Alternate Module Patterns

Kevin Murray

murmeister@hotmail.com

Overview

Data modules

Unit testing modules

Use legacy library as module

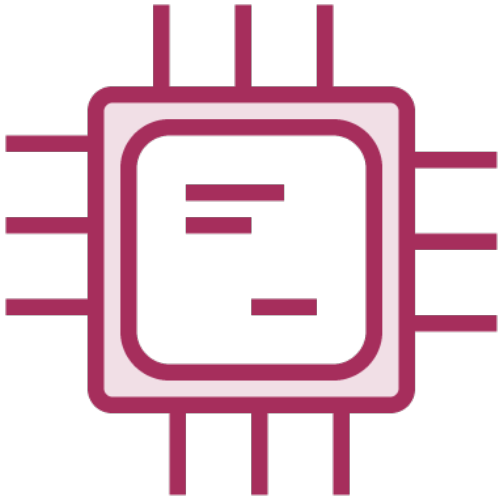
The Pattern so Far

Use “define” or “require” function

Specify dependencies

Include callback function

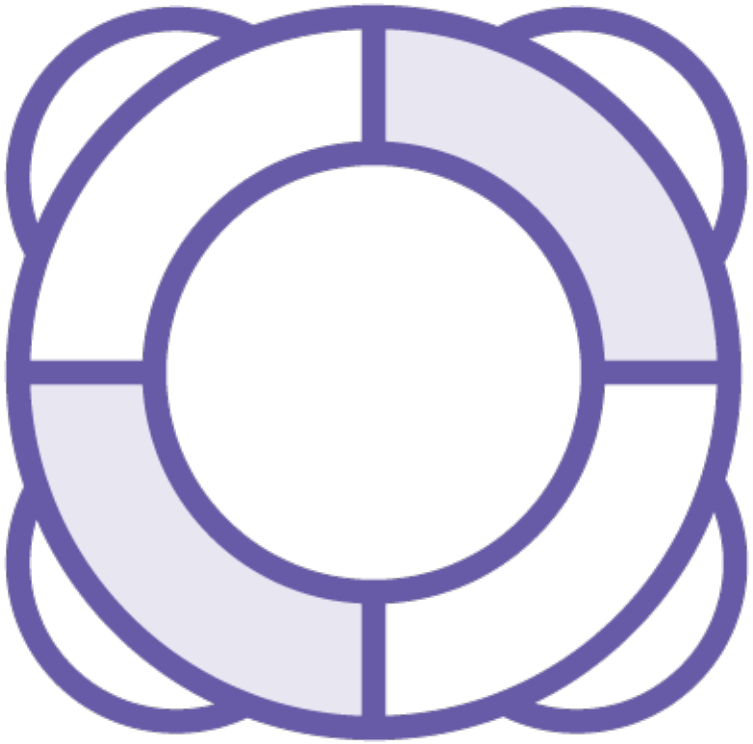
Data Module



Enclose data object in “define”

Data Module

```
define(
{   version: '1.0',
    defaultContext: '.content',
    languageAbbr: 'fr‘
});
```

