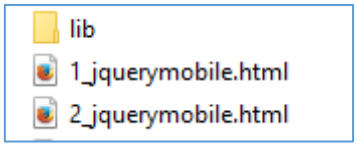
# jQuery-Mobile - 2




## Inhalt:

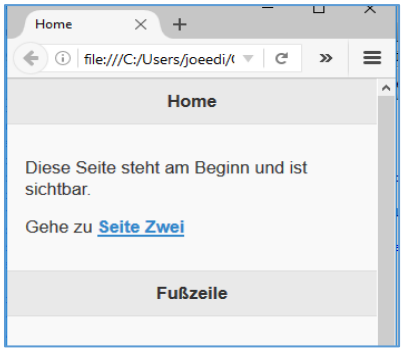


Die grafischen Elemente werden in jQuery Mobile als Widgets entwickelt. Ein Widget ist ein Interaktions-Element einer grafischen Benutzerschnittstelle, das ein eigenes Verhalten aufweisen kann.

## 1)Mehrere Seiten in einem Dokument

a)Erstelle eine neue Website neben dem HTML-Dokument „1_jquerymobile.html“ mit dem Namen „**2_jquerymobile.html**“, der ebenfalls auf den Ordner „lib“ zugreift.

Ergebnis:




```
<title>going mobile</title>
<link rel="stylesheet" href="lib/jquery.mobile-1.4.5.min.css">
<script src="lib/jquery-2.1.4.min.js"></script>
<script src="lib/jquery.mobile-1.4.5.min.js"></script>
</head>
<body>
<!--Start page 1-->
<div data-role="page" id="p1">
    <div data-role="header">
        <h1>Home</h1>
    </div> <!--Ende Header-->

    <div data-role="content">
        <p>Diese Seite steht am Beginn und ist sichtbar.</p>
        <p>Gehe zu <a href="#p2"> Seite Zwei</a></p>
    </div> <!-- Ende content-->

    <div data-role="footer">
        <h4>Fußzeile</h4>
    </div> <!-- Ende -->
</div> <!--Ende page 1-->
</body>
```

## Info:

## Das Page Widget:

Das Page Widget ist der wichtigste Container in jQuery Mobile. Es wird dazu genutzt, den Inhalt in einzelne Seiten (engl. Page) zu gliedern. Eine Page kann ein Header, ein Content und ein Footer Widget enthalten, funktioniert allerdings auch ohne.

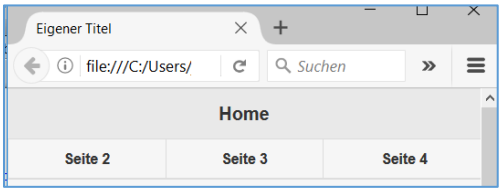
**Syntax**:
Eine Page wird in ein div-Element gekapselt und mit data-role="page" für jQuery Mobile als Page markiert. Es ist immer sinnvoll, einer Page eine sprechende „id" zu geben. Spätestens dann, wenn man von anderen Pages der Anwendung auf sie verweisen/verlinken möchte, wird sie benötigt.

```
<div data-role="page" id="p1">
<!—Inhalt der Seite-->
</div>
```

### data-title:

Damit kann für jede Page ein eigener Titel im Browser angezeigt werden.

```
12 <!--Start page 1-->
13 <div data-role="page" id="p1" data-title="Eigener Titel">
14   <div data-role="header">
```




### Data-Attribute:

In der folgenden Tabelle finden sich einige der wichtigsten und gebräuchlichsten data-Attribute von jQuery-Mobile und deren Bedeutung.

| *Data-Attribut* | Bedeutung |
|---|---|
| data-role | Die Rolle des Elements, in der Regel der Name des Widgets, z.B. button, page, header… |
| data-icon | Falls das Widget ein Icon haben kann, der Name des Icons, z.B. star |
| data-iconpos | Wenn die Position eines Icons vom Standard abweichen soll, kann man mit diesem Attribut die Position bestimmen. |
| data-transition | Bestimmt die Art des Übergangs. |
| data-theme | Bestimmt, welches Theme verwendet werden soll, z.B. a. |

