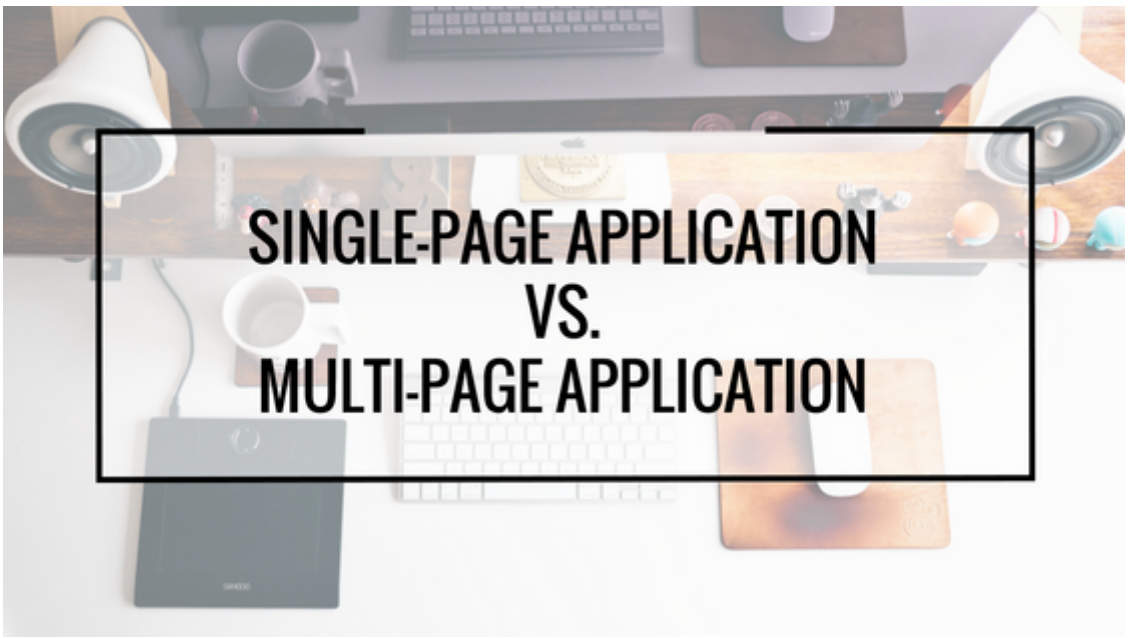

Single-page application vs. multiple-page application

Neoteric
Dec 2, 2016 · 6 min read

Web applications are unwittingly replacing the old desktop applications. They are more convenient to use, they are easy to update, and they are not bound to one device. And even though users are gently moving from browser-based web applications into the mobile ones, the demand for complex and refined apps is already huge and is still growing. If you are thinking about creating your own application, you've probably heard that there are two main design patterns for web apps: multi-page application (MPA) and single-page application (SPA). And of course, both models have their pros and cons.

Before you start turning your idea into the real application, you have to answer a bunch of important questions. To decide what app model is better for you, you should follow content-first approach, which emphasizes the importance of putting your application content before everything else. That's because content is the main reason for which users will or won't use the application. And this leads us to the most important questions: what content do you want to present and what content your users will care about the most.

There are many pros and cons of SPA as well as of MPA. I hope that the lines below will clear the differences between these two design patterns and bring you closer to the point where you will know what kind of application fits your idea more. And make your idea about own application come true.

Single-Page Application

A single-page application is an app that works inside a browser and does not require page reloading during use. You are using this type of applications every day. These are, for instance: Gmail, Google Maps, Facebook or GitHub.
SPAs are all about serving an outstanding UX by trying to imitate a “natural” environment in the browser — no page reloads, no extra wait time. It is just one web page that you visit which then loads all other content using JavaScript — which they heavily depend on.

SPA requests the markup and data independently and renders pages straight in the browser. We can do this thanks to the advanced JavaScript frameworks like AngularJS, Ember.js, Meteor.js, Knockout.js .
Single-page sites help keep the user in one, comfortable web space where content is presented to the user in a simple, easy and workable fashion.

