JavaScript and jQuery

John Resig - March 2008
ACM Northeastern University

JavaScript Language

- Basics
- Functions
- Function Prototypes
- Closures

The Basics

- Most akin to most other major languages
- Looping:
  for (var i = 0; i < 10; i++) {} // array
  for (var i in obj) // obj property/value
  while (true) {}
- Statements:
  if (true) {}

Variable Declaration

- Variable declaration:
  var foo = “bar”;
- Variables are scoped to the function, not to the block
- function test(){
  for (var i = 0; i < 10; i++) {}
  return i;
  }
  test() == 10

Global Object

- JavaScript, by itself, isn't useful
  - Just a 'dumb' scripting language
  - Functionality must be introduced
- All functionality is contained within the global object (named 'window' in browsers)
- var foo = "test";
  window.foo == "test";

Primitives

- Completely dynamic language - no “types”
- A few primitives:
  - Boolean: true / false
  - String: “hello”
  - Number: 4
- Type coersion
  - 0 == false
  - null == false
  - !”foo” == false

- var foo : string = “foo”;
- var num : double = 5.0;
-

Objects

- Object
  var obj = { key: “value”, key2: “value2” };
  obj.key == “value”;
  obj[“key2”] == “value2”;
- Array
  var a = [0, 1, 2, 3];
  a.push(4);
- RegExp
  /he[l]+o/i.test(“hello”)
- and Functions!

Functional

- JavaScript is a functional language
  - Functions are first-class objects
- function test(){}
  test(test);
- Anonymous Functions:
  var test = function(){};
- setTimeout(function(){
    alert(“1 second later”);
  }, 1000);

Function Definition

- Defining a function makes it omni-present, within its scope
- function test() {
  return foo();
  // Will never be reached, but that's ok
  function foo(){ return 1; }
}

Functions as Objects

- Functions can have properties
- function test(){}
  test.a = 1;
  test.b = 2;

Self-Memoization

```
function isPrime( num ) {
  if ( isPrime.answers[ num ] != null )
    return isPrime.answers[ num ];
  var prime = num != 1;
  for ( var i = 2; i < num; i++ )
    if ( num % i == 0 ) {
      prime = false;
      break;
    }
  return isPrime.answers[ num ] = prime;
}
isPrime.answers = {};
```

```
isPrime(5) == true
```

Function Context

- ‘this’ within a function
- var obj = {
  fn: function(){
  return this == obj;
  }
  };
  obj.fn() == true
- function test(){
  return this == window;
  }

Modify Context

- [0,1,2].join(“ “) == “0 1 2”
- [].join.call([0,1,2],” “) == “0 1 2”
- [].join.call(”test”,” “) == “t e s t”
- [].join.apply(”test”, [” “]) == “t e s t”
- var str = “test”;
- str.length
  str[0] == “t”

Instantiation

- No ‘classes’ in JavaScript
- You can instantiate a function
- function User(name){
  this.name = name;
  }
  var user = new User(“John”);
  user.name == “John”;
- User(“John”)
  name == “John”

Function Prototypes

- All functions have a prototype property
- All prototype properties are added to function instances
- function User(){}
  var u = new User();
  User.prototype.setName = function(name){
    this.name = name;
  };
  u.setName(“John”);
  u.name == “John”;

Inheritance

- Use the prototype to chain functions together
- function People(){}
  People.prototype.human = true;

```
function Resig(name){
  this.firstName = name;
}
Resig.prototype = new People();
Resig.prototype.lastName = “Resig”;
var john = new Resig(“John”);
```

instanceof

- var john = new Resig("John");
  john instanceof Resig
  john instanceof Person
  john instanceof Object
- All functions have a base prototype of Object

Native Prototypes

- You can extend the prototypes of native functions
- String.prototype.join = function(array){
  return array.join(this);
  };
  “,”.join([0, 1, 2]) == “0,1,2”
- Properties are added to all obects of the type
- Extending Object.prototype is really bad (can’t loop over properties safely, anymore)

- Object.prototype.keys = function(){
  var a = [];
  for (var i in this)
    a.push(i);
  return a;
};
{a: 1, b: 2].keys().length == 3

Closures

- Referencing an external variable encapsulates it for later access

```
function Ninja(){
  var slices = 0;
  this.getSlices = function(){
    return slices;
  };
  this.slice = function(){
    slices++;
  };
}
var ninja = new Ninja();
ninja.slice();
ninja.getSlices() == 1
ninja.slices // doesn't exist
```

Currying

- Pre-fill in arguments to a function, generating a new function

```
Function.prototype.curry = function() {
  var fn = this, args = Array.prototype.slice.call(arguments);
  return function() {
    return fn.apply(this, args.concat(
      Array.prototype.slice.call(arguments)));
  };
};

String.prototype.csv =
  String.prototype.split.curry(/,\s*/);

var results = ("John, Resig, Boston").csv();
results[1] == "Resig"
```