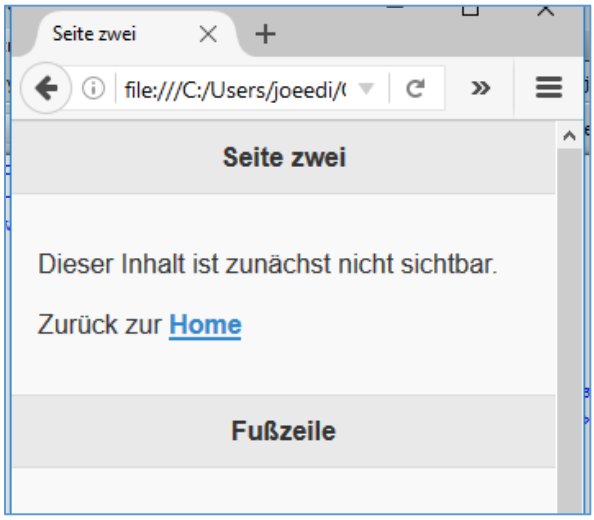
## b)Erstelle darunter eine „Page 2“:

Schreibe nach der Zeile 24 weiter.

```
24  </div> <!--Ende page 1-->
25
26  <!--Start page 2-->
27  <div data-role="page" id="p2">
28      <div data-role="header">
29          <h1>Seite zwei</h1>
30      </div><!--Ende header-->
31
32      <div data-role="content">
33          <p>Dieser Inhalt ist zunächst nicht sichtbar.</p>
34          <p>Zurück zur <a href="#p1"> Home</a></p>
35      </div> <!--Ende content-->
36
37      <div data-role="footer">
38          <h4>Fußzeile</h4>
39      </div><!--Ende footer-->
40  </div><!--Ende page 2-->
41
42  </body>
43  </html>
44
```

Die Verlinkung zwischen den Pages erfolgt durch ein einfaches <a href> und verwendet **nach dem Rautenzeichen die ID auf die man verlinken will**. Z.B. <a href=“#p1“>.

Ergebnis:




**AUFGABE:**

Füge eine beliebige Seite #p3 und #p4 dazu.

## 2)Eine einfache Navigation mit „listview“

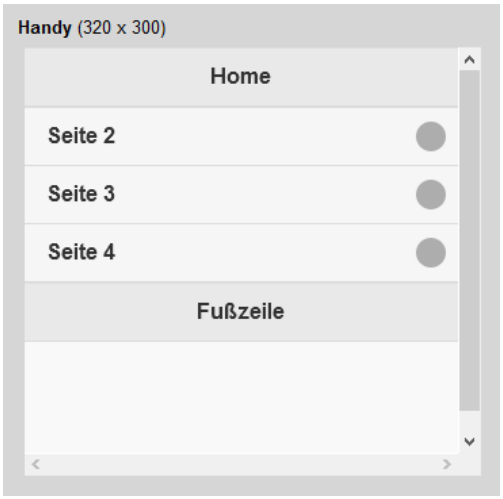
Im Content-Bereich soll eine Möglichkeit für eine Verlinkung erfolgen.

Übung:

Lösche den Inhalt im Content der „page 1“ und füge folgende Zeile ein:

```
16      <div data-role="content">
17          <ul data-role="listview">
18              <li><a href="#p2"> Seite 2 </a></li>
19              <li><a href="#p3"> Seite 3 </a></li>
20              <li><a href="#p4"> Seite 4 </a></li>
21          </ul>
22      </div> <!-- Ende content-->
23
24      <div data-role="footer">
```

Ergebnis im „Multi-Screen-Vorschau“:



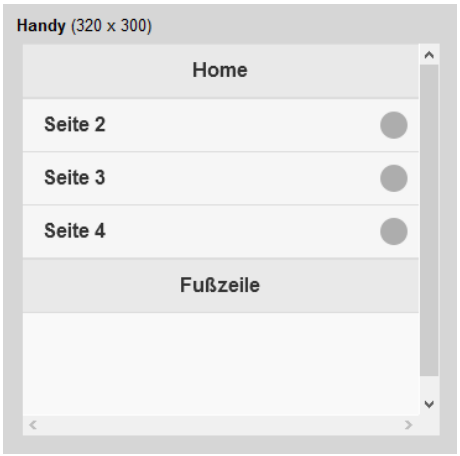

Eine Navigationsliste erhält die data-role „listview“. Dafür erstrecken sich die Schaltflächen über die gesamte Breite des Viewports.

Alternativ kann man auch ein Design wählen, wo ein Rand bestehen bleibt und die Ecken abgerundet erscheinen. Hierfür muss zusätzlich das Attribut **„data-inset“ mit dem Wert „true“** gesetzt werden.

