Becoming a Web Developer

I am constantly asked by desktop developers how to make the transition to web development. Web applications are almost as powerful and as fast as desktop applications these days. Plus, there are no installation hassles as the application resides in just one place: on your server. To use the application, the user simply navigates using a browser to your application's starting point, and they can start working. In this blog post, I provide you with guidance on how experienced developers can get started with web programming. I am not going to go in-depth into each technology and tool. Instead, I will introduce you to terms, technologies, and tools needed for web development, and provide you with links on where you can learn more about each.

Core Technologies to Learn

If you are already programming in .NET, the back-end tools you use will be the same. You are probably already using the Entity Framework, or some other object-relational mapper (ORM) like Dapper, PetaPoco or OrmLite for your data access. I'm sure you already know how to use a database system such as SQL Server or Oracle, or maybe a NoSql. These remain the same for all your server-side processing of data and generation of HTML. In addition to your backend tools, you need to learn how to create the front-end UI.

Hyper Text Markup Language (HTML)

Hyper Text Markup Language (HTML) was developed to describe how to display text and images in a web browser. HTML elements such as , <p>, and <a> tags are some of the building blocks for each web page you want to display. Even if you use a server-side tool that generates HTML for you, you are still going to need to understand how to read HTML to troubleshoot your web pages. HTML5 is the latest standard for HTML. Many new elements are available in HTML5 to provide capabilities such as video playback, rendering shapes, geolocation, and drag and drop. Learn more at https://www.w3schools.com/html/.

Cascading Style Sheets (CSS)

Using HTML elements to layout your web page is just the first step in developing web applications. You want to make it look good, and to be responsive so users can

view the page on any device. This is where Cascading Style Sheets (CSS) come in. CSS helps you style your HTML so the page looks good on any device. CSS is an essential skill you must master to make a professional web application. Learn more at https://www.w3schools.com/css/.

Responsive Design CSS Libraries

Most web pages these days are viewed on either a mobile phone or a tablet. As such, your application needs to look great on these different size screens. This is called responsive design. You can certainly build a responsive design using CSS, but there are a few CSS libraries out there that have done all this for you. Bootstrap (https://getbootstrap.com) is one, and Material Design (https://material.io) is another. These libraries have pre-built styles that automatically change your web pages based on which size screen your user is using.

JavaScript

JavaScript is an interpreted language that runs in any modern web browser. You use JavaScript to program behaviors on your web pages. For example, what happens when the user clicks on a button, a menu, or an image. The syntax of JavaScript is very similar to C, C++ and Java, so for many developers, it is very easy to learn. Learn more at https://www.w3schools.com/js/.

JavaScript Object Notation (JSON)

JavaScript Object Notation (JSON) is a way to exchange data between two different systems. JSON is considerably smaller than XML and is the preferred method to transfer data from a client application to a server. Learn more at https://www.w3schools.com/js/js_json_intro.asp and https://www.json.org/.

Document Object Model (DOM)

The Document Object Model (DOM) is what the browser creates from the HTML elements on a web page (Figure 1) after it is loaded. This object model is a tree-like structure with parent, child and sibling elements. As shown in Figure 1, the document contains a root element named *html*. The *html* element contains two sibling elements; *head* and *body*. The *head* element can have any number of child elements. In Figure 1 there are three children elements; *meta*, *title* and *link*. The *body* element can have any number of child elements. In Figure 1, there is a *header*, a *div* and a *footer* element. The *header* element has a child element that is an *h1* element. The *div* element contains two children; *p* and *img*. The *p* and *img* elements are considered siblings to one another, and children of the *div* element.

This object model is created to allow JavaScript to manipulate elements within the document. Attribute values of the elements can be modified, or new attributes can

be added. You can modify add or remove HTML elements, and change the style of elements. JavaScript also allows you to respond to events on an element.

Each HTML element is created as an object in the DOM. You may then manipulate an element(s) by setting properties or perform an action by invoking a method. Learn more at https://www.w3schools.com/js/js_htmldom.asp.

