TechKnights React Workshop

Presented by Doug Woodrow

http://bit.ly/2EONIZO

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 React?

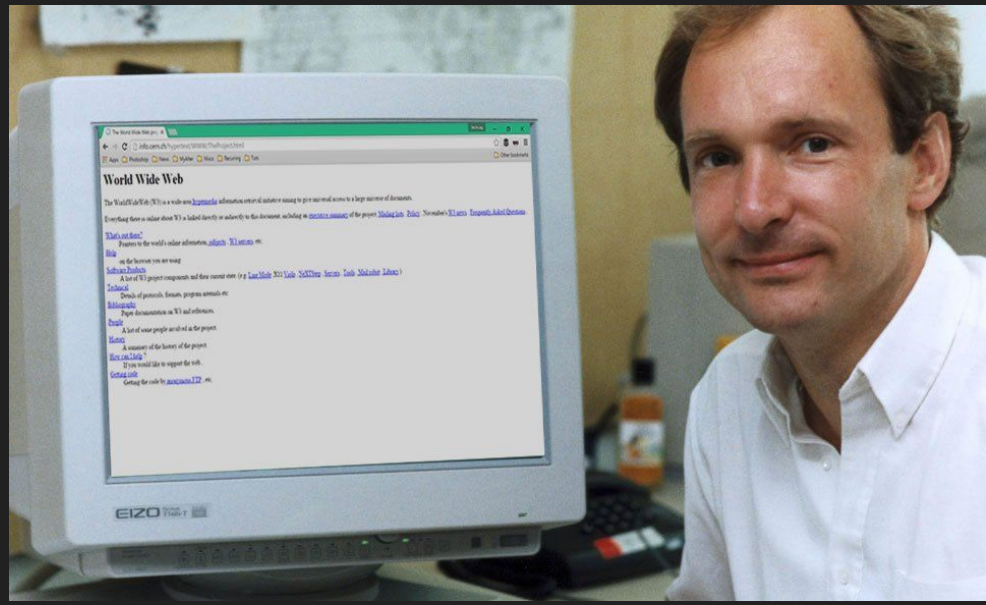
- You’re an experienced developer looking for a multi-functional framework to build experiences in
- You’re likely not a big company
- You’re an established developer trying to create modern, progressive web applications

