Creating Modules

Kevin Murray

murmeister@hotmail.com

Overview

Simple HTML page

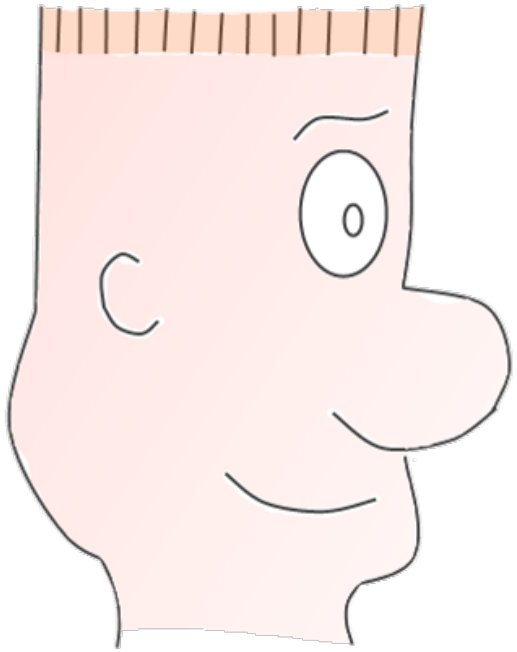
Footer module

- Visible result
- Useful throughout course

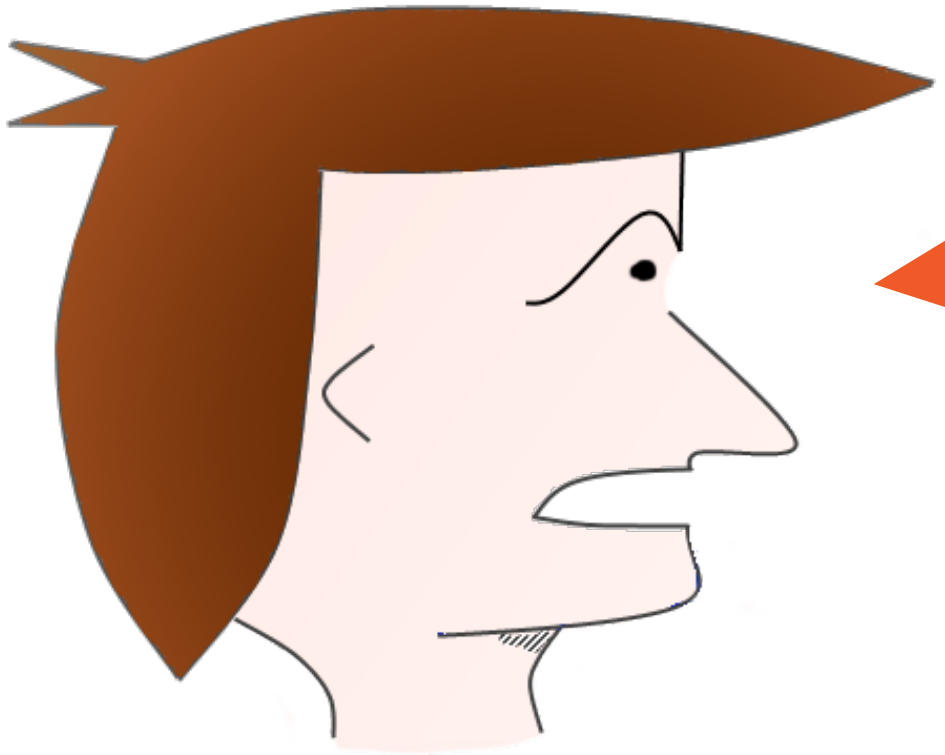
Refactoring into modules

- Language library
- Toolbar library

Demonstrating techniques



Well, how do we use it?



Wait! You forgot the document ready function!

Nice Benefit

Separation of code from HTML

Callback function after dependencies load

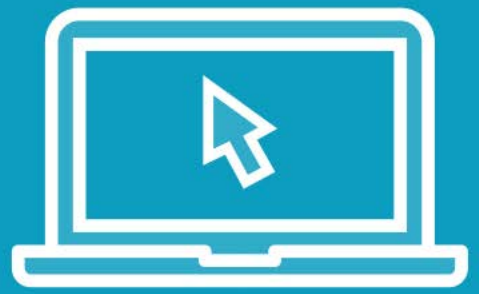
Results

Cleaner HTML

Scripts loaded only as needed

Reusable module

Demo



Back to original toolbar sample

New HTML page

Use startup code

Refactor legacy code

- Language library
- Toolbar library

Common issues

See it work!

Data-Main Attribute

Generally used for startup code that loads other modules

Entry point for client processing

Document ready function

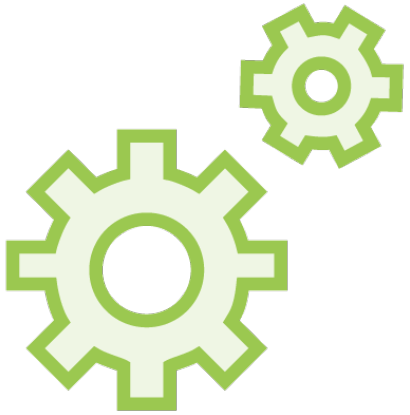
What executes before?

