**What is a Web App?**

**Learn about web apps and what goes into building them.**

**What is a Web Application?**

A web application is an application software that does not require installation and can instead be accessed from a remote server via web browser. Web applications are made for interaction, allowing users to send and consume data between the browser and the web server. This interaction can be as simple as logging in to an account, or as complex as making a payment with your credit card.

**What is the Difference Between a Website and a Web Application?**

While the terms website and web application are often used interchangeably, they can denote somewhat different things. Most commonly, a website is defined as a set of information-carrying pages that are inter-related and accessed through a web browser. A web application, on the other hand, is an application software that runs on a web server and is accessed by the user through a web browser.

If a website could be said to be defined by its content, then a web application would be defined by its interaction with the user. As such, web applications are significantly more complicated than static websites, both in general architecture and features.

**Web Application Architecture**

In order to facilitate this complex flow of data, web applications are usually designed with different layers. The most common design paradigm is a three-layered design consisting of a presentation layer (web browser), application layer (server), and storage layer (database). In this system, the presentation layer is responsible for relaying user data to the application layer, which can process that data and do any number of things, including passing it to the storage layer for “safe-keeping.”

Many times, web applications can grow to be very complex. In these cases, a three-layered design may fall short. This may necessitate the introduction of additional layers to handle this complexity. For instance, the introduction of an integration layer between the application and storage layers can help provide a uniform interface for data access, allowing the application layer to be insulated from changes that occur to the storage layer implementation.

If you’re interested in learning more about the technology that’s behind your favorite web applications, install the [Wappalyzer chrome extension](https://www.wappalyzer.com/" \t "_blank). If you navigate to a site, you can click on it and it will give you a list of the different technologies used to build it.

**What is a SPA?**

**Learn what a Single-Page Application (SPA) is and why you would use it.**

**Introduction**

At this point in your web development journey, you have learned HTML to create web pages, CSS to style those pages, and JavaScript to sprinkle some magic throughout for functionality. With these foundational skills, you are now ready to explore the exciting web development practice of the Single-Page Application (SPA). SPAs combine the content of traditional websites with the smooth user experience of mobile applications. Learning to develop and maintain SPAs is an exciting venture that challenges developers to improve the web experience for users all over the world.

**Multi-Page Applications**

Prior to reading this article, you should be familiar with the concept of a [web app](https://en.wikipedia.org/wiki/Web_application). Whether talking about delivering content through static websites or interactivity through web applications, the article is referring to a multi-page file structure that lives on a server. Each time new data is needed for the browser view, a request is sent to the server, which returns a new set of page files. For a static website, this approach is generally fine, but web applications that require faster and more complicated interaction sometimes struggle to keep up.

Imagine if there was a restaurant where you can only eat one part of your meal at a time. For an order including a burger, fries, and salad, the server(server!) brings out just the burger. After a few bites you want some fries, so the server takes the burger back to the kitchen and brings back just the fries. The server may have even had to wait for the fries to process–I mean prepare, before returning to the table. Repeat the request, prepare, and serve steps with the salad, then the burger again and then more fries and you’ve now eaten at the weirdest, most inefficient restaurant ever. Thankfully this doesn’t happen in the food industry but something similar is happening on the web.

In the following video, a user is navigating around [Wikipedia](https://en.wikipedia.org/wiki/Wikipedia:Contents). On the right, you may recognize the Elements tab of Chrome DevTools displaying an HTML file associated with the page in the browser.

Every time a link is clicked a new page is displayed in DevTools, which means a request to the server was made and a new set of files was sent to be rendered in the browser. This is just like our burger, fries, and salad from our favorite restaurant. In terms of a web application where interactivity is key, these file requests can mean a slow user experience. That’s where the Single-Page Application comes in.

**Single-Page Applications**

[Wikipedia defines](https://en.wikipedia.org/wiki/Single-page_application) a single-page application (SPA) as “a web application that interacts with the web browser by dynamically rewriting the current web page with new data from a web server, instead of the default method of the browser loading entire new pages.” The name single-page application generally refers to the application consisting of one page that is constantly updated by JavaScript. Requests to the server are now quicker since they contain just the data needed to update the view. SPAs are full applications, running in the browser yet still connected to a server to update any application data.

If our favorite restaurant converts to the single-page approach, when we order our meal the server brings the fully prepared burger, fries, and a salad at once. We now have access to devour what we want without the server going to and from the kitchen. At times we may want ketchup for the fries, or some dressing for the salad, but those are delivered quickly and come from the kitchen already prepared. Just like a single-page application, you are given almost everything you need upfront but can still request smaller items to help with each item of the meal.

Now we have a video of user interactions on the [React home page](https://reactjs.org/), which is built as a single-page application. When looking at the Elements panel this time, each interaction changes only parts of the HTML file.

The file stays constant while logic from the client-side JavaScript changes only what is needed to update the view. Requests for data are retrieved a lot quicker than files that need to be processed on a server and then rendered in the browser. SPAs focus on speed when it comes to user interaction and browser rendering times.