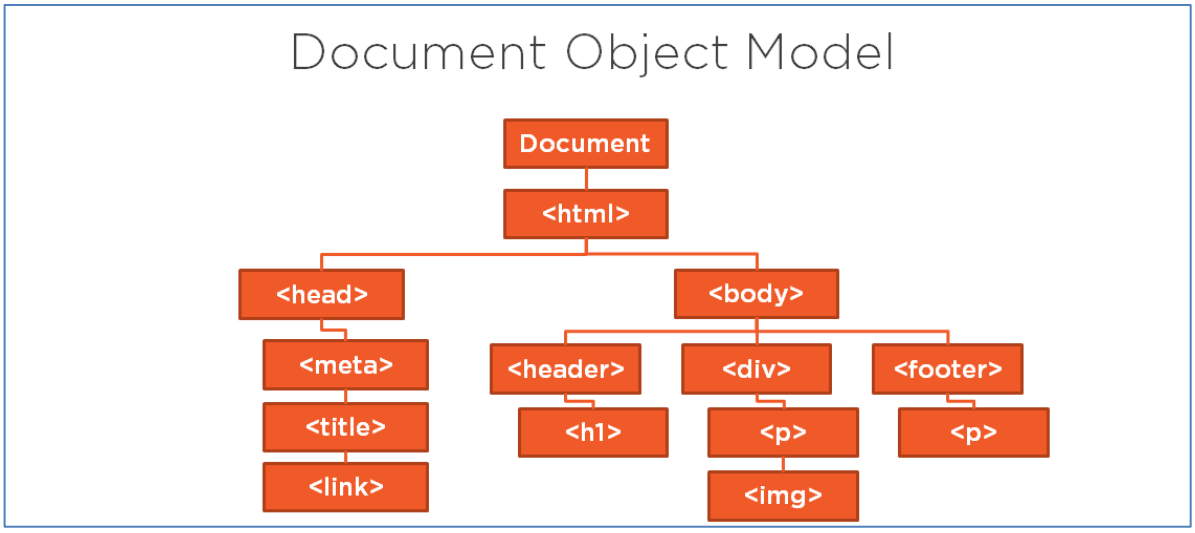

Figure 1: The DOM is a tree structure created by the browser of all elements on a web page

Hypertext Transfer Protocol (HTTP)

I'm sure you use a browser to go to Facebook, Amazon or one of any other of the millions of sites on the world-wide web. You may notice that after you type in www.amazon.com that the browser probably prepends "http" or "https" to the www address you typed in. The Hypertext Transfer Protocol (HTTP) is a standard protocol designed to transmit HTML and other documents. Its purpose is to facilitate communication between web browsers and web servers. HTTP is the epitome of the client-server model. Your browser opens a connection to a web server, makes a request for a web page (or other file) and waits until it receives a response. Once the page is returned, the connection is severed. HTTP is what is called a "stateless" protocol, meaning that the server does not maintain any data (state) about the connection after it is severed. Learn more at https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

Web API

Unless you are writing a very simple web page, at some point you are probably going to want to exchange data between your server and your web page. The data is sent using JSON, but you need a method that is exposed on the world-wide web that can accept data or send data out. A Web API is an HTTP endpoint that listens for requests and runs code on the server. This code can be written using ASP.NET

and C#, Java, or many other back-end technologies. Learn more at https://msdn.microsoft.com/en-us/library/hh833994(v=vs.108).aspx.

Representational State Transfer (REST)

Representational State Transfer (REST) is an architectural paradigm defining how a client can interact with a server to retrieve or manipulate data across the HTTP protocol. For example, you write a Web API that accepts requests from a client, processes that request, and returns a response back using HTTP status codes. The request sent by the client are called verbs, with the most common ones being GET, POST, PUT and DELETE. GET is when you wish to retrieve data, such as a web page or a JSON object. A POST is used to send data a user enters in a web page, or for sending a JSON object built using JavaScript. A PUT is used when you wish to send data as a JSON object to update data on the server. The DELETE verb allows you to send a JSON object to delete data on the server.

