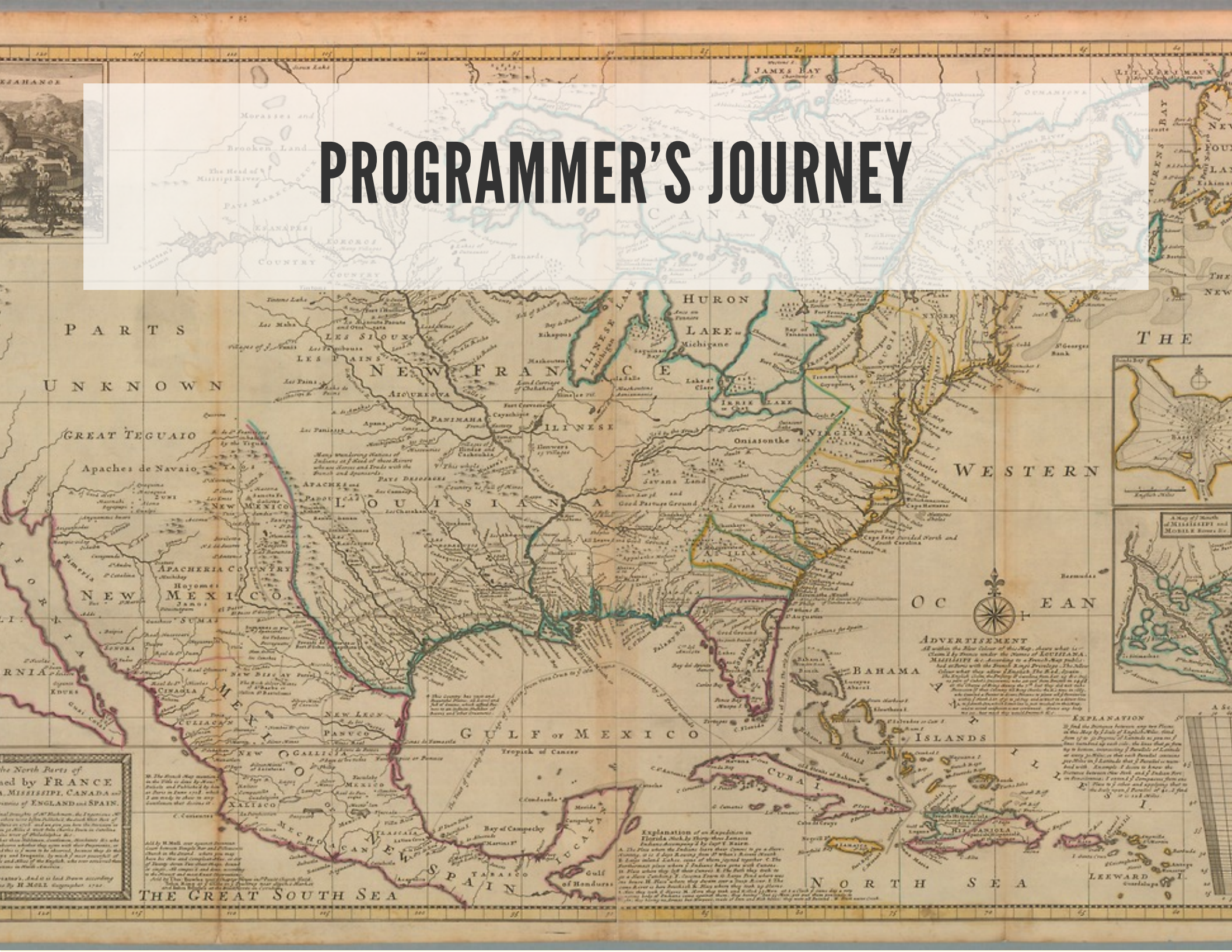

BLIND FAITH AND
BEST PRACTICES
Mike Behnke
@localpcguy

STARTING OUT


PROGRAMMER'S JOURNEY

THERE MUST BE A BETTER WAY

At some point while learning, we've all come to this point, right?

