Web-basierte Anwendungen

Praktikumsaufgabe 5: JavaScript für HTML-Seiten - die Taschenrechner-Aufgabe

Studiengänge AI (4140) & WI (4120)

Taschenrechner: Hintergrund

- Hintergrund:
  - Wir verbessern und erweitern eine bestehende kleine JavaScript-Anwendung schrittweise
  - Dabei legen wir Wert auf eine klare Trennung der Bestandteile HTML, CSS und JS
    - Bei dieser Gelegenheit entfernen wir Code-Redundanzen
  - Wir nutzen die Gelegenheit, erste Versuche mit der JS-Bibliothek JQuery zu unternehmen
  - Nur optional:
    Zur Vereinfachung des Codierungsaufwands entwickeln wir nicht direkt JavaScript-Code, sondern CoffeeScript- oder TypeScript-Code

Taschenrechner: Vorbereitungen

- Erweitern Sie die Ergebnisse aus Aufgabe 1-3
  - Wechseln Sie in den Basisorder **wba1** Ihres Rails-Projekts

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

  - Erzeugen Sie nun leere HAML-Seiten für einen Controller **pr05**:

    ```
    $ rails generate controller pr05 tr
    ```

  - Editieren Sie **./app/controllers/pr05_controller.rb** . Ergänzen Sie den Verweis auf ein neues Layout **pr05** als Zeile 2

    ```
    layout "pr05"
    ```

  - Ergänzen Sie in **./config/initializers/assets.rb** die Zeile

    ```
    Rails.application.config.assets.precompile += %w(pr05.css)
    ```

  - Wechseln Sie in **./app/views/layouts**. Legen Sie ein neues, einfaches Layout **pr05.html.haml** an (Kopie von **application.html.haml**). Ersetzen Sie darin die Zeile

    ```
    = stylesheet_link_tag "application", …
    ```

    durch

    ```
    = stylesheet_link_tag  params[:controller]
    ```

    Wählen Sie ferner als Inhalt von head/title:

    ```
    WBA-Übung 05: Ein Taschenrechner
    ```

Taschenrechner: Ausgangslage schaffen

- Reproduzieren Sie das Taschenrechner-Beispiel
  - Wechseln Sie in den Order **`./public`** Ihres Rails-Projekts
  - Kopieren Sie den HTML-Quellcode des Taschenrechner-Beispiels in die neue Datei **`tr.html`**
    - Quelle: **`~werntges/lv/wba/tr.html`** auf dem Fileserver der Informatik (ext. Zugriff z.B. über `sftp login1.cs.hs-rm.de`)
    - Orginal-Quelle: **selfhtml.org** (Stand 2015/2016)
  - Starten Sie den Web-Server von Rails:

    ```
    $ rails s
    ```

  - Testen Sie die korrekte Funktion der so erzeugten statischen HTML-Seite „Taschenrechner“ unter dem URL **`http://localhost:3000/tr`**.

Taschenrechner: Komponenten einordnen

- CSS:
  - Verlagern Sie den CSS-Code aus dem HTML-Header in die Datei **`app/assets/stylesheets/pr05.scss`**

Hinweis:
Dieser CSS-Code beeinflusst auch frühere wba1-Aufgaben!

- JavaScript-Code:
  - Verlagern Sie den JavaScript-Code aus dem HTML-Header in die Datei **`app/javascript/packs/tr.js`**
  - Ändern Sie in **`app/javascript/packs/tr.js`** alle Funktionsdefinitionen wie in folgendem Beispiel:
    - Vorher: `function Check (Eingabe) { ...`
    - Nachher: `window.Check = function(Eingabe) { …`

- HTML Body:
  - Verlagern Sie nun den Inhalt des gesamten Body-Elements (**`<body>…</body>`**) in die Datei **`app/views/pr05/tr.html.haml`**
  - Bearbeiten Sie die Datei **`tr.html.haml`**. Wandeln Sie den HTML-Code in HAML-Code um, wie bereits in früheren Aufgaben
    - Verwenden Sie dabei die HTML-Variante der Attribut-Codierung, insbesondere bei den **`input`**-Elementen. Beispiel:

```
%input( type="button" value=" 7 " onclick="Hinzufuegen('7')" )
```

