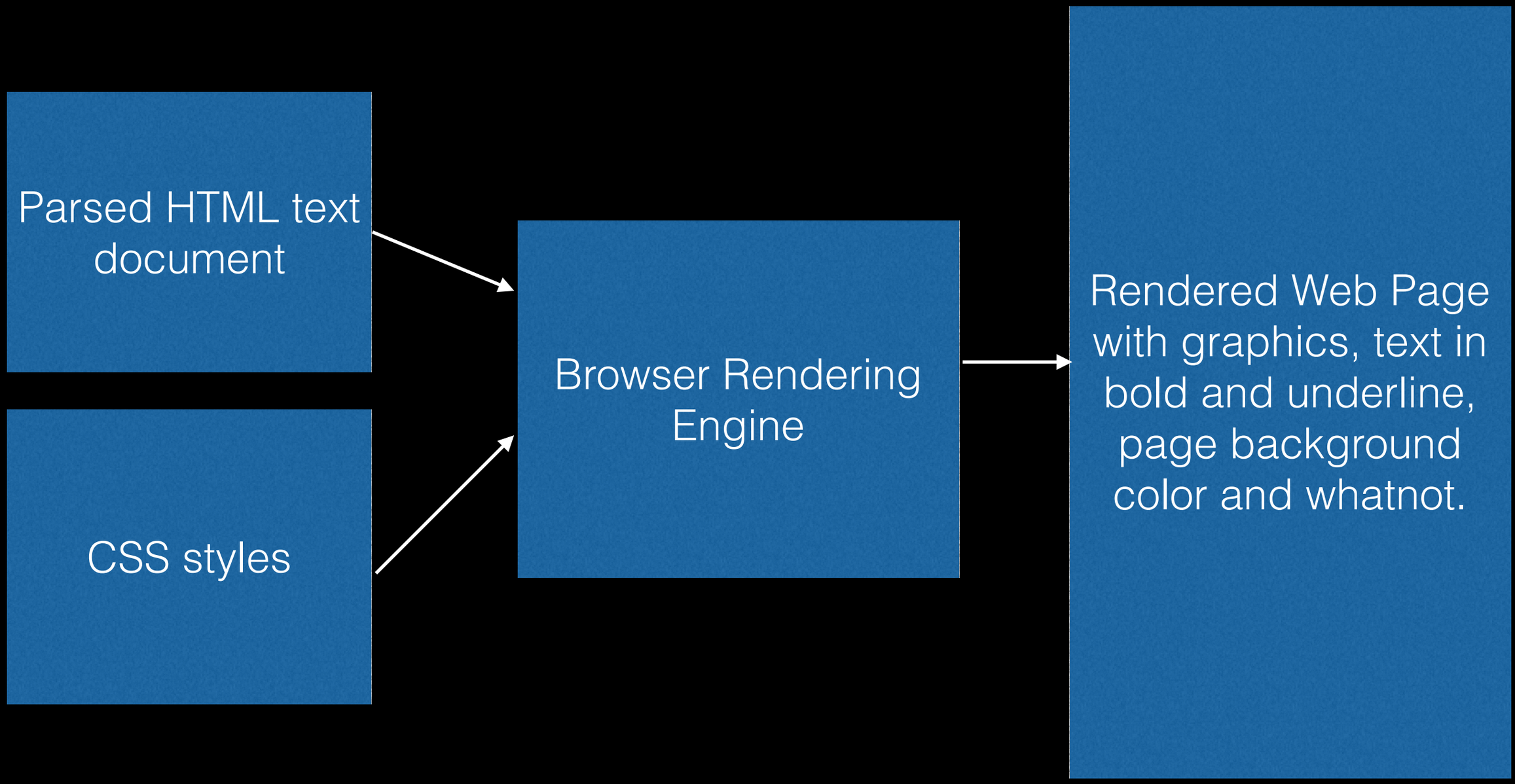
Do’s and Don’ts for adding HTML to Your App

Chris Woodard
Tampa Bay Cocoaheads
Copyright (c) 2014. All Rights Reserved.