The web has changed a lot over time

A quick refresher for those that weren’t here last week

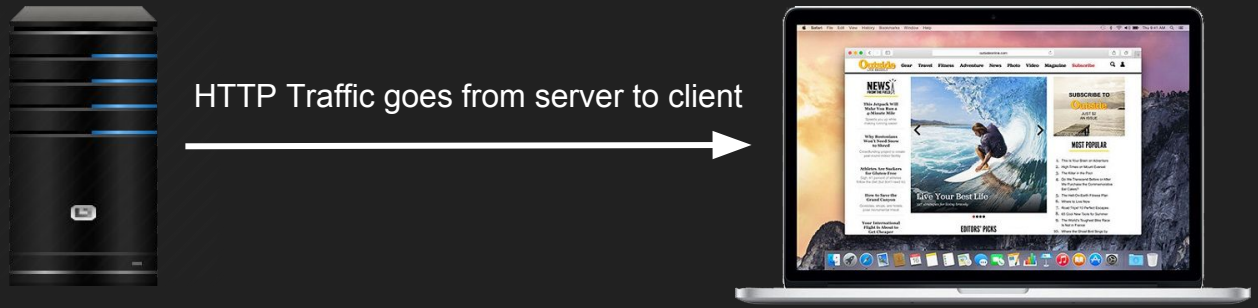
Early days of the web

- 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

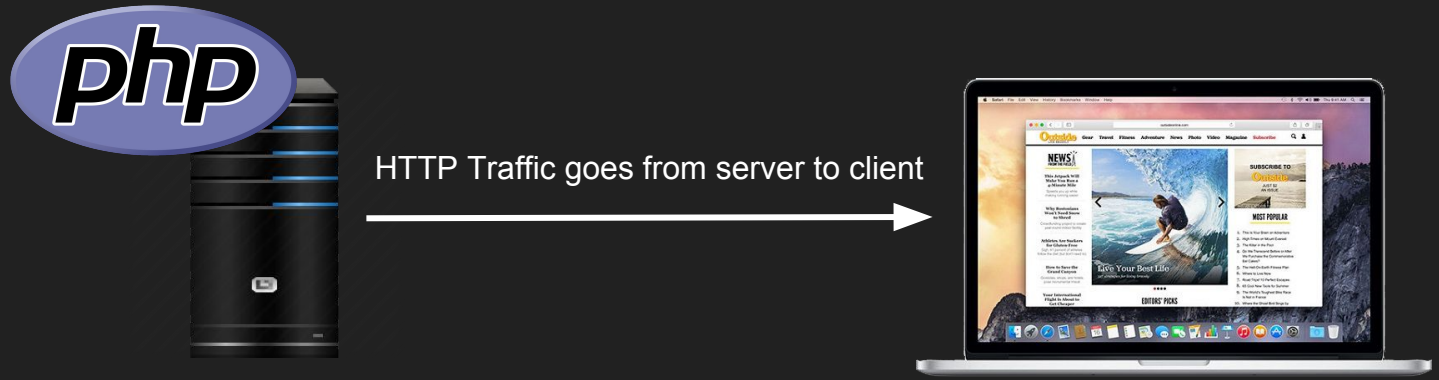
The Lifecycle of a web page circa ‘93



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

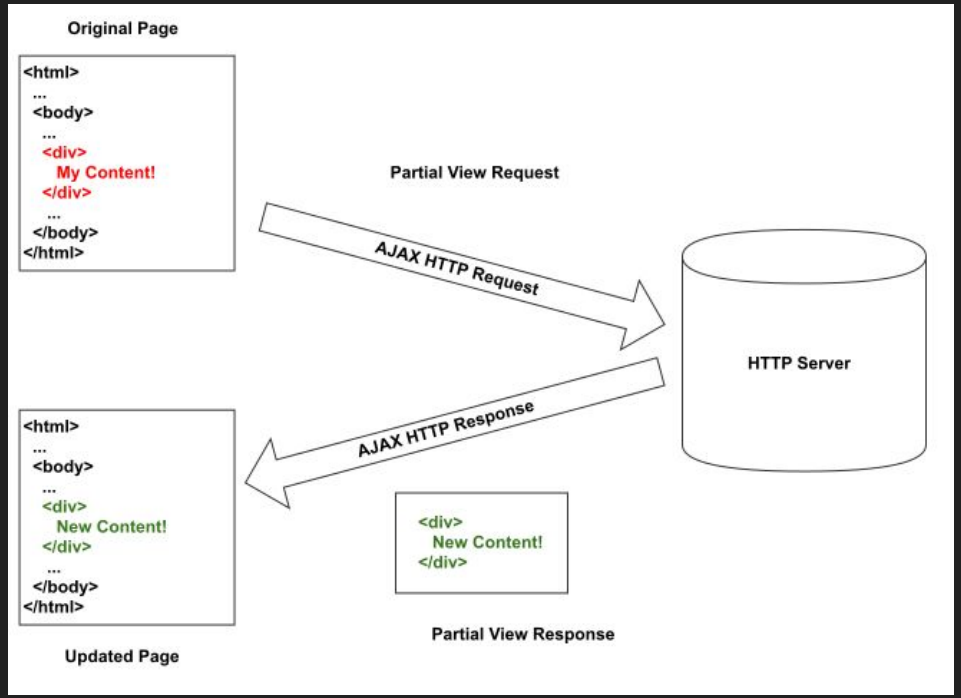


Majestic Monolith - pt. 2

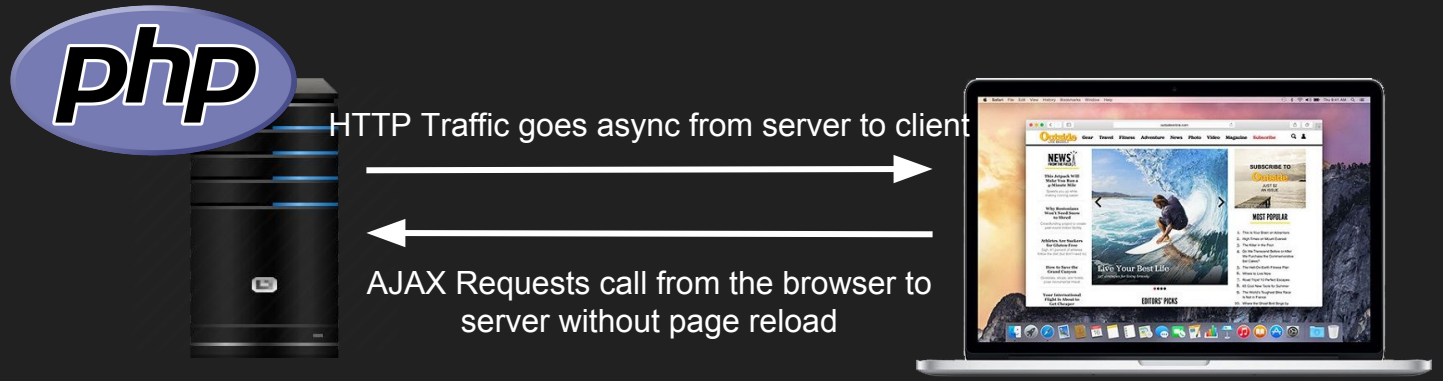
- AJAX calls develop as a primary way to statelessly communicate without reloading the web page
- Iframe shipped in ‘96, Outlook Web App Team develops XMLHttpRequest, in March ‘99 it ships in IE 5
- Webapps need more application power in the backend and OOP in PHP 5 opens a whole new world of applications