Each of these verbs sends back a status code as to the success or failure of the operation. You may receive an HTTP status code of 200 indicating success. You may receive a 404 if the resource you were attempting to update was not found. You are not limited to just sending JSON objects, you may also use XML, HTML or some other format. Learn more at https://en.wikipedia.org/wiki/Representational_state_transfer.

Secure Sockets Layer (SSL)

When you browse to www.amazon.com, your browser immediately prepends "https" to the address. This modifies your browser session to become a secure connection. All data sent back and forth between the server and your browser is encrypted. The web server has a certificate issued by a trusted authority. The server creates a special key which it sends securely to the client. This key is used to encrypt the data sent from the client to the server. Only the server has the other key needed to decrypt the data. The same is true in reverse as well. SSL is a deprecated technology now, but people still use it when talking about a secure web site. The updated standard is called Transport Layer Security (TLS). Learn more at https://en.wikipedia.org/wiki/Transport_Layer_Security.

Tools to Learn

You are probably used to using Visual Studio (VS) to develop Windows Forms or WPF applications. Good news; you may also use Visual Studio to develop web applications too. There are a few other tools that you might wish to become acquainted with as they will be useful in your web development.

NuGet Manager in Visual Studio

If you have not had a chance to use the NuGet Manager in Visual Studio, you most likely will when you start doing web development. There are many free libraries that you will add to your application that are available through this utility. Web developers tend to create some very high-quality libraries and make them available to anyone who wants to use them. The NuGet Manager allows you to search for these libraries and add them directly into your project. NuGet Manager helps you upgrade to newer version of any libraries you have added to your project. Learn more at https://www.nuget.org/.

Browser Developer Tools

Most browsers have a set of "developer tools" built-in that are typically accessed via the F12 key. You often hear these tools referred to as "F12 Tools". These tools are very useful for helping you debug your JavaScript, drilling down into the DOM, and viewing what CSS is currently applied to each element. Take some time to get to know what you can do with these developer tools for your browser. Each browser has similar tools, but how you use them varies from browser to browser. Table 1 has a list of where you can find more information about each browser's developer tools.

Browser	Tools
IE 11 and earlier	https://msdn.microsoft.com/en-us/library/hh772704(v=vs.85).aspx
Microsoft Edge	https://docs.microsoft.com/en-us/microsoft-edge/devtools-guide
Chrome	https://developers.google.com/web/tools/chrome-devtools/
FireFox	https://developer.mozilla.org/en-US/docs/Tools

Table 1: Browser developer tools reference

GitHub

GitHub is a place on the web where you can store code for your libraries and/or your application. It is a source control system that allows multi-developer collaboration. GitHub contains a set of repositories. Each repository can be made public for no charge, or private for a small fee. Learn more at https://github.com/.

Git

Git is a tool used to interact with GitHub. This tool is a command-line interface, and it also integrated into many IDE's such as Visual Studio and Code. This tool allows you to add, edit and delete files from a repository. All changes are tracked within

GitHub so you can revert to an earlier version of a particular file. Learn more at https://guides.github.com/introduction/git-handbook/.

Visual Studio Code

Visual Studio Code (a.k.a. VS Code or just Code) is a lightweight source code editor that is very popular among front-end developers. This editor is available on Windows, macOS and Linux. Support for the most commonly used languages such as JavaScript, TypeScript and Node.js is built right in. In many cases, you will find you can get things done faster with Code rather than Visual Studio. There are many extensions you can add to Code to make it work for your other languages such as C#. Extensions are available to help you debug each of the languages too. By making everything an extension allows this editor to stay lean and provides faster response times. Extensions can be created by anyone, so you may find a few different extensions that perform the same function. You may then choose the one that suits you best. Learn more at https://code.visualstudio.com/.