Not called until all scripts are loaded

Browser evaluates scripts

Using Typical Libraries

Self-executing



Data values

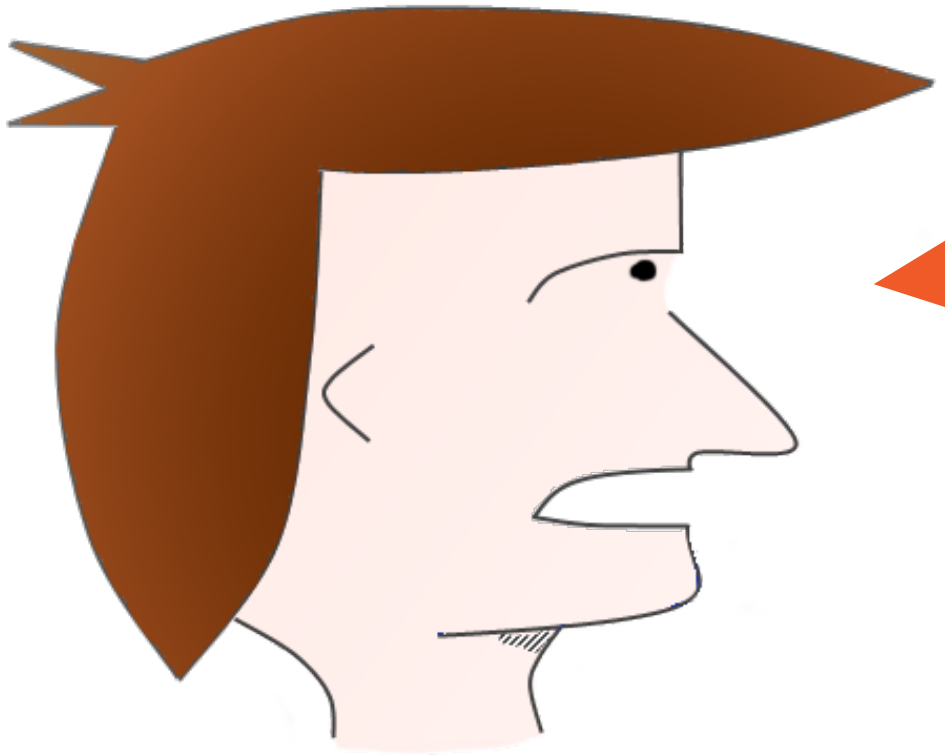
Event handlers

Startup Code

Activity before startup code executes

Load code only when/if needed

Startup code resides in separate file



I found an error in your code! You have 4 dependencies and only have 3 variables!

References to dependent modules loaded with “require” only have scope within the callback function