How AJAX Works & why it's important

AJAX - Asynchronous JavaScript And XML



The Lifecycle of a web page circa ‘99



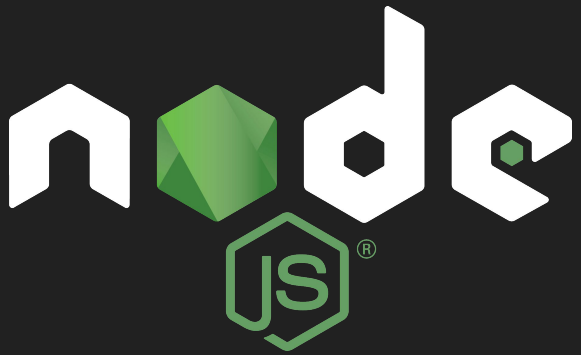
Thinking off the Rails

- 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



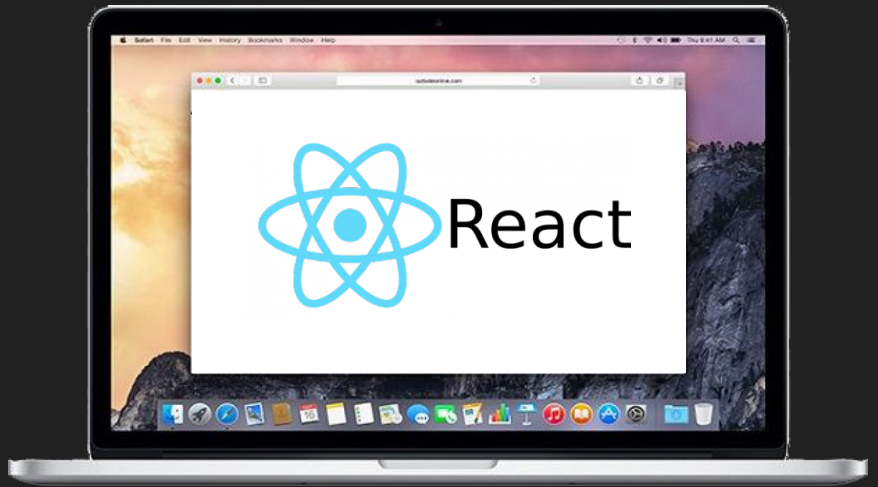
Ryan Dahl

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
- The web can finally develop into JS frameworks

ReactJS, a history of component-based web

In March 2013 (3 years after initial AngularJS release) React is launched by Jordan Walke, as a port of XHP for JS to bring the PHP features of Facebook to a Javascript context



React Bootstraps entire application in the browser using web components

React Core Concepts

React Basics

- Written in **JSX, Flow, Harmony, ES6**, any flavor of JS you want
- Bring your own beer/opinions model
- One-way flow inspires Angular 2+

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 React to bind custom logic to these hook points

One-way data flow

One-way data flow

- A component’s data is a strict one-way flow, defined by props and state, owned by the component
- This reduces a lot of double-checks that need to take place, speeds up the DOM **A LOT**
- React’s implementation of flow inspired Angular 2+

JSX

JSX

- JSX combines both HTML and JS into one consolidated language, allowing programmers to use JS function/expressions inside of HTML
- Makes it easy to combine logic and templating into one consolidated area

Virtual DOM

Virtual DOM

- In-memory DOM (Document Object Model) represents changes, the difference between the two is taken, set of changes are propagated to components, DOM doesn’t need to do full reload.
- Much faster and more efficient than non-virtual (AngularJS) or conventional re-render of entire page/DOM

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!

http://bit.ly/2BiRAy3

The Programmer Accelerator FB Group