ENTER...THE BEST PRACTICE

Commercial or professional procedures that are accepted or prescribed as being correct or most effective.

BEST CODING PRACTICES

==

a set of informal rules that the software development community has learned over time which can help improve the quality of software.

MISGUIDED?

- Most "Best Practices" in Software Engineering are there to keep bad programmers from doing too much damage
- The only "best practice" you should be using all the time is "Use Your Brain"

BLIND FAITH?

==

a belief without true understanding,
perception, or discrimination

Are Best Practices just blind faith in
people that came before us
or that we deem smarter than us?

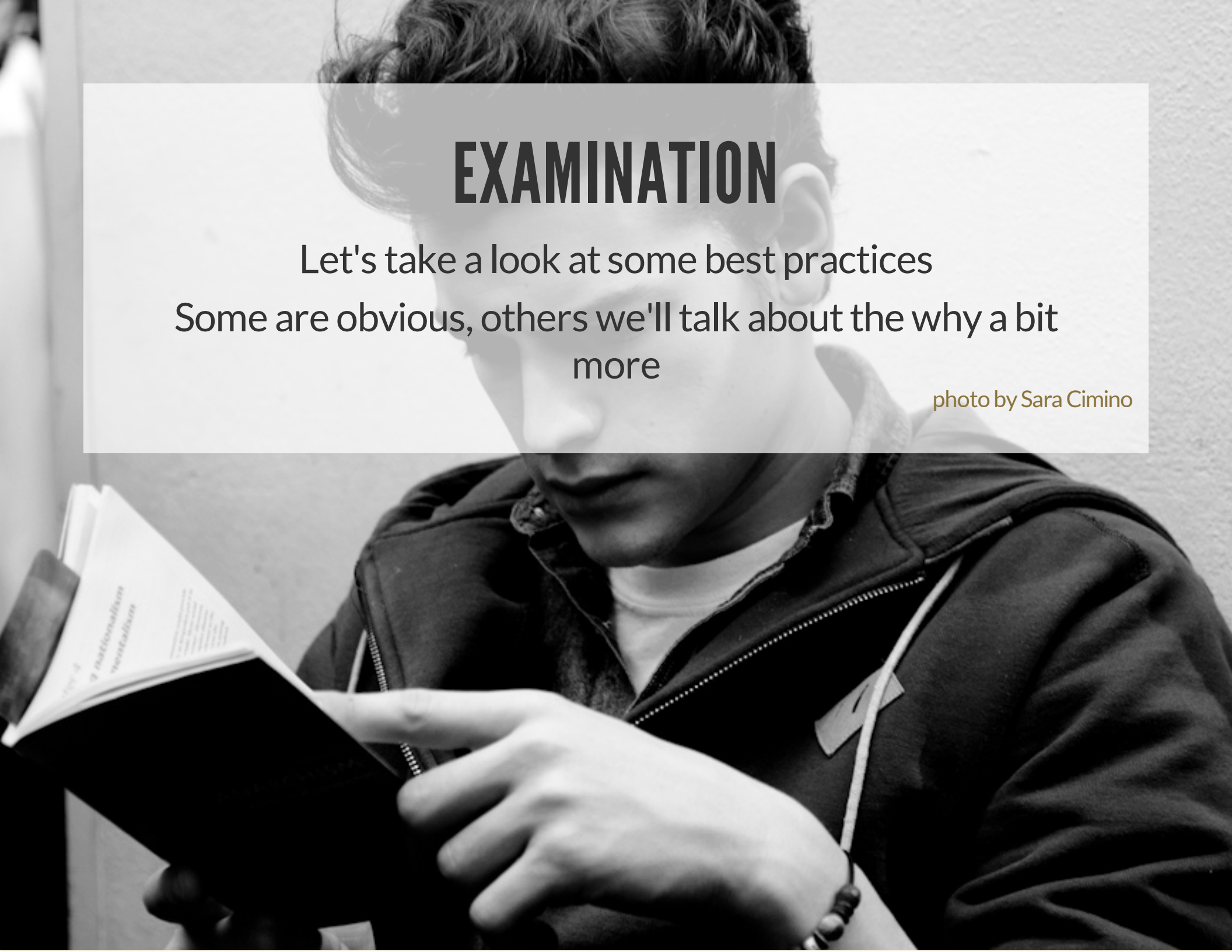
MY GOAL