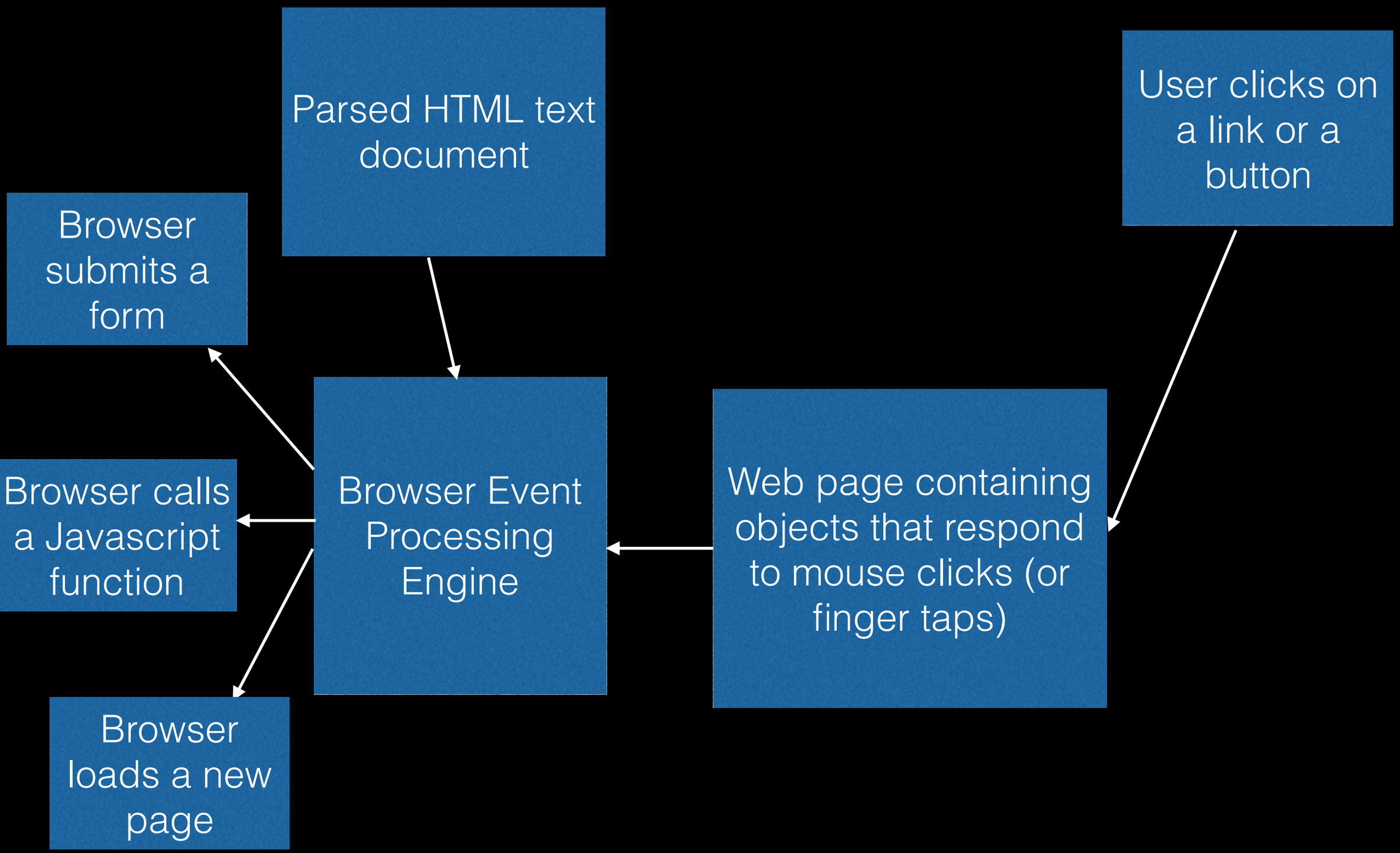
Why would you want to use HTML for an iPhone app in the first place?

