1 Introduction SPA

1.1 Browser-based Applications

Benefits

- Work from anywhere, anytime
- Platform independent, including mobile
- No software update, no application, easy maintenance
- Software can be provided as a service (SaaS - pay as you go)
- Code separation

Liabilities

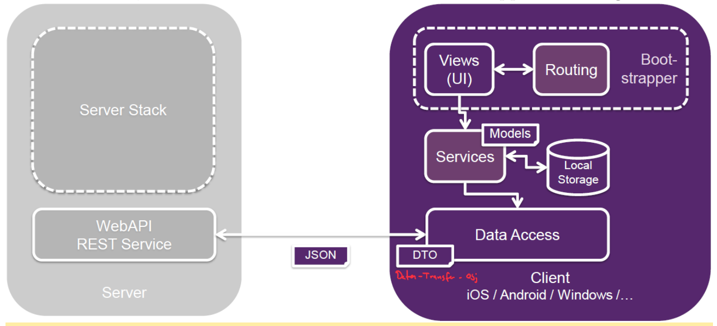
- No data sovereignty (Datenhoheit)
- Limited/restricted hardware access
- SEO - Search engines must execute JavaScript
- More complex deployment strategies

1.2 SPA

A website that fits on a single web page with a user experience similar to that of a desktop application. All code is retrieved with a single page load or resources are dynamically loaded. SPAs use AJAX and HTML5 to create responsive Web apps, without constant page reloads.

1.2.1 Architecture

Website interacts with user by rewriting parts of the DOM. After first load, all interaction with the server happens through AJAX.



1.2.2 Bundling

All JS code must be delivered to the client over potentially slow networks. Bundling and minifying the source leads to smaller SPA footprint. Larger SPAs with many modules need a reliable dependency management. Initial Footprint can be reduced by loading dependent modules on-demand.

1.2.3 WebPack as Bundler

Entry: Start, follows the graph of dependencies to know what to bundle.
Output: Tell webpack where to bundle your application.
Loaders: Transforms these files into modules as they are added to your dependency graph.
Plugins: Perform tasks like bundle optimization, asset management and injection of env variables.
Mode: Enable built-in optimization mechanisms.

1.3 Routing

- Completely on client-side by JS
- Navigation behaves as usual
- Browser needs to fake the URL to change and store page state
- *window.history.pushState*

1.4 Dependency Injection

Benefits

- Reduces coupling between consumer and implementation
- Contracts between classes are based on interfaces
- Supports the open/closed principle
- Allows flexible replacement of an implementation

1.4.1 Decorators

- Provide a way to add annotations / meta-programming syntax
- Can be attached to a class declaration, method, accessor, property or parameter
- Widely used in Angular