Software Engineering 2

Lösungen der Übung JavaScript, Nachladen von Daten im HTML (AJAX)

Dr. F.-K. Koschnick, Sybit GmbH

Aufgabe (Story): Nachladen von Daten im HTML

Als Nutzer der Webseite möchte ich möglichst schnell ohne ein vollständiges Neuladen der Seite, Daten über Maximalgeschwindigkeiten von Tieren angezeigt bekommen, um ein möglichst angenehmes Nutzungserlebnis der Seite zu haben.
Verifiziere, dass

- das Layout von Bootstrap verwendet wird, so dass die nachgeladene Tabelle responsiv dargestellt wird
- es einen Button gibt, der das Nachladen der Daten auslöst
- ein Webservice, der die Daten liefert in der Entwicklungsumgebung eingerichtet wird und dort gestartet werden kann
- das Layout der Seite dem angehängten Bild vor dem Nachladen entspricht
- die dynamisch erzeugte Tabelle zwei Spalten hat, mit den Überschriften "Tier" und "Maximalgeschwindigkeit"
- nach dem Betätigen des Buttons 15 Datensätze als Zeilen angezeigt werden

Technische Aspekte:

- es soll AJAX mit jQuery genutzt werden
- Es sol Bootstrap verwendet werden

Das Layout mit Bootstrap wird mit folgenden Styleklassen bewerkstelligt:
Für den body wird die Styleklasse "text-center" benutzt.
Für den button wird die Styleklasse "btn" und "btn-primary btn-lg" benutzt, also: "<button class=‘btn btn-primary btn-lg‘ ...."
Für die Tabelle wird die Styleklasse "table" und "table-striped" verwenden, also: "<table class = ‘table table-striped‘>...." (da die Tabelle mit Javascript erzeugt wird, bitte daran denken, die Anführungsstriche beim Attribut class mit Backslash zu escapen, wenn man mit einer String-Variable arbeitet oder anstelle der Anführungsstriche einfache Hochkommas zu verwenden.)
Der Source-Code des Webservices ist als Projekt im Repository Vorlesung im Ordner "webService" eingecheckt. Diesen Ordner in den Workspace der Entwicklungsumgebung kopieren und dann als Gradle-Projekt importieren. Dann den Server mit dem Projekt verbinden und starten. Unter der URL http://localhost:8080/webService/api/velocities muss dann ein JSON im Browser sichtbar sein. Dieses analysieren und dann das Javascript entsprechend bauen.

Lösung: HTML

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
          integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
          crossorigin="anonymous">
    <script src="https://code.jquery.com/jquery-3.3.1.js"
          integrity="sha256-2Kok7MbOyxpgUVvAk/HJ2jigOSYS2auK4Pfzbm7uH60="
          crossorigin="anonymous">
    </script>

    <title>AJAX Example</title>
</head>
<body class="text-center">
    <br/>
    <button class="btn btn-primary btn-lg" id="myBtn">Send an HTTP GET request to a page and get the result back</button>
    <br/><br/>
    <p id="paragraph"></p>
    <script src="myscript.js"></script>
</body>
</html>
```

Einbinden von Bootstrap

Einbinden von jQuery, nur für die zweite Lösung (mit jQuery) wichtig.

Style-Klassen aus Bootstrap

In diesen Paragraphen wird die Tabelle mit den Daten dynamisch über JavaScript eingebaut. Falls man hier mit dem Tag <table class="…"> arbeitet, muss man im Javascript mit <tbody> arbeiten, sonst wird table-striped nicht richtig formatiert.

JavaScript am Ende eingebunden. Dass das JavaScript am Ende steht, ist hier nur wichtig für die 1. Lösung (ohne jQuery), da das HTML geladen sein muss, damit das Click-Event für den Button definiert werden kann. (Bei jQuery nutzen wir den document.ready-Event, da ist es egal.)

Lösung: Analyse JSON

Einzelnes Objekt, bestehend aus den Properties: animal und velocity