Pros of the Single-Page Application:

- SPA is fast, as most resources (HTML+CSS+Scripts) are only loaded once throughout the lifespan of application. Only data is transmitted back and forth.
- The development is simplified and streamlined. There is no need to write code to render pages on the server. It is much easier to get started because you can usually kick off development from a file file://URI, without using any server at all.
- SPAs are easy to debug with Chrome, as you can monitor network operations, investigate page elements and data associated with it.
- It’s easier to make a mobile application because the developer can reuse the same backend code for web application and native mobile application.
- SPA can cache any local storage effectively. An application sends only one request, store all data, then it can use this data and works even offline.

Cons of the Single-Page Application:

- It is very tricky and not an easy task to make SEO optimization of a Single-Page Application. Its content is loaded by AJAX (Asynchronous JavaScript and XML) — a method of exchanging data and updating in the application without refreshing the page.
  UPDATE 27.09.2017: In her comment, Iris Shaffer correctly pointed out that it can be done on server side as well. Indeed, it is easier today than it used to be.

- It is slow to download because heavy client frameworks are required to be loaded to the client.

- It requires JavaScript to be present and enabled. If any user disables JavaScript in his or her browser, it won't be possible to present application and its actions in a correct way.
  UPDATE 27.09.2017: In her comment, Iris Shaffer has noticed that with isomorphic rendering / server side rendering, you can render the page on the server already. When the initial render is on the server and can be cached, disabling JS would not be a concern for getting a rendered page. Theoretically, that's right. Obviously you can render on server side. But lack of JS can be a concern for other functionalities. Lots of things can be done in HTML & CSS but from my experience it would be hell to do it this way instead of use JavaScript.

- Compared to the "traditional" application, SPA is less secure. Due to Cross-Site Scripting (XSS), it enables attackers to inject client-side scripts into web application by other users.

- Memory leak in JavaScript can even cause powerful system to slow down

- In this model, back and forward buttons become dysfunctional.
  UPDATE 27.09.2017: Nowadays, it's not an issue anymore.

Multi-Page Application

Multiple-page applications work in a "traditional" way. Every change eg. display the data or submit data back to server requests rendering a new page from the server in the browser. These applications are large, bigger than SPAs because they need to be. Due to the amount of content, these applications have many levels of UI. Luckily, it's not a problem anymore. Thanks to AJAX, we don't have to worry that big and complex applications have to transfer a lot of data between server and browser. That solution improves and it allows to refresh only particular parts of the application. On the other

hand, it adds more complexity and it is more difficult to develop than a single-page application.

Pros of the Multiple-Page Application:

- It's the perfect approach for users who need a visual map of where to go in the application. Solid, few level menu navigation is an essential part of traditional Multi-Page Application.
- Very good and easy for proper SEO management. It gives better chances to rank for different keywords since an application can be optimized for one keyword per page.

Cons of the multiple-page application:

- There is no option to use the same backend with mobile applications.
  UPDATE 27.09.2017: Back then, when I was writing this article, I didn't have much experience with backend and with mobile apps. It's obvious for me now that you can use the same backend for both. And I'd like to thank all the readers who pointed that out.
- Frontend and backend development are tightly coupled.
- The development becomes quite complex. The developer needs to use frameworks for either client and server side. This results in the longer time of application development.

SPA or MPA?

Before deploying a web application, you need to consider the goal of it. If you know you need multiple categories (because, for instance, you run an online shop or publish a lot of other content) — use a multi-page site. If you are sure that your site is appropriate for a pure single-page experience — go for it. And if you like SPA but can just barely fit everything into a single page, consider the hybrid site instead. This is another way I haven't mentioned before. A hybrid application takes what is the best in both approaches and try to minimize the disadvantages. It is, in fact, a single page application which uses URL anchors as synthetic pages enabling more in build browser navigation and preference functionality. But this is the topic for another article.

Perhaps in the future, everyone will use Single Page Application model (including a hybrid app), as it seems to bring a lot of advantages. Many apps on the market are

migrating towards this model. However, as some projects simply cannot fit into SPA, the MPA model is still vivid.

Originally published by Paweł Skólski at neoteric.eu/blog on December 1st, 2016.

Web Development Single Page Applications Application Development Web Applications Software Development