Taschenrechner: Erster Test mit JavaScript

- Test 1:
  - Unter **`http://localhost:3000/pr05/tr`** sollte der Taschenrechner bereits funktionieren.
  - Verwendung findet momentan noch der (nur leicht veränderte) JS-Code von SelfHTML

- Debugging mit Firefox:
  - „Firebug“ einschalten (F12), Reiter „Konsole“ wählen, Seite neu laden
  - Laufzeitfehler von JS-Code werden in der „Konsole“ angezeigt

Taschenrechner: Installation von jQuery

- jQuery-Paket installieren:
  - In **`./wba1`** ausführen:

```
$ yarn add jquery
```

- jQuery-Paket konfigurieren:
  - In **`./wba1/config/webpack/`** eintragen (blaue Texte):

```
const { environment } = require('@rails/webpacker')

const webpack = require('webpack')
environment.plugins.prepend('Provide',
  new webpack.ProvidePlugin({
    $: 'jquery/src/jquery',
    jQuery: 'jquery/src/jquery'
  })
)

module.exports = environment
```

Taschenrechner: jQuery testen

- Test 2:
  - In **`app/javascript/packs/tr.js`** am Ende eintragen:

```
jQuery(document).ready(function() {
  alert("jQuery läuft");
});
```

  - Laden Sie die Taschenrechner-Seite im Browser neu:
    `http://localhost:3000/pr05/tr`
  - Wenn ok, erscheint die Alert-Box mit Text „jQuery läuft“
    - Bem.: alert-Code wird später durch anderen Code zur Initialisierungszeit ersetzt

      jQuery(document).ready() ruft die übergebene Funktion auf, wenn das Laden der HTML-Seite im Browser abgeschlossen ist. Das ist i.d.R. der richtige Moment, um Dinge zu initialisieren und auf Seiteninhalte zuzugreifen, denn zu einem früheren Zeitpunkt sind manche Inhalte evtl. noch nicht vorhanden!
  - Wenn nicht, sehen Sie in der Firefox-“Konsole“ (F12) Fehlermeldungen aus der JavaScript-Ebene mit Hinweisen zu möglichen Ursachen

Taschenrechner: Refactoring

- Code-Redundanzen entfernen:
  - Datei **`pr05.js`**: **Vereinfachen Sie den Code!**
    - Nutzen Sie Variablen (insb. für `window.document.Rechner.Display`), um die wiederholte Verwendung von Ausdrücken wie `window.document.Rechner.Display.value` zu vermeiden
    - Nutzen Sie Möglichkeiten von ECMAScript wie `switch..case..default`
    - Verbessern Sie insbesondere „Sonderfunktion“ so, dass pro Funktion nur eine neue Zeile erforderlich ist
    - Tipp: Code zum Initialisieren gehört in den Codeblock des alert-Aufrufs aus Test 2.
  - Datei **`tr.html.haml`**:
    - Entfernen Sie Leerzeichen aus den `value`-Attributen von **`input`**
    - Entfernen Sie alle Attribute von **`form`** außer `name` (ersatzlos)
  - **Hier STOP für Woche 1 – Fortsetzung folgt. Abgabe von P05 erst in zwei Wochen!**
  - **Ziel für Woche 1: Umstellungen vorführen können.**

Taschenrechner: CoffeeScript (optional!)