Node.js

Node.js is an open-source, asynchronous, event-driven runtime engine typically used to build network applications such as a web server. If you use Code to develop Angular or React applications, you will most likely use the Code Node.js server to run the pages in your application. Learn more at https://nodejs.org/en/.

npm

npm is a package manager for JavaScript code. It is made up of three distinct entities; a website, a Command-Line Interface (CLI), and a registry. The website can help you discover what packages are available for download, however, a google search will work just as well. The CLI, named "npm", assists you with downloading and installing a package into your application. The registry is the database of JavaScript packages and the meta-data associated with each package. Learn more at https://www.npmjs.com/.

Web Application Development

When it comes to web application development, you can approach it in a couple of different ways. You can use a server-side approach, or a client-side approach, or a combination of the two. Both have their advantages and disadvantages. There is no right way or wrong way to which approach you use. Often, the needs of the user will decide how you program each web page to work. What I have found is for pages with not a lot of interaction I use a server-side approach. When I need a page that is very interactive, must respond to the user quickly, and the page's HTML may need

to be modified on the fly, I choose a client-side approach. In the next two sections, I discuss server-side development and client-side development.

Server-Side Development

Ultimately a browser needs HTML to display. However, what if you have a list of products you want display, and that product data is in a product table in an SQL Server database. You don't want to create a product.html page with a list of hard-coded products, because this static page would not change when someone modifies the product table. This is where a server-side development tool such as ASP.NET or PHP can assist you.

Instead of the user requesting a static product.html file using www.mysite.com/product.html, they instead send a GET request to the server to an endpoint such as www.mysite.com/product. Dropping the .html file extension allows the web server to intercept this request and call a program you write to get data from the product table and integrate that data into an HTML template. Together these two pieces come together to create a product.html file that is sent back to the browser to display (Figure 2). Each time the user calls this endpoint, the product data is retrieved from the database, and a new product.html page is sent back to browser.

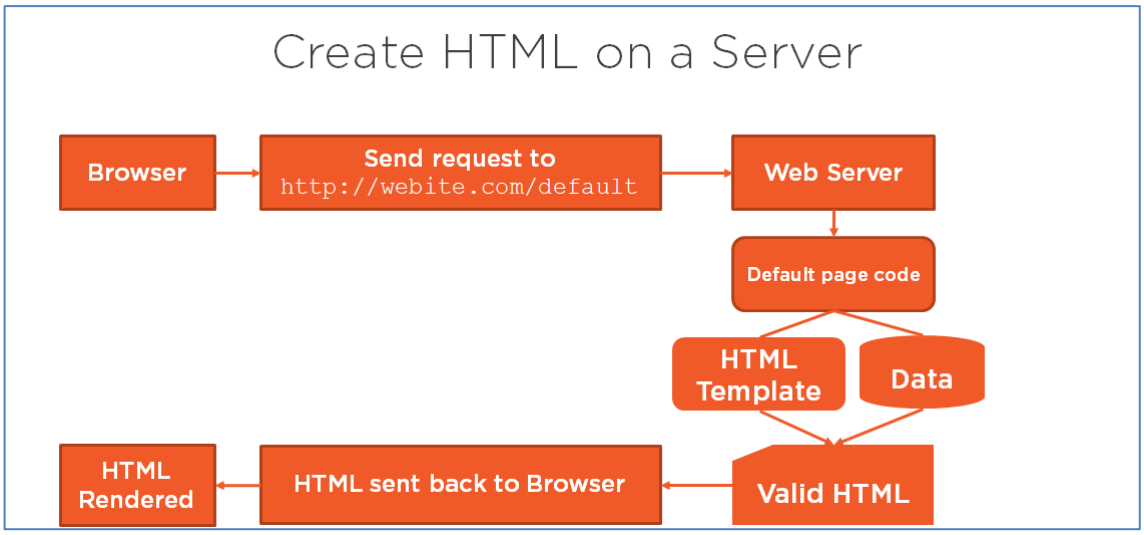

Figure 2: Create HTML on a Server using a programming language like C# or Java.

