Spaß mit JavaScript

Rückblick

SPA

Single and ready to mingle

Die gesamte Applikation wird in einer einzigen Seite dargestellt. Teilbereiche der Applikation werden einzeln ausgetauscht.

Routing

Wo darf's hingehen?

Der Router dient zur Navigation zu unterschiedlichen "Seiten" in einer SPA, sowie zum Abfangen von ungültigen Seitenanfragen.

Translator

Der Polyglot

Die Translation Engine ermöglicht uns die Auslagerung von Ressourcen zur Übersetzung.

Template

Der Gerät

Die Template Engine ermöglicht uns, statische Markup-Templates vorzubereiten, die dynamisch befüllt werden.

Rückblick

Databinding

Einfach unzertrennlich

Die Daten im Model und die sichtbaren Daten in den Views werden immer miteinander synchronisiert.

MV(C)

Most Valuable Concept

Das Model managed die Daten. Die View präsentiert sie. Der Controller regelt die Kommunikation zwischen den beiden.

Async JS

I promise…

Jedes Promise liefert verlässlich ein **Ergebnis** oder einen **Fehler**. Das kann ich euch versprechen!

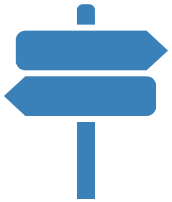
API

Die Datenspeisekarte

Regelt den Austausch von Daten über mehrere Systeme hinweg. Meist **REST** oder **SOAP**.

Routing in Single Page Applications

Der Weg ist das Ziel!



Router - Was ist das?

Navigation in SPA

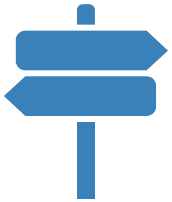
SPA hat ja eigentlich nur eine URL.
Router ermöglicht Navigation in SPA

Ist quasi der View-Manager

Sucht die **passende View** zur Anfrage raus und zeigt diese an.

Unzertrennlich

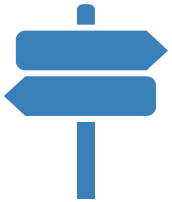
Routes und Router bilden gemeinsam das Konzept des "**Routing**"



Router in echt

- Hat ein **Array mit allen Routen**
- Hat zusätzlich **Spezialrouten** (Startseite, 404-Seite etc.)

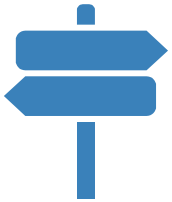
```
export default class KWM_Router{
   constructor(views){
       this.routes = views;
       this.homeRoute = views[0];
       this.init();
   }
```



Router in echt

- Router wird mit **Listener** für das Event **hashchange** ausgestattet.
- Wenn sich **Hash** ändert, tritt der Router mit "**changeView()**" in Aktion.

```
init(){
   window.removeEventListener('hashchange', this.changeView);
   window.addEventListener('hashchange',this.changeView.bind(this));
   this.changeView();
}
```



Routes - Was ist das?

Zielpunkte

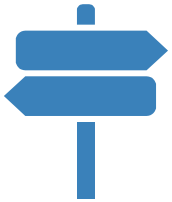
Routes sind **Zielpunkte** in unserer App, **fast so wie URLS**

History API

Kann über **window.history** angesteuert werden.

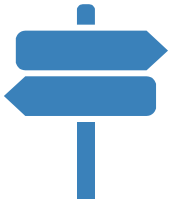
Hash Based

Kann über **window.location.hash** angesteuert werden. Liefert alles **hinter #** in der URL



kwm.JS Routes

- Wir verwenden **Hash-Based Routing**
- z.B. www.kwmjs.at/#**/login**
  - Router fängt "/login" ab und ruft die dazu passende View auf.
- Wie in URLs, können auch **GET-Parameter** übergeben werden
  - z.B. www.kwmjs.at/#/login**?lang=en**



kwm.JS Routes

- Routen haben einen Slug (z.B. **/login** oder **/cats** oder **/anything**).
- Der Slug wird mit der aktiven URL im Browser verglichen.
- Bei Übereinstimmung ist diese Route gerade aktiv.

```
export default class KWM_Route{
   constructor(slug, init){
       this.slug = slug;
       this.init = init;
   }
}

isActive(){...}
```