```
[
    {
        "id": 1,
        "animal": "Gepard",
        "velocity": "122 km/h"
    },
    {
        "id": 2,
        "animal": "Windhund",
        "velocity": "110 km/h"
    },
    {
        "id": 3,
        "animal": "Gazelle",
        "velocity": "90 km/h"
    },
    {
        "id": 4,
        "animal": "Antilope",
        "velocity": "90 km/h"
    },
    {
        "id": 5,
        "animal": "Gabelbock",
        "velocity": "85 km/h"
    },
    {
        "id": 6,
        "animal": "Strauss",
        "velocity": "70 km/h"
    },
    {
        "id": 7,
        "animal": "Pferd",
        "velocity": "70 km/h"
    },
    {
        "id": 8,
        "animal": "Feldhase",
        "velocity": "70 km/h"
    },
    {
        "id": 9,
        "animal": "Baer",
        "velocity": "50 km/h"
    },
    {
        "id": 10,
        "animal": "Flusspferd",
        "velocity": "48 km/h"
    },
    {
        "id": 11,
        "animal": "Afrikan. Elefant",
        "velocity": "40 km/h"
    },
    {
        "id": 12,
        "animal": "Mensch",
        "velocity": "38 km/h"
    },
    {
        "id": 13,
        "animal": "Rennechse",
        "velocity": "29 km/h"
    },
    {
        "id": 14,
        "animal": "Riesen-Schildkroete",
        "velocity": "0,37 km/h"
    },
    {
        "id": 15,
        "animal": "Schnecke",
        "velocity": "0,027 km/h"
    }
]
```

http://localhost:8080/webService/api/velocities

Als GET-Aufruf, z.B. mit Postman, oder da GET einfach im Browser öffnen.

Array von Objekten

Lösung: JavaScript (myscript.js) mit jQuery

Document ready, stellt sicher, dass erst das gesamte HTML geladen ist.

Click-Event dem Button zuordnen und in der anonymen Funktion den AJAX-Get-Aufruf ($.get(…)) definieren.

```
$(document).ready(function(){
    $("#myBtn").click(function(){
        $.get("http://localhost:8080/webService/api/velocities", function(velocities){
            createTable(velocities);
        });
    });
});

function createTable(velocities) {
    var txt="<table class='table table-striped'><tr><th>Tier</th><th>Maximalgeschwindigkeit</th></tr>"
    $.each( velocities, function( key, velocity ) {
        txt+="<tr><td>"+velocity.animal+"</td><td>"+velocity.velocity+"</td></tr>";
    });
    txt+="</table>"
    $("#paragraph").html(txt);
}
```

Die anonyme Funktion bekommt das Ergebnis des AJAX-Aufrufs, den Array velocities mit den velocity-Objekten, übergeben. In der Funktion wird dann createTable aufgerufen, welche dynamisch die Tabelle auf Basis der velocity-Objekte erzeugt.

Style-Klassen aus Bootstrap

Zugriff auf die Properties des velocity-Objekts

Alternative Lösung: JavaScript mit append

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
          integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
          crossorigin="anonymous">
    <script src="https://code.jquery.com/jquery-3.3.1.js"
          integrity="sha256-2Kok7MbOyxpgUVvAk/HJ2jigOSYS2auK4Pfzbm7uH60="
          crossorigin="anonymous">
    </script>

    <title>AJAX Example</title>
</head>
<body class="text-center">
    <br/>
    <button class="btn btn-primary btn-lg" id="myBtn">Send an HTTP GET request to a page and get the result back</button>
    <br/><br/>
    <table id="table" class="table table-striped"><tbody id="tableBody"/></table>
    <script src="myscript_JQueryVelocityAppend.js"></script>
</body>
</html>
```

tbody wird benötigt, sonst stellt der Browser bei Append die bootstrap-Klassen nicht richtig dar.

Alternative Lösung: JavaScript mit append

(myscript_jQueryVelocityAppend.js)

```
$(document).ready(function(){
    $("#myBtn").click(function(){
        $.get("http://localhost:8080/webService/api/velocities", function(velocities){
            createTable(velocities);
        });
    });
});

function createTable(velocities) {
    var t = document.getElementById("table");
    t.removeChild(document.getElementById("tableBody"));
    var tb = document.createElement("tbody");
    var tr = document.createElement("tr");
    var elem = document.createElement("th");
    var node = document.createTextNode("Tier");
    elem.appendChild(node);
    tr.appendChild(elem);
    elem = document.createElement("th");
    node = document.createTextNode("Maximalgeschwindigkeit");
    elem.appendChild(node);
    tr.appendChild(elem);
    tb.appendChild(tr);

    $.each( velocities, function( key, velocity ) {
        tr = document.createElement("tr");
        elem = document.createElement("td");
        node = document.createTextNode(velocity.animal);
        elem.appendChild(node);
        tr.appendChild(elem);
        elem = document.createElement("td");
        node = document.createTextNode(velocity.velocity);
        elem.appendChild(node);
        tr.appendChild(elem);
        tb.appendChild(tr);
    });
    t.appendChild(tb)
}
```

Remove, damit bei mehrmaligen Laden, die Tabelle nicht länger wird.

Vergleich beider Lösungen

Die Lösung ohne Append mit einer Stringvariable ist deutlich einfacher.