Web-basierte Anwendungen

Praktikumsaufgabe 6: Umgang mit Daten & JSON, Nutzung aktueller JavaScript-Bibliotheken

Studiengänge AI (4140) & WI (4120)

JS-Bibliotheken: Hintergrund

- Hintergrund:
  - Die WBA-Entwicklung ist inzwischen weit fortgeschritten. Für zahlreiche – auch komplexe – Routinevorgänge auf Client-Seite gibt es inzwischen leistungsfähige JS-Bibliotheken
  - Die Eigenentwicklung von JS tritt daher zugunsten der Auswahl und **Nutzung** solcher **Bibliotheken** immer mehr in den Hintergrund
  - In dieser Übung soll daher mit zwei (auf jQuery beruhenden) derartigen Bibliotheken gearbeitet werden. Die eigene JS/CS-Entwicklung reduziert sich dabei auf die **Integration und Bereitstellung von Daten**.
  - Als Datenformat wollen wir dabei **JSON** anwenden
    - Das schließt die Beschäftigung mit **Arrays und Hashes** ein

JS-Bibliotheken: Ziele

- Ziele:
  - Wir erzeugen den neuen Controller „pr06“ mit einen View „graphs“
    - Verwenden Sie wieder das Standard-Layout „application“ (voreingestellt)
    - Zur Erinnerung:

```
$ rails generate controller pr06 graphs
```

  - Der View soll eine Seite mit zwei Inhalten erhalten
    - Eine Tabelle mit den im JSON-Format bereitgestellten Daten über ökonomische Eckdaten von 27 EU-Ländern (Stand: Jahr 2010)
    - Eine Grafik zur Visualisierung jeweils einer ausgewählten Datenspalte

JS-Bibliotheken: Ziele

- Besonderheiten, *client*-seitig mit den JS-Bibliotheken zu lösen
  - Tabelle
    - Sortierung der Spalten (durch Anklicken der Spaltenüberschriften)
    - Filterung (Eingabe eines Strings → Anzeige nur der „passenden“ Zeilen)
    - Paginierung (Anzeige nur der ersten 10/25/50 Zeilen; evtl. Weiterblättern)
  - Grafik
    - Kuchendiagramm („*pie chart*“) für die Spalten, deren Angaben sich zu einem Gesamtwert für die EU addieren lassen
    - Säulendiagramm („*column chart*“) für die anderen Spalten (wo sinnvoll)
    - Jeweils mit Legende und ggf. sinnvoll beschrifteten Achsen

JS-Bibliotheken: Lösungsweg

- Woche 1:
  - Einrichtung eines sehr einfachen HTML-Grundgerüsts ohne Daten
  - Dynamische Erzeugung von HTML mit JS/CS:
    - Einlesen der JSON-Daten (AJAX-Technik),
    - dynamischer Aufbau einer Tabelle mit DOM- bzw. jQuery-Funktionen
- Woche 2:
  - Installation und Benutzung zweier JS-Bibliotheken
  - Tabelle
    - „Animierung“ durch Herstellung einer Verbindung mit JS-Bibliothek
    - Ggf. Installation mitgelieferter CSS-Dateien, I18N-Konfiguration
  - Grafik
    - Erzeugung einer Grafik (Basis HTML5, Element „canvas“!) mithilfe einer darauf spezialisierten JS-Bibliothek
    - Ggf. Installation mitgelieferter CSS-Dateien
    - Die Grafik soll erst entstehen bzw. jeweils erneuert werden, wenn Anwender eine der Datenspalten-Überschriften anklicken
    - Die Daten der Grafik extrahieren Sie dazu per DOM oder jQuery aus der jeweils angezeigten Tabelle, sodass sich die Grafik automatisch nach der gerade angezeigten Zeilenzahl und Sortierung der Tabelle richtet!

