THE SHORTCUTS TO JS PARADISE

PAIR PROGRAMMING AND PRACTICE

WHAT WE'LL COVER

- Stucturing JS
- Javascript Libraries
- JS Frameworks vs DIY
- Using multiple libraries
- Debugging / logging
- Testing (manual, automated and semi-automated)
- Production tools and techniques
- Deploying to production
- Any other best practices

" JAVA IS TO JAVASCRIPT WHAT CAR IS TO CARPET. "

Chris Heilmann

NO NEED FOR INTRODUCTIONS

- I don't endorse the illegal sharing of digital resources

NO NEED FOR INTRODUCTIONS

- I don't endorse the illegal sharing of digital resources
- Professional JavaScript for Web Developers
    - http://m.friendfeed-media.com/95a8434720c64a63e96ff8c4364fb595d9e98c36
- JavaScript the Good Parts
    - http://bdcampbell.net/javascript/book/javascript_the_good_parts.pdf
- What do you know?

FIRST, SOLVE THE PROBLEM. THEN, WRITE THE CODE.

Structuring JS

WHAT YOU SHOULD CONSIDER

- System architecture–The basic layout of the codebase. Rules that govern how various components, such as models, views, and controllers, interact with each other.
- Maintainability–How well can the code be improved and extended?
- Reusability–How reusable are the application's components? How easily can each implementation of a component be customised?
- The Design of Code: Organising JavaScript - http://alistapart.com/article/the-design-of-code-organizing-javascript
- Learning JavaScript Design Patterns - https://addyosmani.com/resources/essentialjsdesignpatterns/book/

STRUCTURING JS EXERCISE

MANIPULATING THE DOM

- document.getElementById()

- document.getElementsByTagName()

- document.getElementsByClassName()

```
for(var i = 0; i < document.getElementsByClassName("hello").length; i++) {
    console.log(document.getElementsByClassName("hello")[i].innerHTML)
}
```

MANIPULATING THE DOM

- document.querySelector()
- document.querySelectorAll()
- document.querySelectorAll("h1")[0].innerHTML = "Hey Funnelback"

MANIPULATING THE DOM EXERCISE

RESTRUCTURING... OR REFACTORING EXERCISE

WALKING ON WATER AND DEVELOPING SOFTWARE FROM A SPECIFICATION ARE EASY IF BOTH ARE FROZEN.

JS Libraries - An Introduction

WHAT ARE LIBRARIES?

- Libraries are levels of abstraction on top of vanilla JS
- They provide:
  - Helpers
  - Guidelines
  - Solutions to common tricky problems

WHAT DO THEY DO?

- Simply - Libraries do all things
- This list will help to show the variety of library functions:
    - Manipulating the DOM
    - Saving to local storage
    - Creating routes
    - Graphing and Data Visualisation
    - Making further requests after page load
    - Provide templating systems for data structures

POPULAR LIBRARIES

- jQuery
- Handlebars
- Underscore/LoDash
- D3/DC
- Angular
- React

WHY SHOULD YOU USE LIBRARIES?

- Don't Reinvent The Wheel
  - Someone already wrote this once, they tested it in real life scenarios and debugged all of the issues already
- Do More With Less Code
  - There are many helper libraries and one off functions out there, make use of them
  - E.g. Updating query string by JS
    - http://stackoverflow.com/questions/5999118/add-or-update-query-string-parameter
- Save Time -- You Don't Code Your Own OS, Do You?
  - "Look at my shiny new machine, just 453 weeks until I can use it"
- Chances Are, You Aren't The Expert
  - This links back to not reinventing the wheel, someone has painstakingly tested and debugged this exact issue before
- Speed Thrills
  - Again, with testing comes refactoring and reworking, this often leads others to think about performance
- Avoid Cryptic JavaScript Base Code
  - Using common helper functions will help your future self and other developers work with the code base in future projects

WHY SHOULDN'T YOU USE LIBRARIES?

- Libraries are awesome. They make the arduous tasks quicker so you can get onto the fun stuff.
- Because of this, many developers will default to using libraries in every build.
- Scenario: 'I need to select one element on one page and hide it'...

LOADS JQUERY!!!

WHY SHOULDN'T I USE LIBRARIES?