To encourage everyone to look at best practices and know WHY they use the ones they do, and WHEN to discard ones that no longer are effective.

AND TO ENCOURAGE BEST PRACTICES...BUT

Make sure they really are, and know when to follow them

And when to ignore them



EXAMINATION

Let's take a look at some best practices

Some are obvious, others we'll talk about the why a bit more

photo by Sara Cimino

HTML

SEMANTICS MATTER

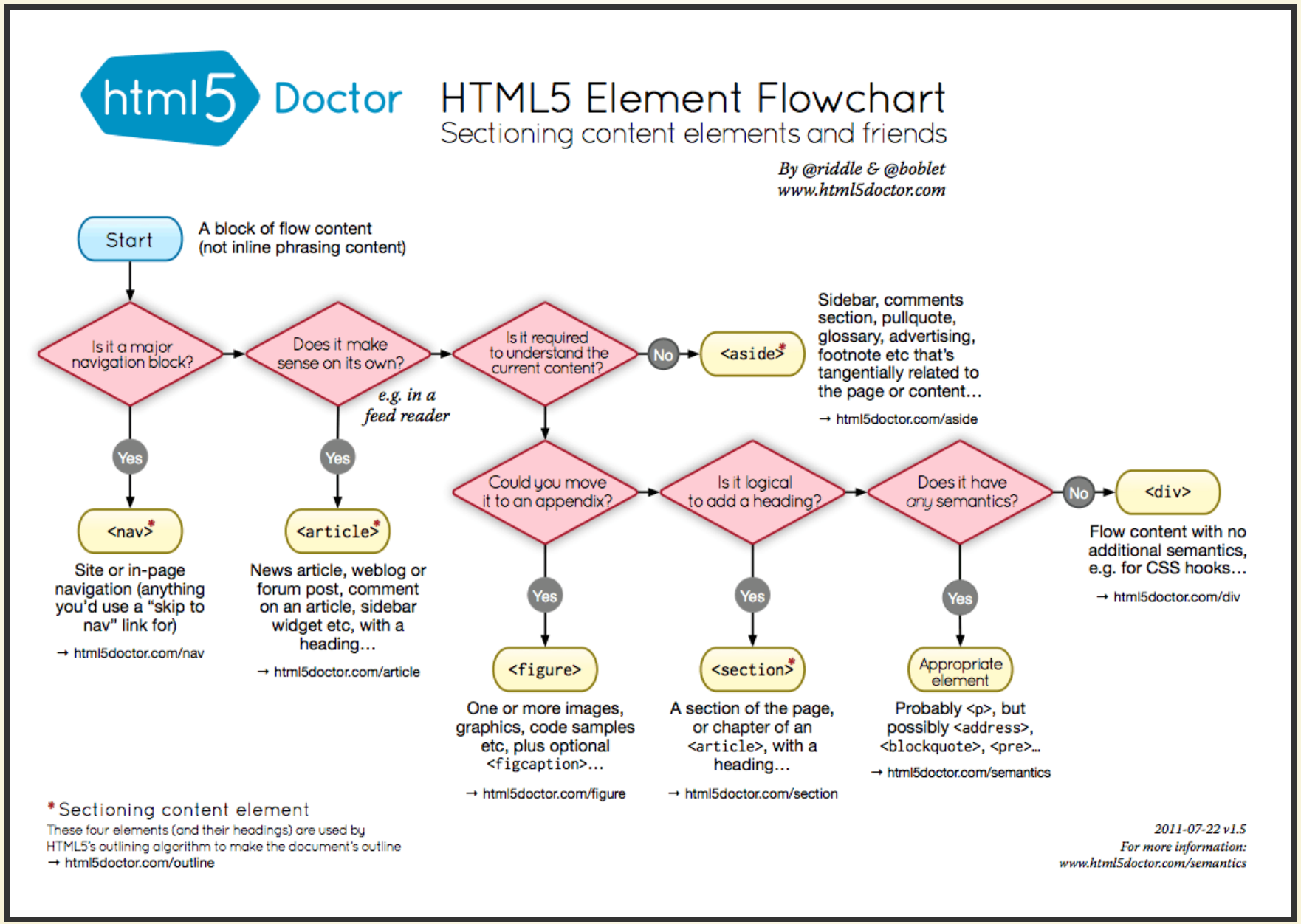
Help screen readers, Google, and other machines/bots parse meaning from markup

```
<div class="header">
    <ul class="nav"></ul>
</div>
<div class="body"></div>
<div class="footer"></div>

<!-- vs -->

<header>
    <nav></nav>
</header>
<section></section>
<footer></footer>
```

