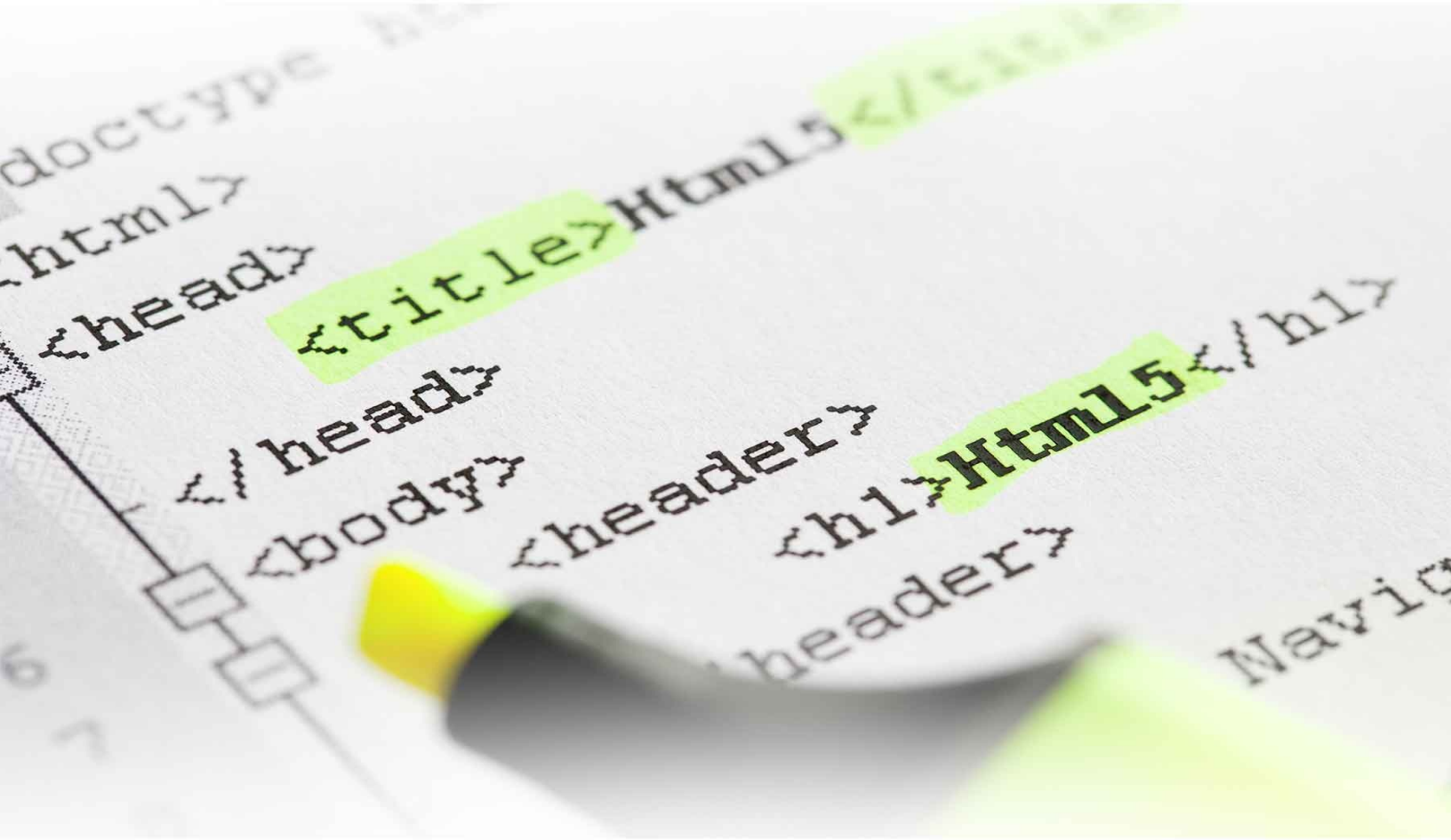
DEVELOPMENT OF WEB WITH REACT



MICHAEL REDCAR

TABLE OF CONTENT

INTRODUCTION

THE REACT

STEPS FOR LEARNING REACT APPLICATION DEVELOPMENT

INCONCLUSION

INTRODUCTION

What is a Web-Developer?