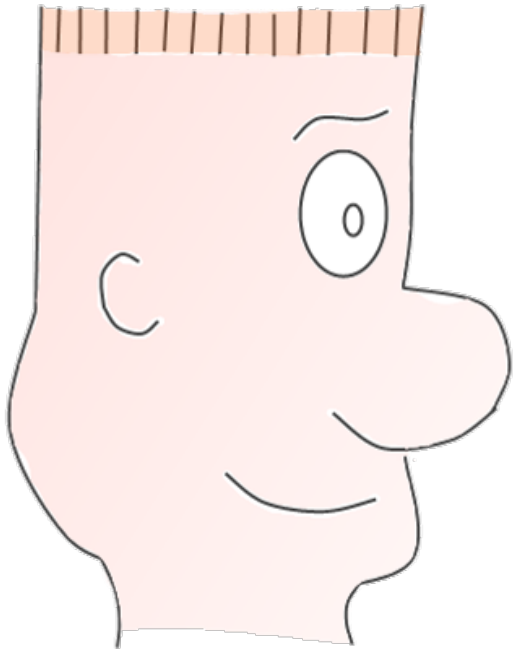


Centralized definition of values

Global namespace is not required

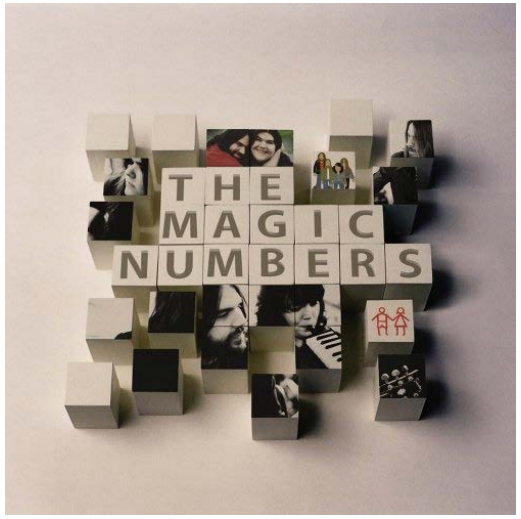
Readable code

Easier code reviews



I know about Magic Numbers! They're an English pop rock band, right?

Magic Number Information

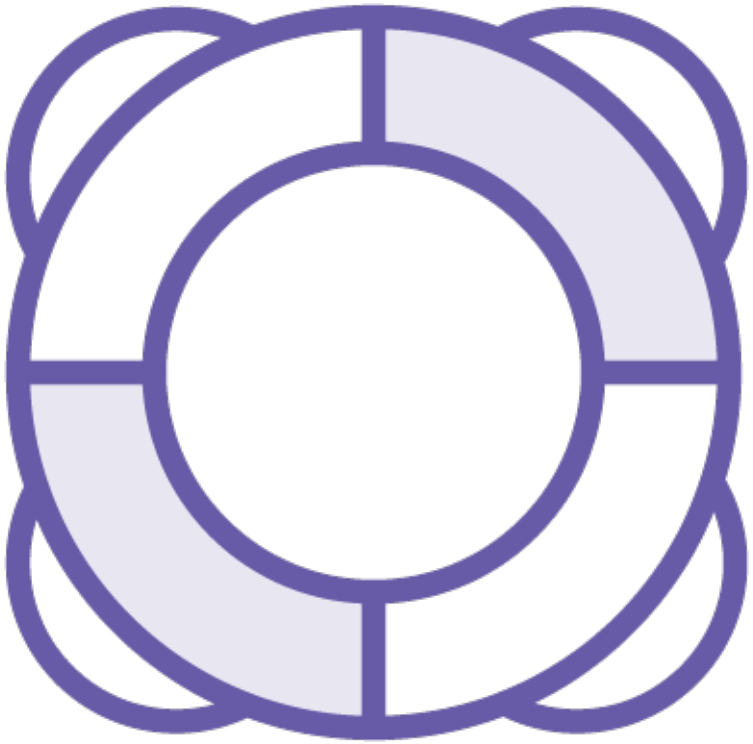


English Pop Group

https://www.facebook.com/TheMagicNumbers

Programming

https://en.wikipedia.org/wiki/Magic_number_(programming)



Centralized definition of values

Global namespace is not required

Readable code

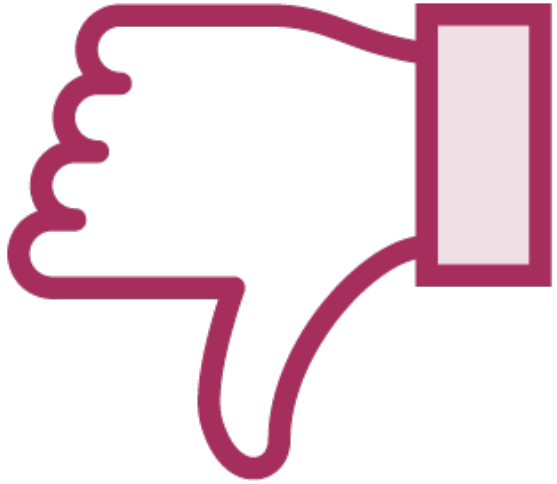
Easier code reviews

Able to include constant values

Consistent values for test and production

Development vs. Production

Development Page



Production

Library Order Importance

```
...
<script src="jquery.js"></script>
<script src="KSM_Language.js"></script>
<script src="KSM_Toolbar.js"></script>
...
```

