DBWT2 Praktikum Meilenstein 3:

A1.)

Google Photos API:

Mit der API kann eine App „Google Photos“-Funktionalitäten einbinden, welche das lesen, schreiben und teilen von Bildern über die Google Services ermöglicht.

REST Prinzipien:
- Client-Server Struktur
- Zustandslos (?)
- Caching möglich, wegen einzigartiger URI pro API-Ressource
- Mehrschichtig (?)

Beispiel für Aufruf:
https://www.googleapis.com/apiName/apiVersion/resourcePath?parameters

Richardson Modell:
- Level 0: Single Service Endpoint → ja, da in einem Aufruf ein bestimmter „apiname“ mit Versionsnummer angegeben sein muss – dadurch entsteht für jeden Aufruf ein bestimmter und einzigartiger Endpunkt
- Level 1: gilt auch, da jede Resource über einen „resourcePath“ identifizierbar ist
- Level 2: GET, POST, PUT, DELETE unterstützt
- Level 3: Der „resourcePath“ lässt darauf schließen, dass Level 3 gegeben ist, allerdings konnten wir keine Beispiele finden, die das bestätigen

Versionierung:
Wird ebenfalls über den Aufruf als „apiVersion“ übergeben.

Twitter API:

https://developer.twitter.com/en/docs/api-reference-index

Mit der API kann auf sämtliche Twitter-Funktionen zugegriffen werden, wie z.B. User suchen, diesen folgen, Accounts verwalten, oder eigene Tweets verfassen.

REST Prinzipien:
- Client-Server Struktur
- Zustandslos (?)
- Caching möglich und erwünscht
- Mehrschichtig (?)
- einheitliche Schnittstellen

Anmerkung: Google nennt die API „RESTful“, daher sind die beiden Fragezeichen wahrscheinlich erfüllt, allerdings konnten wir in der Doku nicht die Stellen finden, an denen diese explizit beschrieben werden.

Beispiel für Aufruf:
'https://api.twitter.com/1.1/users/show.json?screen_name=twitterdev'

Richardson Modell:
- Level 0: Single Service Endpoint → ja, da in einem Aufruf ein bestimmter „apiname“ mit Versionsnummer angegeben sein muss – dadurch entsteht für jeden Aufruf ein bestimmter und einzigartiger Endpunkt
- Level 1: gilt auch, da jede Resource über einen „resourcePath“ identifizierbar ist
- Level 2: GET, POST, PUT, DELETE unterstützt
- Level 3: nicht in der API erwähnt, daher möglicherweise nicht unterstützt

Versionierung:
Wird ebenfalls über den Aufruf als übergeben.

A6.)

- V8 (Google) ist meistbenutze JavaScript Engine. Wird in Google Chrome benutzt.
- SpiderMonkey wird von Mozilla für Firefox verwendet.
- JavaScriptCore wird von Apple für den Safari Browser verwendet.

Array.prototype.flat(depth):

- Gibt ein verschachteltes Array als ein „flaches“ Array zurück
    - [1, 2, [3, 4]] → [1, 2, 3, 4]

Laut https://test262.report/features/Array.prototype.flat/built-ins/Array/prototype/flat?date=2019-08-23 von allen drei Javascript-Engines unterstützt.

Object.fromEntries():

Macht eine beliebige Key-Value-Zuordnung (Map, Liste, Array, etc.) zu einem Objekt.

Ebenfalls laut https://test262.report/features/Object.fromEntries von allen JavaScript-Engines außer Chakra (Internet Explorer) unterstützt.

Let a = () => {…}; a.toString():

Nicht ganz sicher, wie gemeint, aber Code wie

```
let a = {
        toString() {
            return „a“;
        }
    }
Console.log(a.toString());
```

Funktioniert auf allen JavaScript-Engines - vielleicht Aufgabe missverstanden?

A9.)

Motivation:
Gestensensitive(bewegliche) Apps nachbauen – allerdings mit Webelementen statt IOS/Android

Backbone:
Sehr restriktiv bei der Codegenerierung. Experimente und „freie“ Projekte brauchten mehr Flexibilität

Laravel:
Ceo von Laravel: „React zu schwierig“. Meisten andere Umgebungen haben sehr komplizierte Kompilationsvorgänge und Toolkits die man benutzen lernen muss. Vue sehr einfach – auch ohne externe Tools

Wechsel Angular → Vue:
Etwas schwammige, größtenteils emotionale Begründungen, z.B. dass sich Vue besser anfühlt, da es nicht so „coperate“ ist und es sich besser anfühlt mit einer Art „passion project“ einer einzelnen Person zu arbeiten.

Erfolg:
Keine Sprachbarriere für chinesische Programmierer, die Doku in eigener Sprache suchen (da von Gründer selbst geschrieben und übersetzt). Dadurch auch eine Art „Fanbase“ in China.