- Your boss read an article in a magazine and thinks this is the “next big thing”.
- A client insists.
- You think you can write most of your app in HTML and have it run on iOS and Android.
- **You have content that needs to be pushed out more quickly than the app store release cycle.**

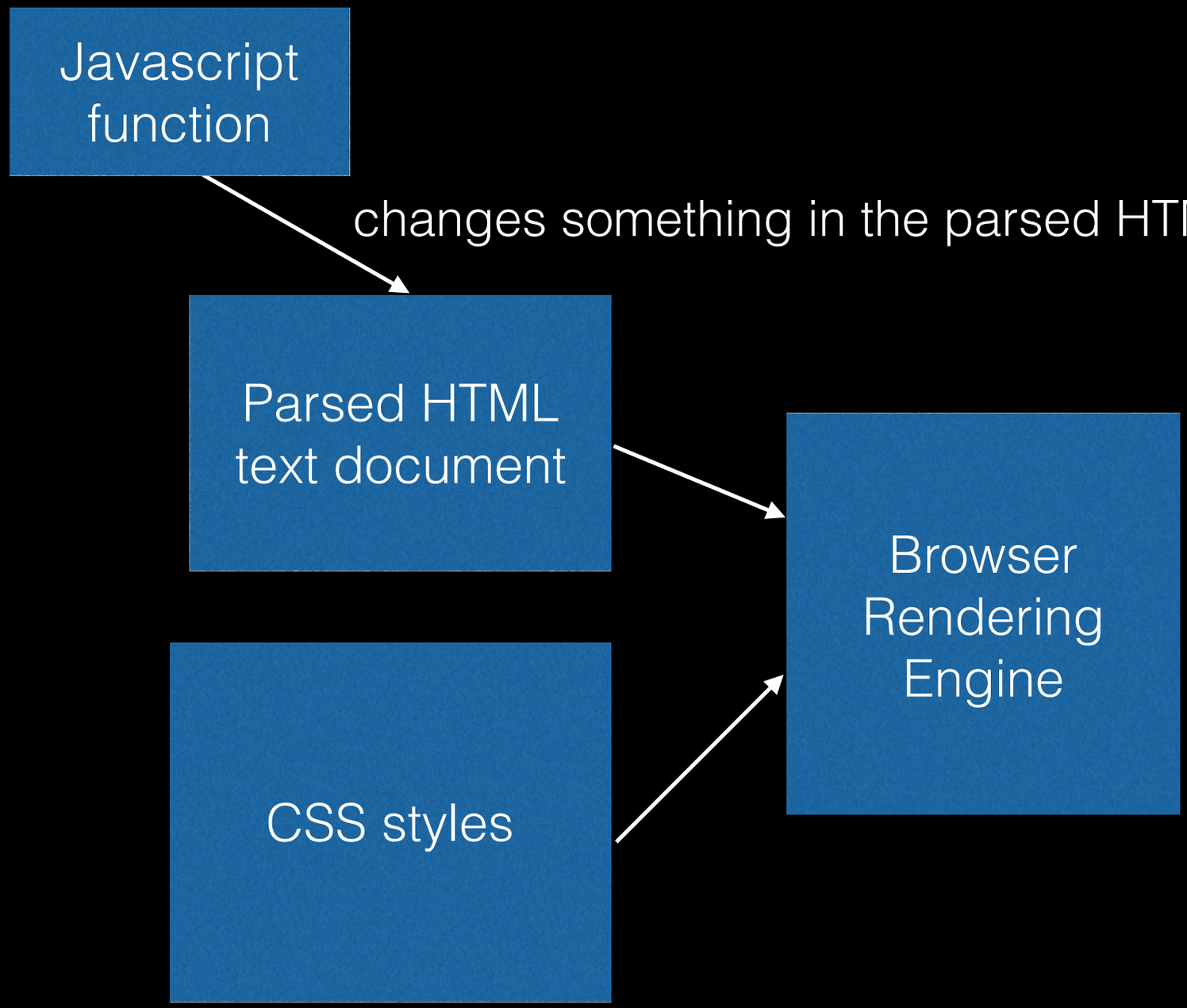
HTML Overview - Rendering



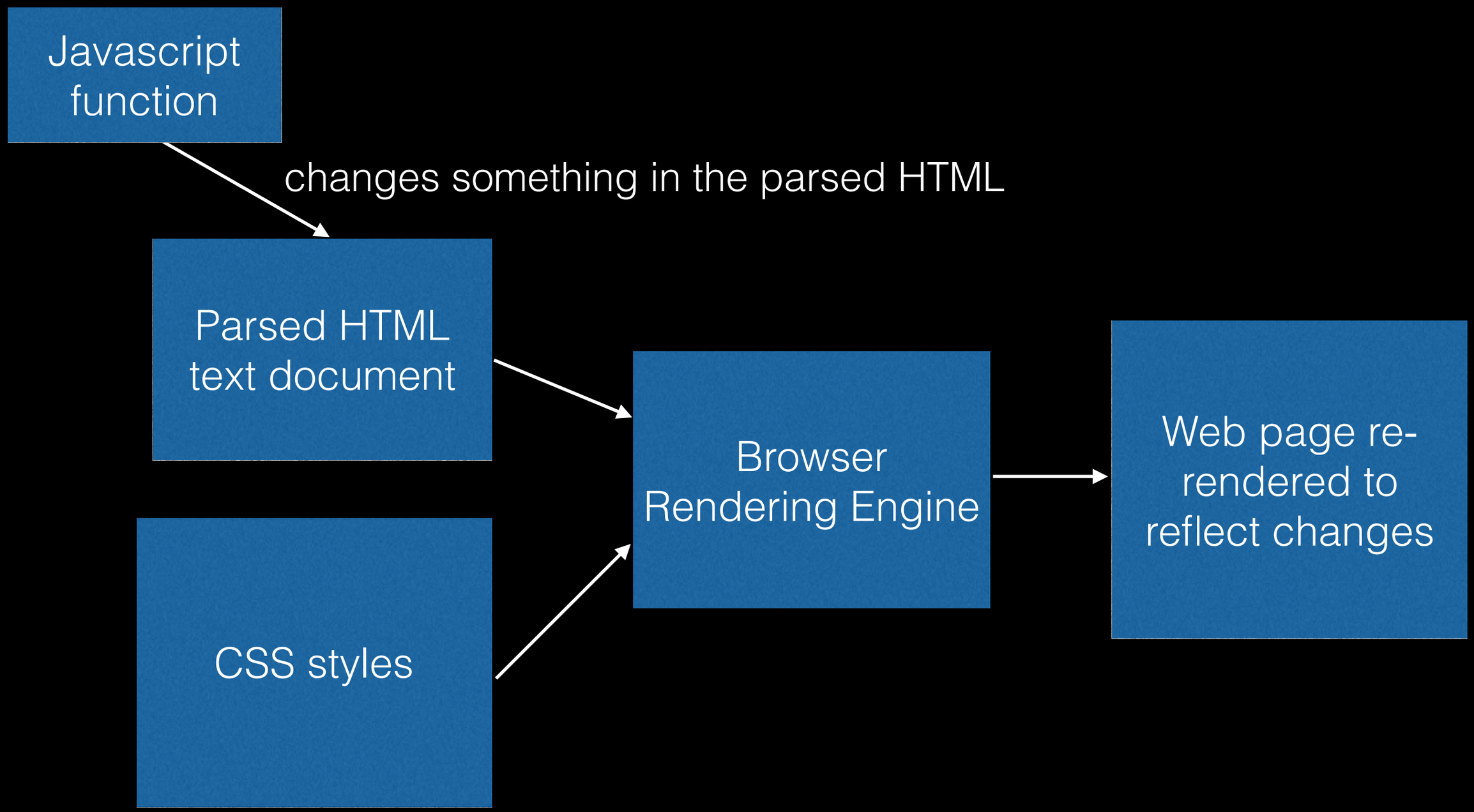
HTML Overview - Responding



HTML Overview - Modification



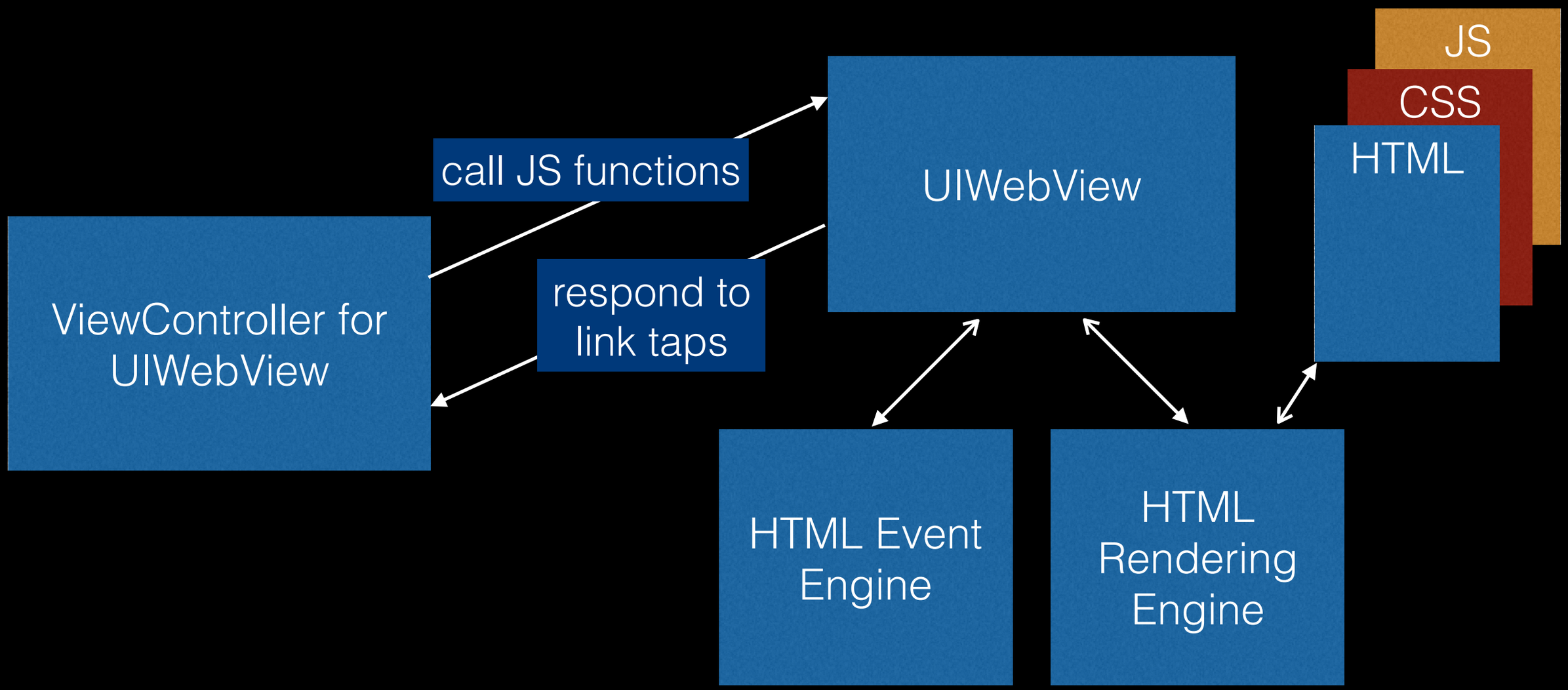
