SPA

Single Page Application

History - Birth of web

-
- The web was born in the early 90’s
- Started to become popular with the first graphical browser in 93 which then became netscape
- Internet explorer followed on 95
- apple released safari in 2003
- chrome was released in 2008 and today is the most popular browser
- At first the web could only get static html pages with anchor links to other pages.

History - Birth of JS

- JS was born by netscape in 1995
- JS was part of ECMAScript in 96
- AJAX was born in 1998
- after AJAX it started to become more popular to download data from the server without refreshing the page
- more logic was thrown to JS side
- libraries like jQuery DOJO were born to help js developers
- in 2009 angular.js was born and other SPA frameworks like backbone, ember emerged
- in 2013 React was released as open source
- in 2016 ES6 was released
- in 2016 angular2 was released

What is SPA

- Single Page Application
- simulate behaviour of desktop applications we remain in a single page and content is loaded dynamically
- **initial load -** The first load of the page we grab the HTML and other resources referenced from the HTML
- after the **initial load** all other load from the server is based on AJAX
    - This doesn’t mean that ALL the resources have to be loaded on the initial load
- URL can still change using history api in HTML5
- reloading the page should lead to the same state

Advantages of SPA

- better UX
- less server requests to download a new full page
- server works less

Disadvantages of SPA

- requires JS
- initial load can be slow
- SEO
- using SPA frameworks makes larger JS files
- usually more memory cpu consuming sites

- Initial load and SEO can be improved by using Server Side Rendering

Server Side Rendering

Usually in SPA the initial HTML we get from the server looks similar to this:

```
<html>
...
<body>
        <script src=”my-spa-app.js”></script>
</body>
</html>
```

- the body is empty and contains a download of a script
- the script is running and in charge of presenting the page to the user
- initial load can be slow - we need to grab the html and js and only after that we need to run the js to render the page
- Unfriendly to search engines

Server Side Rendering

- with server side rendering the initial HTML we get from the server looks like this:

```
<html>
          …
          <body>
                    <h1>The full page is rendered by the server</h1>
                    <p>we get a full html page like it was when there was no SPA</p>
                    <p>We can also use server caching so the html will be sent faster</p>
                    ….
                    <script src=”my-spa-app.js”></script>
          </body>
</html>
```

- the first html is sent by the server, after that the spa takes control and everything is loaded dynamically
- modern frameworks/libraries like angular2/react support SSR - we have to make sure that the code we write is universal

Challenges of SPA

- Templates
- Routing
- SSR
- binding from inputs to JS
- Taking care of forms
- rerender page when needed

SPA Framework

- It was challenging to create SPA so as a result frameworks were released to help us create SPA
- Frameworks direct us to their way of developing SPA
- Frameworks usually have a very large code base which may have a substantial effect on memory and cpu usage

Angular6

- SPA Framework
- built with TypeScript
- has its own templating language based on HTML
- you extend that templating language by adding tags or classes that creates UI components
- take care of re rendering the components when needed
- works for node server side as well

Angular6 Advantages

- typescript
- reusable components
- easy testing
- much faster then AngularJS

Disadvantages of Angular6

- High memory usage
- Large JS files
- High CPU
- Performance is slower than React

React - SPA library

- React is open source js library maintained by facebook
- Library helps us create UI components
- React manage the state of component and re render the component when needed
- React is doing changes to the DOM very fast using VirtualDOM
- React is a library and not a framework and does not constrain you to a certain way of developing things
- React is fast with a minimal footpring on resources
- Not possible to create large SPA with just react and not using some other libraries like Redux or Flux
- React has a large open source community that on top of the library built frameworks and other libraries to help us create SPA application

React VS Angular

- The question should change to: When React and When Angular
- using npm you can split your frontend application to multiple packages
- you can use in your project Angular and React without mixing the framework/libraries and keep separation of frameworks in each project

Angular VS React

- Angular is a framework while React is a library
- Angular as a result is more opinionated
- Performance: React
- Memory CPU usage: React
- loading time: React
- TypeScript: Angular
- Testability: Angular & React
- Code reusability: Angular
- Code structure: Angular
- Easier to learn: React
- 3rd libraries support: React
- Large Teams: Angular

Summary

- SPA is better UX and it is the future of web development but still you have to ask yourself two important questions:
  - How big is the project?
  - How big is the team working on the project?
- If it's a small project like for example a landing page, it might be an overkill to use angular or react, I would just recommend using jQuery, and JS
- For more complex projects choose either React or Angular
- With a large team I recommend using angular
- for more smaller projects with smaller team I recommend using React