Language Library First

Error in web page

```
...
<script src="jquery.js"></script>
<script src="KSM_Toolbar.js"></script>
<script src="KSM_Language.js"></script>
...
```

Toolbar Library First

Unit testing worked

Verify Data Module Contents

```
define(
{  version:        '1.0',
   defaultContext: '.content',
   languageAbbr:   'fr',
});
```

Production

defaultContext is “.content”

```
define(
{  version:        '1.0',
   defaultContext: '.body',
   languageAbbr:   'fr',
});
```

Development

defaultContext is “.body”

Using Data Module

Modify behaviors by changing configuration, not code

What about unit testing?

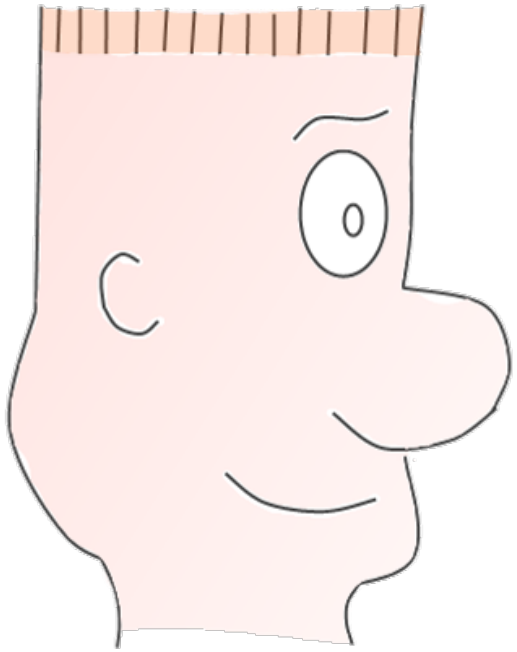
Unit Testing

Using Jasmine

Strengthening integration skills

Uses Global Namespace

Legacy code makes use of global namespace



We can just wrap the Jasmine library in a “define” function, right?

Changing Third-Party Code

Avoid temptation

Assuming all responsibilities

Good sample library

Revisit specs

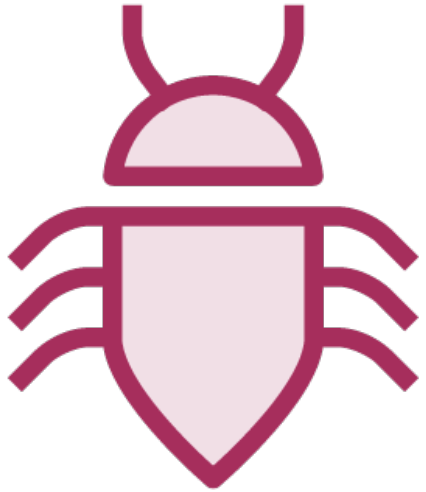
Unit tests in exercise files

Intermediate solution

Potential Problems

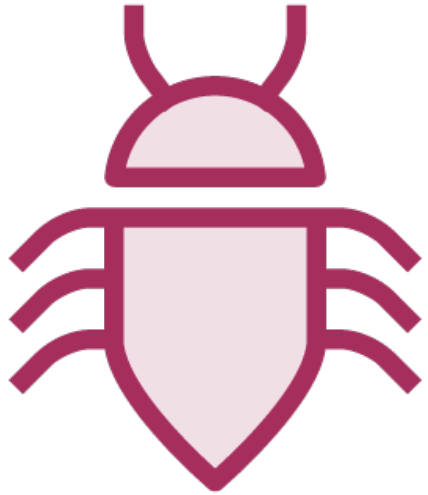
Use RequireJS to load all JavaScript libraries