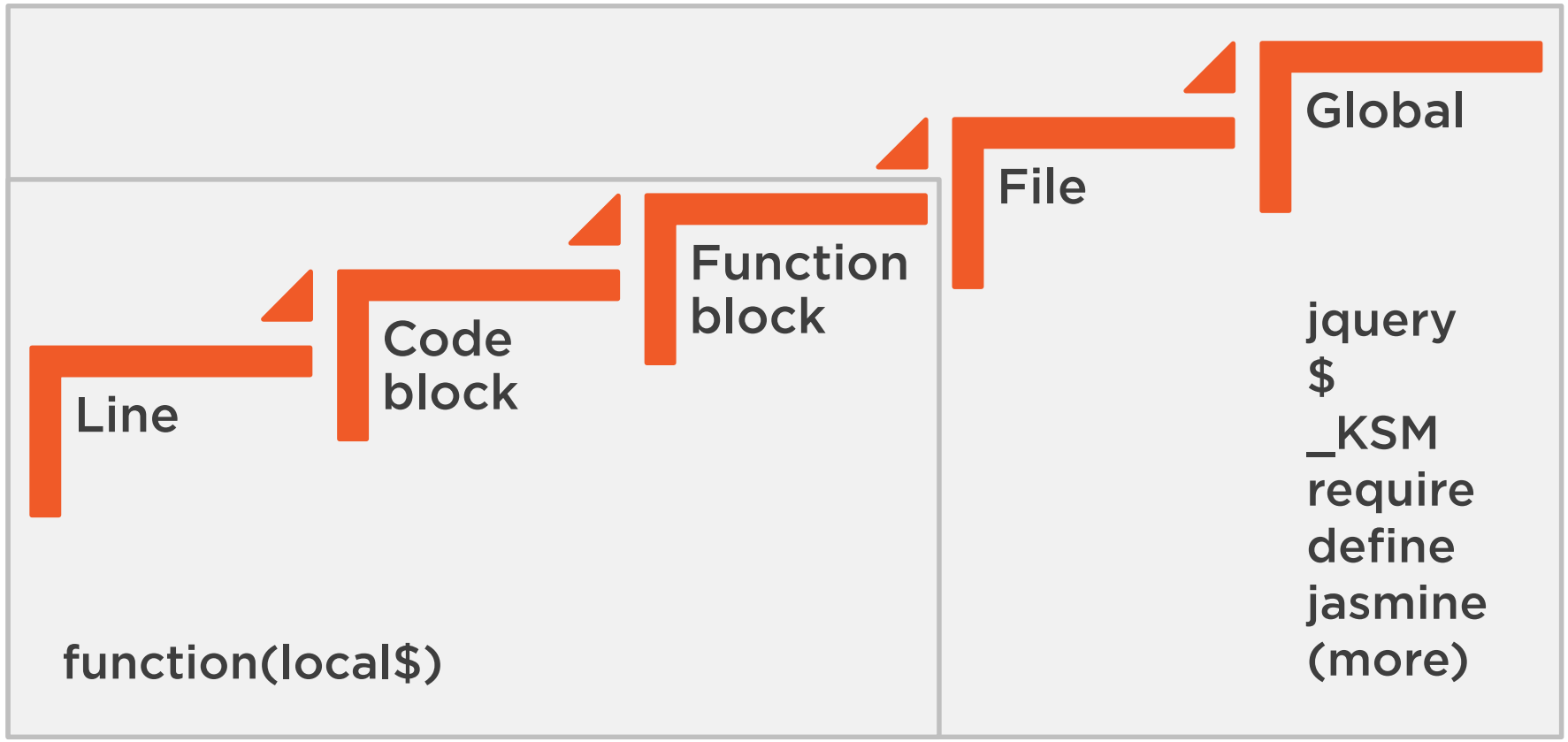
Global Namespace vs. Local Scope

Local function parameter name could be anything

jQuery still available globally after loaded

Variable may still exist in global namespace

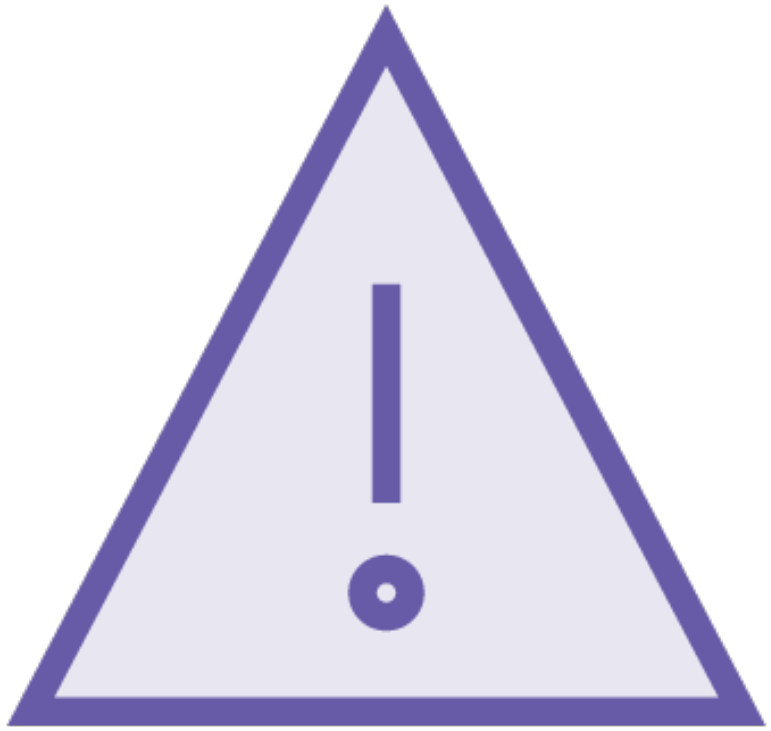
Variable Resolution



Global Variables

References to module return values are only available locally

Legacy libraries will still populate global namespace even when used with RequireJS



