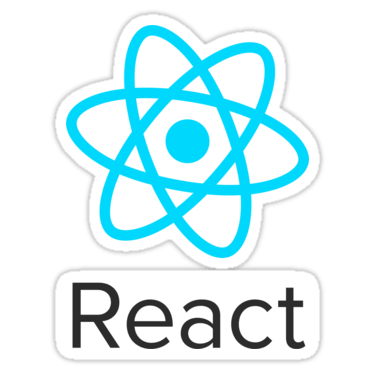
**Period-5 Single Page Applications with React**

Note: This description is too big for a single exam-question. It will be divided up into separate questions for the exam

https://camjackson.net/post/9-things-every-reactjs-beginner-should-know

Describe the term Single Page Application (SPA) and why it’s relevant for modern web-applications

A **single-page application** (**SPA**) is a [web application](https://en.wikipedia.org/wiki/Web_application) or [web site](https://en.wikipedia.org/wiki/Web_site) that interacts with the user by dynamically rewriting the current page rather than loading entire new pages from a server. This approach avoids interruption of the [user experience](https://en.wikipedia.org/wiki/User_experience) between successive pages, making the application behave more like a [desktop application](https://en.wikipedia.org/wiki/Desktop_application). In an SPA, either all necessary code – [HTML](https://en.wikipedia.org/wiki/HTML), [JavaScript](https://en.wikipedia.org/wiki/JavaScript), and [CSS](https://en.wikipedia.org/wiki/CSS) – is retrieved with a single page load,[[1]](https://en.wikipedia.org/wiki/Single-page_application#cite_note-Flanagan2006-1) or the appropriate resources are [dynamically loaded](https://en.wikipedia.org/wiki/Dynamic_loading" \o "Dynamic loading) and added to the page as necessary, usually in response to user actions. The page does not reload at any point in the process, nor does control transfer to another page, although the [location hash](https://en.wikipedia.org/wiki/Fragment_identifier) or the [HTML5](https://en.wikipedia.org/wiki/HTML5) [History API](https://en.wikipedia.org/wiki/Comparison_of_layout_engines_(HTML5)" \l "APIs" \o "Comparison of layout engines (HTML5)) can be used to provide the perception and navigability of separate logical pages in the application.[[2]](https://en.wikipedia.org/wiki/Single-page_application#cite_note-2) Interaction with the single page application often involves dynamic communication with the [web server](https://en.wikipedia.org/wiki/Web_server) behind the scenes.

Describe fundamental differences between the SPA-framework/libraries AngularJS and React

https://rubygarage.org/blog/react-vs-angularjs

First, take a look at the React vs AngularJS comparison in the table below.

|  |  |  |
| --- | --- | --- |
| **Technology** | **AngularJS** | **React** |
| Developer | Google | Facebook |
| Technology type | Full-fledged MVC framework written in JavaScript | JavaScript library (View in MVC; requires Flux to implement architecture) |
| Concept | Brings JavaScript into HTML Works with the real DOM Client-side rendering | Brings HTML into JavaScript Works with the virtual DOM Server-side rendering |
| Data binding | Two-way data binding | One-way data binding |
| Dependencies | Manages dependencies automatically | Requires additional tools to manage dependencies |
| Language | JavaScript + HTML | JavaScript + JSX |
| Last version | AngularJS 1.6.0 RC2 | React 15 |
| Suits best | Best for SPAs that update single view at a time | Best for SPAs that update many views at a time |

Angular:

The main differences between AngularJS (the framework) and React (the library) are in the following aspects: componentization, data binding, performance, dependency resolution, directives, and templating. Let’s look at each of these aspects separately.

componentization : AngularJS has a very complex and fixed structure because it's based on the three layers — Model, View, and Controller — typical of single-page applications.

React:

Facebook, the creator of React, chose an architecture different from that of AngularJS and similar MVC frameworks. In short, there is no “correct" structure for applications built with React.

React is a large JavaScript library that helps us update the View for the user. But React still doesn't let us create applications on its own. The library lacks the model and controller layers. To fill in the gap, Facebook introduced Flux, which has numerous variants nowadays, to control the application workflow.

React offers a freedom that AngularJS doesn’t. But this freedom comes at the cost of additional time spent designing the structure of an application. Before we start a new project, we have to think about what instruments we are going to use. When you have to pick a tool from among 100 options to resolve a single task, this choice becomes cumbersome.

