TechKnights Angular Workshop

Presented by Doug Woodrow

http://bit.ly/2BfWxHX

To follow along, clone this repo! Slides and all code examples are included

Quick intro

- My name is Doug Woodrow
- Developer for 13 years
- Started with Visual Basic, C++, Java
- Then got into the web and learned Wordpress
- Became a graphic designer and got my foot in the door a lot of places
- Was in college (CS) at the same time, growing my knowledge base
- Then quit that (graphic design *sucks* as a job) and went to development, got a job downtown developing PHP apps
- Developed a TON in between then
- Now develop big data tools in Java, DAPPS, AI, Logistics platforms for a company I co-own, Knightspeed Moving, & everything else under the sun

Why you would use Angular?

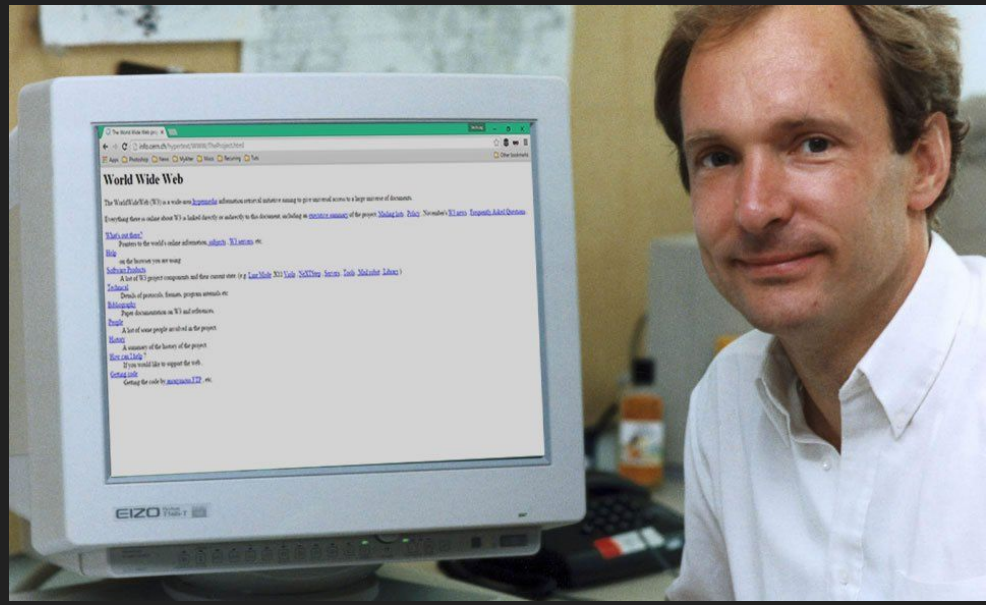
- You’re a developer looking for an entry point to the web
- You’re a big company looking for a framework with opinions
- You’re an established developer trying to create modern, progressive web applications

The web has changed a lot over time

In order to show why Angular is so great, let’s take it back a few steps to the days of the good ole web...

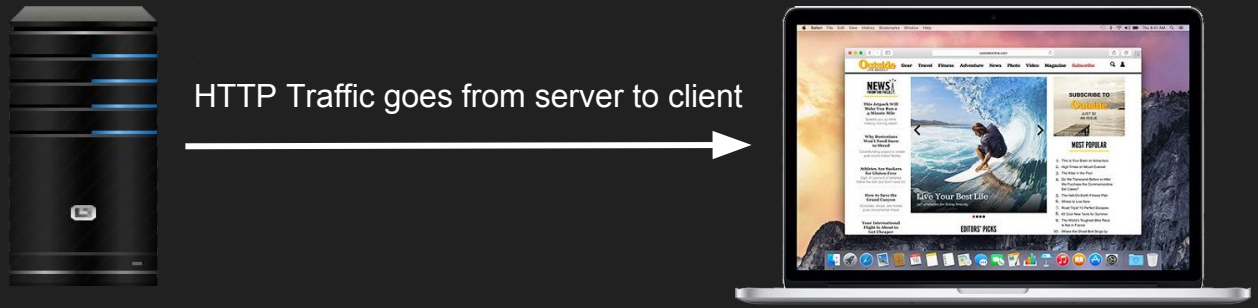
Early days of the web

- Tim Berners Lee creates a language that makes creating web pages easily
- These web pages use basic stateless transactions and send basic plain data, interpreted by the user’s browser
- Speeds around 56kbps