The window Object

Ein Objekt sie alle zu binden…

Window Object

Offenes Fenster

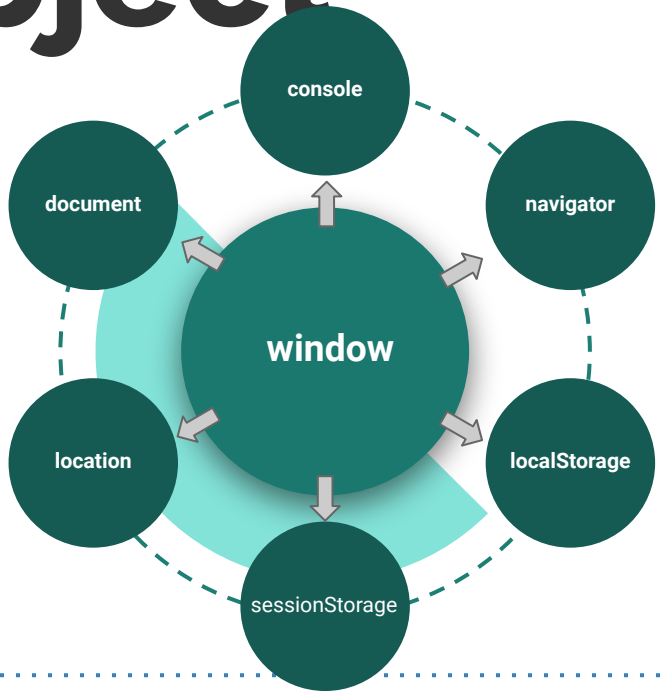
Ein window Objekt repräsentiert ein offenes Browser-Fenster

Mehrere Rahmen

Enthält ein Dokument mehrere Frames (bspw. <iframe>), so enthält jeder iframe ebenfalls ein eigenes window Objekt.

Window Object

- Globales Objekt
- Immer vorhanden
- Erweiterbar
- Unique



Software Flavor

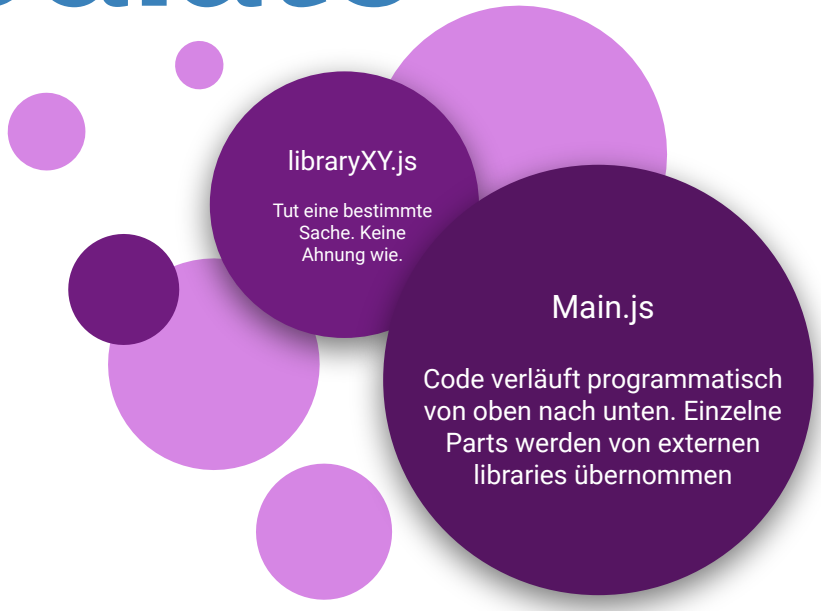
Schmeckt nach Freedaten-Suppe!