HTML Overview - Modification



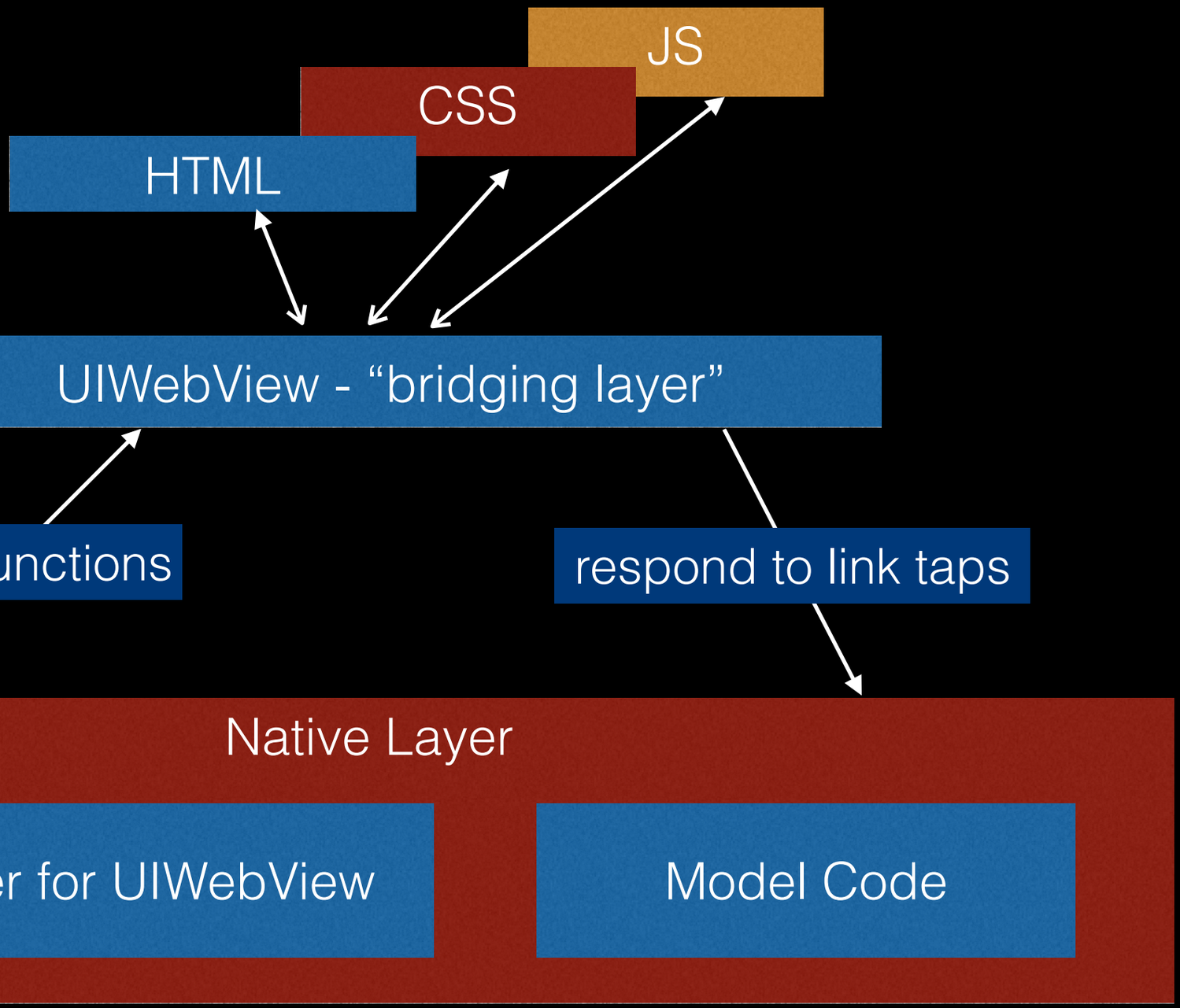
HTML on an iOS Device

- Web site displayed in Safari Mobile
- Web Clip - shortcut displayed on device home screen, tapping it launches Safari Mobile and loads the web site specified in the shortcut
- Hybrid app - native code shell that houses a **UIWebView** (embedded browser) that the HTML and Javascript runs in

UIWebView and the rest of the code



Schematically…



How the Native Layer talks to the Web Layer

How the Native Layer talks to the Web Layer

You *could* change the HTML and reload it each time you make a change. However, that means the same “watch while it builds the UI piece by piece" experience you get with a regular web page.

How the Native Layer talks to the Web Layer (cont.)

The advent of “Web 2.0” brought techniques for using Javascript functions (or methods on Javascript objects) to make individual changes in the displayable elements. Those changes are then applied and the page is re-rendered in place, *without reloading it*.

How the Native Layer talks to the Web Layer (cont.)

To do this in a hybrid app, you use the ***stringByEvaluatingJavaScriptFromString*** instance method on UIWebView.

- To use this method you have to
  - Have Javascript functions in the HTML you're communicating with
  - Construct the Javascript call as an NSString
  - Call [_webView stringByEvaluatingJavaScriptFromString:jsCodeStr];

How the Native Layer talks to the Web Layer (cont.)

