

Advancing JavaScript with Libraries

John Resig

ejohn.org / jquery.com / mozilla.com

Two Hats

- **Mozilla Corp**
- **FUEL**
 - JS Lib for Ext. Devs
 - Reduce Complexity
- JS Lib Test Suites
 - Integration, better coverage
- **jQuery JavaScript Library**
 - Released Jan 2006
 - Small Filesize
 - Short, concise, code
 - Extensible via plugins

Libraries are...

- ...abstractions away from existing APIs
- ...giving us new APIs to interface with

Hypothesis

- APIs breed patterns
- Libraries build new patterns on top of existing APIs
- New library patterns advance development in meaningful, considerable, ways

Why Create A Library?

- Distance ourselves from repetition
- In JavaScript:
 - Distance from browser differences
- In C (stdlib):
 - Distance from platform differences

DOM API

- Implemented in every modern browser
- Most stable of all the JavaScript APIs
- Very well documented

Fails in IE 7

JavaScript:

```
document.getElementById("obj")  
  .getElementsByTagName("*")
```

HTML:

```
<object id="obj">  
  <param name="src" value="test.mov"/>  
  <param name="title" value="My Video"/>  
</object>
```

Fails in Safari 2

JavaScript:

```
document.getElementById("opt").selected
```

HTML:

```
<div style="display: none;">  
  <select><option id="opt" value="test"/></select>  
</div>
```

Genuine Bugs

Fails in Opera & IE

JavaScript:

```
document.getElementById("q")
```

HTML:

```
<form action="" method="POST">  
  <input type="text" name="q"/>  
  <span id="q">Search</span>  
</form>
```

Fails in Opera & IE

JavaScript:

```
var f = document.getElementsByTagName("input");  
for ( var i = 0; i < f.length; i++ ) { ... }
```

HTML:

```
<form action="" method="POST">  
  Num Rows: <input type="text" id="length" value="22"/><br/>  
  <input type="submit" value="Generate"/>  
</form>
```

Re-interpretation of the Spec

Fails in All

JavaScript:

```
document.getElementById("name")  
  .setAttribute("disabled", false);
```

HTML:

```
<input type="text" id="name" disabled="disabled"/>
```

Fails in All

JavaScript:

```
document.body.setAttribute("class", "home");
```

HTML:

```
<body> ... </body>
```

What you expect

Why are Libraries Created?

- Browsers have a large number of strange differences...
 - IE has a lot of well documented bugs
 - Safari has less, but more obscure, bugs
- APIs don't do what you expect
- When the API is impossible to learn, but through experience

Break down into Meta-Problems

- Waiting for the document to load
- Traversing the DOM
- Injecting HTML into a page

Waiting for the DOM to load

- A DOM document must be fully loaded before you can work with it
- Waiting for the window to load causes flashes of un-effected content

Focusing an input

```
<input type="text" id="test"/>  
<script>document.getElementById("test").focus();</script>
```

```
<head><script src="script.js"</head>
```

```
$(document).ready(function(){  
    $("#test").focus();  
});
```

Traversing the DOM

- `getElementsByTagName`, `getElementById`, `getAttribute`, `nextSibling`, `previousSibling`, `parentNode`, `className`, etc.
- A lot to learn and get right
- Has the potential to be very tricky (and long-winded)

DOM Selectors

- Goal: Improve the DOM Traversal API
- Provide an API on top of the existing DOM API
- Look to standards for inspiration

Types of Selectors

- XPath Selectors
 - `//div/span[2]`
 - `/html[@id='home']`
- CSS 3 Selectors
 - `div > span:nth-of-type(2)`
 - `html:root#home`

Non-trivial Selectors

```
$("#menu > li:not(:first-child)").hide();
```

```
$("#ul[ul]").show();
```

```
$("#tr:even").addClass("even");
```

Injecting HTML

- Very frustrating problem
- Browsers are very temperamental
- (Inserting table rows, select options, etc.)