HTML5 SEMANTICS



EVERYTHING IN THE PROPER PLACE

- JavaScript:
  - just before body close
  - with exceptions for scripts like Modernizr
  - analytics?

- CSS
  - in the head - prevents layout readjustment
  - before any JavaScript to prevent possible blocking

HTML5 BOILERPLATE

- Great starting point.
- "A professional front-end template for building fast, robust, and adaptable web apps or sites."
- Great reference guide
- Performance - check out the various server configs

HTML5 BOILERPLATE
THE CODE

```
<!doctype html>
<html class="no-js" lang="">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title></title>
        <meta name="description" content="">
        <meta name="viewport" content="width=device-width, initial-scale=1">

        <!-- Place favicon.ico and apple-touch-icon(s) in the root directory

        <link rel="stylesheet" href="css/normalize.css">
        <link rel="stylesheet" href="css/main.css">
        <script src="js/vendor/modernizr-2.8.0.min.js"></script>
    </head>
    <body>
        <!--[if lt IE 8]>
            <p class="browsehappy">You are using an <strong>outdated</strong>
```


CSS

GENERAL RULES

- Use a reset (or normalize) style sheet
- Avoid browser hacks
  (use HTML Conditional Comments to set body classes for old IEs)

```
<!doctype html>
<!--[if lt IE 7]>      <html class="no-js lt-ie9 lt-ie8 lt-ie7" lang=
<!--[if IE 7]>         <html class="no-js lt-ie9 lt-ie8" lang=""> <![
<!--[if IE 8]>         <html class="no-js lt-ie9" lang=""> <![endif]-
<!--[if gt IE 8]><!--> <html class="no-js" lang=""> <!--<![endif]-->
```

CLEAN STYLESHEETS

Provide Stylesheet information and indicate structure

```
/* stylesheet for Corp Bar
File created date: 09.15.2010
Last modified date: 06.04.2012
By: [name]

For: [section]
*/

/* Table of Contents
- Typeography
- Link Styles
- Other sitewide styles
- Actions
- Layout
- HEADER
- TOP NAV
- MAINCONTENT
- FOOTER
```

SELECTORS

- Avoid IDs for styling - specificity problems, not reusable
- As short as possible, reduce specificity
- Use shorthand properties when possible

```
/* Bad */
#sidebar {
    background-color: #fff;
    background-image: (bg.png);
    background-position: 0 0;
    background-repeat: repeat-x;
    border-width: 1px;
    border-style: solid;
    border-color: #ffff00;
    font-family: Georgia, serif;
    font-size: 1.33em;
    line-height: 1.33em;
    font-weight: normal;
    margin: 10px 20px 10px 20px;
    padding: .1em;
}

/* Better */
```

MULTIPLE FILES

Prefer multiple, smaller CSS files over a single monolithic file.