JS-Bibliotheken: Werkzeuge & Tipps

- Debugging-Werkzeuge:
  - Wie bereits in Aufgabe 05 sollten Sie Firebug (Firefox) o.ä. eingebaute Debugging-Werkzeuge verwenden, um JS-Fehler zu bemerken und zu analysieren
    - Ein/Ausschalten meistens mit Taste „**F12**“
  - Mit dem Einbau von alert-Aufrufen lässt sich leicht prüfen, ob Ihr Code auch wie erwartet ausgeführt wird und welche Werte bestimmte Variablen an der Stelle des alert-Aufrufs haben. Beispiele:

```
alert("Hier");  // Javascript
alert "Wert von 'my_var': " +  my_var  # Coffee
```

- Stimmt Ihr „Timing“?
  - JavaScript-Programmierung zeichnet sich oft aus durch Reagieren auf oder Auslösen von Ereignissen. Überprüfen Sie sorgfältig, ob erwartete Events auch eintreffen, und wann und wo Sie diese erwarten. Machen Sie sich den Ereignis-Ablauf klar, einschließlich des Seitenaufbaus bzw. der Initialisierungen. Testen Sie mit alert-Boxen!
  - Die jQuery-Methode zum Abholen der JSON-Daten arbeitet mit einem asynchronen AJAX-Aufruf. Das hat Folgen für Sie als Entwicker/in! Machen Sie sich klar, wann Sie mit den Daten weiterarbeiten können.

JS-Bibliotheken: Organisatorisches

- Zeitlicher Ablauf
  - Die Aufgabe erstreckt sich über zwei Wochen.
  - Abgabe & Abnahme finden erst später statt, aber Sie sollten im eigenen Interesse nach einer Woche den Tabellenteil fertig haben!

- Aufteilung
  - Woche 1: Tabelle, DOM/jQuery
    - Einlesen der JSON-Daten
    - Aufbau der Tabelle mittels DOM/jQuery-Methoden
  - Woche 2: Tabelle mit Interaktionen ausstatten, Grafik erzeugen
    - Einbau der DataTables-Bibliothek für Paginierung, Filterung & Sortierbarkeit
    - Generierung der benötigten Daten für die Grafik (DOM & jQuery nutzen!)
    - Erzeugung einer Grafik mit der jqChart-Bibliothek
    - Programm-Logik, die Klicks auf die Spaltenüberschriften bemerkt und je nach Spalte die „richtige“ Grafik erzeugt

- Bewertung
  - 2 Punkte - für beide Teile gibt es je einen Punkt
  - Hinweis: Der erste Teil ist der einfachere, dennoch erhält er 50% der Punkte. Bei Zeitnot empfiehlt es sich daher, zumindest diesen Teil anzufertigen.

Woche 1:

Aufbau der HTML-Tabelle aus JSON-Daten

JS-Bibliotheken: Die Tabelle

- Vorbereitungen
  - Wechseln Sie in den Basisorder „wba1“ Ihres Rails-Projekts

    ```
    $ cd mein/pfad/zu/wba1
    ```

  - Kopieren Sie die bereitgestellte JSON-Datei vom Fileserver in den Ordner **public** :

    ```
    $ scp login1.cs.hs-rm.de:~werntges/lv/wba/06/eu-data.json ./public
    ```

  - Wechseln Sie in „./app/views/pr06“. Editieren Sie die Datei graphs.html.haml.
    - Entfernen Sie den Platzhalter-Code
    - Erzeugen Sie als erste Zeile folgendes:

      ```
      = javascript_pack_tag "pr06"
      ```

    - Erzeugen Sie dann eine Überschrift (h1) „Kennzahlen der EU-Länder (2010)“
    - Erzeugen Sie eine leere HTML-Tabelle („table“) mit ID „pr06table“ und den Unterelementen „thead“ (ID: „pr06thead“) und „tbody (ID: „pr06tbody“)
  - Fügen Sie an:
    - Eine Unter-Überschrift (h2) „Ausgewähltes Diagramm“
    - Zwei ineinander geschachtelte div-Elemente.
    - Das innere div-Element erhält die ID „pr06chart“

JS-Bibliotheken: Die Tabelle

- Option 1 (falls Sie CS installiert hatten): Entwickeln Sie nun CS-Code
  - Wechseln Sie in den Ordner „app/javascript/packs“. Öffnen Sie Datei „pr06.coffee“, fügen Sie den CS-Code dieser Aufgabe hier ein. Beim „ready“-Event soll alles Folgende wirksam werden:

```
$(document).ready ->
  # Ihr Code…
```

  - Laden der JSON-Datei
    - Verwenden Sie das folgende jQuery-Code-Fragment:

```
$.getJSON '/eu-data.json'
.done (data) ->
  $.each data, (key, value) ->
    # Ihr Code zum Verarbeiten der EU-Daten …
.fail (d, textStatus, error) ->
  alert "getJSON gescheitert, Status: " +
        textStatus + ", Fehler: " + error
```