Maintain consistency

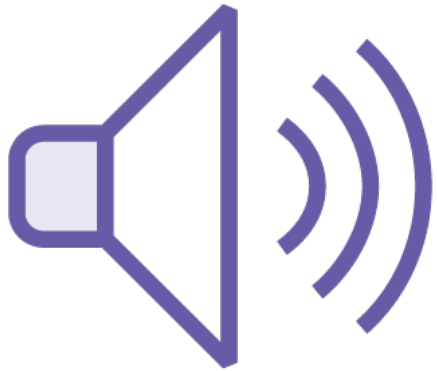
Adhere to standards

Write code for humans

Two Choices

Refactor

Configure RequireJS



Stay tuned

Using jQuery with AMD

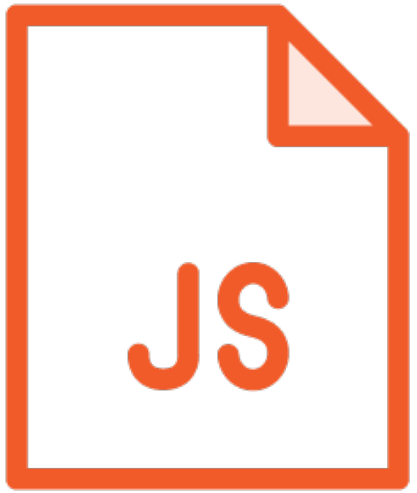
Supports AMD

Robust support

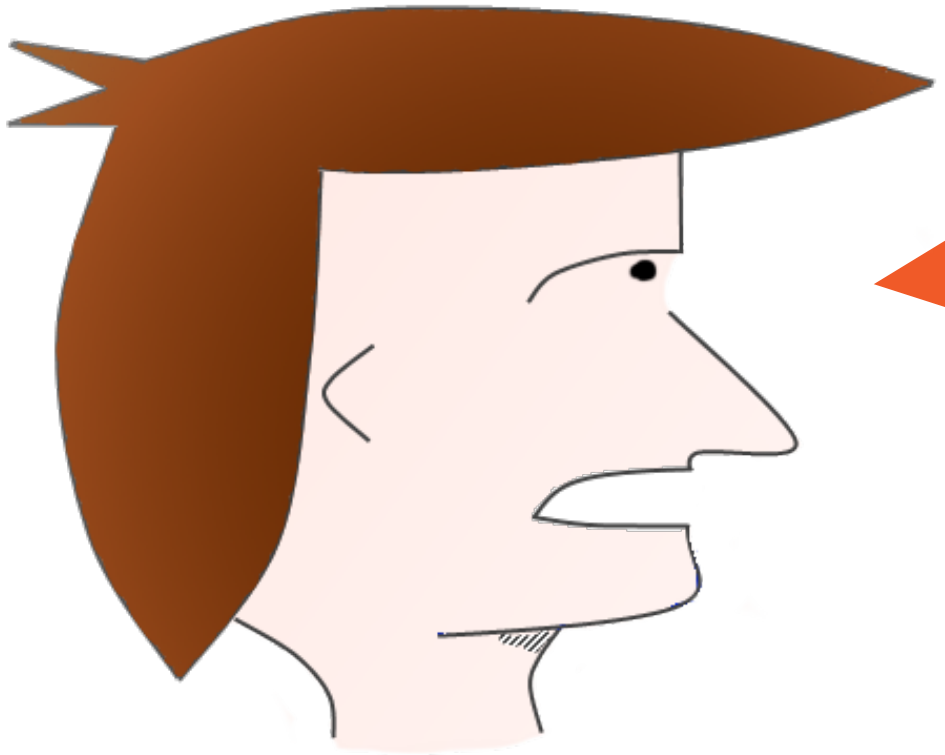
Version 1.7 and above

Changing Our Libraries

Refactoring



Changing names



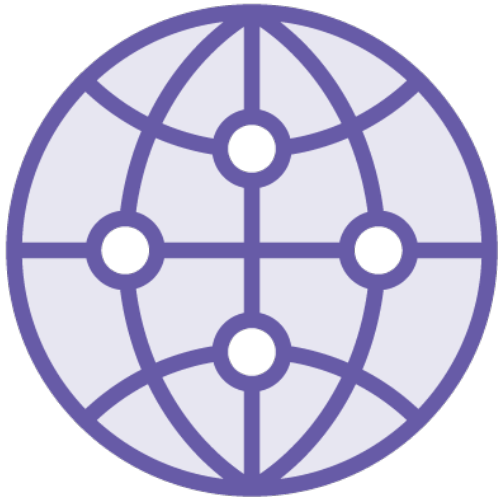
Define? What happened to Require? Will you make up your mind?