**Übung**: Erstelle dies für die Seite 2:
Nenne die Seite Eins „Home“

```
35      <div data-role="content">
36          <ul data-role="listview" data-inset="true">
37              <li><a href="#p1"> Home </a></li>
38              <li><a href="#p3"> Seite 3 </a></li>
39              <li><a href="#p4"> Seite 4 </a></li>
40          </ul>
41      </div> <!--Ende content-->
```

Seite 1:




Seite 2 mit <data-inset="true">:

## 3)Navbar Widget

Es gibt die Navbar grundlegend in zwei Varianten, einmal als Buttonleiste innerhalb einer Page und einmal als persistente Navigationsleiste zwischen verschiedenen Seiten. Beiden gemein ist die Konfiguration und Syntax, sodass die persistente zu 100 Prozent auf der grundlegenden Variante aufbaut.

## 3a) Navbar-Button-Liste

Um eine Navigationsleiste (Navbar) zu erzeugen, benötigt man zwei Dinge, ein Container-div und eine Liste mit Hyperlinks. Im jQuery Mobile mitzuteilen, dass es eine Navbar ist, muss dem „div“ das Attribut data-role mit dem Wert „navbar“ mitgegeben werden.

```
<div data-role="navbar">
   <ul>
        <li><a href="#p2">Seite zwei</a></li>
        ……..
  </ul>
</div>
```

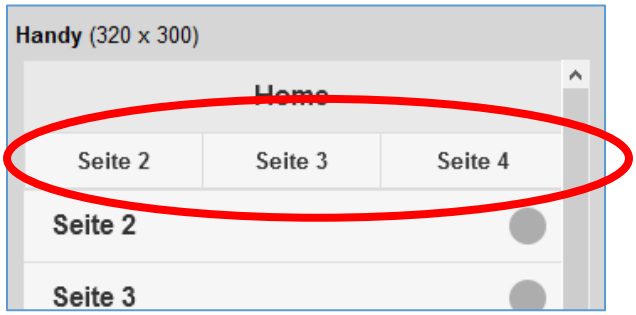
### Navbar im Header

Ein üblicher Ort für eine derartige Button-Leiste ist der Header:

**Übung**: Füge im Header folgenden Code ein:

```
<!--Start page 1-->
<div data-role="page" id="p1">
    <div data-role="header">
        <h1>Home</h1>
        <div data-role="navbar">
            <ul>
                <li><a href="#p2"> Seite 2 </a></li>
                <li><a href="#p3"> Seite 3 </a></li>
                <li><a href="#p4"> Seite 4 </a></li>
            </ul>
         </div> <!--Ende navbar-->
    </div> <!--Ende Header-->
```

Ergebnis:



Bildet man eine Button-Leiste teilen sich bis zu einer Höchstzahl von fünf Buttons diese die gesamte Breite. Ab fünf Elementen werden sie in Zeilen mit jeweils zwei Buttons dargestellt.

Um zu definieren, wie viele Buttons/Links sich in einer Zeile befinden, gibt es das Attribut „data-grid“, das die Werte a,b,c oder d enthalten kann, um ein zwei-, drei-, vier- oder fünfspaltiges Navbar Widget zu erzeugen.

Hierzu füge im  z.B. data-grid=“b“ ein:

<ul>….</ul>

## Button-Leiste mit Icons

Hierfür stehen folgende Attribute zur Verfügung:

