JavaScript Libraries: Choose to Reuse!

@jeremyckahn

Let's make this work everywhere

```
require(['library'], function (Library) {

  var library = new Library();

});
```

Hi! Who am I?

Jeremy Kahn

Web Developer at YouTube

I spend a lot of time writing open source JavaScript libraries!

* Opinions are my own

github.com/jeremyckahn

twitter.com/jeremyckahn

"No problem should ever have to be solved twice."

How To Become A Hacker
(http://www.catb.org/esr/faqs/hacker-howto.html)

Code reuse is king

Copy/paste

!==

Code reuse

Frameworks

Libraries

Framework: A mechanism for structuring an application

Examples: Ember, Angular, Knockout, Closure

Library: A module for solving a specific problem

Examples: jQuery, Backbone, Underscore, Require

Frameworks vs Libraries

Inversion of Control

Frameworks control

application code

Application code controls libraries

A library is a wrench, a framework is the entire workshop

Libraries!

Reusable solutions for common problems

Very good ideas

- vs -

Extremely bad ideas

Extremely bad ideas

Modifying built-in prototypes

Environment assumptions

Excessive dependencies

Very good ideas

Minimal global variables

Small and focused scope

"It just works"

Module compatibility

CommonJS

AMD

UMD

ES Harmony

Module compatibility

CommonJS - Server

AMD - Browser

UMD - CommonJS + AMD

ES Harmony - proposed standard

Testing

QUnit

javascript unit testing.

Jasmine

mocha

Semantic Versioning

AKA: I like my package versions like I like my markup

semver.org

major.minor.patch

semver.org

0.12.6

semver.org

0.12.6

semver.org

```
0.12.6
```

Optimizing for the modern developer

npm

Ender

Jam

Bower

Yeoman

npm

Ender

Jam

Bower

Yeoman

So many details!

lib-tmpl: A template for unobtrusive, scalable JavaScript libraries

github.com/jeremyckahn/lib-tmpl

Modeled after Rekapi and Shifty

(and more!)

Anatomy of lib-tmpl

Static properties

`prototype` (public) properties

private properties

```
var library = new Library();
```

Anatomy of lib-tmpl

Components are broken into multiple module files and wrapped in a closure

```
library.intro.js
library.init.js
library.core.js
library.module.js*
library.outro.js
```

```
;(function (global) {
    // code
} (this));
```

```
;(function (global) {
    // code
} (this));
```

```
library.intro.js
library.init.js
library.core.js
library.module.js*
library.outro.js
```

`library.init.js`

Bootstraps the library

Normalizes AMD and non-AMD environments

library.intro.js

library.init.js

library.core.js

library.module.js*

library.outro.js

`library.core.js`

Creates **`Library`** Object

Defines library utilities

```
// INTERNAL LIBRARY METHODS
function aPrivateMethod () {};

// LIBRARY CONSTANTS
var A_CONSTANT = true;

function initLibraryCore (context) {

  var Library = context.Library
    = function (opt_config) {};

  Library.prototype.aPublicMethod
    = function () {};
}
```

```
// INTERNAL LIBRARY METHODS
function aPrivateMethod () {};

// LIBRARY CONSTANTS
var A_CONSTANT = true;

function initLibraryCore (context) {

  var Library = context.Library
    = function (opt_config) {};

  Library.prototype.aPublicMethod
    = function () {};
}
```

```
// INTERNAL LIBRARY METHODS
function aPrivateMethod () {};

// LIBRARY CONSTANTS
var A_CONSTANT = true;

function initLibraryCore (context) {

  var Library = context.Library
    = function (opt_config) {};

  Library.prototype.aPublicMethod
    = function () {};
}
```

library.intro.js

library.init.js

library.core.js

library.module.js*

library.outro.js

Library modules!

Library modules

!==

AMD/CommonJS modules

Anatomy of a library module

A function

`Library` Object decoration

```
function initLibraryModule (context) {

  // MODULE-LEVEL CONSTANTS
  var ANOTHER_CONSTANT = 5;

  function localPrivateFunction () {};

  Library.aStaticMethod = function () {};

  Library.prototype.additionalPublicMethod
    = function () {};
}
```

```
function initLibraryModule (context) {

  // MODULE-LEVEL CONSTANTS
  var ANOTHER_CONSTANT = 5;

  function localPrivateFunction () {}

  Library.aStaticMethod = function () {};

  Library.prototype.additionalPublicMethod
    = function () {};
}
```

lib-tmpl freebies

Sane default directory structure

Unit testing bootstrap (QUnit)

Build script

lib-tmpl freebies

Sane default directory structure

Unit testing bootstrap (QUnit)

Build script

Building with UglifyJS

Compress many JavaScript files into one minified binary

Create custom binaries

Compiler directives

Or, how I learned to stop worrying and love the binary

```
if (typeof DEBUG === 'undefined') {
   DEBUG = true;
}


if (DEBUG) {
   window.testHook = function () {
      // code...
   };
}
```

```
if (typeof DEBUG === 'undefined') {
  DEBUG = true;
}
```

```
if (DEBUG) {
  window.testHook = function () {
    // code...
  };
}
```

```
var
   uglifyJS = require('uglify-js'),
   jsp = uglifyJS.parser,
   pro = uglifyJS.uglify,
   ast = jsp.parse(
      _fs.readFileSync(
       'path/to/file.js', 'utf-8'));
```

```
ast = pro.ast_mangle(ast, {
      'defines': {
         'DEBUG': ['name', 'false']
      }
   });
ast = pro.ast_squeeze(ast);
```

```
var
  uglifyJS = require('uglify-js'),
  jsp = uglifyJS.parser,
  pro = uglifyJS.uglify,
  ast = jsp.parse(
    _fs.readFileSync(
     'path/to/file.js', 'utf-8'));

ast = pro.ast_mangle(ast, {
    'defines': {
      'DEBUG': ['name', 'false']
    }
  });
ast = pro.ast_squeeze(ast);
```

```
if (typeof DEBUG === 'undefined') {
   DEBUG = true;
}
```

```
if (DEBUG) {
   window.testHook = function () {
      // code...
   };
}
```

```
if (typeof NODE_JS === 'undefined') {
  NODE_JS = true;
}
```

```
if (NODE_JS) {
  global.nodeJSSpecificMethod () {
    // code...
  }
}
```

```
var
  uglifyJS = require('uglify-js'),
  jsp = uglifyJS.parser,
  pro = uglifyJS.uglify,
  ast = jsp.parse(
    _fs.readFileSync(
     'path/to/file.js', 'utf-8'));

ast = pro.ast_mangle(ast, {
    'defines': {
      'DEBUG': ['name', 'false'],
      'NODE_JS': ['name', 'true']
    }
  });
ast = pro.ast_squeeze(ast);
```

The point is...

The point is...

Build better tools.

The point is...

Build a community.

microjs.com

Thanks!

Questions?

@jeremyckahn

Resources

https://github.com/jeremyckahn/lib-tmpl

http://addyosmani.com/writing-modular-js/

https://github.com/umdjs/umd

https://github.com/wilmoore/frontend-packagers

http://microjs.com/

http://jeremyckahn.github.com/blog/2012/07/05/a-javascript-library-template/