JavaScript Asynchronous Module Definition (AMD Explained

INTRODUCTION

Kevin Murray

murmeister@hotmail.com

Project Expectations

Organized

Asynchronous Module Definition (AMD)

Goals

- Trainable
- Explainable
- Maintainable

Code review

Informational slides

Progressive samples

Chapters build on previous information

Tools used

- RequireJS
- jQuery
- Browser
- Text editor
- Jasmine

RequireJS

- Version 2.1.22 (requirejs.org)

Notepad++

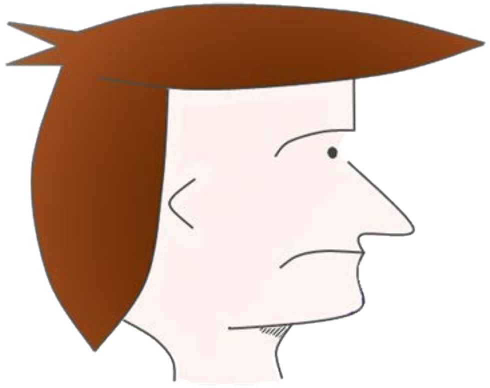
- (notepad-plus-plus.org)

FireFox

- other browsers ok

Web server not required

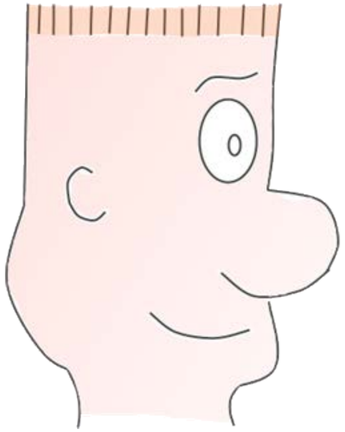
Additional Participants



Harold

Very bright

A bit edgy



Doug

Friendly

Sometimes careless

Overview

Reviewing traditional web design

Module creation and refactoring

Data modules and 3rd-party libraries

RequireJS configuration

Project organization

RequireJS optimizer

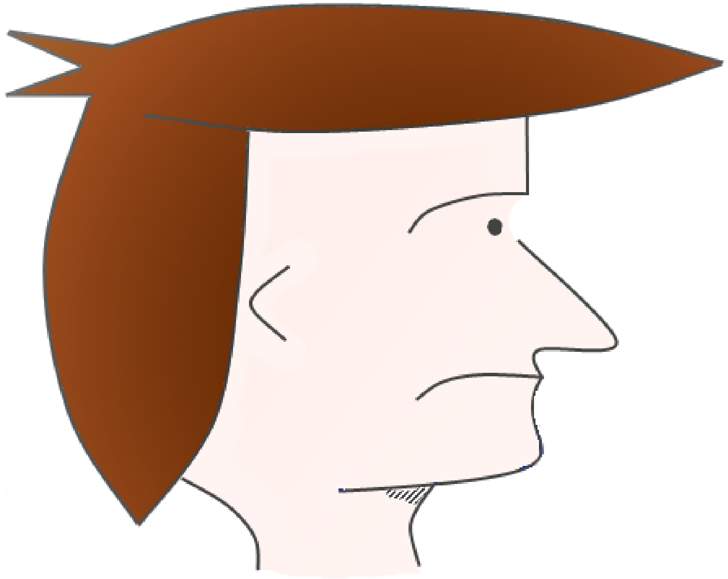
Demo

Code from actual projects

Not building another "Hello World"

- Uses CSS
- JavaScript files loaded by browser
- Some embedded JavaScript on page

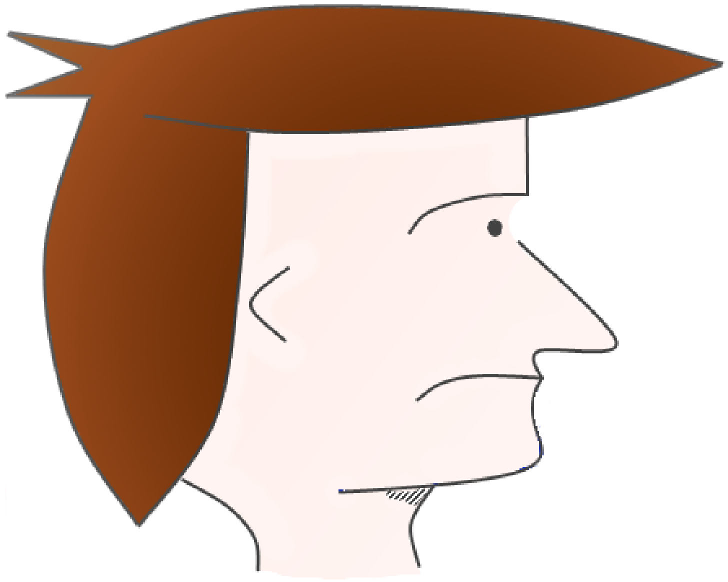
Expanding baseline



Is anyone else as confused as I am?

Expecting Global Namespace

```
(function()
{   if (typeof _KSM === 'undefined')
        _KSM = { defaultContext: 'body' };
...
```



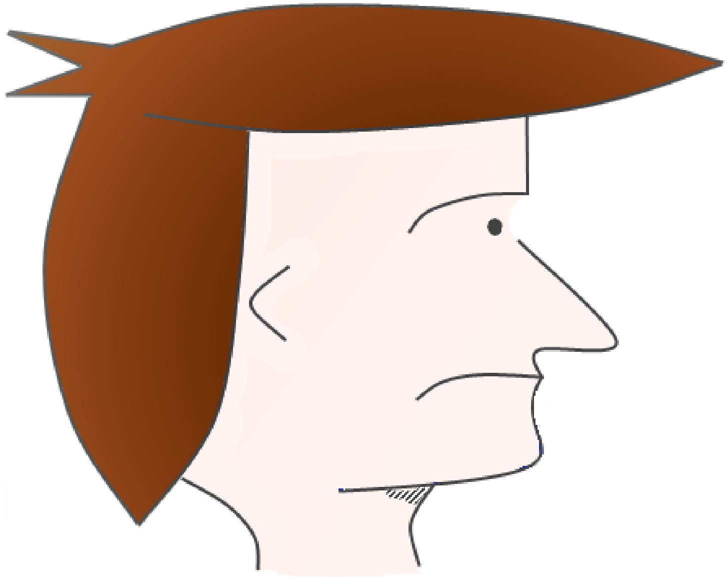
Globally accessible objects just pollute the global namespace!

Inspecting Bugs

_KSM namespace is not defined

_KSM.Language.load is not a function

With traditional JavaScript libraries, the order they are loaded can be critical



Ok, you have a point, I guess.

Coding Problems Arise

Interconnected libraries

New team members

Changing libraries

Code evolution

Unit Testing

👍

Test Page

👎

Production

Including unit tests

- Options available
- Learning opportunity

Jasmine

- http://jasmine.github.io/

Gaining more information

- https://www.pluralsight.com/

Unit testing can give false results if libraries are loaded in an unexpected order

Expecting Global Namespace

```
(function()
{   if (typeof _KSM === 'undefined')
        _KSM = { defaultContext: 'body' };
...
```

Global Namespace

Acting as container

Existing references

Migrating tips

Coupling

Dependencies loaded in proper order

Common failure point

Running Jasmine with RequireJS

Thinking of possibilities

Coming up later

Summary

Going simple

Fragile dependencies

Unit testing