Fundamentally, web-developers help you build your website; again, they deal with the unseen coding aspect of a website, known as the back-end, and are primarily concerned with making a website function quickly, efficiently, and with the greatest stability. Moreover, many web-developers oversee and direct the construction of a website from its birth - again, it is up to them to build the backbone of the website, and ensure it functions properly.Although this process is not actually visible to the visitors of the website, it is absolutely vital to the visitor's experience while browsing the website - for example, a web-developer can optimize and dramatically reduce loading times, so that a visitor can access anything q uickly and without frustration. Typically, a web-developer will be responsible for developing and coding server side applications, databases, complex e-commerce transactions, and for search engine optimization - naturally, a web-developer must make certain that these are done in an accessible and sensible fashion, while being efficiently coded and highly compatible.

Developing Web Apps: Getting Started

Web apps do not differ much from traditional web applications, except for three details:

Can be installed on the local device (mobile, tablet, desktop)

Can be performed offline

Have access to APIs and manipulate local device capabilities like camera, accelerometer, GPS, etc.

In addition, web apps are written with open and responsive technologies that are based on the triad HTML5, CSS3 and JavaScript. Thus it is possible, for example, run a web application on an Android device like a native application with access to resources that the machine offers!

THE REACT

React (also known as React.js or ReactJs) is a powerful JavaScript library that uses server-side rendering (SSR) with a uni q ue twist—one that allows it to provide a flexible, performance-oriented, componentized solution for the “V” in MVC (Model View Controller). If you’re looking for instant page loads, or the ability to handle long lists of dynamic content changing within the view, React may be the JavaScript library for you.

THE REACTIVE REVOLUTION

Harken back to a simpler time, when the World Wide Web was young—PHP ruled the back-end, and the client-side was little more than a window to content stored in static HTML pages housed on humming servers. The web was nowhere near as dynamic or as interactive as it is today, but the path between server request and an immediately viewable webpage was simple, direct, and even theoretically faster (if it weren’t for the obvious technological limitations of the time), thanks to SSR.

Fast forward to the present, and the client-side has since experienced a renaissance—dynamic front-ends, interactive content, and sleek desktop-like user experiences have become the norm. But all of this advancement did not come without a cost: added complexity. Bloated client-side implementations led to loading screens, blank white pages and other lags in performance while waiting for the browser to download the JavaScript needed to render a page, something that was once a simple output from the server. The inherent limitations of client-side rendering (CSR) were starting to show at the seams.

Enter React, an isomorphic JavaScript library released by Facebook in 2013, that allowed developers to use SSR to build modern web apps. One of the reasons developers had started to drift away from SSR was that it was necessary to have to load the entire web page every time something changed within the view. React

introduced a workaround called the Virtual DOM that allowed developers to take advantage of the inherent performance advantage of SSR, without having to update the entire view every time a user made a small change to the UI.

SERVER-SIDE RENDERING (SSR)

The beautiful thing about SSR is that the app is able to pre-render the initial state of a view before showing it to the user. When a user loads a webpage, they get to see it as the designers and developers intended, without having to wait for their browser to download the JavaScript necessary to render the page. SSR is also great for search engine optimization (SEO), since most search engines have an easier time crawling and understanding pre-rendered HTML views.

THE VIRTUAL DOM

The Virtual DOM works by modeling two copies of the DOM, the original and an updated version that reflects changes received from the view. React takes note of the differences and outputs the DOM operations necessary to only update the parts of the UI that actually changed. In this way, React overcomes a previous shortcoming of SSR, where it was necessary to recreate the entire updated view.

JSX FILES

React goes against the traditional grain of separating form (HTML) and function (JavaScript), instead opting to prioritize encapsulation by mixing JavaScript and HTML into a single JSX file. This makes sense considering the component-oriented future of the web—the convenience of having all the JavaScript and HTML needed to define a single UI element like a slider neatly packaged into a reusable component is worth the switch.

REACT NATIVE

React Native brings the power of React to mobile app development. Unlike other JavaScript mobile development frameworks like Cordova or Ionic, React Native allows you to write UI components in JavaScript that compile to native code for

iOS and Android—that means you can create mobile apps that perform and function as if you built them in native Java (for Android) and Objective-C or Swift (iOS). Better still, because React treats the view layer as a pure output of state, switching from web to mobile is often as easy as swapping React tags for React Native tags, making it possible to build projects where you only had to write once for iOS, Android, and the web.