Over Simplifying

"define" is used to establish a module for future and current use

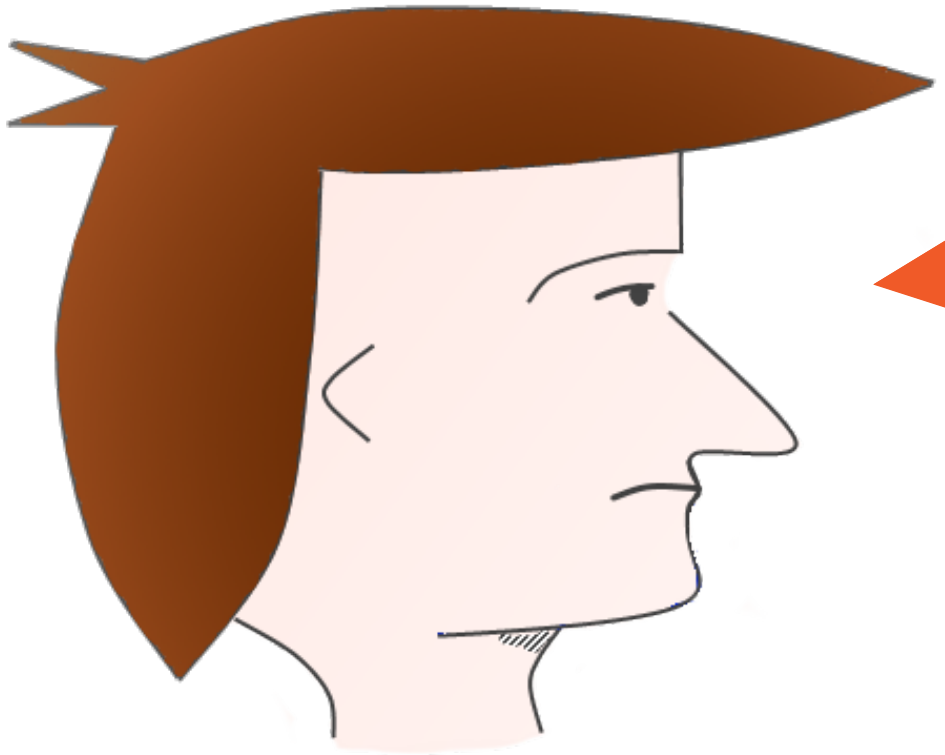
"require" is used for single use needs

Local References



No conflicts with global values

Interesting possibilities



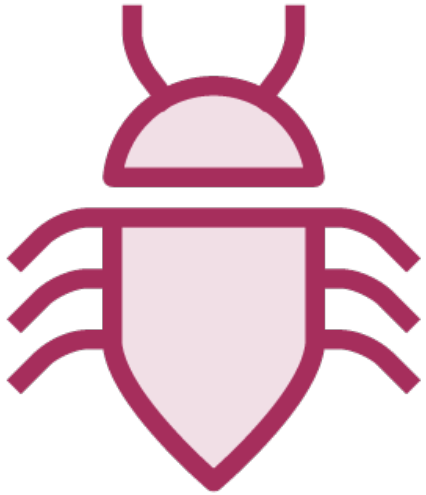
Fine, but I still think you should have shown me.

No conflicts with

- Global code
- Other modules
- Legacy libraries

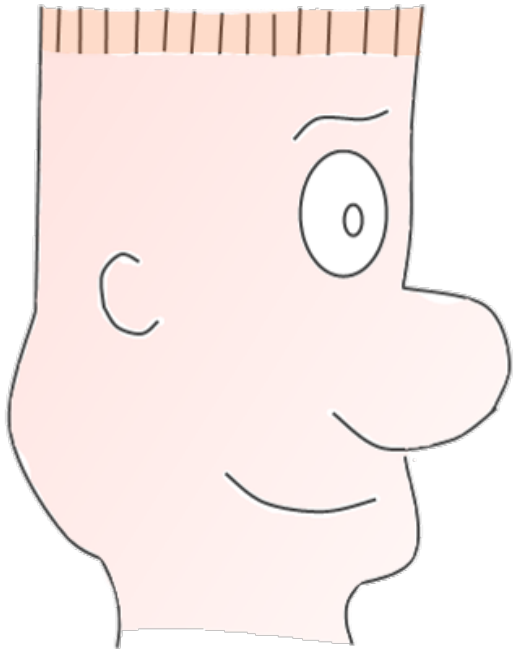
Awesome!

Using Locally Scoped Object

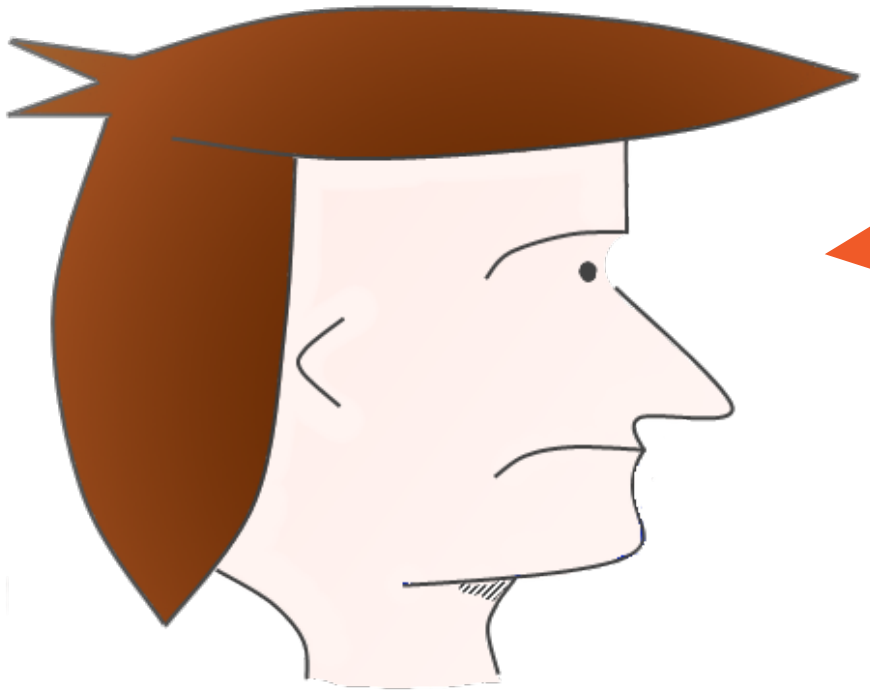


Namespace undefined

Container needed



Just change the code, it's no big deal!



Why extend values onto an empty object that only has local scope? It seems completely unnecessary!