There are a few different options you have when using Microsoft tools; Web Forms, MVC or Core MVC. There are other server-side tools you can use such as Java, PHP, Ruby on Rails, Python and many others. Let's look at some of the most popular choices

PHP

PHP is a popular language and web framework that has been around for quite some time. PHP is easy to use and helps the novice web developer get a website up and running in a short time. Created in 1994, it is one of the oldest web development languages.

Advantages

1. Can be run on many platforms, including Windows, macOS and Linux.
2. PHP is open-source which means it is developed and maintained by a large group of PHP developers
3. Since it is maintained as open-source, when bugs are found, you can fix them yourself
4. It uses a C-like syntax, so many developers can get up-to-speed with it quickly
5. There are many libraries that integrate into PHP because it has been around so long
6. Connecting to a database to manipulate data is easy to do

Disadvantages

1. There are a lot of legacy coding practices that can make for hard to maintain code
    a. You can either use procedural style coding or object-oriented style of coding
    b. Sometimes one application uses both making maintaining that code very difficult
2. Since it is open sourced, people can read the code and find ways to exploit websites written in PHP
3. Not a strongly typed language
    a. This can lead to unexpected bugs
4. It is an interpreted language, not compiled

Ruby on Rails

Ruby on Rails is a web development framework that makes developing web applications easy. GitHub, Shopify and Airbnb are just some of the websites built using Ruby on Rails. Rails is a framework and is very opinionated about how you write a web application. If you write the way they want you to, then you will have a good time, if not, then you are probably not going to like programming using this framework.

Advantages

1. Good tools for development

2. Quick to get a web site up and running
3. There is a large amount of third party libraries available
4. Good support for testing

Disadvantages

1. The load time for the framework can be slow
2. The runtime can be slow
3. Not very flexible as you must follow their way of doing things
4. Can be hard to find good developers
5. Hard to find good documentation for what you want to do

ASP.NET Web Forms

Web Forms was Microsoft's way of easing desktop developers into web development. They tried to make Web Forms react in a similar fashion to Windows Forms development. Web Forms binds data retrieved from your data store to server-side controls, and those controls render the appropriate HTML. While this makes for some quick development, it is not as easy to customize the HTML that it produces. In addition, the runtime for this engine is very large because of the event model that is used. Web Forms is a legacy technology and should not be used for new development. You might still run into this technology at many companies however, so learning the basics of it might be appropriate. Learn more at https://www.asp.net/web-forms.

Advantages

1. Rapid application development
2. Drag & drop for designing pages
3. Data binding
4. Lots of third party support
5. Easier for desktop developers to learn

Disadvantages

1. Little control over the HTML that is rendered
2. Can't have multiple form tags on a page
3. Large page lifecycle makes it slow
4. Working with the lifecycle events can be tricky
5. Hard to automate the testing process

ASP.NET MVC

ASP.NET MVC (Model-View-Controller) shares the same engine as ASP.NET Web Forms, but the pipeline has been simplified which makes it faster than Web Forms. There are several advantages to using MVC over Web Forms.

Advantages

1. Separation of concerns
2. Data binding
3. Lots of third party support
4. Faster execution speed
5. Multiple form tags allowed on a page
6. Similar programming model to other web development paradigms
    a. Easier for non-Microsoft developers to learn
7. Designed to be stateless
8. Can automate the testing process
9. Full control over HTML that is rendered
10. Uses HTTP correctly

Disadvantages

1. Developers need to understand OOP and MVC
2. Can't see the design of the form during development
3. Harder to follow the flow of the logic because of the separation of concerns

Learn more at https://www.asp.net/mvc

ASP.NET Core MVC

The ASP.NET Core framework is an open-source web framework for creating web applications. This is a significant rewrite of the .NET framework targeted for cross-platform development. An ASP.NET Core web application runs on Windows, macOS and Linux servers. This framework has basically the same advantages and disadvantages as ASP.NET MVC but has two additional advantages; it is faster and cross-platform. Learn more at https://www.asp.net/learn.