FLUX

Earlier, we called React the “V” in MVC, mostly to help newcomers familiar with front-end web development concepts to understand how they might integrate this awesome library into their web projects. But if your goal is to embrace the “componentized” future of web development, Facebook develops all its apps using an application architecture called Flux that’s better suited for React. The Flux programming pattern places an emphasis on unidirectional data flow and consists of three parts: the dispatcher, the stores, and the views.

• Stores are similar to the models in MVC, except they manage the application state for a particular domain within the application.

• The Dispatcher is a simple single registry of callbacks to the stores within the application. It also manages the dependencies between stores.

• Views are the same as the view in MVC, except in the context of React and Flux, also include Controller-Views, which listen for change events and retrieve application state from stores as re q uired.

Basically, all data in the application flows through the dispatcher, which acts as a central hub. This data is tracked as Actions, which are provided to the dispatcher in an action creator method, often as a result of a user interacting with the view. The dispatcher invokes the registered callback, effectively dispatching the action to all stores that have registered with that callback. The stores in turn relay that change event to the controller-views to alert them of the change. The controller-views listen for events, retrieve data from the appropriate stores as required and re-render themselves and all their children in the component tree accordingly.

If you’re interested in learning more about Flux, check out this article from Facebook itself. You may also want to checkout Redux, a library that lets you code in Flux, but simplifies things by providing a single store.

SHOULD YOU USE REACT FOR YOUR PROJECT?

React is a powerful UI library that brings the power of SSR, isomorphic JavaScript, and component-based web development to modern dynamic web applications. It really shines when you need to render large lists of dynamic, data-heavy, content within a single view a la Facebook or Instagram.

Consider using React if…

• You're already a fan of ClojureScript and the Om Project.

• You're looking for a performance boost for the V in MVC for your app.

• You like the concept of Flux and unidirectional data flow.

• You don't mind learning a new technology that's already become a major part of front-end web development.

• You embrace the componentized future of web development.

STEPS FOR LEARNING REACT APPLICATION DEVELOPMENT

Getting going with React can sometimes be overwhelming. Or, at least, this is what people are saying. The consensus seems to be that the difficulty in getting going is mostly due to the following realities or obstacles.

1. The information available is not written for average developers.

2. Massive updates to the JavaScript language in 2015, and the potential future updates being actively used today, have muddied the water. i.e. learning ES6 & ES* and React at the same time can involve a mountain of change when coming from ES3 and script includes.

3. React is actually a rather small abstraction that typically is present as a cog in a system of many other complex moving parts. Grokking the parts and React at the same time is a complicated affair. I find that most developers agree:

Redux, react, flux, es6/7, webpack, jspm, tackle them one by one and everything should be okay, but combine all those at once and learning how to do apps with Redux suddenly becomes much harder. – Cedric Dugas from Thoughts about React, Redux & javascript in 2016

In this article, I’m going to prescribe a set of thoughtfully curated steps for learning React that I hope might bring some relief to these three difficulties.

That was the good news. The bad news is that productive learning paths, taught by someone other than you, are rarely free. Learning React from others will cost you some cash. If you follow the steps below it will cost you around $73.00 USD (or around $30.00 if you use 10 day trials). This amount will get you one month of access to several sites providing React learning materials (i.e. $29 a month for Pluralsight or 10 day free trial + $29 a month for Evented Mind + $15 a month for tutsplus.com or 10 day free trial).

Prerequisites

Let's discuss some prere q uisites.Steps one and two will re q uire that you have some HTML and JavaScript chops. These steps focus on learning React without learning how to build an actual React app.

When you reach step three you'll need a good understanding of Node, npm, ES6 (including ECMAScript modules and either Webpack, systemJS or Browserify), and Babel. If you lack a basic understanding of each of these, I would suggest you brush up on the basics before moving to step three. If you spend the money on the learning resources I mentioned above, catching up shouldn't be too difficult, given most of these resources also provide Node, npm, Webpack, SystemJS, ES6, and Babel screencasts.

Now, before you go complaining and crying fatigue realize these technologies are the minimal barrier to entry with most of the competing solutions (i.e. Angular 2, Ember, and Aurelia). If you think learning these extra parts isn't worth your time, I'd like to suggest you back up from React and have a look at vue.js, riot.js, or mithril (i.e. download one(ish) script file, include it in HTML page, and go!). After all you might not need React, or need to learn it. And the last thing you want to do is use React when something simpler will suffice.