- **OPTIONAL**: Arbeiten mit CS statt mit JS
  - Wenn Sie JavaScript (JS) durch CoffeeScript (CS) ersetzen möchten,
    - installieren Sie nun CoffeeScript in Rails 6,
    - ändern Sie Ihren Code von JS um in CS
  - CS nachinstallieren: Führen Sie in ./wba1 aus:

    ```
    $ rails webpacker:install:coffee
    ```

  - CS testen: Ergänzen Sie in views/layouts/pr05.html.haml folgende Zeilen:

    ```
    = javascript_pack_tag 'hello_coffee'
    = javascript_pack_tag 'tr' # Vorbereitung für pr05.coffee
    ```

    Wenn Sie die tr-Seite nun neu laden, sollte in der Browser-Konsole (→ F12) der Text ‘Hello world from coffeescript’ erscheinen.
  - Wandeln Sie nun die Datei **`pr05.js`** `um in` **`pr05.coffee`**:

    Datei umbenennen, JS-Syntax durch CS-Syntax ersetzen.

    Verwenden Sie dazu die Hinweise von coffeescript.org und folgende Quelle:

    http://railscasts.com/episodes/267-coffeescript-basics?view=asciicast

Taschenrechner: TypeScript (optional!)

- **OPTIONAL**: Arbeiten mit TS statt mit JS
  - Wenn Sie JavaScript (JS) durch TypeScript (TS) ersetzen möchten,
    - installieren Sie nun TypeScript in Rails 6,
    - ändern Sie Ihren Code optional von JS um in TS

  - TS nachinstallieren: Führen Sie in ./wba1 aus:

    ```
    $ rails webpacker:install:typescript
    ```

  - TS testen: Ergänzen Sie in views/layouts/pr05.html.haml folgende Zeile:

    ```
    = javascript_pack_tag 'hello_typescript'
    ```

    Wenn Sie die tr-Seite nun neu laden, sollte in der Browser-Konsole (→ F12) der Text ‘Hello world from typescript’ erscheinen.

  - Wandeln Sie nun die Datei **`pr05.js`** `um in` **`pr05.ts`**:

    Datei umbenennen, JS-Syntax ggf. durch TS-Syntax erweitern.

    Da TS abwärtskompatibel zu JS ist, müssen Sie zunächst nichts ändern.

(DISCLAIMER: DIESE SCHRITTE WURDEN VOM DOZENTEN NICHT GEPRÜFT)

Taschenrechner: Unobtrusive JavaScript

- Stellen Sie um auf „Unobtrusive JavaScript“:
  - Datei tr.html.haml:
    - *Entfernen* Sie alle `onclick`-Attibute der input-Elemente *für normale* Eingaben
    - *Ersetzen* Sie das `onclick`-Attribut des Buttons *für* „=“ durch das in HTML5 zulässige Attribut `data-cmd="result"`
    - *Ersetzen* Sie die `onclick`-Attribute der Buttons für Sonderfunktionen durch das Attribut `data-cmd="function"`

Taschenrechner: Unobtrusive JavaScript

- Stellen Sie um auf „Unobtrusive JavaScript“ (**JS**-Version):

  - Datei **`pr05.js`**: Hier reagieren wir nun auf die Events!
    - Ergänzen Sie folgenden Code direkt unterhalb von `jQuery ->` :

```
$(document).ready (function() {
  # Hier Ihre Initialisierungs-Codezeilen
});
```

    - Ersetzen Sie die Kommentarzeile durch Ihre gewünschten Kommandos. Diese werden einmal, nach dem Laden der HTML-Seite, aufgerufen.

    - Ergänzen Sie darunter folgenden Code (im Codeblock von $(document).ready):

```
$('input[type="button"]').click (function(event) {
  switch($(this).attr('data-cmd')) {
    case 'result':  Ergebnis(); break;
    case 'function': Sonderfunktion(this.value); break;
    default: Hinzufuegen(this.value);
  }
});
```

    - Entfernen Sie das nun überflüssige Präfix „windows.“ wieder von den Funktionen „Ergebnis“ usw. (diese Funktionen werden jetzt nur noch lokal aufgerufen).