- This is not a good use of a library.
    - We've now loaded and initiated around 11,000 lines of code, good job
    - Alternatively we could have used

```
var el = document.getElementById(".search");

el.style.display = 'none';
```

    - We've saved around 10,998 lines of code here!
- I would classify good use of a library as 'the use of at least 50% of its functions and concepts'.
- http://youmightnotneedjquery.com/
- https://twitter.com/iamdevloper/status/710833889825001472

JQUERY EXERCISE

IF DEBUGGING IS THE PROCESS OF REMOVING SOFTWARE BUGS, THEN PROGRAMMING MUST BE THE PROCESS OF PUTTING THEM IN. .

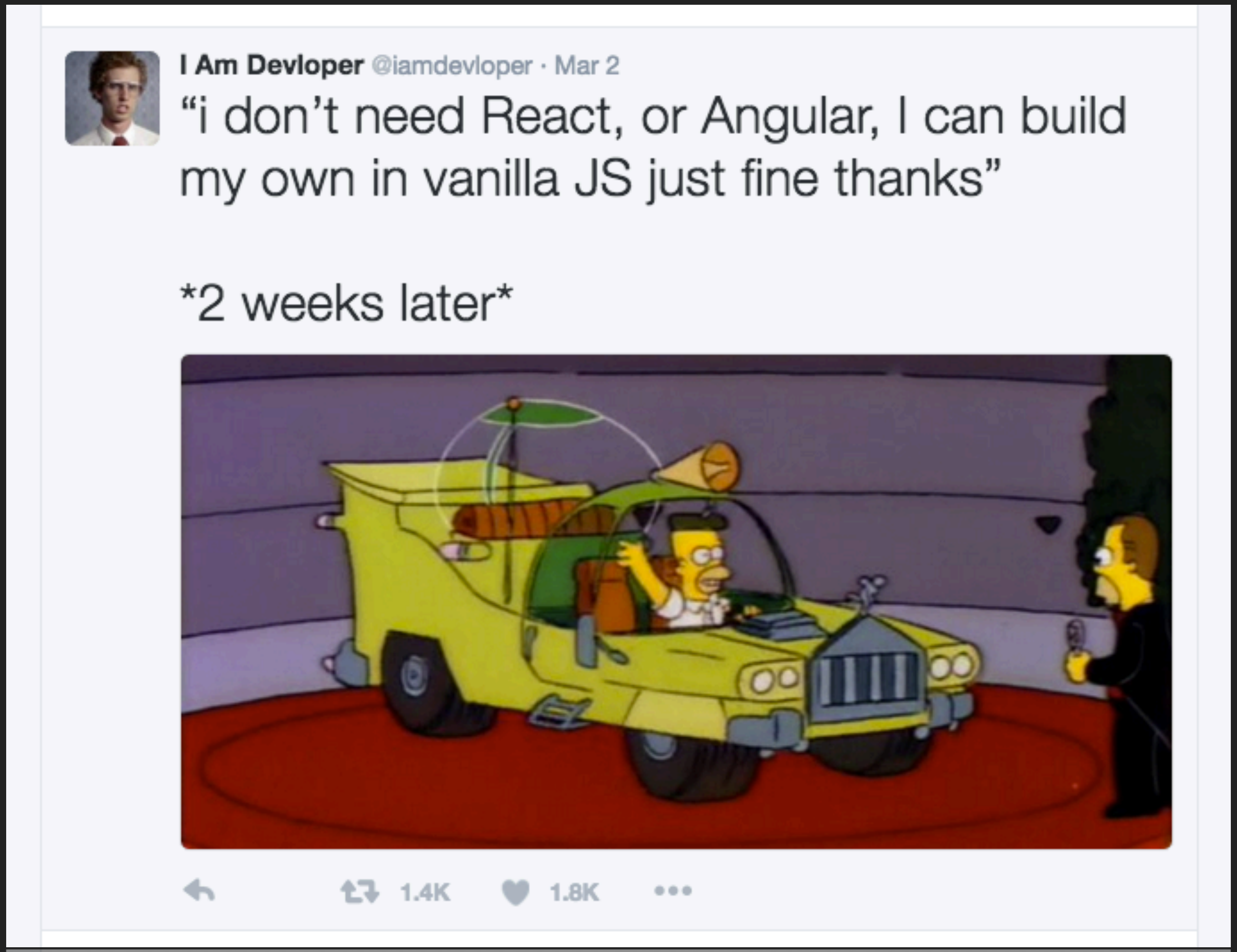
Debugging and Logging

WHAT IS DEBUGGING AND LOGGING?