Concatenate and minify for performance

PREPROCESSORS

Quickly becoming best practice to use a preprocessor

- Adds to developer efficiency
- Reduces errors
- Prevents repetition
- Careful: Don't over-complicate things
- Careful: Don't over-nest selectors

Oh, which one, Sass, Less or Stylus? Flip a coin, draw straws...or just base it on your framework

MODULARIZE YOUR CSS

SMACSS is a popular style

- Make elements portable, reusable
- No longer tied to a page or section

```
<div class="fldr fldr-callout">
    <h2 class="fldr-name">Folder Name</h2>
    <span class="fldr-items">(32 items)</span>
</div>
```

FRAMEWORKS

Bootstrap and Foundation are very popular now

They are NOT best practices by themselves however

BUT THEY DO ENABLE BEST PRACTICES IN MANY CASES


JAVASCRIPT

EASY ONE? == VS ===

All together now...

Know the difference, and use them appropriately

... wait, what? You thought it was always use ===?

```
// Use === when both type and value equality matter
// or when inconsistent input may cause problems
if(1 === '1') //Returns false
if(1 == '1') //Returns true

if(0 === '') //Returns false
if(0 == '') //Returns true

// Use == when the types are predetermined and known
// Or when it is more concise and clear than otherwise
if (typeof foo == 'undefined') // typeof always returns a string
if (foo.indexOf(bar) != -1) // indexOf always returns a number
if (foo != null)
// more concise than checking null and undefined, but still clear
```

USE YOUR SEMICOLONS!

```
var a = obj
[a].forEach(logProp)    // 'fail' : var a = obj[a].forEach(logProp)
```

Source - Ben Alman

ALWAYS USE CURLY BRACES

Goto fail bug

```
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
hashOut.length = SSL_SHA1_DIGEST_LEN;
if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
// ...
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;  /* MISTAKE! THIS LINE SHOULD NOT BE HERE */
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;

err = sslRawVerify(...); /* SSL certs never fully verified */
```

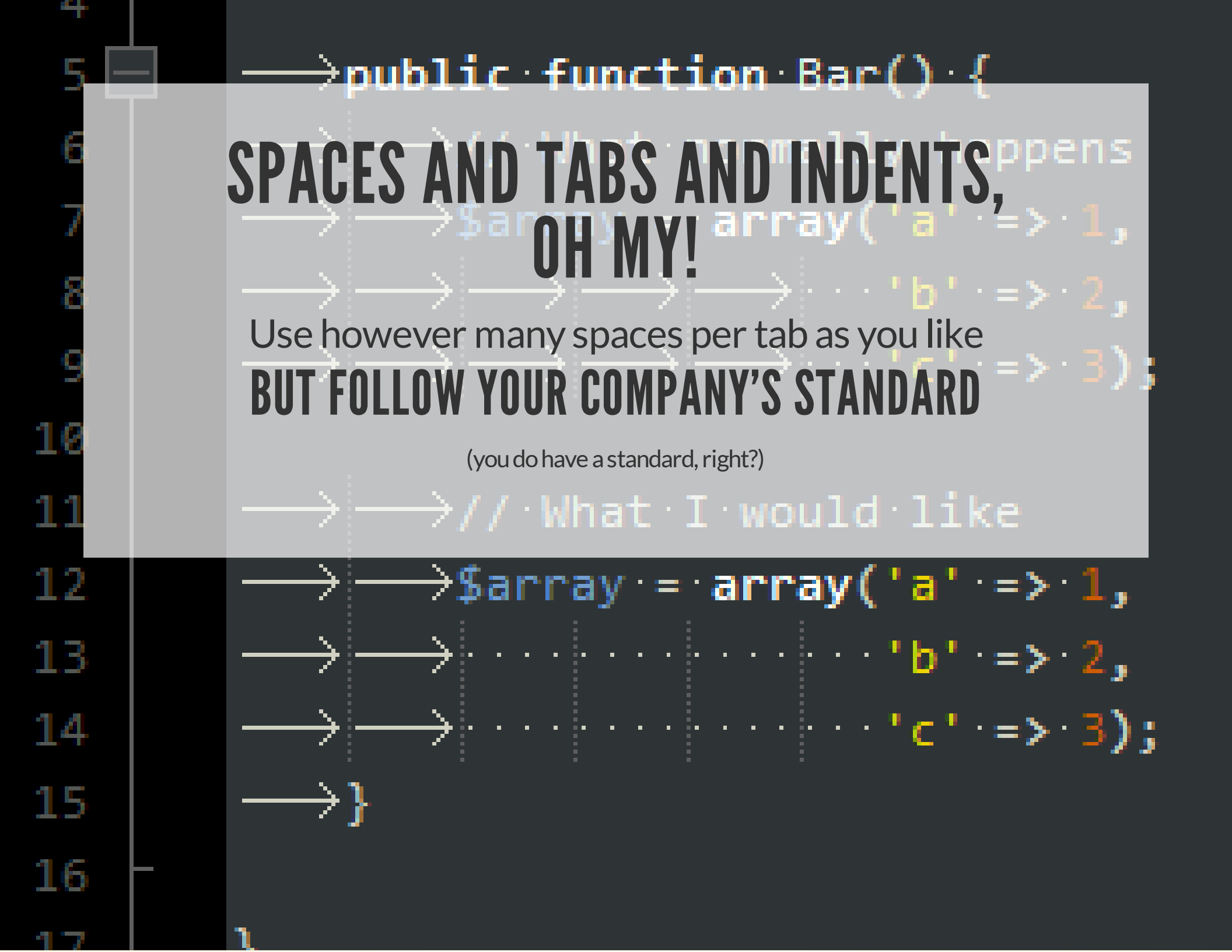
Yes...this is C, not JavaScript...example was too good to pass up