- **data-icon**
- **data-iconpos (top, left, right und bottom)**

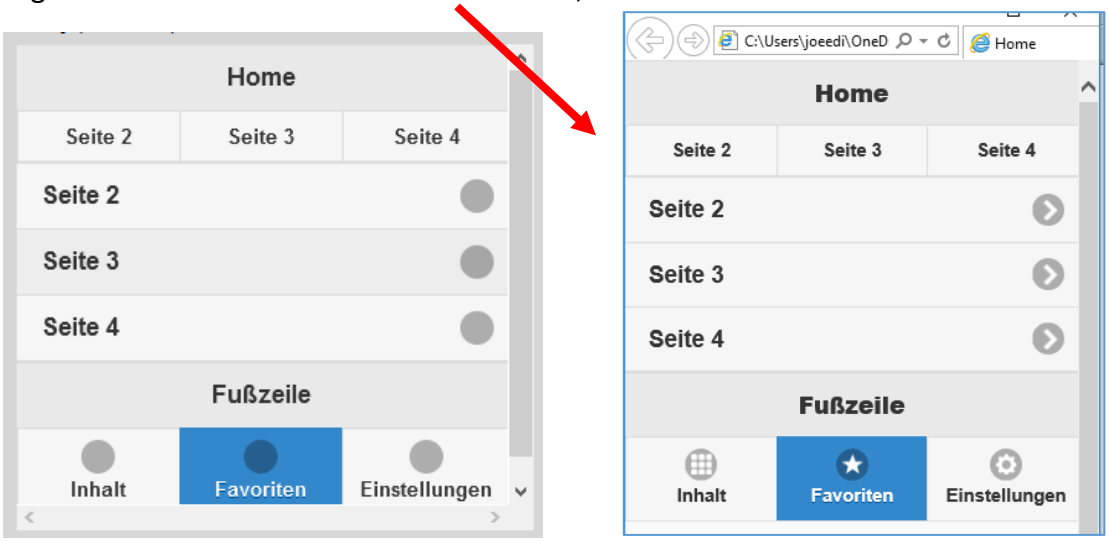
Soll ein Button bereits ausgewählt (vorselektiert) erscheinen, kann man ihm die Klasse „**ui-btn-active**“ geben.

Beispiel für Icons: „grid“, „star“ und „gear“…. 

**Übung**:
Erstelle im Footer eine Navbar für „Inhalt“, „Favoriten“ und „Einstellungen“ inklusive Icons.

```
    <div data-role="footer">
    <h4>Fußzeile</h4>
        <div data-role="navbar">
            <ul>
                <li><a href="#p2" data-icon="grid" data-iconpos="top"> Inhalt </a></li>
                <li><a href="#p3" data-icon="star" data-iconpos="top"
                class="ui-btn-active"> Favoriten </a></li>
                <li><a href="#p4" data-icon="gear" data-iconpos="top">
                Einstellungen </a></li>
            </ul>
        </div> <!--Ende navbar-->
    </div> <!-- Ende -->
</div> <!--Ende page 1-->
```

Ergebnis: Die Icons sind im Browser sichtbar, nicht in der Multi-Screen Vorschau:




## 3b)Persistente Navbar

Im eigentlichen Sinn ist es nicht die Navbar, die persistent ist, sondern der jeweilige Header oder Footer, in dem sie sich befindet. Dieser muss bestimmte Eigenschaften haben, um eine persistente Navigationsleiste zu beherbergen.

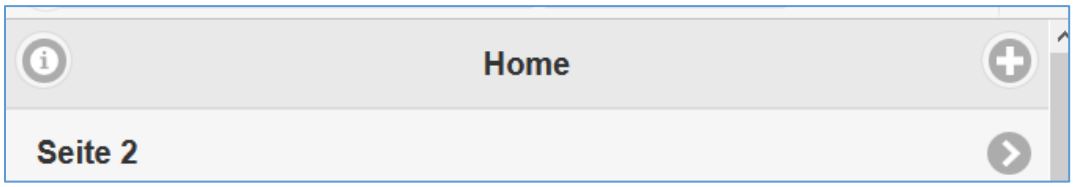
Das data-Attribut „data-position“ muss auf den Wert „fixed“ eingestellt sein. Zusätzlich muss sichergestellt sein, dass der Footer-Bereich der über die Navbar verlinkten Seiten immer die gleiche „data-id“ vergeben bekommt, z.B. data-id=“fixedNavbarFooter“. Um ein Highlighting des aktuell gewählten Navbar-Buttons darzustellen, müssen diesem Link die CSS-Klassen „ui-btn-active“ und „ui-state-persist“ zugewiesen werden.

**Übung:**
Erstelle folgende persistente Navbar für den Footer der „page 2“

```
  <div data-role="footer" id="navbarFooter" data-id="fixedNavbarFoter"
  data-position="fixed">
  <div data-role="navbar">
    <ul>
        <li><a class="ui-btn-active ui-state-persist" href="#">Adam</a></li>
        <li><a href="#">Bertl</a></li>
        <li><a href="#">Cäsar</a></li>
    </ul>
    </div>
  </div>
  <!--Ende footer-->
</div>
<!--Ende page 2-->
```