AngularJS connects Document Object Model (DOM) values to Model data through the Controller using two-way data binding.

React uses one-way data binding, meaning we are able to direct the flow of data only in one direction. Because of this, it’s always clear where the data was changed. It’s worth noting that two-way data binding was available in React before v15 thanks to ReactLink.

Both AngularJS and React are great for writing single-page applications. But they are completely different instruments. Some programmers may say that React is better than AngularJS or vice versa. What’s really best for a given project, however, depends on how you use these instruments.

Working with React may seem a bit easier starting out, because you write old-school JavaScript and build your HTML around it. But there are many additional tools you'll have to grasp, such as Flux. In turn, AngularJS implements a different approach organized around HTML. That's why we may see unusual syntax and solutions that seem questionable at first sight. But once you get used to AngularJS, you will certainly benefit from its features.

Describe the overall principles used in React to create a SPA

https://reactjs.org/docs/design-principles.html

Explain, using examples, the fundamental building blocks in React Applications like:

* JSX

JSX - Tillader os at skrive HTML-lignende syntaks, der bliver forvandlet til lette JavaScript-objekter.

* Rendering Elements

render (metode) - Hvad vi gerne vil have, at vores HTML-skabelon ligner.

* Components and Props

De data, der sendes til barnets komponent fra moderkomponenten.

React.createClass - Den måde, hvorpå du opretter en ny komponent.

her visualiserer, hvordan vi bruger createClass til at oprette en komponent. createClass tager en objekt ind. Dette objekt er, hvad \*\* vil \*\* angive de forskellige egenskaber (render, getInitialState, propTypes) af komponenten.

var HelloWorld = React.createClass({

render: function(){

return (

<div>

Hello World!

</div>

)

}

});

ReactDOM.render(<HelloWorld />, document.getElementById('app'));

* State and LifeCycle

state - Den interne datalager (objekt) af en komponent.

getInitialState - Den måde, hvorpå du indstiller den oprindelige tilstand for en komponent.

setState - En hjælper metode til ændring af tilstanden af en komponent.

* Handling Events
* onClick
* onSubmit
* onChange
* List and Keys
* Working with Forms
* Lifting State Up

I Reakt udføres delingstilstand ved at flytte den op til nærmeste fælles forfader af de komponenter, der har brug for det. Dette kaldes "løftestatus op".

Describe tools like *Babel*, *WebPack* and *create-react-app* and how they fit into the React-world

https://www.robinwieruch.de/minimal-react-webpack-babel-setup/

man bruger **Webpack** som modulbundt og bygge værktøj. Desuden vil du bruge webpack-dev-server til at betjene din bundne app i et lokalt miljø. Ellers kunne du ikke se det i browseren for at udvikle det.

Man instillerer dem ved at bruge npm.

npm install --save-dev webpack webpack-dev-server

Fra rodmappe: touch webpack . config . js

npm start

man kan åbne appen i en browser . Derudover skal man se console.log()i konsollen.

[Med **Babel**](https://babeljs.io/) kan man skrive koden i [ES6 (ES2015)](https://babeljs.io/docs/learn-es2015/) . Med Babel kodes koden tilbage til ES5, så hver browser, uden at alle ES6-funktioner er implementeret, kan fortolke den. Babel tager endda det et skridt videre. Man kan ikke kun bruge ES6-funktioner, men også de næste generationer af ES.

Fra rodmappe:

npm installer - gem - dev babel - core babel - loader babel - preset - es2015

For at bruge **React** har man brug for yderligere to nodepakker. Reakt- og reakt dom packages skal rette din npm-start.

npm install --save react react-dom

i *src/index.js* kan man implementere første hook in i react verden.

*src/index.js:*

import React from 'react';

import ReactDOM from 'react-dom';

const title = 'My Minimal React Webpack Babel Setup';

ReactDOM.render(

<div>{title}</div>,

document.getElementById('app')

);

module.hot.accept();

Explain, using examples, about Class Components, versus pure JavaScript functions in React, and when to use them.

En JavaScript-funktion er en blok kode designet til at udføre en bestemt opgave og en JavaScript-funktion udføres, når "noget" kalder det, en event opstår eller automatisk.

var x = myFunction(4, 3);        // Function is called, return value will end up in x  
  
function myFunction(a, b) {  
    return a \* b;                // Function returns the product of a and b  
}

Komponenterne har egenskaber og metoder til at styre deres udseende og bevægelser.