Creating New Functionality

Make incremental and easily testable changes

Focus on enhancements rather than debug what used to work

Using Identical Object Structures

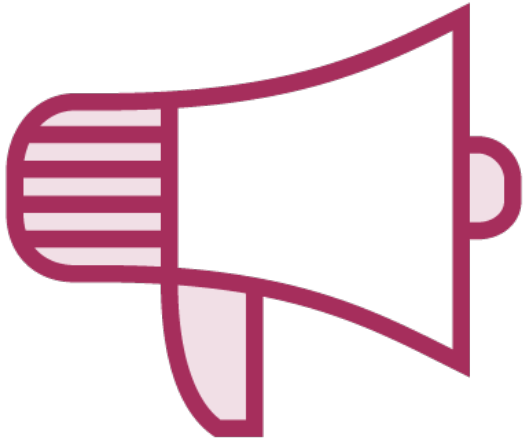
New modules don't have to support legacy object structures

Useful technique when migrating or refactoring

Variables resolve locally instead of globally

Watch out for references to "window" objects

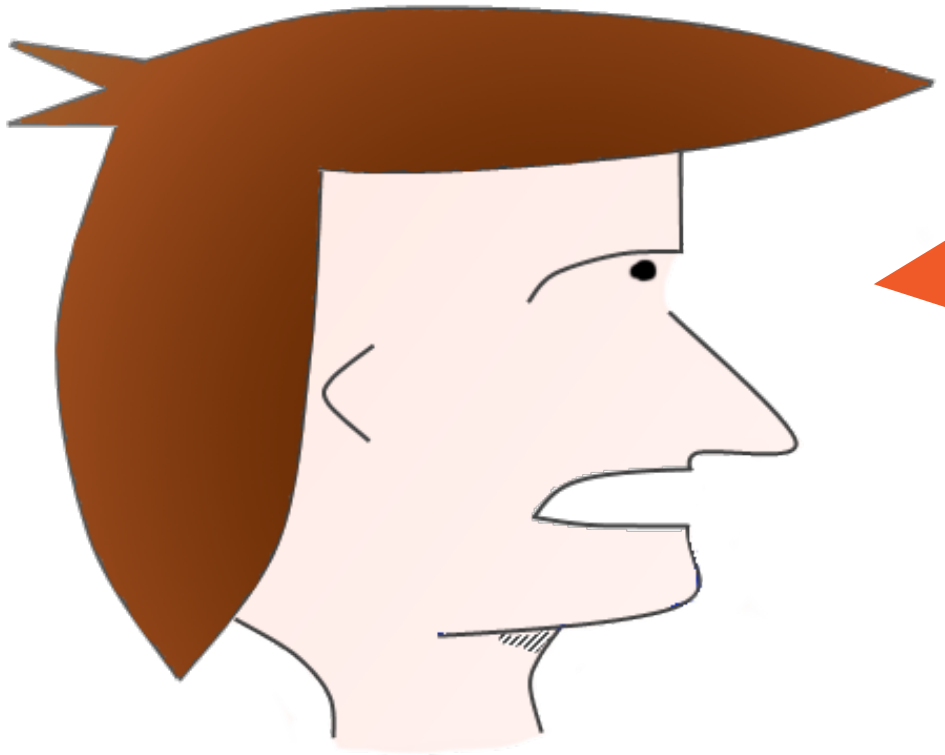
A Note About “define”



Return a value from “define”

Be useful

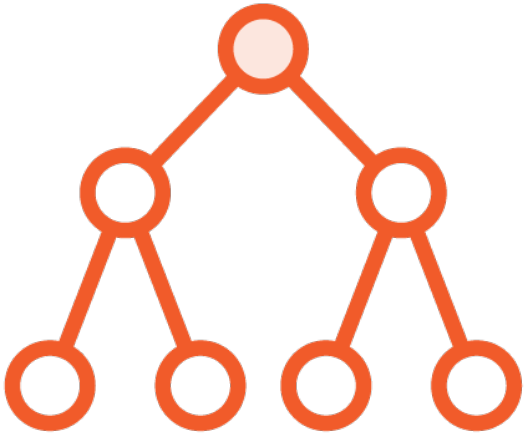
There are other options



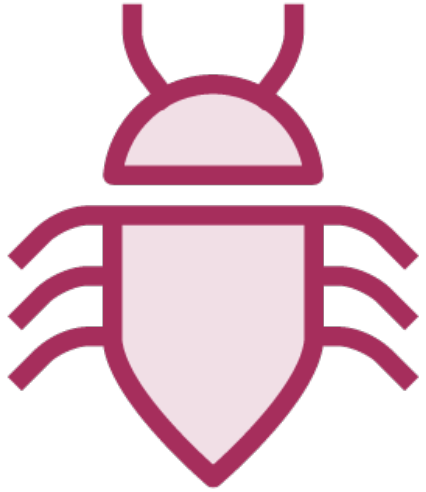
Why don't you just return a reference to _KSM?