## 4)Buttons im <header>

Füge im <header> zwei Icons hinzu, eines links und eines rechts. Noch vor der Navbar.




```
<!--Start page 1-->
<div data-role="page" id="p1" data-title="Eigener Titel">
  <div data-role="header" data-position="fixed">
    <h1>Home</h1>
    <a href="neu.html" data-rel="dialog" data-role="button"
    data-shadow="false" data-iconpos="notext" data-icon="plus"
    class="ui-btn-right" id="neueWertung">
    </a>
    <a href="ueber.html" data-rel="dialog"
    data-shadow="false" data-icon="info"
    data-role="button" data-iconpos="notext"
    class="ui-btn-left" id="ueber"></a>
  </div>
  <!--Ende Header-->
```

```
  <h1>Home</h1>
  <a href="neu.html" data-rel="dialog" data-role="button"
  data-shadow="false" data-iconpos="notext" data-icon="plus"
  class="ui-btn-right" id="neueWertung">
  </a>
  <a href="ueber.html" data-rel="dialog"
  data-shadow="false" data-icon="info"
  data-role="button" data-iconpos="notext"
  class="ui-btn-left" id="ueber"></a>
 </div>
```

## 5)Grid-Widget – Inhalte anordnen

Normalerweise nehmen Inhalte im Content-Bereich die volle Breite des Screens ein. Möchte man Spalten darstellen, aber nicht auf Tabellen zurückgreifen, kann man das Grid-Widget (Layout Raster) verwenden, die jQuery Mobile anbietet.

Das Verfahren ähnelt dem „data-grid-Attribut“. Aber hier wird das Raster jedoch rein über CSS-Klassen namens „**ui-grid**“ definiert. Dieses Attribut weist einem Container eine bestimmte Zahl von Spalten zu, z.B.:

```
<div class="ui-grid-a">
....hier sind zwei Spalten
</div>
```

| CSS-Klassen | Wirkung |
|---|---|
| ui-grid-solo | Eine Spalte mit 100% Breite |
| ui-grid-a | Definiert zwei Spalten mit 50% Breite |
| ui-grid-b | drei Spalten mit 33,3% Breite |
| ui-grid-c | vier Spalten mit 25% Breite |
| ui-grid-d | fünf Spalten mit 20% Breite |

Diese Klassen kann man dann jedem Container zuweisen und deklariert ihn damit zum Grid.

Fehlen Blöcke zur definierten Spaltenzahl, so bleibt die entsprechende Anzahl von Layoutblöcken frei.

Ein Grid „ui-grid“ muss zumindest die entsprechende Anzahl an Blöcken enthalten, um eine Zeile zu füllen. Die A-B-Folge kann beliebig wiederholt werden, um mehrzeilige Raster zu erhalten.

**Beispiel:**

ui-grid-a hat zwei Blöcke „**ui-block-a“ und „ui-block-b**“

Der A-Block ist immer der Startblock und die anderen füllen diese Zeile auf. Der A-Block hat nämlich ein „clear:left“ „eingebaut“.

| CSS-Klassen | Wirkung |
|---|---|
| ui-block-a | Erster (einzeiliger) Spaltenblock |
| ui-block-b | Zweiter Spaltenblock |
| ui-block-c | Dritter Spaltenblock |
| ui-block-d | Vierter Spaltenblock |
| ui-block-e | Fünfter Spaltenblock |

## Übung:

Erstelle auf der ersten Seite im <content> einen zweispaltigen Inhalt mit den Bildern aus dem Ordner „img“.

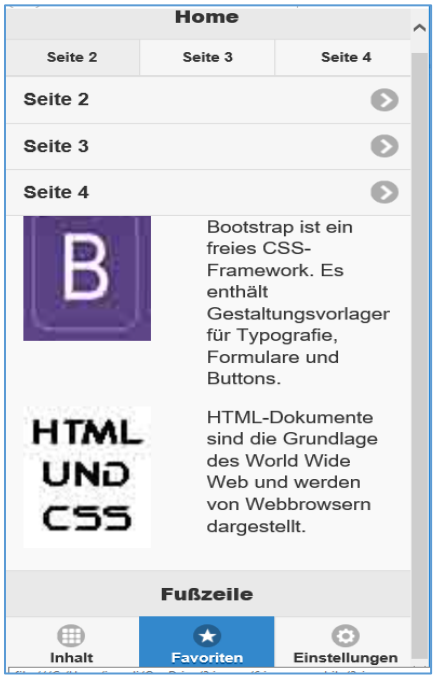

**Text zum Kopieren:**

Bootstrap ist ein freies CSS-Framework. Es enthält Gestaltungsvorlagen für Typografie, Formulare und Buttons

HTML-Dokumente sind die Grundlage des World Wide Web und werden von Webbrowsern dargestellt.