In the updated sample app the call is made like this:

```
NSString *json = [currentWeather toJSON];
NSString *javascriptCall = [NSString stringWithFormat: @"setInitialValues(%@)", json];
[_webView stringByEvaluatingJavaScriptFromString:javascriptCall];
```

How the Web Layer talks to the native layer

How the Web Layer talks to the native layer

Each UIWebView has a *delegate* that you assign, either in IB or in code. The important method for that delegate here is shouldStartLoadWithRequest, which gets invoked when you tap on a link or in JS set *document.location*.

```
-(BOOL)webView:(UIWebView *)webView shouldStartLoadWithRequest:
(NSURLRequest *)request navigationType:
(UIWebViewNavigationType)navigationType
{
    // return YES if you want the web view to load the HTML in the request
    // return NO if you want to process the request in the native layer but leave the
existing web view content alone.
}
```

Things to Understand

Things to Understand:

- UIWebView interfaces between the native layer and the HTML layer.
- UIWebView loads HTML from a string or a URL and the loading looks just like a browser loading a web page.
- UIWebView instances function as individual browsers, so HTML/JS applets running in them can’t talk directly to each other.

Things to Understand (cont.):

- UIWebView isn't hardware accelerated so Javascript execution is going to be much slower in a hybrid app.
- Each time anything changes the size of any displayable element in a web page, the browser has to recalculate the layout of the entire page. *This takes time*, and you should keep it under your control as much as possible.

Things to Understand (cont.):

- UIWebView instances can cache images and data. You have to manage that caching yourself to make sure that there is enough room so that your hybrid app doesn’t hit the built-in size limit.

- The more Javascript your app runs and the more HTML (actually Document Object Model or DOM) changes it makes in the same UIWebView, the slower your app will run.

Things to Understand (cont.):

- Although you *can* migrate AJAX code and have it run inside of a UIWebView to do asynchronous network I/O, it's better to do your network I/O natively and crunch the data first, then feed it to the HTML layer.

Things to Understand (cont.):

- Hybrid apps have to be able to work offline, which means you will need to be able to load and display the HTML without an active network connection.

- Some developers solve this by packaging canned HTML/Javascript/CSS and shipping it with the app.

- Other developers solve this by caching the HTML and associated assets as they are downloaded so that they can be retrieved later.

What to do Where

Reacting to NSNotifications

Delegates and notifications are two of the main ways that parts of a Cocoa app communicate. If the HTML layer needs to be notified of something by part of the native layer, that has to be proxies by the *stringByEvaluatingJavaScriptFromString* method.

- Notification handlers go in the *native* layer
- The handlers call code that assembles data and call *stringByEvaluatingJavaScriptFromString*
- There has to be a Javascript function (or method) in the HTML to receive and process the notification being proxies into it.

Displaying New Web Content

If your app fetches new content in the native layer, you can cache it and change the parts of the HTML that reference the displayable element's CSS properties, any downloaded and cached image assets, or any element content.

- Structured this way, you're fetching data and images from an API rather than loading a standard web page.

- CSS properties and element content can be changed directly using Javascript.

- Images need to be cached in the file system and paths generated for them, and those *paths* get assigned via Javascript to the relevant <img> element, after which the new image will be loaded.

Letting the HTML Layer Request Data from the Native Layer

Using the *shouldStartLoadWithRequest* delegate method and *stringByEvaluatingJavaScriptFromString*: UIWebView method call you can have the UI request data from the native layer.

- Using a structured fake URI like @“data:page=1” and the *shouldStartLoadWithRequest*, your code can parse the URI and use it to fetch whatever data corresponds to “page=1”.

- Your code can then dispatch_async() a block containing the raw fetched data and in that block construct a JSON string to send to the HTML layer via the *stringByEvaluatingJavaScriptFromString:* method

- If you need data for page=2, then you can repeat the first step with a structure URI like @“data:page=2” and then dispatch another block.

General Do's and Don'ts

- Do keep the HTML small and in its proper place.
- Don't expect the same user experience you get with native code.
- Do understand how the native and HTML layers communicate with each other.
- Don't buy into "write once, run anywhere". It's a fantasy if you care about your UX.

Resources

- https://www.cocoacontrols.com/posts/a-primer-on-hybrid-apps-for-ios
- http://www.kinvey.com/blog/3421/how-to-spot-a-hybrid-app-on-ios
- http://stackoverflow.com/questions/19937424/hybrid-ios-web-and-native-app-events-communication
- http://www.nngroup.com/articles/mobile-native-apps/
- http://www.amazon.com/Developing-Hybrid-Applications-iPhone-JavaScript/dp/0321604164