Global _KSM Object

Provides organization

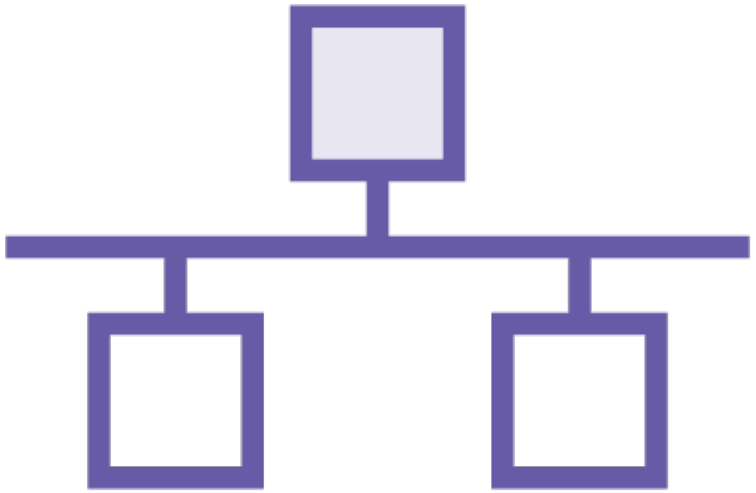


Requires structural knowledge

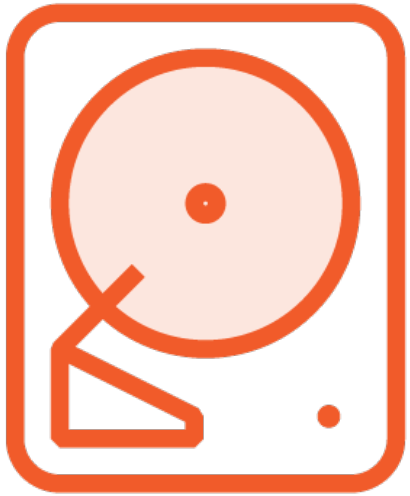


Structure change causes errors

Benefits of Modules



Independent of other code



Location of file doesn't matter

Loading Dependencies

Should they be listed in some order?

Order of dependencies is unimportant

Sibling dependencies load asynchronously

Nested dependencies properly loaded

Complex dependencies can be configured

Legacy dependencies may be less obvious

The order that asynchronous modules are loaded is unknown unless dependencies are specified

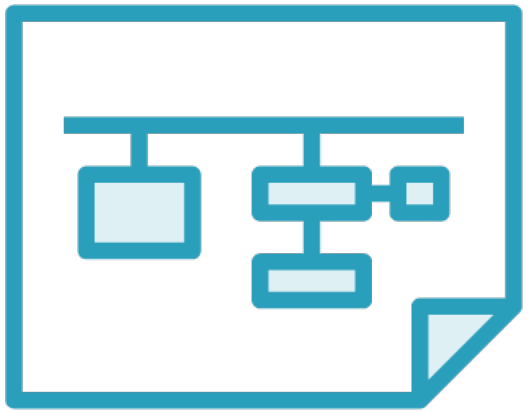
Varying Results

“It works on my machine”

During the customer demonstration

Callback function variables are presented in the same order as they are listed in the dependency array

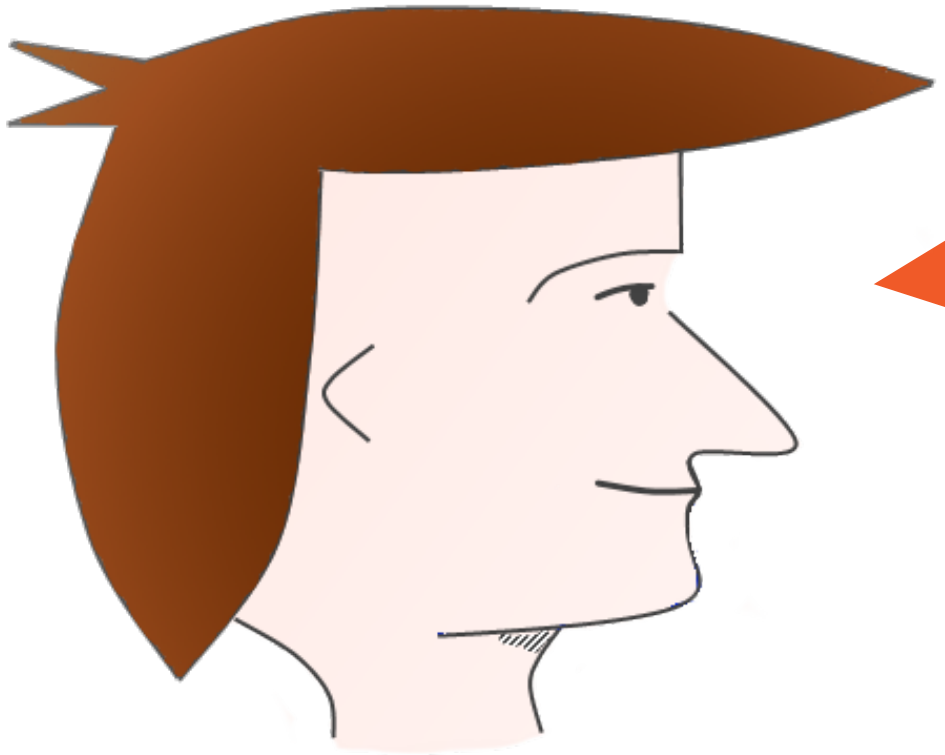
Homework Assignment



Dependencies exist

Is jQuery required?