BLOCKS

Opening curly brace for blocks should go on the same line as the declaration (in JavaScript)

```
function foo() {
    var bar = 1;

    return // Oops! Return == undefined
    {
        bar: bar
    }
}
```


SPACES AND TABS AND INDENTS, OH MY!
Use however many spaces per tab as you like
BUT FOLLOW YOUR COMPANY'S STANDARD
(you do have a standard, right?)

JSLINT (OR HINT) YOUR CODE

Catch errors before you refresh.

SublimeLinter

USE STRICT

```
(function(){
    'use strict';
    name = 'Mike'; // Throws an error, no "var"
    var obj = { foo: true, foo: false }; // Error, duplicate property names
}());
```

Catch potential errors quicker, no side effects for old browsers.

[ref]

- Prevents global variables
- Prevents duplicate properties
- Prevents accidental function overwrite
- Makes eval (a bit) less evil

ENCAPSULATE YOUR CODE

```
(function(){
    // Do stuff here
}());
```

Along with "use strict", keeps your variables from leaking into the global scope, makes your code more modular

USE THE MODULE PATTERN

```
var app = (function(){
    'use strict';

    var settings;

    return {

        init: function(initialSettings) {
            settings = initialSettings;
        },

        getOptions: function () {
            return settings.options;
        }
    };
}());
```

Set a namespace, then expose just the parts of your module that you need exposed

History of JavaScript module patterns and variety of examples and links to more reading


JQUERY

STILL USE JQUERY?