Seeding React Ideas

I'd like to plant some broad React ideas (in no particular order) into your head before you get dizzy and overwhelmed. This might all be gibberish right now but reading them will help you connect some dots in the future.

1. React components are virtual (i.e. written in JS) until they are rendered (i.e. render()) to the virtual DOM which, in turn, updates the actual DOM with the

fewest changes re q uired. The idea is that implicit DOM interactions (i.e. jQuery DOM interactions) are avoided and given over to an abstracted DOM (i.e. the virutal DOM).

2. React delivers on the merits and value of web components today! (i.e. structuring your app into a tree of single purpose elements/components minus web component standards.)

3. React doesn't want you to write HTML. It wants you to write JSX inside of JavaScript files. It even wants your styles to be written in JS. The React way starts with JS and ends with JS. It's JS all the way down. It doesn't have to be, but that is what is presented in the React docs.

4. React doesn't use a templating engine, it uses JSX. Think of JSX as a magical, string-less type of HTML written in JavaScript requiring a compiling step. Yes, JSX is optional, but the alternative is writing painfully verbose functions. JSX is easier to read and write in a JavaScript file. Just remember that it must be transformed into JavaScript functions before running in the browser.

5. The momentum behind React doesn't lie in its focus on the UI or the virtual DOM. The value of React is found in the fact that it provides the tools needed to create a tree (i.e. composition) of components where data flows in a sane manner (i.e. one-way reactive data flow). That's the secret sauce!

6. A React element is equal to an HTML element. A React component can minimally be a single React/HTML element but when the term component is used, you should think of it as a grouping of React elements that make up a distinct region of your UI. For example, a login component. A login component is not one element. It is several elements that all make up one login component.

7. React "Props" are likened to HTML properties. They are used to declaratively pass configuration to components and child components. Props shouldn't be mutated, that is what "state" is for.

8. React "state" is the mechanism to deal with a components uni q ue data changes over time. If you like a photo (or unlike), then that photo needs to know that you have liked it (or unliked it) and contain that state.

9. You'll be really bewildered by what looks like HTML elements sprinkled through your JavaScript files. Don't forget, these HTML/React elements get transformed into JavaScript so a browser can parse the React code.

JSX goes from this:

```
var HelloMessage = React.createClass({
  render: function() {
    return <div>Hello {this.props.name}</div>;
  }
});

ReactDOM.render(<HelloMessage name="John" />, mountNode);
```

To this (note the createElemen t function replaces JSX):

```
"use strict";

var HelloMessage = React.createClass({
  displayName: "HelloMessage",

  render: function render() {
    return React.createElement(
      "div",
      null,
      "Hello ",
      this.props.name
    );
  }
});

ReactDOM.render(React.createElement(HelloMessage, { name: "John" }), mountNode);
```

10. While it's possible to have dumb or stateless components, the focus of React is on building stateful components containing stateless child components relying on the top-most parent for state (i.e. data). Thus, React is more than the V in MVC. It is also the C in most cases. The top most component is often refereed to as a controller component because this is where the state is managed. Consider the explanation from the React docs:

A common pattern is to create several stateless components that just render data, and have a stateful component above them in the hierarchy that passes its state to its children via props. The stateful component encapsulates all of the interaction logic, while the stateless components take care of rendering data in a declarative way.

As you learn, return to these ideas to see if what I am saying becomes more meaningful.

Step 1: Understand why Facebook built React

This is the most important step that you'll be tempted to skip. Don't! Before

going off and trying to understand what exactly React is, first you should strive to understand why it is.

Step 1.a – Watch: JS Apps at Facebook or Introduction to React.js

Step 1.b – Read: Why did we build React?

Step 1.c – Watch: Pete Hunt: React: Rethinking best practices

Step 2: Get the React Gist, Before Getting To The API

This step will lay the foundational mental model for React. Read this material knowing that the next step will fill in the missing details.

Step 2.a – Read (but skip if too basic): React.js Introduction For People Who Know Just Enough jQuery To Get By

Step 2.b – Read: ReactJS For Stupid People

Step 2.c – Read: The React Quick Start Guide

Step 2.e – If you are feeling brave, read: Removing User Interface Complexity, or Why React is Awesome

Step 3: Learn React Fundamentals i.e. How to Use The API

It’s time to stretch those fingers and play around with some code. In this step, you should strive to gain the fundamental concepts through real code comprehension (i.e. learn to write React code and know what it is doing).

