1 SPA

What are the benefits of a browser based application? A browser based application (web application) has various benefits:

- You can work from anywhere at anytime
- It is platform independent (even mobile)
- No software update nor installation => easy maintenance
- Software can provided as a Services (SaaS)
- Can be cross-compiled to different ecosystems
  - Client app: electron.io
  - Moile app: Native Script / Ionic
  - Server app: "Universal" Compilation

What are the liabilities of a browser based application? Browser based applications do not have only benefits (What are the benefits of a browser based application?) but also downsides, such as:

- no data sovereignty
- limited / restricted hardware access (no OS access, may be less efficient)
- Search Engine Optimization (SE must execute JS)
- More complex deployment strategies
- Overhead

What is a Single Page Application? An SPA is a special kind of Web apps.

> A Single Page Application (SPA) is a web site [...] that fits on a single web page with the goal of providing a user experience similar to that of a desktop application. In an SPA, either all necessary code [...] is retrieved with a single page load or the appropriate resources are dynamically loaded and added to the page as necessary.
>
> SPAs use AJAX and HTML5 to create responsive Web apps, without constant page reloads. -- Wikipedia

The traditional web application architecture

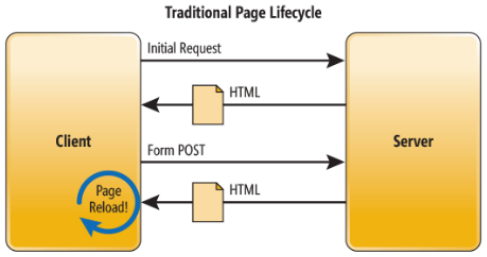

Figure 1: Traditional Architecture

The SPA architecture

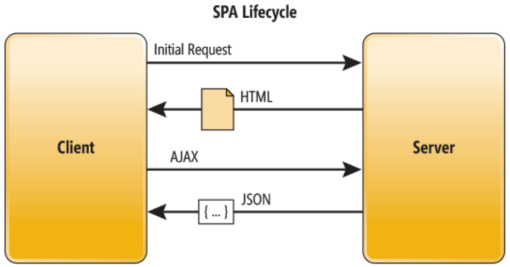

Figure 2: SPA Architecture

What are characteristics of an SPA? An SPA has the following properties:

- Plan HTML5 / CSS and JavaScript
  - no plugins like SilverLight or Flash
- no page reloads
- Working Back-Button
- Bookmarkable Links
- Provides (limited) offline functionality
- Uses (REST)-API services for data access

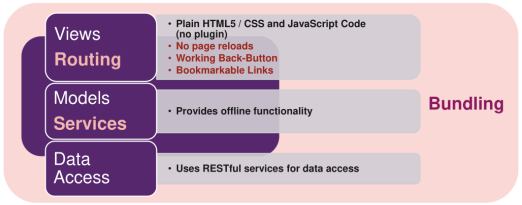

Figure 3: Logical Overview of an SPA

When would you prefer an SPA to a classic web application from the customer's point of view?

> As soon as a desktop (native) app with a similar user experience is required. The page feels like an application. An SPA also offers more options for complex web applications with lots of animations/-graphical elements.

What do you see as the technical benefits of an SPA?
The server application is separated from the display by a structured interface (e.g. REST / ODATA / WSDL). This opens up various advantages:

- Seperation of Concerns
- Better maintainability of the client code
- Division into different teams / competence centers

What does a typical layering in an SPA look like? The Views are connected using a routing in the browser (no new request to the server). The Business Logic provides data over servies and only the data layer will communicate directly with the server.

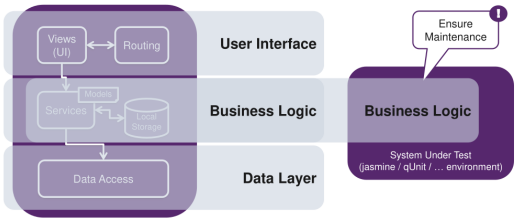

Figure 4: Layering in SPA

Why do we use bundling in an SPA?
An SPA may consist of many single JS files, which may or may not dependt on each other. To include them manually in your HTML is error pronce and tedious.
With bundling we achive the following things:

- All JS code must be delivered to the client over potentially metered/slow networks
- Bundling and minifying the source leads to smaller SPA footprint (e.g. using Tree Shaking)
- Bundling leads to a reliable dependency management
- Usage of pre and post processors during bundling

The initial footprint caused by bundling can be reduced by loading dependent modules on-demand.

2 React

What is JSX? JSX is an extension to JavaScript. It is used to write markup for an SPA. The JSX is transpiled during building into standard ECMAScript

> JSX is an XML-like syntax extension to ECMAScript without any defined semantics. It's NOT intended to be implemented by engines or browsers. It's NOT a proposal to incorporate JSX into the ECMAScript spec itself. It's intended to be used by various preprocessors (transpilers) to transform these tokens into standard ECMAScript. -- facebook.github.io/jsx/

How is JSX desugared in React?

How do you make props available for all child components? Some props must be availabel in all components (e.g. color scheme). It does not scale well, if you have to pass all props from the root component. To solve this problem, we can use contexts. However, you should only use contexts for read-only variables and limit the number of different contexts.

```
const themes = {
  light: {
    foreground: "#000000",
    background: "#eeeeee",
  },
  dark: {
    foreground: "#ffffff",
    background: "#222222",
  },
};
const ThemeContext = React.createContext(themes.

function App() {
  return (
    <ThemeContext.Provider value={themes.dark}>
      <Toolbar />
    </ThemeContext.Provider>
  );
}

function ThemedButton() {
  const theme = useContext(ThemeContext);
  return (
    <button style={{
              background: theme.background,
              color: theme.foreground }}>
      {" "}I am styled by theme context!{" "}
    </button>
  );
}
```

How does Redux work? The state in Redux is represented as trees of objects. The tree is immutable. When you change something in the tree, a new tree will be created (Functional Programming).

A state action is communicated using a *Redux Action*. A *Reducer* takes the action and the current state an applies the action on the state to generate a new state.

3 Angular

4 ASP.NET