Mixed loaders while refactoring



Frustrating errors

Common Error

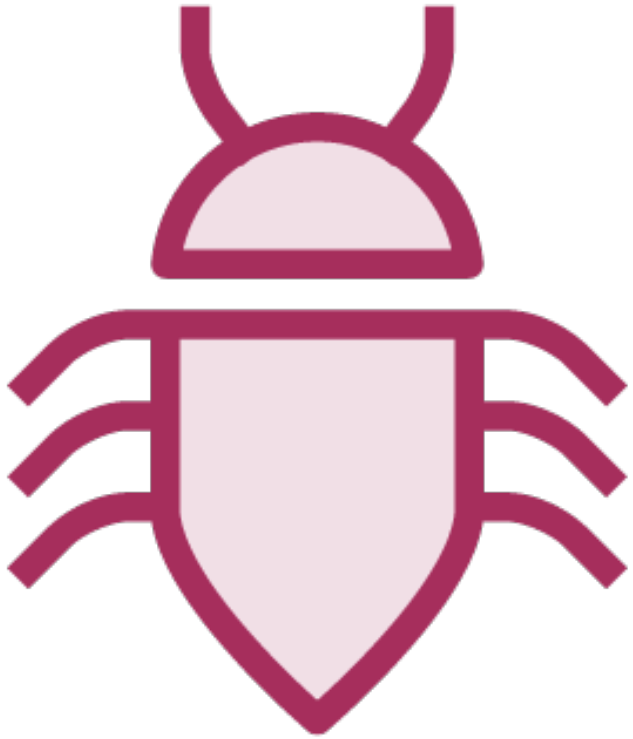


“define is not defined”

What?

Causing “define is not defined” Error

```
<!DOCTYPE html>
 <html lang="en">
  <head> <meta charset="utf-8" http-equiv="encoding">
   <link rel="stylesheet" href="./css/KSM_toolbar.css">
   <link rel="stylesheet" href="./css/KSM_master.css">
   <script src="KSM_FooterAMD.js"></script>
   <script src="./require.js"
           data-main="KSM_Start-06"></script>
  </head>
  <body>
...
```



