```

```
$( "p.surprise" ).show( "slow" );
```

(If you make custom mouse cursors, I will hunt you down)

Good Libraries help Keys to Rapid Development

1. Maintainability
2. Extensibility
3. Flexibility

Which library should I use?

"It depends"

Comparison

moo.fx	6k	uncompressed with prototype lite
behavior	8k	uncompressed
jquery	15k	compressed
prototype	46k	uncompressed
dojo	129k	compressed
script.aculo.us	159k	uncompressed including prototype
Yahoo	323k	compressed
Yahoo	860k	uncompressed

Rapid Ajax

(could it *BE* any faster?)

JSON

JSON parsers exist for many server-side languages

JSONT = JSON Transformations

<http://goessner.net/articles/jsont/>

JSON

```
var jsonData = {  
  "guitars": {  
    "GSG": "Gibson SG",  
    "GLP": "Gibson Les Paul",  
    "FSC": "Fender Stratocaster",  
    "FTC": "Fender Telecaster"  
  }  
}  
  
alert(jsonData.guitars.GSG);  
alert(jsonData['guitars']['GSG']);
```

<http://muffinresearch.co.uk/archives/2006/07/19/json-selects/>

Use innerHTML

It's Rapid!

Do transformations on the server

Premade HTML Shells

The shell exists on page at load time

Populate with data when ready

It's even faster than innerHTML

<http://particletree.com/notebook/changing-the-dom>

Use a Library

Many facilitate JSON, innerHTML

Keys to Rapid Development

1. Maintainability
2. Extensibility
3. Flexibility

Where to from here?

Usability

Accessibility

Any Questions?

<http://rapiddom.com/wv2006/>