Übungsblatt 3

11 Mit jQuery eine Vorschau der Sendedaten erstellen*

Binden Sie nun jQuery als externe JavaScript Quelle ein. Verändern Sie das Formular von Blatt 2 sobald es abgeschickt wird, so dass es nicht direkt an die im action Attribut definierte URL geschickt sondern nur geändert[1], d.h. der DOM manipuliert wird. Die Änderung soll sichtbar alle `<input>` Felder in `<span>`-Tags ändern. Intern soll jedoch folgendes geschehen:

- Die Input Felder sollen über jQuery als nicht sichtbar gesetzt werden:

```
1 $('input').hide();
```

- Vor jedes gefundene `<input>`-Element soll über `insertBefore()` ein neues `<span>`-Element gesetzt werden.
- Der Inhalt des neuen Elementes soll dem Wert des Eingabeelementes entsprechen.
- Die Zeilen für die Passworteingabe sind komplett aus dem Sichtfeld zu entfernen.
- Sie können jQuery z.B. von hier einbinden: `http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.js`.

Jetzt sollte der Submit Button wie man es eingangs erwartet hat funktionieren, d.h. Abschicken schickt die Daten des Formulars an die angegebene Url. Zum Testen können Sie folgende die Url `http://html5.florian-rappl.de/submitted.html` verwenden.

12 Die Nullstelle finden*

Sie erstellen ein sehr einfaches Formular mit zwei `type=number` ($-\infty$ bis $+\infty$) Elementen und einer `type=range` (genannt N, von 2 bis 10000) Eingabe. Es soll eine live, d.h. nicht erst beim Drücken des Submit Buttons, Validierung ausgeführt werden. Hierbei wird überprüft, ob die erste Nummer (genannt x_i) kleiner ist als die zweite Nummer (def. x_f). Sollte dem nicht so sein, so ist der Submit Button deaktiviert.
Die verwendete Funktion f soll direkt im JavaScript-Code eingebaut werden. Folgender Code soll implementiert werden:

1. Zunächst berechnen Sie $\delta = (x_f - x_i)/N$.

[1] Indem Sie z.B. `onsubmit='return false;'` oder `onsubmit='return myfoo();'`, mit `function myfoo() { return false; }`, schreiben

2. Nun gehen Sie von $x = x_i$ mit N Schritten in Richtung x_f.

3. Beim Vorzeichenwechsel, d.h. $f(x_{i-1}) \cdot f(x_i) < 0$, soll der Punkt x_i einer Liste hinzugefügt werden.

4. Die so gefundenen Punkte sollen am Ende ausgegeben werden (z.B. in einer Textbox).

13 Für Tüftler

Sie können Aufgabe 12 durch Einbau einer Textbox über `<textarea>` erweitern. In diesem Eingabefeld soll dabei eine (beliebige) Methode eingegeben werden können, die dann auf Nullstellen untersucht werden kann. Dies können Sie über folgenden Trick erreichen:

- Beim Laden des Dokumentes legen Sie sich bereits eine Variable f an, die einfach eine leere Methode mit Rückgabe von 0, d.h.

```
1 var f = function(x) { return 0; };
```

  darstellt.

- Beim `onsubmit` des Formulars erstellen Sie über `createElement()` ein neues Script-Element.

- Dieses Script Element soll folgenden `innerHTML` Wert erhalten (`tai` steht symbolisch für den Inhalt der Textbox):

```
1 script.innerHTML = 'f = function (x) { var y = 0.0; ' +
      tai + '; return y; };';
```

- Nun dem body das Script-Element hinzufügen und die Auswertung wie in Aufgabe 12 durchführen.

14 Zahlenraten*

Sie entwickeln ein Spiel mit HTML und JavaScript. Gehen Sie folgendermaßen vor:

- Basteln Sie eine HTML Seite mit drei `<div>`-Containern, wobei im ersten Container drei numerische Boxen (Minimum, Maximum, Zeit in Sekunden) und ein Button zum Absenden sein sollen.

- Im zweiten Container sollen zwei `<span>` für die verbleibende Zeit, Anzahl der Versuche, sowie ein `<select>`-Feld mit Optionen “Zahl wählen” und den verfügbaren Zahlen (von Minimum bis Maximum) angezeigt werden.

- Der dritte Container beinhaltet ein Button ("Neustarten") und ein `<span>` zur Anzeige des Ergebnisses.
- Schreiben Sie JavaScript in einer externen Datei (**.js*). Beim Starten des Scriptes soll eine anonyme Methode selbstständig aufgerufen werden, die einige Veränderungen am HTML-Code durchführen: Die letzten beiden Container sollen verborgen werden (entweder mit jQuery über Standardmethoden):

```
1 // jQuery
2 $('#...').hide();
3 // Standard
4 getElementById('...').style.display = 'none';
```

- Außerdem sollen Ereignisse gesetzt werden. Der Button im ersten Container soll einen `onclick`-Callback erhalten, genauso wie der Button im dritten Container. Das `<select>`-Feld im zweiten Container soll einen `onchange`-Callback erhalten.
- Überlegen Sie sich die Callbacks geschickt zu implementieren, so dass folgende Funktionalität erzeugt wird: Es ist immer nur ein Container sichtbar, und es ist immer die aktuelle Anzahl an verbleibenden Sekunden bis zum Spielende sichtbar.
- Im 3. Container soll als Zusammenfassung die richtige Zahl (zufällig mit:

```
1 Math.floor(Math.random() * (Maximum - Minimum)) +
    Minimum
```

  erzeugt), die zur Lösung verstrichene Zeit und die Anzahl der Versuche angezeigt werden.
- Binden Sie Ihre erstellte JavaScript Datei über einen `<script>` Tag vor dem Ende des `<body>` ein.

15 Bestehendes erweitern

In JavaScript können Sie über die `prototype` Eigenschaft nicht nur Ihre Klasse um öffentliche Methoden und Eigenschaften erweitern, sondern auch bereits bestehende Klassen. In dieser Aufgabe sollen Sie die Array-Klasse um eine Methode `shuffle()` erweitern. Diese Methode soll die Elemente des Arrays zufällig umsortieren.
Zum Testen Ihrer Implementierung bauen Sie ein Formular das über ein Textfeld zur Eingabe und zwei Buttons (*Add*, *Shuffle*) verfügt. Beim Drücken auf *Add* soll die Eingabe einem Array hinzugefügt werden und das Textfeld anschließend geleert werden. Beim Drücken auf Shuffle soll das Array über Ihre Methode umsortiert werden und anschließend ausgegeben werden. Die Ausgabe sollte unterhalb der Eingabe in einem `<div>`-Container erfolgen. Dieser Ort würde sich auch eignen um nach jedem Ausführen des *Add*-Callbacks den aktuellen Inhalt des Arrays anzuzeigen.