Taschenrechner: Unobtrusive JavaScript

- Stellen Sie um auf „Unobtrusive JavaScript“ (**CS**-Version):
  - Datei **`pr05.coffee`**:
    - Initialisierungs-Code

```
$(document).ready ->
  # Hier Ihre Initialisierungs-Codezeilen
```

    - Ersetzen Sie die Kommentarzeile durch Ihre gewünschten Kommandos. Diese werden einmal, nach dem Laden der HTML-Seite, aufgerufen.

    - Ergänzen Sie darunter folgenden Code (im Codeblock von $(document).ready):

```
$('input[type="button"]').click (event) ->
  switch $(this).attr('data-cmd')
    when 'result' then Ergebnis()
    when 'function' then Sonderfunktion(this.value)
    else Hinzufuegen(this.value)
```

    - Entfernen Sie das nun überflüssige Präfix „windows.“ wieder von den Funktionen „Ergebnis“ usw. (diese Funktionen werden jetzt nur noch lokal aufgerufen).

Taschenrechner: Unobtrusive JavaScript

- Erläuterungen zu „Unobtrusive JavaScript“:
  - Im HTML/HAML-Code ist nun kein JS-Code mehr enthalten!
  - Im JS/CoffeeScript-Code stellt man Verbindungen dank JQuery mit dem HTML-Code über ähnliche Selektor-Ausdrücke her wie in CSS. Dies ist viel einfacher als über den üblichen DOM-Weg: Der String in **`$(…)`** ist ein normaler CSS-Selektor!

```
$('input[type="button"]')
```

  - Das Ergebnis ist i.a. eine Menge von Elementknoten. Für jedes Element wird die folgende Methode **`click`** angewendet. Das bedeutet: Die auf **`click`** folgende Funktion wird beim Anklicken ausgeführt.
  - Mithilfe des Attributs **`data-cmd`** haben wir nun eine zentrale, erweiterbare Dispatcher-Funktion eingeführt, die zu den drei ursprünglichen Funktionen weiterleitet.

Taschenrechner: Erweiterungen

- Erweiterungen um technisch-wissenschaftliche Funktionen
  - Der Rechner soll bei Bedarf weitere Tasten einblenden, s.u.

- Unschalt-Tasten:
  - Eine Umschalt-Taste wechselt beim Anklicken die Beschriftung und bewirkt eine Zustandsänderung des Taschenrechners
  - Umschalttasten sollen sich optisch von anderen abheben
  - Vorgabe: Umschalttasten sollen den aktuellen Status anzeigen (nicht den gewünschten)
  - Der Wert von Attribut `data-cmd` sei „**`switch`**“ bei Umschalttasten

  - **Taste „Sci“/“Std“**
    - Umschaltung zwischen Standard-Modus („**Std**“) und technisch-wissenschaftlichem Modus („**Sci**“)
    - Beim Anklicken sollen drei neue Tastenreihen ein- bzw. ausgeblendet werden, von „ln“ bis „Deg“ (vgl. Abbildung unten)

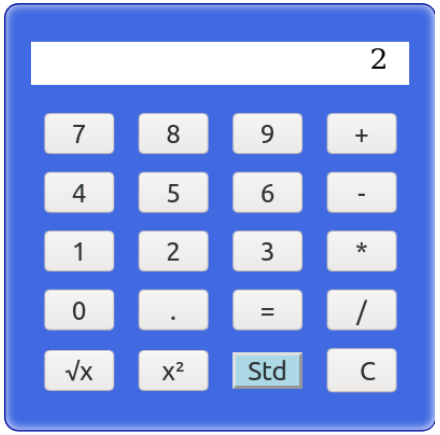
  - **Taste „Deg“/“Rad“**
    - Umschaltung zwischen „Grad“ bzw. „Bogenmaß“ für die Winkelfunktionen
    - Einstellung „**Deg**“:
      - sin(90) ergibt 1, asin(1) ergibt 90.0,
    - Einstellung „**Rad**“:
      - sin(π/2) ergibt 1, asin(1) ergibt π/2