JS-Bibliotheken: Die Tabelle

- Option 2 (falls Sie Javascript bevorzugen): Entwickeln Sie nun JS-Code
  - Wechseln Sie in den Ordner „app/javascript/packs“. Öffnen Sie Datei „pr06.js“, fügen Sie den JS-Code dieser Aufgabe hier ein. Beim „ready“-Event soll alles Folgende wirksam werden:

```
$(document).ready( function() {
  // Ihr Code…
)};
```

  - Laden der JSON-Datei
    - Verwenden Sie das folgende jQuery-Code-Fragment:

```
$.getJSON('/eu-data.json')
.done(function(data) {
  $.each(data, function(key, value) {
    // Ihr Code zum Verarbeiten der EU-Daten …
  });
})
.fail(function(d, textStatus, error) {
  alert("getJSON gescheitert, Status: " +
        textStatus + ", Fehler: " + error);
});
```

JS-Bibliotheken: Die Tabelle

- Entwickeln Sie JS- bzw. CS-Code
  - Schreiben Sie eine Funktion **`append_row`**. Sie soll in die Tabelle eine neue Zeile einfügen. Übergabewerte:
    - **parent** (Eltern-Element, entweder das thead- oder das tbody-Element)
    - **ary** (ein Array von Strings) mit den Zeilenwerten
    - **element_name** (soll ‚td' oder ‚th' annehmen)
  - Verwenden Sie dazu DOM- oder jQuery-Methoden (jQuery empfohlen)
  - Erzeugen Sie mit zwei **`append_row()`**-Aufrufen nun zwei Überschriften-Zeilen im thead-Element:
    - Übergeben Sie (den Inhalt von) **data['Spaltentitel']** und ‚**th**' für die erste Zeile
    - Übergeben Sie (den Inhalt von) **data['Einheit']** und ‚**td**' für die zweite Zeile
  - Erzeugen Sie mit weiteren **`append_row()`**-Aufrufen für alle Arrays außer „Titel", „Spaltentitel" und „Einheit" nun Zeilen im tbody-Element (‚**td**'-Elemente).
    - Ergänzen Sie hier den Wert von „key" als ersten Array-Wert (!)

JS-Bibliotheken: Die Tabelle

- Ende von Woche 1:
  - Ihr Browser sollte nun ungefähr folgendes Bild anzeigen:

Kennzahlen der EU-Länder (2010)

Ländername	Fläche	Einwohner	Währung	BIP	BIP	Staatsverschuldung	Neuverschuldung	Wachstum	Arbeitslosigkeit
-	km²	Mio.	-	Mrd. Einh.	Mrd. EUR	% BIP	% BIP	%	%
Belgien	32545	10.8	EUR	352.3	352.3	96.8	4.1	2.1	8.3
Bulgarien	110994	7.6	BGN	70.5	36	16.2	3.2	0.2	10.2
Tschech. Rep.	78866	10.4	CZK	3670	145.9	38.5	4.7	2.4	7.3
Dänemark	43098	5.5	DKK	1746	234.7	43.6	2.7	2.1	7.4
Deutschland	357093	82.2	EUR	2499	2499	83.2	3.3	3.6	7.1
Estland	45227	1.3	EUR	14.5	14.5	6.6	0.1	3.1	16.9
Irland	70273	4.6	EUR	153.9	153.9	96.2	32.4	-1	13.7
Griechenland	131957	11.3	EUR	230.2	230.2	142.8	10.5	-4.5	12.6
Spanien	504645	46.7	EUR	1063	1063	60.1	9.2	-0.1	20.1
Frankreich	543965	62.6	EUR	1948	1948	81.7	7	1.6	9.7
Italien	301336	60	EUR	1549	1549	119	4.6	1.3	8.4
Zypern	9251	0.8	EUR	17.5	17.5	60.8	5.3	1	6.5
Lettland	64589	2.2	LVL	12.7	17.97	44.7	7.7	-0.3	18.7
Litauen	65301	3.3	LTL	94.6	27.4	38.2	7.1	1.3	17.8
Luxemburg	2586	0.5	EUR	41.6	41.6	18.4	1.7	3.2	4.5
Ungarn	93030	10	HUF	27120	98.4	80.2	4.2	1.2	11.2
Malta	316	0.4	EUR	6.2	6.2	68	3.6	3.7	6.8
Niederlande	41526	16.5	EUR	591.5	591.5	62.7	5.4	1.8	4.5
Österreich	83871	8.4	EUR	284	284	72.3	4.6	2	4.4
Polen	312685	38.1	PLN	1416	353.7	55	7.9	3.8	9.6
Portugal	92345	10.7	EUR	172.5	172.5	93	9.1	1.4	11
Rumänien	238391	21.3	RON	513.6	121.9	30.8	6.4	-1.3	7.3
Slowenien	20253	2	EUR	36.1	36.1	38	5.6	1.2	7.3
Slowakei	49034	5.4	EUR	65.9	65.9	41	7.9	4	14.4
Finnland	338144	5.3	EUR	180.3	180.3	48.4	2.5	3.1	8.4
Schweden	449964	9.3	SEK	3301	346.1	39.8	0	5.5	8.4
Ver. Königreich	242910	62	UKP	1454	1694	80	10.4	1.3	7.8