Browser evaluates footer module

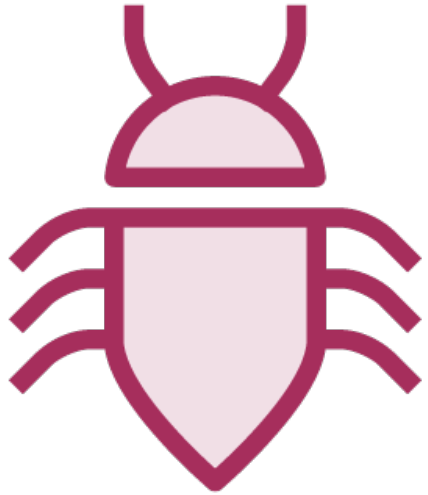
Footer code is modular

“define” is a RequireJS function

Browser can’t find the “define” function

Accurate – but odd – error message

Another Common Error

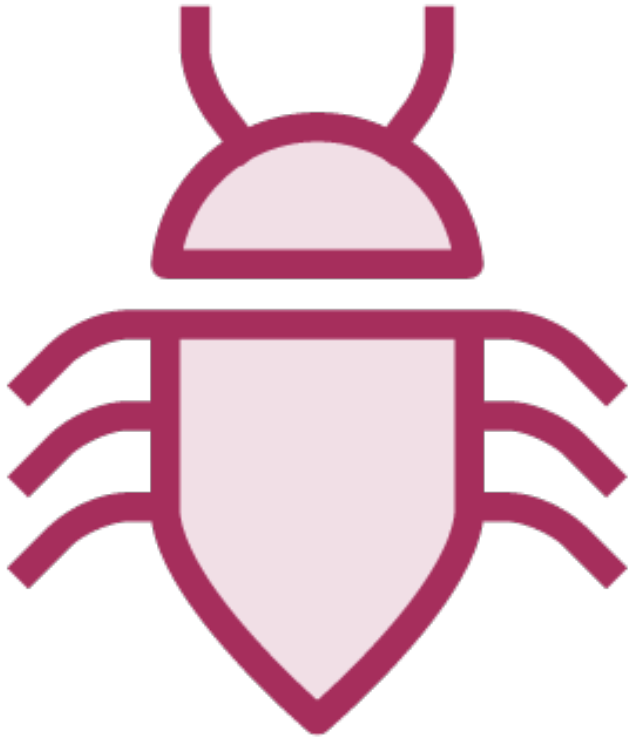


“Mismatched anonymous define() module”

Are you serious?

Causing “mismatched anonymous” Error

```
<!DOCTYPE html>
 <html lang="en">
  <head> <meta charset="utf-8" http-equiv="encoding">
   <link rel="stylesheet" href="./css/KSM_toolbar.css">
   <link rel="stylesheet" href="./css/KSM_master.css">
   <script src="./require.js"
           data-main="KSM_Start-06"></script>
   <script src="KSM_FooterAMD.js"></script>
  </head>
  <body>
...
```



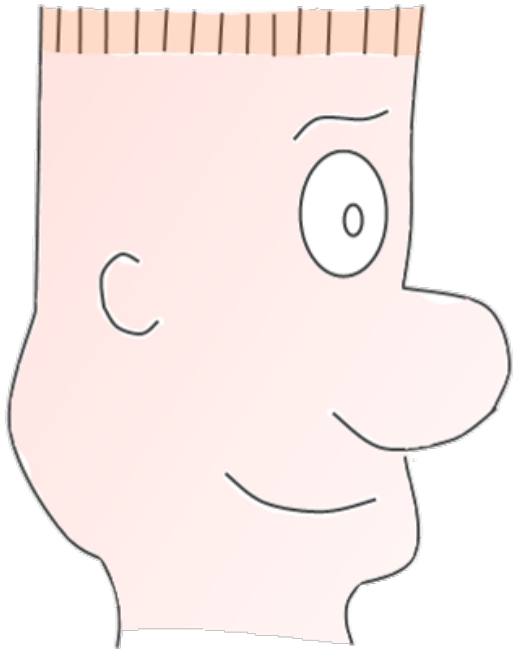
“define” function

- Already loaded
- Used to register modules

RequireJS uses filename as module name

Browser doesn’t register module

Unnamed module is anonymous

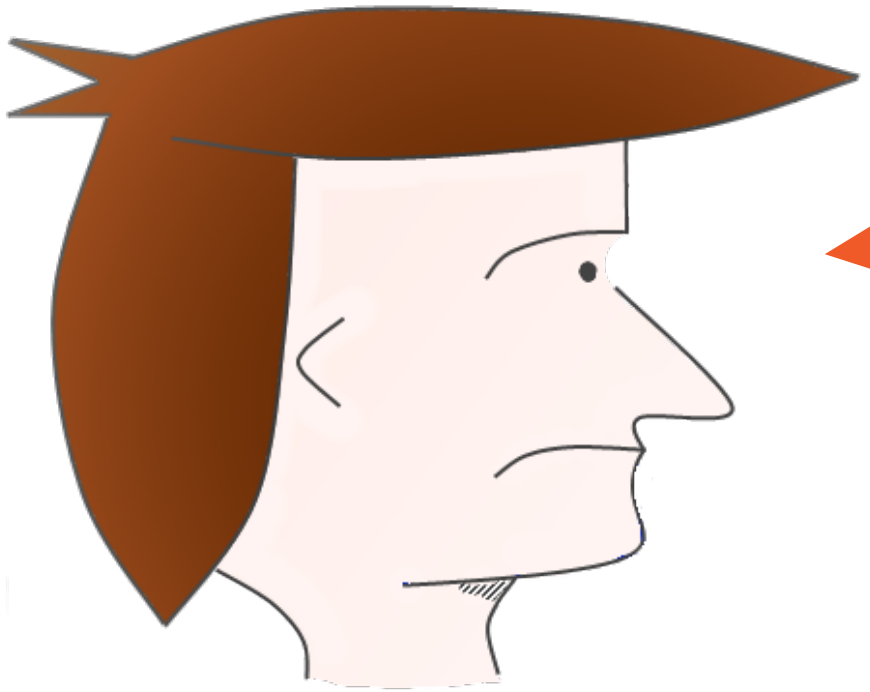


Those error messages are confusing!