Taschenrechner: Erweiterungen

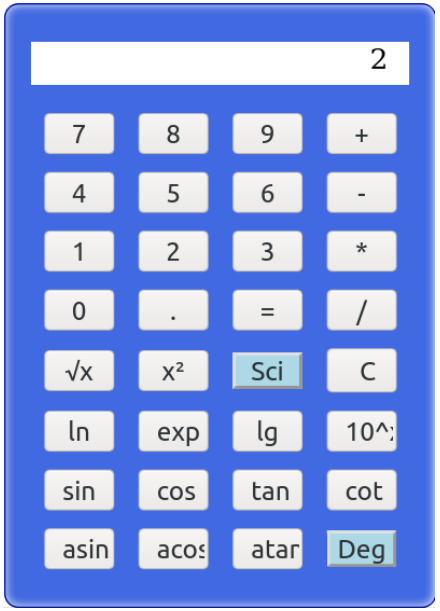
- Unschalt-Tasten, CoffeeScript oder JavaScript:
  - Ergänzen Sie den Dispatcher-Code um einen Fall für den neuen Wert „**`switch`**“. Rufen Sie von dort die neue Funktion „Umschalten“ auf!
  - **Implementieren Sie die Funktion „Umschalten“.**
    - Als Argument erhält sie die geklickte Taste (this bzw. $(this)), damit Sie die Tastenbeschriftung dieses Objekts (“value”) später ändern können.
    - Beim Wechsel auf „**Sci**“ sollen die drei Zusatzzeilen eingeblendet werden (Dauer: 0.5 s), beim Wechsel zurück sollen sie entsprechend wieder ausgeblendet werden. Die Button-Beschriftung wechselt passend.
    - Beim Wechsel auf „**Deg**“ soll eine globale Variable zur Umrechnung von Grad auf Bogenmaß geeignet belegt werden. Verwenden Sie wieder den Namensraum „window.“, wie bei den Funktionen „Ergebnis“ usw. Beim Wechsel auf „**Rad**“ soll diese Variable wieder auf 1.0 gesetzt werden. Die Button-Beschriftung wechselt entsprechend, zeigt also immer den aktuellen Modus an.
  - Implementieren Sie die neuen math. Funktionen (in „Sonderfunktionen“)
    - Tipp: **`switch / when`** verwenden!
    - Details: Nächste Folie

Taschenrechner: Erweiterungen

- Neue Tasten: Ändern und erweitern Sie die Tastatur des Taschenrechners wie folgt.
  - Tasten der bisherigen letzten Zeile lauten nun:
    - **√x, x², Std , C**
      (Quadratwurzel, Quadrat, Umschalttaste „Std/Sci“, Löschtaste)
    - Verwenden Sie genau diese Tastenbeschriftungen!



  - Tasten der ersten neuen Zeile im „Sci“-Modus
    - **ln, exp, lg, 10^x**
      (Natürl. Log., e-Funktion, 10er-Log, 10 hoch x)
  - Tasten der zweiten neuen Zeile im „Sci“-Modus
    - **sin, cos, tan, cot**
      (Die Winkelfunktionen; cot(x) = 1/tan(x))
  - Tasten der dritten neuen Zeile im „Sci“-Modus
    - **asin, acos, atan, Rad**
      (Die Arcus-Funktionen / Umkehrfunktionen zu den Winkelfunktionen; die Umschalttaste zwischen Bogenmaß und Grad)
      Achten Sie hier darauf, die Umrechnung in Grad bzw. Bogenmaß korrekt anzupassen.
  - Voreinstellungen beim Öffnen der Seite seien: „**Std**“, „**Deg**“