DOM

DOM Difficulty

- JavaScript programming can be hard
  - Problem isn't the JavaScript language
  - Cross-browser issues are frustrating
  - DOM can be clunky
- DOM is an object representation of an XML structure (in most cases, an HTML document)

DOM Methods

-
  document.getElementsByTagName(“div”)
-
  document.getElementById(“test”)
- removeChild()
- appendChild()
  insertBefore()

JavaScript Libraries

- Come to the rescue!
- Drastically simplify DOM, Ajax, Animations
- Use a library, like jQuery!

jQuery

What is jQuery?

- An open source JavaScript library that simplifies the interaction between HTML and JavaScript.

Ideal for Prototyping

- Completely unobtrusive
- Uses CSS to layer functionality
- Easy to separate behavior
- Quick, terse, syntax

- div { color: red; }
- div p { }
-

The Focus of jQuery

Find Some Elements Do something with them

```
$(“div”).addClass(“special”);
```

jQuery Object

The jQuery Object

- Commonly named '$'
- Also named 'jQuery'
- Completely valid:
  - jQuery("div").addClass("special");

Find Some Elements...

- Full CSS Selector 1-3 Support
- Basic XPath
- Better CSS Selector support than most browsers

$(“div”)

```
<div id=”body”>
    <h2>Some Header</h2>
    <div class=”contents”>
        <p>...</p>
        <p>...</p>
    </div>
</div>
```

$(“#body”)

```
<div id=”body”>
    <h2>Some Header</h2>
    <div class=”contents”>
        <p>...</p>
        <p>...</p>
    </div>
</div>
```

$(“div#body”)

```
<div id=”body”>
    <h2>Some Header</h2>
    <div class=”contents”>
        <p>...</p>
        <p>...</p>
    </div>
</div>
```

$(“div.contents p”)

```
<div id=”body”>
    <h2>Some Header</h2>
    <div class=”contents”>
        <p>...</p>
        <p>...</p>
    </div>
</div>
```

$(“div > div”)

```
<div id=”body”>
    <h2>Some Header</h2>
    <div class=”contents”>
        <p>...</p>
        <p>...</p>
    </div>
</div>
```

$(“div:has(div)”)

```
<div id=”body”>
    <h2>Some Header</h2>
    <div class=”contents”>
        <p>...</p>
        <p>...</p>
    </div>
</div>
```

Do something with them

- DOM Manipulation (append, prepend, remove)
- Events (click, hover, toggle)
- Effects (hide, show, slideDown, fadeOut)
- AJAX (load, get, post)

DOM Manipulation

- $(“a[target]”)
  .append(“ (Opens in New Window)”);
- $(“#body”).css({
  border: “1px solid green”,
  height: “40px”
  });

Events

- $(“form”).submit(function(){
    if ($(“input#name”).val() == “”)
      $(“span.help”).show();
  });
- $(“a#open”).click(function(){
    $(“#menu”).show();
    return false;
  });

Animations

- $(“#menu”).slideDown(“slow”);
- Individual properties:

```
$(“div”).animate({
    fontWeight: “2em”,
    width: “+=20%”,
    color: “green” // via plugin
});
```

- Callbacks:

```
$(“div”).hide(500, function(){
    // $(this) is an individual <div> element
    $(this).show(500);
});
```

Ajax

- $(“#body”).load(“sample.html div > h1”);
  - Before:

  - After:

    <h1>Hello, world!</h1>

- $.getJSON(“test.json”, function(js){
  for (var name in js)
  $(“ul”).append(“
” + name + “”);
  });

Chaining

- You can have multiple actions against a single set of elements
- $(“div”).hide();
- $(“div”).hide().css(“color”,”blue”);
- $(“div”).hide().css(“color”,”blue”).slideDown();

Chaining (cont.)

- $(“ul.open”) // [ul, ul, ul]
  .children(“li”) // [li, li, li]
  .addClass(“open”) // [li, li, li]
  .end() // [ul, ul, ul]
  .find(“a”) // [a, a, a]
  .click(function(){
  $(this).next().toggle();
  return false;
  }) // [a, a, a]
  .end(); // [ul, ul, ul]

Why jQuery?

- Fully documented
- Great community
- Tons of plugins
- Small size (14kb)
- Everything works in IE 6+, Firefox, Safari 2+, and Opera 9+

Questions?

http://ejohn.org/
http://jquery.com/

- [a, b, c] = [0, 1, 2]
- [a for (var a = 0; a < 10; a++)]
- let foo = 5;