Sir Tim Berners Lee

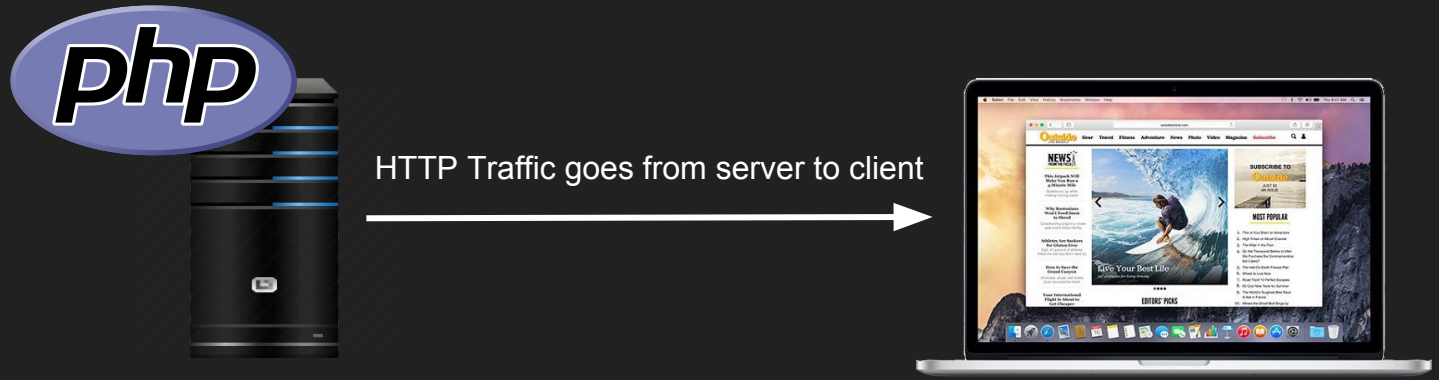
The Lifecycle of a web page circa ‘93



Majestic Monolith - pt. 1

- The need for logic in web page design drives programming language PHP, Perl to become popular.
- PHP is ugly (and in many ways...*still is*) but it’s easy syntax and predictable server/browser behavior excite users to create websites that have now have servers that don’t just simply serve data, but also perform logic.