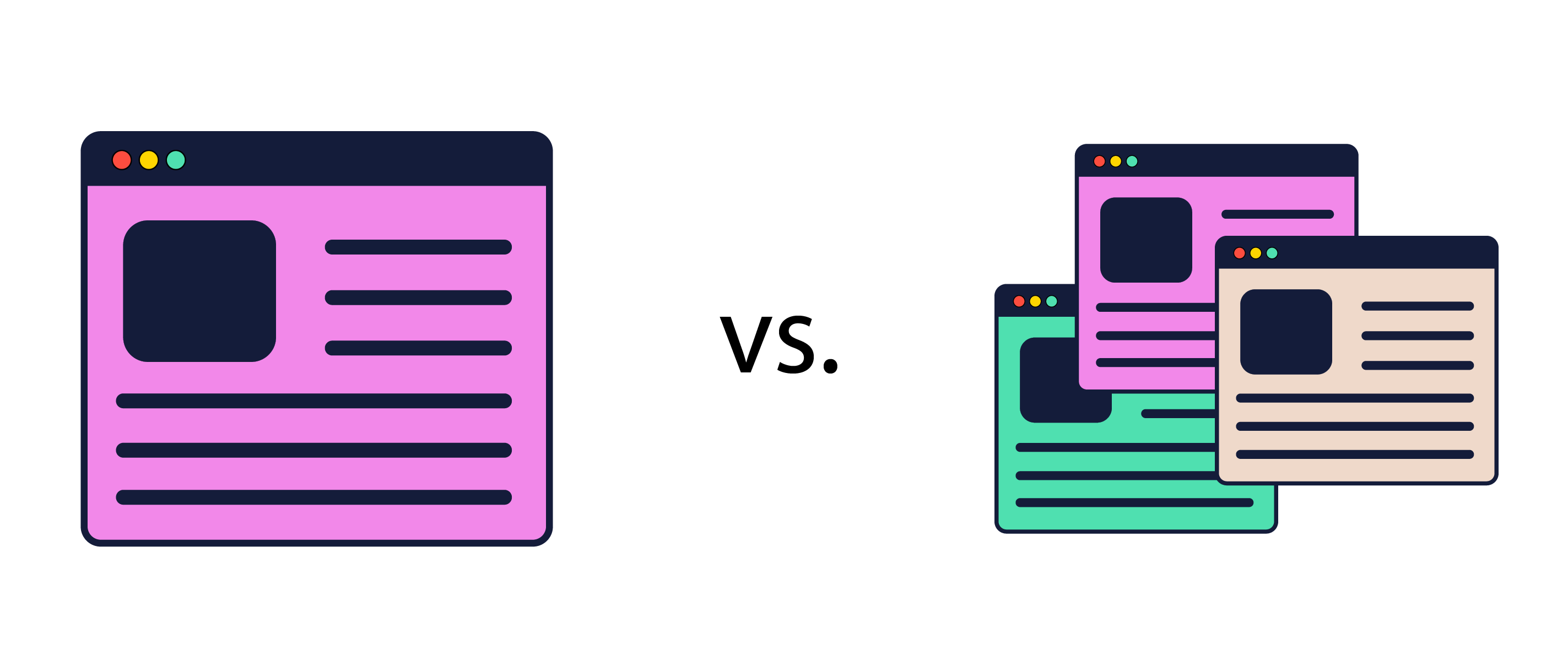
**SPA Frameworks**

To create a SPA you can use vanilla JavaScript for controlling all the required logic. In practice, the complexity of SPAs scales rapidly so just using JavaScript is not recommended. Luckily, there are several tools available to help with the creation of a SPA. These tools help with many tasks from controlling the view of the page to managing the application build.

* [React](https://reactjs.org/) is a popular JavaScript library for building single-page applications. It focuses on creating components that can render themselves differently based on an application’s current state and user data.
* [Vue.js](https://vuejs.org/) is a framework that uses templating within a single HTML file while the application logic controls what is rendered. This approach is sometimes thought to be more traditional and therefore easier to learn.
* Other libraries and frameworks include [AngularJS](https://angular.io/), [Ember.js](https://emberjs.com/), [ExtJS](https://www.sencha.com/products/extjs/" \t "_blank), [Knockout.js](https://knockoutjs.com/), and [Meteor.js](https://www.meteor.com/). While all of these share similar goals, they each take different approaches to building SPAs.

**SPA Pros and Cons**

So, should your next project be a single-page application? Maybe. Maybe not. Like most development endeavors you must consider the pros and cons before deciding how you will implement a new project.



**Pros**

* SPAs are fast. The main selling point of a SPA is that it feels like a desktop or mobile application. By eliminating requests for new files and only relying on smaller amounts of data from the server, SPAs provide a real-time interface with their users.
* Reuse of code is a big bonus when using SPAs because it saves time within a project and across multiple projects. Many SPA libraries and frameworks advise that components be general enough that they can be reused from project to project.
* SPAs provide an easier path to migrate code to a mobile application. With a SPA, the back-end of the application feeds data to the decoupled front-end interface. This separation of tasks allows the creation of a mobile app UI while maintaining the back-end logic of the application.

**Cons**

* SPAs require more files to run at startup so the load time of the application can be longer. This is something to consider if a user will not want to visit a site that takes too long to load. SPA load time can be minimized through strategically loading resources throughout the run of an application.
* Search Engine Optimization (SEO) has some pitfalls when it comes to SPAs. Search engines, like Google or DuckDuckGo, index pages of a website to rank the content. This can be difficult with only one page that may not have content until it is loaded by JavaScript. SEO is an ever-changing world so strategies already exist to mitigate these downsides.
* SPAs may not function as expected within the browser. For example, the back button or browsing history can act differently while using a single-page application. This can be frustrating for users who are expecting certain functionality within their browsers.

**Conclusion**

Single-page applications provide a better user experience while running within a web browser. They are the right choice for applications that need to provide real-time or complex interaction with their users. Creating a SPA involves more than just a handful of HTML, CSS, and JavaScript files, but their complexity continues to be minimized by frameworks such as React and Vue.js. Even if your next application is not a SPA, knowing how to implement them is a must-have skill for a front-end developer.