HTML Solution

- Allow users to insert HTML strings
- Convert to DOM nodes on the fly
- Inject into the document correctly

```
$(“table tr”).append(“<td>test</td>”);  
$(“select”).prepend(“<option>test</option>”);
```

- DOM Ready + Selectors + HTML Injection
- The dream of true unobtrusive DOM scripting

```
$(document).ready(function(){  
    $("select").append("<option>None</option>");  
});
```

New Expectations

- What new expectations emerge out of this new API?

```
$(“ul > li”).click(function(){  
    $(this).load(“menu.html”);  
});
```

```
<ul>  
  <li>item a</li>  
  <li>item b</li>  
</ul>
```

The Perception

- When the document was ready:
 - We bound an event to a set of elements
- We loaded some new elements
 - Why don't they have the event?

New Pattern: Behaviors

```
function handleClick() {
    $("li", this).click(loadMenu);
}

function loadMenu() {
    $(this).load("menu.html", handleClick);
}

$(document).ready(function(){
    $(document).each(handleClick);
});
```

Behaviors

```
$(document).ready(function(){  
  $("li").behavior("click",function(){  
    $(this).load("menu.html");  
  });  
});
```

Pure DOM

```
function handleClick( elem ) {
    var li = elem.getElementsByTagName("li");
    for ( var i = 0; i < li.length; i++ )
        li[i].addEventListener( "click", loadMenu, false );
}

function loadMenu() {
    var elem = this;
    var xhr = new XMLHttpRequest();
    xhr.open( "GET", "menu.html", false );
    xhr.onreadystatechange = function(){
        if ( xhr.readyState == 4 ) {
            elem.innerHTML = xhr.responseText;
            handleClick( elem );
        }
    };
    xhr.send( null );
}

window.onload = function(){
    var ul = document.getElementsByTagName( "ul" );
    for ( var i = 0; i < ul.length; i++ )
        handleClick( ul[i] );
}
```

Doesn't exist in IE

IE sets the wrong context

Doesn't exist in IE

Leaks in IE

Doesn't happen immediately

FUEL

- JavaScript Library for Firefox Extension Development
- Designed to be used by Web Developers
- Current API is very C++ centric

Mozilla: Preferences

```
var prefs = Components.classes["@mozilla.org/preferences-service;1"]
    .getService(Components.interfaces.nsIPrefBranch);

var str = Components.classes["@mozilla.org/supports-string;1"]
    .createInstance(Components.interfaces.nsISupportsString);
str.data = "some non-ascii text";
prefs.setComplexValue("preference.with.non.ascii.value",
    Components.interfaces.nsISupportsString, str);
```

Preferences w/ FUEL

```
Application.prefs.setValue("some.pref", "some non-ascii text");
```

SQL

```
SELECT * FROM users WHERE id = 5;
```

Programming Language
Interface

```
$res = mysql_query("SELECT * FROM users WHERE id = 5;");  
while ( $row = mysql_fetch_assoc($res) ) {  
    // etc.
```

Abstraction
Layer (ORM)

```
User.find(5)
```

```
class Account < ActiveRecord::Base
  has_many :people do
    def find_or_create_by_name(name)
      first_name, last_name = name.split(" ", 2)
      find_or_create_by_first_name_and_last_name
(first_name, last_name)
    end
  end
end

person = Account.find(:first).people
  .find_or_create_by_name("David Heinemeier Hansson")
person.first_name # => "David"
person.last_name  # => "Heinemeier Hansson"
```

Conclusion

- Libraries build new patterns on top of existing APIs
- New library patterns advance development in meaningful, considerable, ways

Bonus: Meta-Libraries

- Domain Specific Languages (DSLs)
- Build upon existing libraries building new languages out of their APIs
- “jQuery2” meta language:
<http://ejohn.org/apps/jquery2/>

More info...

- <http://ejohn.org/>
- <http://jquery.com/>
- <http://wiki.mozilla.org/FUEL>
- <http://ejohn.org/apps/jquery2/>

- Contact:
jesig@gmail.com