The Lifecycle of a web page circa ‘99

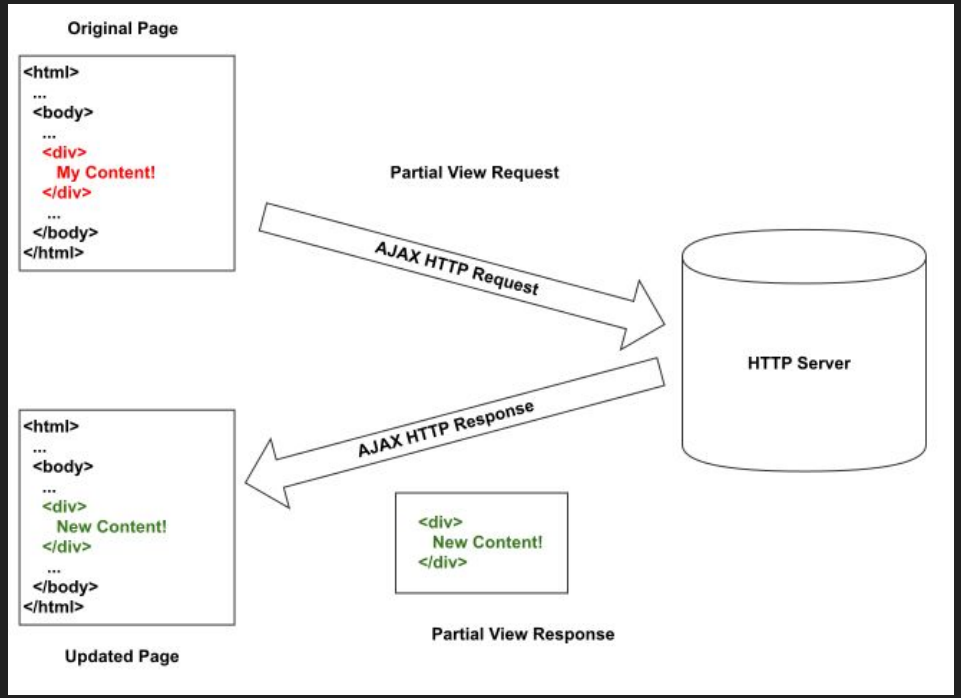


Majestic Monolith - pt. 2

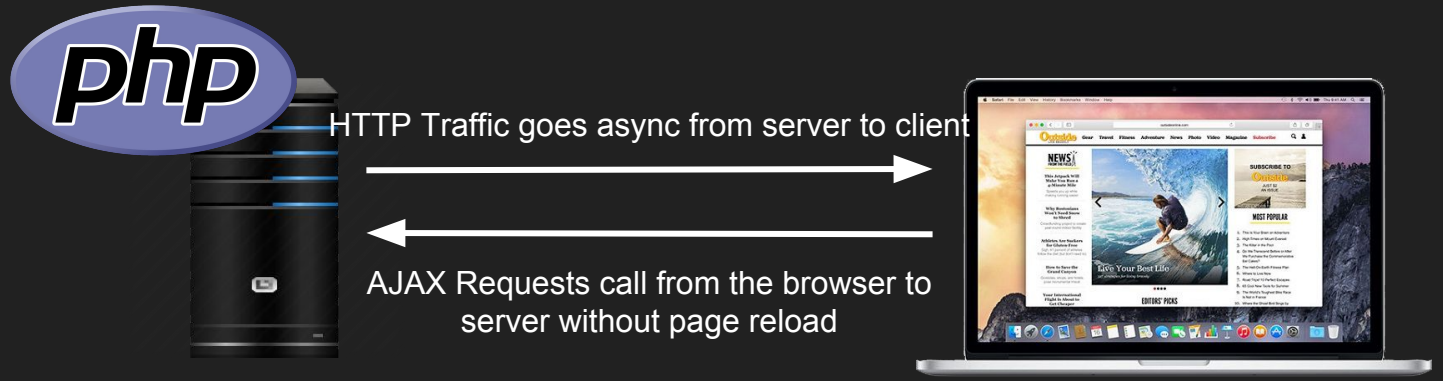
- AJAX calls develop as a primary way to statelessly communicate without reloading the web page
- Iframe shipped in ‘96, Outlook Web App Team develops XMLHttpRequest, in March ‘99 it ships in IE 5
- Webapps need more application power in the backend and OOP in PHP 5 opens a whole new world of applications

How AJAX Works & why it's important

AJAX - Asynchronous JavaScript And XML



The Lifecycle of a web page circa ‘99



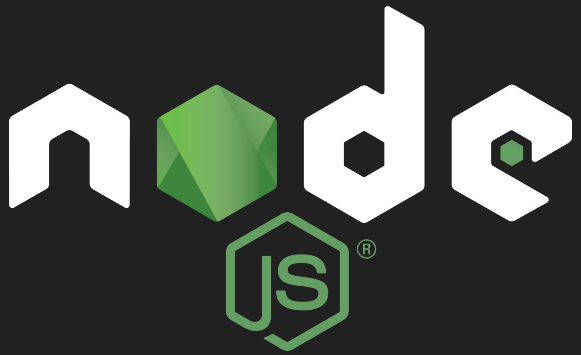
Thinking off the Rails

- 2004 Ruby on Rails brings new fundamental ideas to the table about how to build, structure, and maintain webapps, Symfony in ‘05
- Web apps like Gmail start to show up, using AJAX, pushes need for modern web development
- Enterprise features become readily available
  - Stable, predictable ORM (ActiveRecord)
  - DRY principles
  - Strong use of MVC
- Ruby on Rails fundamentally brings into question how we build web applications

David Heinemeier Hansson

The browser becomes a big player

- **2008** Google makes the V8 Engine open-source, a C++ implementation of the Javascript interpreter in the browser
- **2009** sees Node.js first release, Ryan Dahl creates a server-side version of the V8 JS Interpreter that is able to run Javascript server-side
- Suddenly, we realize that the browser can now efficiently process JS code, a big step forward for modern web apps



Ryan Dahl

Now, finally, all the pieces are in place

- AJAX is the preferred method of get data to and from the browser beyond the initial page load
- The browser is powerful now & Javascript becomes a relatively “fast language” with V8
- The community around Node is quickly growing & Developers feel comfortable with JS’ un-strict syntax
- Now it’s finally time...