- Debugging and Breakpoints
    - debugger;
    - * Adding breakpoints on the fly *
- Call Stack Pane - Shows everything that has been executed up until that breakpoint
    - This helps to give an idea of what has helped to form a particular object within the current function
    - The async option includes any asynchronous JS actions like XHR request callbacks
- Scope Pane - Shows the scope of both local and global variables. It also shows the state of 'this'.
- Breakpoints Panes - Shows any active and not active breakpoints

MOST SOFTWARE TODAY IS VERY MUCH LIKE AN EGYPTIAN PYRAMID WITH MILLIONS OF BRICKS PILED ON TOP OF EACH OTHER, WITH NO STRUCTURAL INTEGRITY, BUT JUST DONE BY BRUTE FORCE AND THOUSANDS OF SLAVES.

Alan Kay



I Am Devloper @iamdevloper · Mar 2

"i don't need React, or Angular, I can build my own in vanilla JS just fine thanks"

2 weeks later

1.4K 1.8K

JS FRAMEWORKS

WHAT ARE JS FRAMEWORKS?

- At the core of a JS Framework, it's just a library
- JS frameworks are more than just functions and helpers though, they're whole concepts!
- They encourage a certain way of writing JS
- They attempt to introduce a more strict Object Oriented construct.
- Common concepts across all modern frameworks include: MVC, 2 way data binding, RESTful HTTP packages, routing, templating and much more.
- Features and Browser Support: https://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks#Features
- http://i.imgur.com/kxglKG0.png

WHY SHOULD YOU USE FRAMEWORKS?

- For large JS based applications
- Frameworks impose a strict but flexible set of rules on a developer, straying from these guidelines usually cause pain
- One rule is that of a more component based approach
  - Editing one part of an application should never have a detrimental impact on another section of the application
  - Multiple developers can work on an application at the same time because they will be working within the context of their own component
  -

WHY SHOULDN'T YOU USE FRAMEWORKS?

- As I mentioned before, frameworks are just libraries. The same rules apply…
- If you're not using over 50% of the concepts within a framework then do not use it.
- Web Applications !== Web Sites
  - Frameworks offer functionality to accommodate common application based conundrums, the same dilemmas are usually not present for a website
- http://community.algolia.com/instantsearch.js/examples/e-commerce/
- What this example does well:
  - Modularisation
  - RESTful requests by the react framework module
  - 2 way data binding
  - Single page
- What this example doesn't do well:
  - Go to any other pages
  - Create routes for filtering and searching

I Am Devloper @iamdevloper · Jan 11

When people say "GUI interface"

From Development to Production

OVERVIEW

- In this section we'll cover all of the tools that we use from the start of a project build right through to a production release.
- These include:
  - A suitable task runner and build process
  - Unit Testing
  - A versioning and release system, git and GitLab.
  - Continuous Integration

TASK RUNNERS AND BOILERPLATES

- Task Runner Candidates:
  - Grunt
  - Gulp
- Its not always necessary to right your own build file from scratch. Much of the time you should start with a configuration that already exists.
- Foundation 6 Boilerplate
  - http://foundation.zurb.com/sites/docs/installation.html
  - Ships with Gulp + Lots of useful packages
- Creating a new foundation project

FOUNDATION PACKAGES

- Build processes are always different for every project, however, there are some similar aspects like:
    - HTML Parser (Ships with Foundation 6 - Panini)
    - SASS Parser (Ships with Foundation 6 - gulp-sass)
    - Script concatenation, reduction and minification tools (Ships with Foundation 6 - gulp-concat, gulp-uglify and gulp-uncss)
    - Documentation tools (Custom addition to build process - sc5-styleguide and gulp-jsdoc)
    - Live Reload (Ships with Foundation 6 - BrowserSync)

UNIT TESTING

- Testing is the most difficult part of writing JS
- To write effective unit tests the JS itself must be in a suitable state
- There are two occasions where unit tests can run
  - Unit tests can run as part of the build process on your local machine (This runs your local code only)
    - There are many gulp packages for local unit testing one of which is called gulp-mocha, which allows full mocha testing
    - https://mochajs.org/
    - We are starting to use this set up on new clients and ensuring that the code base is written in a way that is unit testable
  - Or on GitLab via GitLab Runner (This runs all contributed code - which could be a potential release candidate)

CONTINUOUS INTEGRATION (CI)

- "Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early. By integrating regularly, you can detect errors quickly, and locate them more easily."
- http://www.yegor256.com/2014/10/08/continuous-integration-is-dead.html