Ausgewähltes Diagramm

Bedingungen

- Abgabe
  - Nach zwei Wochen Laufzeit, also in Kalenderwoche 25
  - Jeweils vor Beginn Ihrer Praktikumsgruppe in der Abgabewoche
- Art des Leistungsnachweises
  - Zu vergeben: **2 Punkte + 0,25 SoPu.**
    - 1 Punkt für den Tabellen-Teil
    - 1 Punkt für den Grafik-Teil
    - 0,25 Sonderpunkte für die Internationalisierung der DataTables-Texte
  - **Einzel-Arbeit – keine Teams!**
  - Abgabe der Dateien:
    - Im Verzeichnis „wba1“ ausführen:

      Neu!

      ```
      $ rake webpacker:clobber
      $ rake log:clear
      $ rake tmp:clear
      $ cd ..
      ```

      Hinweis: Lösungen mit JavaScript und jQuery sind der Normalfall. CoffeeScript-Lösungen sind willkommen, weil sie Arbeit sparen und übersichtlicheren Code fördern. DOM statt jQuery und TypeScript sind zulässig – bei der Abnahme bitte darauf hinweisen.
    - Sie sind nun im Elternverzeichnis von „wba1“. Jetzt noch ausführen:

      ```
      # Ordner ./wba1 verpacken, ohne unnötige Unterordner:
      $ tar czf 06-wba1-<matnr>.tar.gz --exclude wba1/tmp \
            --exclude wba1/node_modules --exclude wba1/log  ./wba1
      ```

      Fortsetzungszeile

      Datei „06-wba1-<matnr>.tar.gz“ abgeben
    - Achten Sie auf die Entfernung unnötiger Daten – Dateigröße von ca. 350 kB ist ok. Dateien > 1 MB werden nicht angenommen!
  - Online-Abnahmegespräch / Online-Demo der korrekten Funktion / Code-Stichproben

Literatur

- **Zu CoffeeScript**
  - **https://coffeescript.org/**
    - Projekt-Homepage, mit Übersicht & vielen Code-Beispielen
    - Beachten Sie auch den Umgang mit Arrays [..] und Hashes {..} !
- **Zu jQuery**
  - **http://jquery.com/**
- **Zu DOM: Vorlesungsskript und darin genannte Quellen**
- **Zu DataTables**
  - **https://datatables.net**
    - Projekt-Homepage, mit Übersicht & vielen Code-Beispielen
    - Empfehlung: examples / „Zero configuration“
- **Zu jqChart**
  - **http://www.jqchart.com**
    - Projekt-Homepage, mit Übersicht & vielen Code-Beispielen
    - Empfehlung: „samples“

Anhang

- Eventuell korrigieren (i.d.R. nicht nötig):
  - Wechseln Sie in den Ordner „app/javascript/packs“. Öffnen Sie Datei „pr06.coffee“, ändern Sie den CS-Code zum Starten nach erfolgreichem Laden einer Seite in:

```
$(document).on "turbolinks:load", ->
  # Ihr Code…
```

statt

```
$(document).ready ->
  # Ihr Code…
```

- Hintergrund:
  - Wenn Rails seine Technik “turbolinks” zum beschleunigten Laden einer Seite verwendet, muss Event “ready” ersetzt werden durch “turbolinks:load”.
  - Beim ersten bzw. vollständigen Laden einer Seite (F5) verwendet Rails den normalen Ladeprozess, daher sollte “ready” hier genügen.