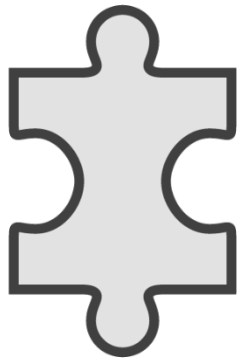
Accessing the jQuery variable



I bet the $ variable is still available since it's in the global namespace

Discussing _KSM Object

Numerous existing references



Returns a module reference

Not changing legacy code

Consistent naming conventions

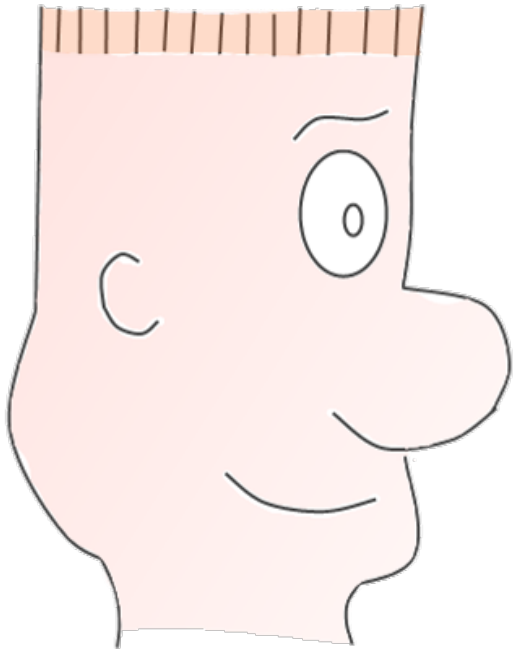
Repeatable coding patterns

Easier comprehension

No surprises!

Potential Error

Configuration settings no longer in global object



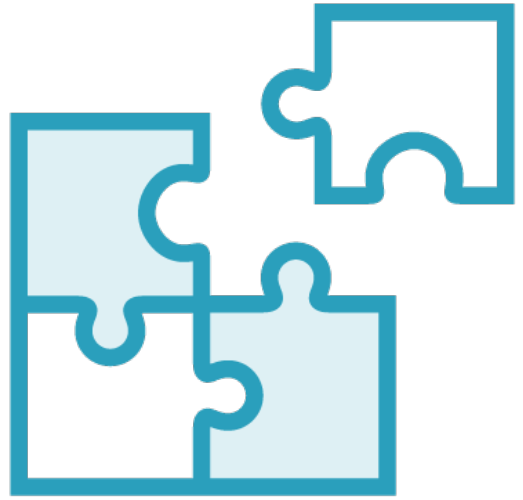
Since there is a test for the event handler, it's probably ok if it doesn't exist. I wouldn't worry about it.

Refactoring Legacy Code

Opportunity to review expectations

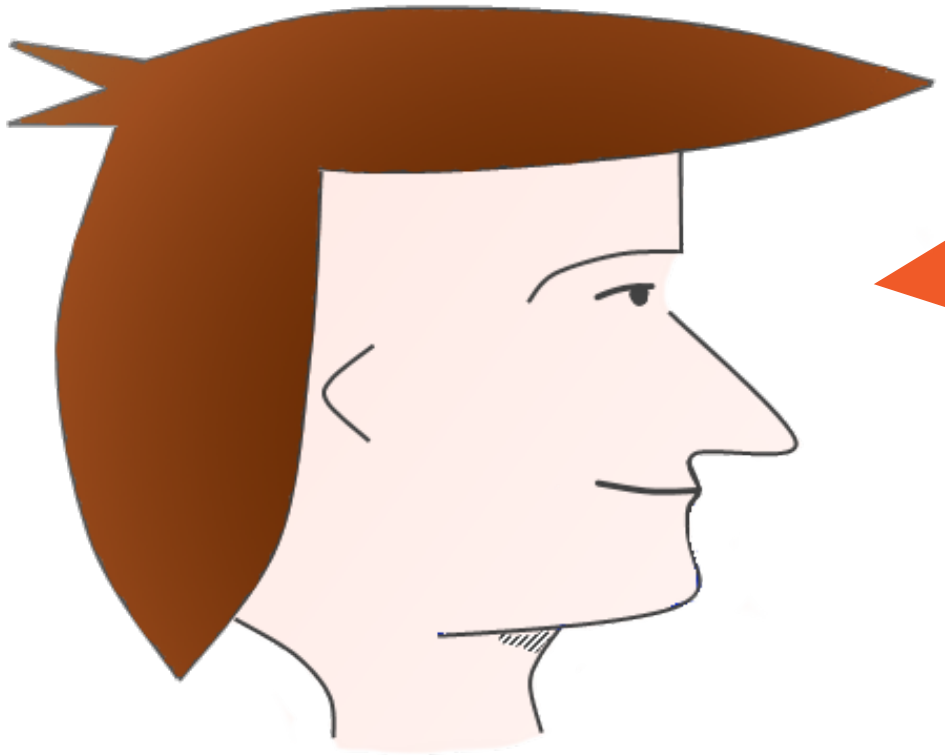
Deadlines may limit changes

Not always an easy task



Research to prevent breakage

Pop Quiz



Nice try! I didn't fall for your trap. I had to think about it, though.

HTML and Code

HTML for presentation

Code for logic

Additional Learning Challenge

Adding another supported language to the existing pattern

Making sure changes produce the expected and desired results

Summary

Created footer module

Startup code

Refactored legacy code

- Language library
- Toolbar library