- Makes the DOM tolerable
- On a diet lately (only 32KB minified/gzipped)
- Cross-browser compatibility, events, ajax
- Plugins
- Trusted by >50% of websites online

JQUERY IS JAVASCRIPT

So LEARN JavaScript first (at least the basics)

lpg.io/web-dev/javascript-resources

ISOLATE JQUERY

Pass jQuery into your module

```
(function ($, undefined) {
    // ... Do Stuff with $
    // Now jQuery scoped locally to $
}(jQuery));
```

Bonus: undefined really is undefined

JQUERY READY

Don't put all your code here, just use the
DOM Ready event listener to kick off your init function

```
(function ($, undefined) {
    var app = {
        init: function() {
            // Do init stuff here
        }
    }

    $(app.init); // app.init called on DOM ready
    // Same as: $(function() { app.init(); });
    //      or: $(document).ready(function() { app.init(); });
}(jQuery));
```

CACHE YOUR SELECTOR

If you are using a jQuery lookup more than once, cache the lookup in a variable

```
$('#item').text('Lorem Ipsum');
$('#item').css('color': '#c00');

// vs.

var $item = $('#item');

$item.text('Lorem Ipsum');
$item.css('color', '#c00');
```

Use a $ to designate that it is a jQuery object. Controversial?

CHAIN WHEN POSSIBLE

Use chaining

Format it nicely, more readable, better for debugging also

```
$item
    .text('Lorem Ipsum')
    .css('color', '#c00');
```

But don't abuse chaining

```
$(this).html("Back")
    .siblings("ul")
    .stop()
    .css({"top":0,"opacity":1.0})
    .fadeOut(500)
    .focus()
    .find("li:last")
    .addClass("lastTestimonial")
    .parent()
    .parent()
    .parent()
    .addClass("viewAll")
    .animate({width:697+"px"}, 1000)
    .find(".testimonialsCntnr")
    .animate({width:697+"px"}, 1000,
        function(){
            $(".seeAllTestimonials")
             siblings("ul")
```

PREFER .FIND() AND .CLOSEST()

More robust and succint than .children() or .parent()/.parents() for traversing the DOM

```
<div class="wrap">
    <ul>
        <li>
            <p><span class="elem">stuff</span></p>
        </li>
    </ul>
</div>
<script>
    var $elem = $('.wrap').find('.elem');
    var $wrap = $('.elam').closest('.wrap');
</script>
```

PREFER DEFINED FUNCTIONS TO ANONYMOUS WHEN POSSIBLE

- Cleaner, easier to read code
- Less nesting

```
(function ($, undefined) {
    var theater = {
        init: function() {
            // Setup control events
            $('.thtrWrap').on('click', '.theaterNavLink', theater.changeSlide
            $('.thtrWrap').on('click', '.theaterLeft', theater.slideNavPrev);
            $('.thtrWrap').on('click', '.theaterRight', theater.slideNavNext)
        },
        changeSlide: function(e) {
            // ...
        },
        slideNavPrev: function(e) {
            // ...
        },
        slideNavNext: function(e) {
            // ...
        }
    }
```

Bonus: Event delegation. Tip: listen on the closest parent element that isn't dynamic

DON'T PRE-OPTIMIZE

jQuery Sizzle (selector engine) is very fast

BUT BE WARY OF .EACH()

Loops can be slow with .each(),
consider using a for loop

TOOLS

Not Best Practices

BUILD

- Grunt
- Gulp
- Mimosa
- Ant, Make, Jake

SCAFFOLDING

- Yeoman
- Lineman

LANGUAGES

for tools to work

- Node.js
- Ruby

FRAMEWORKS

- Backbone
- Angular
- Ember
- Bootstrap
- Foundation

There is a reason we have so many people touting best practices, new frameworks, even new languages

FOR LOVE OF KITTEN PHOTOS, RIGHT?

IT'S ABOUT PASSION

We love this business, we love making things.

We set best practices to ensure the business
we love presents it's best face forward.

'BLINDLY FOLLOWING BEST PRACTICES IS NOT A BEST PRACTICE.'

- DAVE MARKEL

ABOUT

MIKE BEHNKE

- Front End Focused Engineer, dabble on the server-side
- JavaScript & Android Enthusiast
- Tech geek, futurist

- Site: Local-PC-Guy.com
- Tweets: @LocalPCGuy
- GitHub: Github.com/LocalPCGuy
- Slides: http://lpg.io/best-practices
- Employed: Enlighten Agency
  (We are hiring)