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

What are the liabilities of a browser based application? Browser based applications do not have only benefits 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 are characteristics of an SPA? An SPA has the following properties:

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

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.

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. The Business Logic provides data over servies and only the data layer will communicate directly with the server.

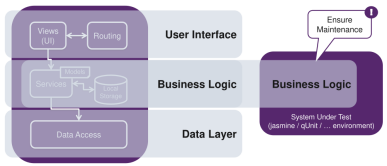

Figure 1: Layering in SPA

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

How do you make props available for all child components? However, you should only use contexts for read-only variables and limit the number of different contexts.

- `React.createContext()`
- `<TContext.Provider value={value}></TContext.Provider>`
- `useContext`

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.

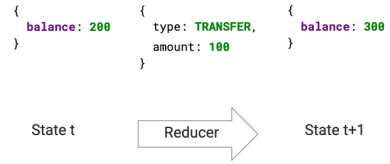

Figure 2: Action Reducer State change

3 Angular

Angular Architectural Parts ngModules, directives, components, templates, metadata, services

Angular Module Declaration

- declarations: belongs to this module
- exports: Subset of declarations
- imports: imported into this module
- providers: Creators of services, contributes to the global DI container
- bootstrap: Main application view (only in root module)

Binding Syntax

```
<!-- Two Way Binding -->
<input
  type="text"
  [(ngModel)]="counter.team">

<!-- One Way (View to Model) -->
<button
  (click)="counter.eventHandler($event)">
```

```
<!-- One Way (Model to View) -->
<p>{{counter.team}}</p>
```

Listing 1: Bindings variants in Angular

Directives Declared as a TypeScript class with an `@Directive()` function decorator. **Structural** directive: modifies structure of DOM (`*ngIf` / `*ngFor`), **Attribute** directive: alters appereance / behavior of an existing element (`[ngStyle]` / `[ngClass]`)

Services The `@Injectable` decorater is used to register the service in the DI container. `@Injectable({ providedIn: 'root' })export class CounterService {}`

Forms

```
<form #f="ngForm"
  (ngSubmit)="doLogin(f)">
</form>
```

Listing 2: Angular Form Markup

```
@Component({})
export class SampleComponent {
  public doLogin(f?: NgForm) {
    if (f?.form.valid) {}
    return false;
  }
}
```

Listing 3: Angular Form Logic

Routing `.forRoot()`: declare routes on root level, `.forChild()` declaring sub-routings

Module Type / Example Architecture Root / App Module, Feature Modules, Shared Module, Core Module

RxJS - Observable Types Hot Observables, Cold Observables (triggerd on subscription)

Pipes Pure Pipes (change to the input expression), Impure Pipes (executed on every change detection cycle)

4 ASP.NET

PWA PWA use modern web APIs along with traditional progressive enhancement strategy (fells like native app)

For what is Firbase used? To develop a PWA we used the following features from Firebase:

- Cloud Firestore / Realtime (DB)
- Cloud Functions (Serverless Functions)
- Authentication
- Hosting / Cloud Storage

Front Controller The Front Controller is the Entry Point and executes logging and routing (actions that are required for all incoming requiest). After that, the front controller forwards the request the the responsible controller.

Middleware Middleware is a "function block" and has a certain task. When a middleware has finished its task it will call the next following middleware or finishes the request.

- `app.Use()`: non terminating
- `app.Map()`: mapping
- `app.Run()`: terminating
- `app.UseMiddleware<>()`: register class middleware

Context contains all information for the request as well for the response.

Dependency Injection - Lifetime Register services with different lifetimes:

- Transient, are created every time when used
- Scoped, are created once for every request (used most often)
- Singelton, it's just a singelton (only for Readonly, configs, options)

Register a transient service: `builder.Services.AddTransient<>()`

Captive dependency The term "captive dependency" refers to the misconfiguration of service lifetimes, where a longer-lived service holds a shorter-lived service captive.

- Transient requires Singelton -> OK
- Singelton requires Transient -> Not OK
  - When the singleton is used a 2nd time (e.g. parallel) it uses the same reference to the transient service.
  - Transient services are normally **NOT** thread-safe and an error can occuer when used in a not thread safe context

Web Assembly File Type

- *.WASM (WebAssembly)
  - Compiled Web Assembly
  - Can be transformed back to WAT
- *.WAT
  - Text Based Web Assembly
  - Can be compiled to WASM
- Web Assembly
  - Is a stack machine (similar to the JVM)
  - Evaluation Stack

Key Concepts

- Module
  - Represents a WebAssembly binary that has been compiled by the browser into executable machine code
  - Stateless
  - Import / Export
- Memory
  - shared memory section between JS and web assembly
- Table
  - A resizable typed array of references (e.g. to functions) that could not otherwise be stored as raw bytes in Memory (for safety and portability reasons).
- Instance
  - A Module paired with all the state it uses at runtime including a Memory, Table, and set of imported values.