Beginner

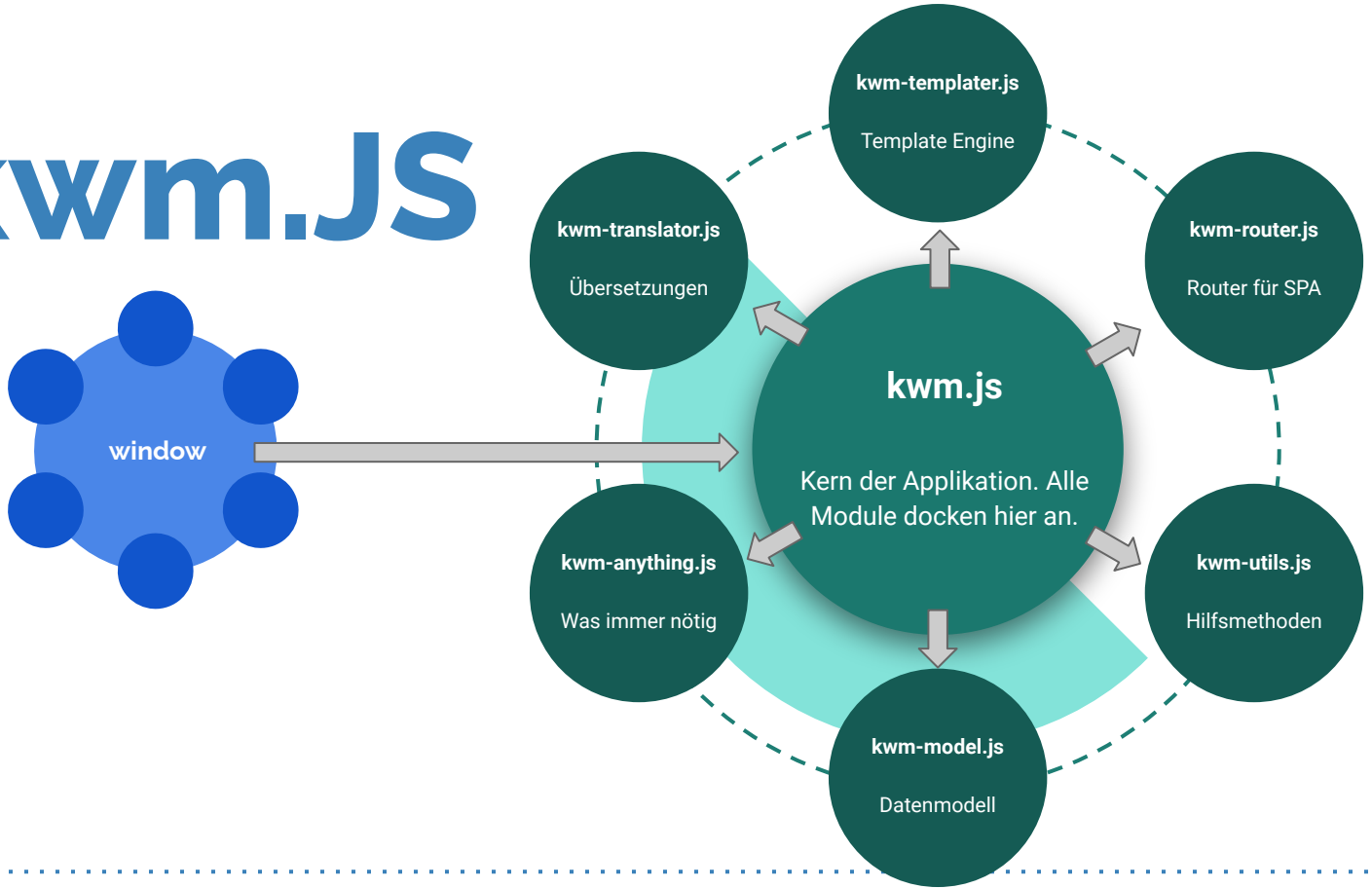
Main.js

Alles ist hier drin. Es gibt nur dieses File.

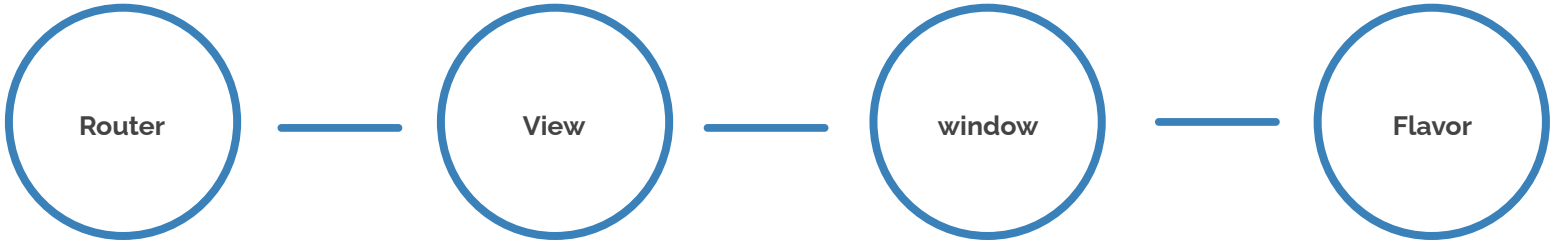
Intermediate



kwm.JS


window
kwm.js
Kern der Applikation. Alle Module docken hier an.
kwm-templater.js
Template Engine
kwm-translator.js
Übersetzungen
kwm-router.js
Router für SPA
kwm-anything.js
Was immer nötig
kwm-utils.js
Hilfsmethoden
kwm-model.js
Datenmodell

Auf einen Blick



Routen-Manager

Router hat Überblick **über alle Views** und **sucht die richtige** View raus, wenn die Route verändert wird.

Die Präsentation

Jede View ist mittels **Slug** einer Route zugewiesen. Jede **View bestimmt selbst**, was passiert, wenn sie gerendert wird.

Sehr durchsichtig

Globales Objekt, an das **mehrere Komponenten** drangehängt sind. Besteht für jedes Browser-Fenster.

Geschmackssache

Bezeichnet den **Stil eines Frameworks.** Nicht nur, wie es **geschrieben** ist, sondern vor allem auch, wie es zu **verwenden** ist.