Simple Solution

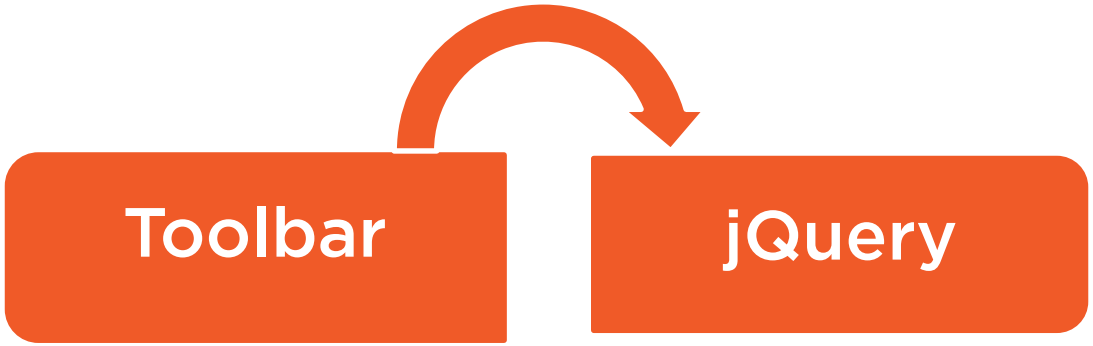
Load all scripts with RequireJS

Use RequireJS to load modules and JavaScript libraries to avoid “define not defined” and “mismatched anonymous define() module” errors



Wait! You didn't include jQuery! How are we asking for it to be loaded?

Dependency




I knew it!

