Chapter 1: Change Begets Change

The current state of the internet is drastically different from when it dawned over 30 years ago. Fundamentally, the internet still works the same way as it always has. A client sends a request for an HTML file and the server responds with the resolution to that request. In its early years, the server simply responded to a request with a static webpage stored on the site's servers, which was useful for informational and social media purposes. In recent years, the web has become a lot more involved than just requesting and sending a file back and forth. Unlike HTML, JavaScript can be run either on the client-side or the server-side to achieve processing tasks that were not possible on the hardware at the time. Now, websites can run entire applications using server resources and allow the user's machine to handle querying for those resources. For example, Xbox has recently released their Game Pass streaming service that allows gamers to either install games from a library of hundreds, or stream them from the internet from a device that can't run the games itself. An iPhone has the ability to stream games that only expensive graphical hardware was able to render previously. Since the web is constantly evolving, it's important to ensure that our web applications evolve with it. Using a modern client-server approach to web development allows for endless expandability for the developer community. Modern client-server applications respond only with minimal information for the browser to handle, which is useful for preserving bandwidth on a home network that typically has an issue with congestion, such as mine. Prototyping is a vital aspect to web development using this approach, as it allows for the kinks of a product to be ironed out prior to release. It can also be delegated to the open-source community to find bugs and security issues that need to be patched immediately.

Chapter 2: JavaScript and JavaScript Tools

JavaScript is a browser-oriented language that spawns from Java and was created in 1995 by Brendan Eich. This language borrows the syntax frameworks from neighboring languages, such as Perl, C++, Scheme, and lisp to create a combination of functional programming and Object-Oriented programming. Similar to Perl, JavaScript adopts a broad and adaptable approach based on the "there's more than one way to do it" ideology. In python, many developers use a "There's only one way to do it" ideology that is in contrast to how JavaScript functions. Upon its conception, Eich was recruited by Netscape to develop a Scheme-based language to be used in the browser, which later became JavaScript. While this is based on Java, it's more focused on function-based programming opposed to Java's focus on object-oriented programming. One issue with the architecture of JavaScript is that hoisting may move a variable to be inaccessible by the functions. This is mitigated by adding a var at the very top of the scope that tells the rest of the functions what variables will be used throughout the process. JavaScript functions are built of four parts, the function operator, an optional name, open/closed parentheses, and open/closed brackets. An anonymous function in a JavaScript program is one that does not have a name and can be assigned to any variable. These allow for internal loops to access the function in a different way from the external loops. In order to call a function, four methods can be used: as a function, as a method, as a constructor, and by using call() or apply(). JavaScript differs from Java in that it is an interpreted language and does not need to be compiled. This can save developers time and resources that may be used for higher priority tasks. Due to the dependencies that JavaScript employs, debugging can be difficult for unexperienced developers that are unfamiliar with the fundamentals of those dependencies.

Chapter 3: REST and JSON

REST is an architecture that defines how the web should run and enacts constraints that allow for a seamless synonymy across various websites and platforms. The architecture is considered standard due to its reliance on the fundamentals of HTTP, which is what the entire internet is based on. Many view REST as a web services protocol in relation to its ability to run on loose, free-form messages and URLs. Resources were previously defined as the specific document that is being requested from the client to the server, but the definition has since changed to include a broader scope of any object accessible via the internet. This can include movies, music, program files, social network profiles, etc. When it comes to URL handing, REST uses a slightly different definition of resources compared to HTTP. In REST, the resources are path elements and the slashes are used to denote resources and express their relationships in regard to the hierarchal structure.

JSON stands for JavaScript Object Notation, is a multi-purpose and light-weight alternative to XML that is a subset of JavaScript. Commonly, this language model is seen as advantageous to XML because it uses less data, allows for interoperability between web pages, and allows for the ease of Ajax integration into the web resources. In JSON, there is no way to denote a comment, can be taken by eval() run from JavaScript, and is prevented from being evaluated from a separate site by the same origin policy enacted by many modern web browsers. Only a few data types are available in JSON: strings, numbers, Booleans, objects, arrays, and null. There is no data type to handle hyperlinks as many other API's use, which is incongruent with the schema defined by the REST constraints called Hypermedia as the Engine of Application State (HATEOAS).