**function component(width, height, color, x, y) {  
    this.width = width;  
    this.height = height;  
    this.x = x;  
    this.y = y;   
    ctx = myGameArea.context;  
    ctx.fillStyle = color;  
    ctx.fillRect(this.x, this.y, this.width, this.height);  
}**

Normalt kan du definere en React komponent som følger :

|  |
| --- |
| const List = React.createClass({ |
|  | render: function() { |
|  | return (<ul>{this.props.children}</ul>); |
|  | } |
|  | }); |

Eller ved hjælp af ES6-klassesyntaxen:

|  |
| --- |
| class List extends React.Component { |
|  | render() { |
|  | return (<ul>{this.props.children}</ul>); |
|  | } |

Explain the purpose of *Client Side Routing* in a SPA

en af ​​de mest nødvendige dele for at opbyge SPA er routeren. Den del, der ved, hvordan man justerer indholdet i adresselinjen og meddeler resten af ​​systemet til URL-ændringer.

Der er flere opgaver af client-siderouting: man downloader mindre data for at vise nyt indhold, man kan genbruge DOM-elementer, vise indlæsningsmeddelelser til bruger osv. webapps, der genererer DOM'en på serversiden, er meget lettere at gennemgå (ved søgning motorer), hvilket gør SEO optimering nemmere.

Explain, using an example of your own, the basic “building blocks” in react-router:

<http://www.mycodesmells.com/post/setting-up-react-with-router>

Ruter bygget med react-router består af to kerne komponenter, som du skal huske:

* Route er den mest almindelige, og den indeholder en grundlæggende definition af hvilken vej der skal udføres af hvilken komponent
* IndexRouteer en genvej til at definere en rute, der skal monteres på *skråstreg* , det er et indeks for den givne vej

herboprette en simpel ruters struktur, hvor vi ønsker at have Tre visninger: Login Page , Logout Page og Welcome Page

// Common for all components files

import React from 'react';

// src/components/Login.js

const Login = () => <div>Login</div>

export default Login;

// src/components/Logout.js

const Logout = () => <div className="logout">Logout</div>

export default Logout;

// src/components/Welcome.js

const Welcome = () => <div className="welcome">Welcome</div>;

export default Welcome;

For at lave en fælles skabelon på begge sider for indloggede brugere, skal vi oprette en anden visning, som kan kaldes App. Det er lidt anderledes, da vi skal fortælle React hvor skal vi lave indlejrede ruter, passeret som childrenejendom. Vi kan stadig bruge kortere React komponent syntaks og bruge det faktum, der propser bestået som et første argument:

// src/components/App.js

const App = ({ children }) => <div><h3>Hello, Cool User!</h3>{ children }</div>

export default App;

Sidst men ikke mindst skaber vi en Rootkomponent (med en meget lignende fremgangsmåde som for App):

// src/components/Root.js

const Root = ({ children }) => <div><h1>Root level</h1>{ children }</div>;

export default Root;

Når vi har vores komponenter klar til brug, kan vi begynde at definere ruter. Vores Rootkomponent vil faktisk fungere som en rod:

<Route path="/" component={ Root }>

...

</Route>

Derefter kan vi definere to børneleder: Appog henholdsvis Logoutindlogget og anonyme brugere:

...

<IndexRoute component={ App }>

...

</IndexRoute>

<Route path="login" component={ Login } />

...

Da vi ønsker at betjene vores ansøgning på en skråstrejse , definerer vi vores Apprute som en IndexRoute. Hvis brugeren tilføjer logindet, bliver de omdirigeret til vores login side .

Endelig tilføjer vi to sidste ruter for logget bruger:

...

<IndexRoute component={ Welcome } />

<Route path="logout" component={ Logout } />

...

I sidste ende skal vores rute definition se sådan ud:

<Router history={ browserHistory }>

<Route path="/" component={ Root }>

<IndexRoute component={ App }>

<IndexRoute component={ Welcome } />

<Route path="logout" component={ Logout } />

</IndexRoute>

<Route path="login" component={ Login } />

</Route>

</Router>

Explain what is required to use react-router with a *create-react-app* project built from scratch

Explain, using examples, how JavaScript array methods, like filter, map and (reduce) are used to generate dynamic HTML structures (tables, ul's etc.), and explain about React Keys.

<https://github.com/cph-cs241/Pd5_REACT>

https://github.com/KongBoje/Hand-in-5-React