Unlisted Dependency

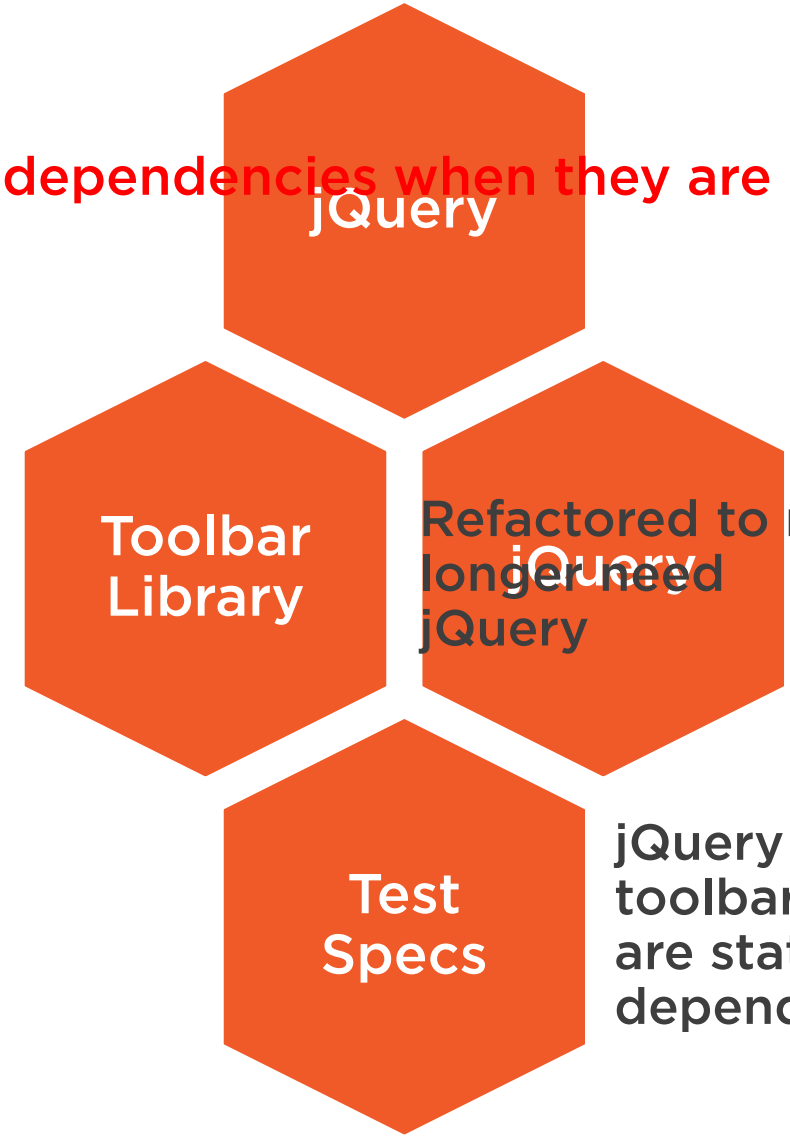
Commonly used –
even accidentally

Unclear code requires
extra research

Unstated dependency

Dependencies

List all dependencies when they are needed



Uses jQuery for current version

jQuery and toolbar library are stated dependencies

Uses jQuery but counts on the toolbar library to load it

Migrating to Modules

Placing objects in global namespace retains backwards compatibility

Legacy code expects objects to be defined and contain certain values

Using “window” Namespace

Temporary bridge while migrating to modular patterns

Nesting “require” Calls

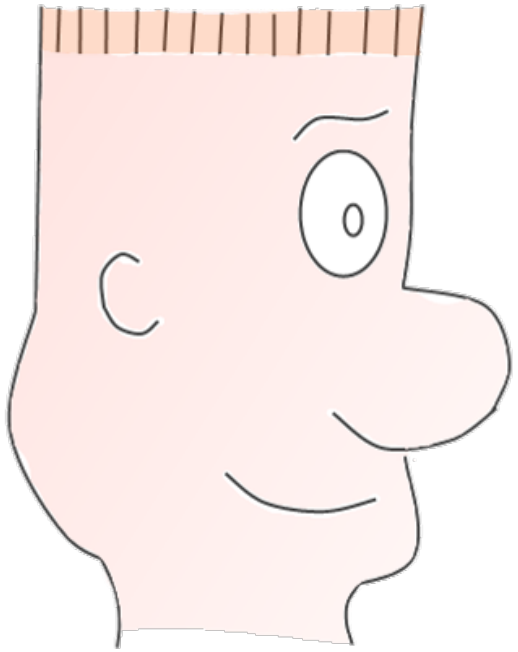
Only load files when needed

Use if/else logic to avoid loading

Only load files when they are “Required”


Groan!

Modules can be required
from within other modules
as part of the logic flow



Shouldn't the onload function kick off when the page loads?