Client-Side Development

Client-side development means starting with a product.html file that only has an empty HTML element such as a *div* tag. It is into this tag you build the HTML using data retrieved via a call to a Web API and mixing that into a pre-defined HTML template. There is typically only one HTML file containing the *html* and *body* tags. All other pages are built by replacing the contents of the *div* element with data retrieved from a Web API call, and other HTML templates as shown in Figure 3.

This type of development can be faster than server-side rendering as you are only transferring JSON data from the server to the client, not the HTML plus the data within that HTML. There are some instances where server-side rendering can be faster, so you need to decide which approach is best for your situation. Remember,

you don't need to do all one or the other. You can mix and match these methods just fine.

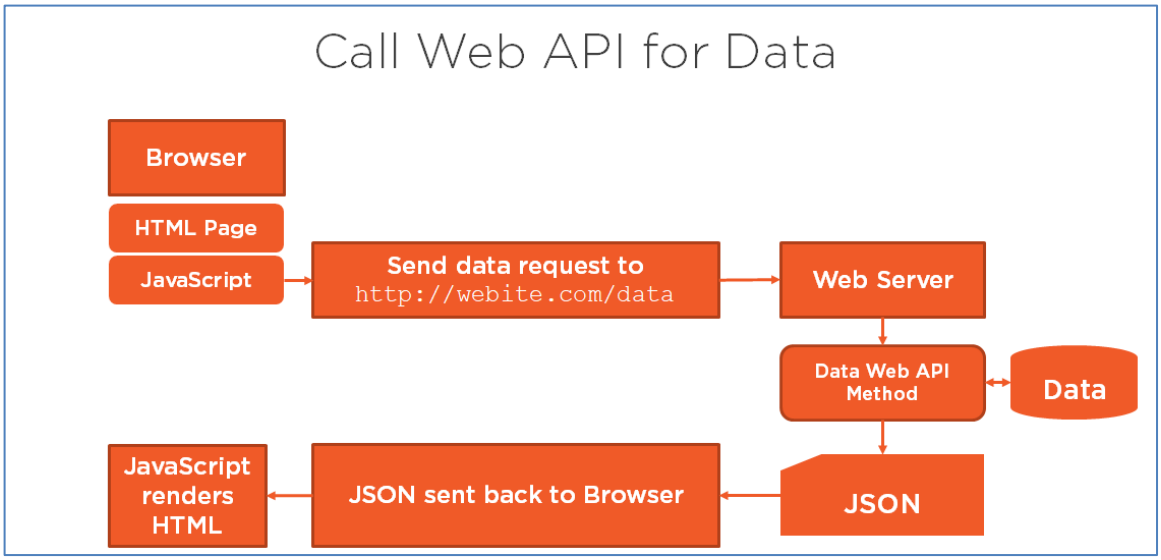

Figure 3: Call Web API to retrieve data to render.

Libraries versus Frameworks

Before looking at some of the various tools you can use to create client-side web applications, let's first talk about libraries and frameworks.

Library

A library is a package of JavaScript that performs a single task and performs it very well. For example, you may have a library that just knows how to display a drop-down calendar or display a toast notification. There are many libraries available on GitHub and through npm. What is nice about a library, is they are usually easy to integrate into an application. You can use several libraries together to create your application. The downside is you might have to use several libraries to create your application. I know, that is both an advantage and disadvantage, but sometimes libraries can conflict with one another. Sometimes libraries have different styles of programming, for example, one library uses callbacks, while another library uses promises. This can be lead to some confusing code. In addition, there are usually a lot of libraries that provide the same functionality. You might have to spend a lot of time working through the different ones to find one that works for your situation.