Taschenrechner: Erweiterungen

- Tipps
  - Nützliche Funktionen und Konstanten finden Sie im JavaScript-Modul „Math“, siehe **`https://wiki.selfhtml.org/wiki/JavaScript/Objekte/Math`**
    - Auch W3Schools bietet eine gute Dokumentation zu „Math“
    - Benötigte Umrechnungen (z.B. zu lg, 10^x) finden Sie in üblichen mathematischen Formelsammlungen, falls benötigt
  - Vergeben Sie eine eigene HTML/CSS-Klasse für die 3 x 4 Tasten mit technisch-wissenschaftlichen Funktionen, oder gruppieren Sie diese in einem geeigneten Block.
  - Mit den JQuery-Funktionen **`show`** und **`hide`** ist das Ein- und Ausblenden dieser Tasten bzw. ihres Blocks sehr einfach möglich
  - Generell: Je besser der Code *vor* den Erweiterungen „aufgeräumt“ wurde, desto einfacher fallen die Erweiterungen.

Taschenrechner: Erweiterungen

- Tipps
  - **`this`** oder **`$(this)`** ?
    - **`this`** ist i.d.R. ein DOM-Objekt und versteht nur dessen Methoden & Attribute
    - **`$(this)`** ist ein jQuery-Objekt (Wrapper). Es ist mit anderen, meist bequemeren Methoden ausgestattet.
  - Beispiele mit gleichem Ergebnis:
    - Fall **`$(.button).click`** :

      `this.getAttribute("data-cmd")` und **`$(this).attr("data-cmd")`**

    - Fall Passwort-Vergleich:

      `$('#params_password').val()` und **`$('#params_password')[0].value`**

Taschenrechner: Erweiterungen (freiwillig)

- Freiwillige Zusatzaufgaben für die Formulare aus Pr03
  - Sorgen Sie dafür, dass die Abfrage des Hochzeitsdatums nur dann erscheint, wenn der Familienstand „verheiratet“ gewählt ist
    - Ein-/Ausblenden analog zu „Sci/Std“
    - Noch besser: + Löschen des Hochzeitsdatums, falls „nicht verheiratet“
  - Prüfen Sie vor dem Absenden des Formulars (also auf Client-Seite), ob das wiederholte Passwort mit den Passwort übereinstimmt. Im Fehlerfall unterdrücken Sie das Absenden des Formulars (Rückgabewert: false) und erzeugen stattdessen eine Fehlermeldung („alert“)
    - Tipps: **`$("form").submit -> …, $('#params_password').val`**
- *Diese Aufgabenteile wenden sich an Teilnehmer:innen mit bereits vorhandenen JavaScript-Kenntnissen! Ihr Hauptteil besteht in der Suche nach passenden JQuery-Funktionen*

Bedingungen

- Abgabe
  - Wegen des Feiertags am 3.6. erst in Kalenderwoche 23
  - Jeweils vor Beginn Ihrer Praktikumsgruppe in der Abgabewoche
- Art des Leistungsnachweises
  - Zu vergeben: **2 Punkte + 0,5 SoPu.**
    - 1 Punkt für Reproduktion und Umorganisation des Taschenrechners
    - 1 Punkt für die Erweiterungen
    - 0,5 Sonderpunkte für die beiden Zusatzaufgaben zu den Formularen (pr03)
  - **Einzel-Arbeit – keine Teams!**
  - Abgabe der Dateien:
    - Im Verzeichnis „wba1“ ausführen:

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

    - Sie sind nun im Elternverzeichnis von „wba1“. Jetzt noch ausführen:

```
# Ordner ./wba1 verpacken, ohne unnötige Unterordner:
$ tar czf 05-wba1-<matnr>.tar.gz --exclude wba1/tmp \
      --exclude wba1/node_modules --exclude wba1/log  ./wba1
```

      Fortsetzungszeile

      Datei „05-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

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.

Literatur

- **Zu CoffeeScript**
  - **https://coffeescript.org**
    - Projekt-Homepage, mit Übersicht & vielen Code-Beispielen

- **Zu jQuery**
  - **https://jquery.com/**
    - Einstiegsseite
  - **https://api.jquery.com/category/selectors/**
    - CSS-artige und erweiterte Selektoren – sehr nützlich!“
      (liefert Seite mit allen FormHelper-Methoden)