This is the step where everything will break down if you don’t have knowledge about Node, npm, ES6(including ECMAScript modules and either webpack, systemJS, or Browserify), and Babel.

Step 3.a – Read: Learn Raw React — no JSX, no Flux, no ES6, no Webpack… then consider reading part two and part three as well.

Step 3.b – Watch: Getting Started With React.

Step 3.c – Watch: React.js: Getting Started. This is a bit outdated, but worth an hour of your time. Watch and digest the concepts discussed.

Step 3.d – Watch: Understanding React. This is also a bit outdated, but worth an hour of your time. Watch and digest the concepts discussed. Then read the official React tutorial.

Step 4: Build An App

This is the step where you take all of your React knowledge and add in some additional players to make an actual application.

Step 4.a – Read: All the official mini guides starting from “Displaying Data”.

Step 4.b – Watch: Building a Wiki With React and Firebase

Step 4.c – Read: Flux For Stupid People

Step 4.d – Watch: Building Applications with React and Flux

Step 4.e – Watch: Build a Microblogging App With Flux and React

Step 4.f – Watch: Building a Real-time App with React, Flux, Webpack, and Firebase

Step 5: Go Build Something

A lot of people will skip forming a foundational amount of knowledge and just start build something for the sake of learning. I personally think this is a waste of time. After all, is it more efficient to sink, then learn not to drown or, learn to swim and then go swimming. If you followed the steps above, you’ve learned to swim and you should be ready to go and build something.

When building you need to be aware of the following resources:

• starter kits

• editor integrations

• react-devtools

• reactcheatsheet.com

INCONCLUSION

What is the magic behind it all?

Some mobile operating systems, such as Firefox OS, perform the web apps directly. Others like Android and iOS, require applications to be packaged in any form and compiled with specific APIs so they can be installed and run.

Anyway, web apps can not be written without discretion. They must follow certain patterns that involve, for example:

HTML5/CSS3 specific code design patterns that fit the various screen resolutions (responsive design);

Possible to be performed offline;

Support 'touch';

Standards and conventions dictated by each platform.

Below we examine some tools, libraries and frameworks that help in the development of web apps, among IDEs, libraries, frameworks, and other resources.

IDEs with support for HTML5

NetBeans is undoubtedly one of the best HTML editors to develop apps. It has extensive support for HTML5, CSS3, JavaScript and other languages. Furthermore, it comes with a built-in WEB server and many facilities for debugging and automatic integration with various JavaScript libraries.

Sublime Text: this editor has one of the best implementations of the feature "Auto Completion" I've ever seen. Also has a new feature: The "minimap", which presents a miniaturization of the entire source code, making navigation easier.

NotePad++ and Eclipse are two other excellent options for editing HTML code.

JavaScript libraries for general use

HTML5 Boilerplate is a set of best practices and HTML / CSS / JavaScript settings for creating responsive websites that includes features such optimizations, reconciliations cross-browser, compression and so on. There is also the Mobile Boilerplate version.

JQuery Mobile is a touch version of consecrated JavaScript library compatible with a wide variety of mobile and desktop browsers. See also other alternatives: Zepto and JQTouch.

oCanvas is a JavaScript library that greatly facilitates the handling of HTML5 CANVAS object-oriented manner. It has also supported Mobile.

Buzz allows easy manipulation of the HTML element SOUND.

Hammer a library that implements multi-touch support in their applications.

Frameworks for game development

Construct2 is a complete environment for creating games consists of a JavaScript framework and an IDE. The schedule is totally visual, via drag-and-drop, in a logic-based events and behaviors. Game Maker and Game Salad are other software that follow the same line of approach.

Crafty provides a framework for building HTML5 games. Among its features are: cross browser, choice of Canvas or DOM maps sprites, collision detection and modularization. Quintus, EnchantJS and ImpactJS are other software in the same category.

Hybrid Mobile Frameworks

Are frameworks that act as a bridge between the web app and the device, which

can be Android, Windows Phone, Apple, etc..

PhoneGap allows you to create cross-platform native apps using HTML5, CSS3 and JavaScript triad, but with access to own the various mobile devices such as the accelerometer, camera, compass, geo-location, sound files and other resources. Their "rivals" are: CocoonJs and Intel XDK.