Dependency Chain

HTML File

RequireJS

Jasmine-Start-03

Test spec

Dependency Loading

AMD

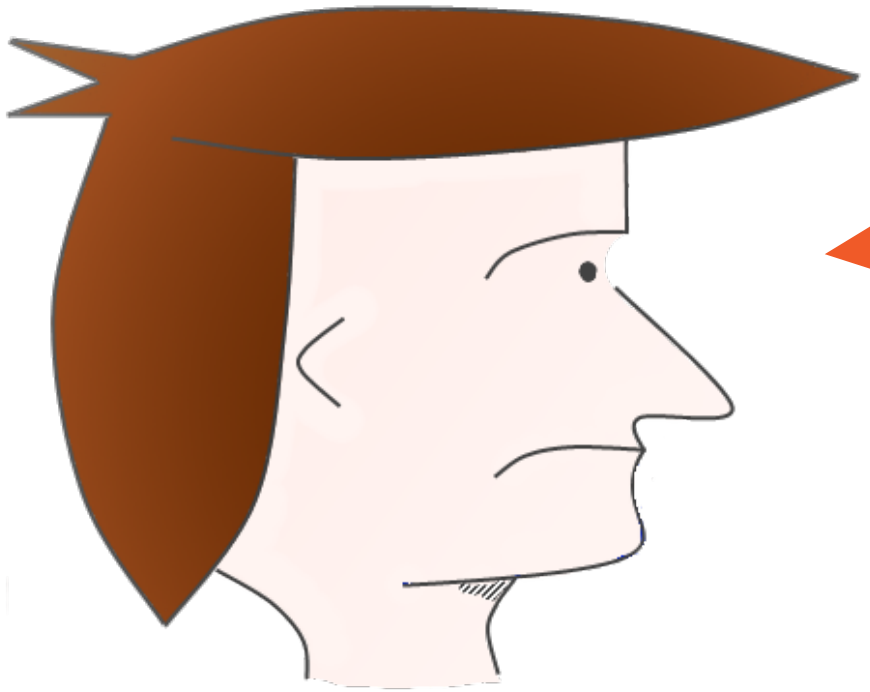
Asynchronous

Events process quickly

Window “onload” event fires early

Page Load Events

Involved discussion



I have a feeling there's more...

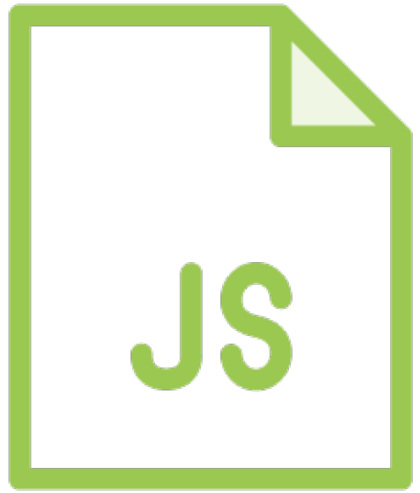
Testing code works

Using RequireJS to load Jasmine

Learning opportunity

Loading Libraries with RequireJS

Demonstration library



Use for any libraries not coded as modules

Jasmine in Modules

Indicative of other libraries

Helpful solution

“describe” and “it” on global namespace

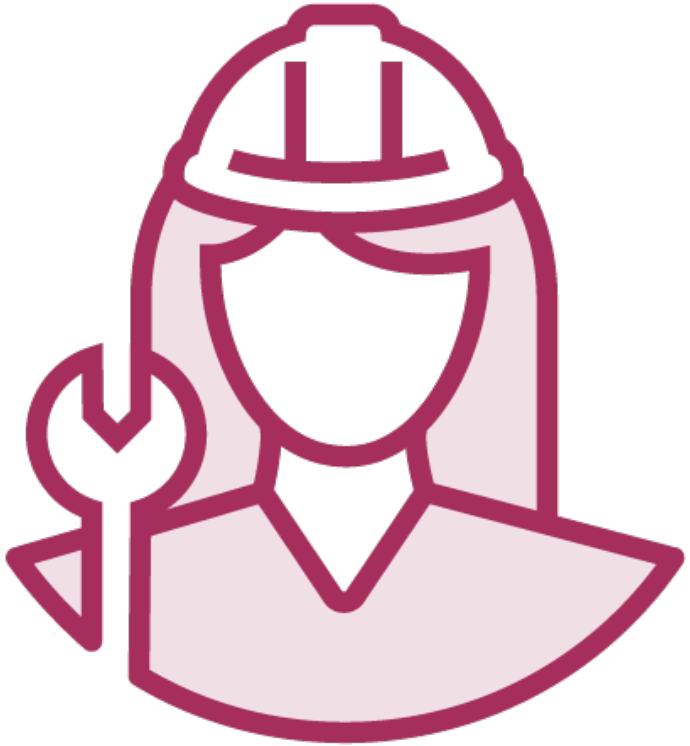
Leave core code alone

Testing code identical

Extra indent

New filename

Change Jasmine start script



Upgrade jQuery

Errors with Jasmine library

jQuery 1.8 resolved errors


Global namespace?! Polluter!

Over Simplifying

Summary

Data module

Unit testing

- Modules!

Used Jasmine library as modules

Configuration hints