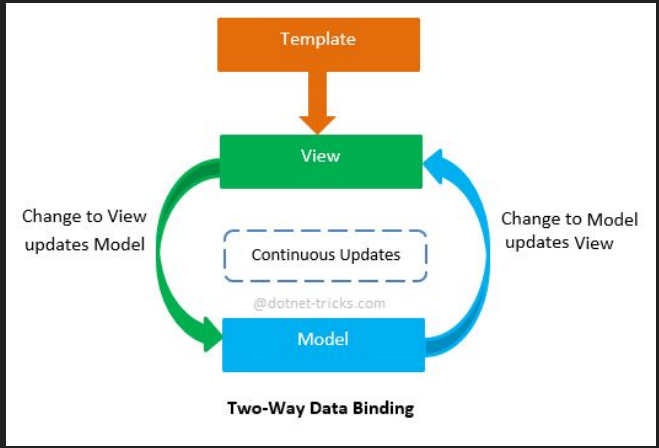
AngularJS, radically re-thinking the web

- In October 2010 AngularJS is launched by Misko Hevery and Adam Abrons, AngularJS is one of the first frameworks that questions how a frontend web framework works, 6 months after creating bets manager can rewrite in 2 weeks...actually takes 3 weeks (from 17,000 loc -> 1,500)
- Bootstraps entire application in the browser from Javascript runtime, not just plain HTML or server-side rendered pages
- Uses two-way binding to provide programmers the ability to update the view and changes are propagated to the controller and vice versa, but this is a big issue...

Angular Bootstraps entire application in the browser from Javascript runtime

AngularJS forces a web revolution

- AngularJS shows us that the browser can do a lot more than expected
- Community + AngularJS realizes that the first iteration of how to do that was a bit naive
- Alongside the competition from React, Angular realizes it needs a rewrite
- Many issues arise from two-way data binding and overly heavy re-renders on DOM change



Angular beyond AngularJS

- Angular 2 brings the rewrite, using a virtual DOM to efficiently re-render pages on data change
- Finally Angular is ready for the big time, maintained by Google and backed by a majority of the developers, some have bad taste due to the major architectural decisions made from ng1 to future Angular versions

Angular Core Concepts

Angular Basics

- Written in **TypeScript**
- **MVC**-based
- **Component-based** architecture

Components

Components

- A Component is a Javascript module that defines its own behavior and does not have knowledge of other components in the system
- A Component moves through a life cycle, the Component is able to implement to interfaces defined by Angular to bind custom logic to these hook points

Component Code Example

Services

Services

- A Service is a type of a component. Services typically manage the persistence of data to external sources.
- Services are injectable, meaning at runtime, dependencies on that service are satiated via the Angular ecosystem.

Service Code Example

RxJs

RxJs

- Rx (or Reactive) programming is a new programming model that manages data as streams. Reactive has libraries every major lang.
- RxJs is a core concept now in the Angular environment.
- Similar to functional language concepts in Haskell, whereby you are basically just modifying data as it passes through your app.
- RxJs allows you to declaratively determine modifications to your data stream, making the Angular/Rx Ecosystem very pleasant, avoiding callback hell, and most importantly, maintaining the application as an asynchronous environment.

RxJs Code Example

Tons of awesome stream modifications

- You can run ops on your stream like concat, forkJoin Observables, maps, flatMaps to create Observables from items in an Observable stream and then emit those resulting items to a flat array...as you can see Rx is pretty badass. (imagine a metrics aggregator service!)
- Possibilities are endless, **Rx is a way of thinking, not just a framework.**

MVC

Is Angular an MVC framework?

- For the most part, yes!
- If you are familiar with MVC, *Controllers* in Angular are implemented with Components.
- The *View* is defined in a HTML template with Javascript expressions that may be evaluated on runtime.
- The *Model* is defined as an interface and a concrete implementation of that interface...which I will show in the example.

Let’s write some code!

Questions?

Wanna follow me? (oooh that sounds weird doesn't it?)

- The website guy doesn't have a website...yet, it's in progress ;)
- Would love for anyone seriously interested in exclusive programming to join my FB group, **The Programmer Accelerator** where we talk about all kinds of tech & programing. Add me on FB and I will add you to the group, or request access!
- React Workshop two weeks from today (Feb 22), same app but in React instead, more in-depth conversation about the web.

http://bit.ly/2BiRAy3

The Programmer Accelerator FB Group