Framework

A framework is a collection of JavaScript libraries that have been written in a consistent manner and provide functionality that would be the equivalent of combining several libraries together, but someone has already done the job of making the usage of all of them the same. A framework makes the choice for you of what components are used. This means you no longer need to take the time to

research a bunch of different libraries to see which one works the best. Of course, the downside of using a framework is you are typically locked into their way of doing things. If you need to do something out of the ordinary, it can sometimes be very difficult, if not impossible to do.

Which One to Choose

There is no easy answer to whether you use a library or a framework. It is pretty much a personal choice, but can also be dictated by your corporate environment, and by the type of application you are creating. I find that for smaller applications, written by one, or just a few, developers that a library approach is usually best. For large (enterprise) applications, with a larger development team, a framework approach generally works best.

jQuery

While JavaScript is powerful, it can sometimes be a little verbose. jQuery is a JavaScript library that simplifies JavaScript coding significantly. Sometimes reducing several lines of code to one line. Learn more at https://jquery.com/.

TypeScript

TypeScript is a superset of the functionality found in JavaScript. The most notable feature of TypeScript is it provides type-safety and OOP features in a syntax that is familiar to C# and Java developers. There is no browser that supports TypeScript, thus TypeScript must be converted to JavaScript before your web page is sent down to a user. This conversion process is called "transpiling". It is not required that you learn TypeScript, but many developers are moving to it, and if you are using Angular, you will use it. Learn more at https://www.typescriptlang.org/.

Vue.js

Vue.js is a popular JavaScript library that purports to make creating client-side web applications faster and easier to develop. The main theme behind Vue.js is to help you build user interfaces. This library provides the HTML templating and data-binding to allow you to create web pages quickly. This library only requires that you know HTML, CSS and JavaScript to use, so the learning curve is easy. You use any other library you want for other features of an application. There are several other features in the Vue.js library, so you can learn more at https://vuejs.org/.

React

React is a popular JavaScript library that makes it easy to build user interfaces. React uses templating and data-binding to layout your HTML and inject data within that HTML. This library only requires that you know HTML, CSS and JavaScript to use, so the learning curve is easy. You use any other library you want for other

features of an application. There are several other features in the React library, so you can learn more at https://reactjs.org/.

Angular

Angular is a full featured framework with a large API to provide everything you need to build an enterprise application. Since it is so large, and is a framework, the learning curve is more significant than Vue.js or React. You also use TypeScript instead of JavaScript to program an Angular application. However, once you learn the framework, Angular is very powerful and makes creating a full-featured web application easy. More information about Angular can be found at https://angular.io.

Server-side or Client-side?

So, between server-side and client-side development, which one should you choose? Unfortunately, there is no easy answer. A lot of it depends on what kind of application you are writing, what your corporate culture has dictated, how many developers will be working on your application, your (and your team's) skillset and many other factors. Of course, if you choose server-side technology like MVC for your development, you can easily integrate a few pages that use client-side technology when you need it. So, the good news is you don't really need to decide until your application demands it. Start with either technology, and adjust what you use based on what is required.

Server-side Advantages

1. Best approach for static web pages
2. Search engines can crawl the pages for better Search Engine Optimization (SEO)
3. Pages are loaded faster initially

Server-side Disadvantages

1. Harder to make pages that are very interactive
2. After the initial page load, frequent post-backs are slow since the whole page is typically redrawn

Client-side Advantages

1. Pages can have a lot of rich interaction
2. Pages are very fast to render
3. Wide variety of JavaScript libraries to choose from

Client-side Disadvantages

1. The first page tends to load more slowly
2. SEO can be difficult to implement
3. You will generally need multiple libraries to provide a great user experience

Summary

In this blog post you were introduced to the different technologies you need to learn to become a web developer. Various tools for creating web applications were explored such as NuGet, GitHub, browser developer tools and Code. You learned the different kinds of tools you can use to create a server-side web application. If you want to do client-side development, you are going to need to choose between using a library or a framework. The great thing is you can use a combination of both server-side and client-side coding techniques within one application.

Sample Code

There is no sample code for this blog post, but to find other blog posts, visit http://www.pdsa.com.