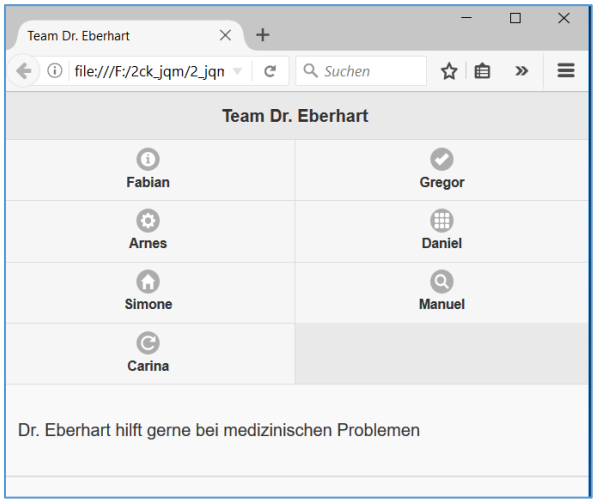
JavaScript (kurz JS) ist eine Skriptsprache, die ursprünglich für dynamisches HTML in Webbrowsern entwickelt wurde, um Benutzerinteraktionen auszuwerten.

```
23      <div data-role="content">
24          <ul data-role="listview">
25              <li><a href="#p2"> Seite 2 </a></li>
26              <li><a href="#p3"> Seite 3 </a></li>
27              <li><a href="#p4"> Seite 4 </a></li>
28          </ul>
29          <div class="ui-grid-a">
30              <div class="ui-block-a">
31                  <img src="img/boot.jpg" width="100" height="135" alt="bootstrap">
32              </div>
33              <div class="ui-block-b"><p>Bootstrap ist ein freies CSS-Framework. Es enthält
34              Gestaltungsvorlagen für Typografie, Formulare und Buttons.</p>
35              </div>
36              <div class="ui-block-a">
37                  <img src="img/html.jpg" width="100" height="135" alt="html">
38              </div>
39              <div class="ui-block-b">HTML-Dokumente sind die Grundlage des World Wide Web
40              und werden von Webbrowsern dargestellt.
41              </div>
42          </div>
43      </div> <!-- Ende content-->
```




**Übung:**

Erstelle folgende Website



# 6)Collapsibles – platzsparende Inhalte

Ein Nachteil mobiler Endgeräte besteht in der kleinen Displayfläche, die verfügbar ist.

## 6a)Collapsibles

Damit kann man Inhaltselemente als ein- und ausklappbar deklarieren und im eingeklappten Zustand laden.

Man muss folgende Eigenschaft verwenden:

**data-role="collapsible"**

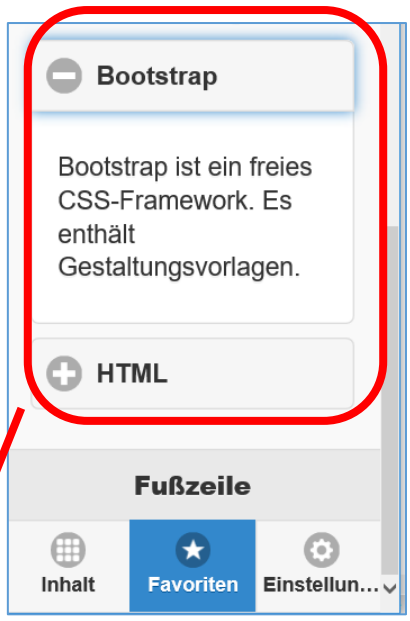
Von Haus aus ist ein Collapsible allerdings sichtbar, also muss es gegebenenfalls über

**data-collapsed="true"**

als eingeklappt angemeldet werden.

**Wichtig:**

Das erste Element eines Collapsibles muss eine **Überschrift** sein. Diese bildet den klickbaren Button, der das Collapsible öffnet und schließt. Ließe man sie weg, so hätte man keine Klickfläche; im eingeklappten Zustand wäre das Collapsible unsichtbar und daher nicht verwendbar. Der Grad der Überschrift spielt hierbei keine Rolle.




Befinden sich mehrere Collapsibles auf einer Mobile-Page, so arbeiten sie dennoch nicht automatisch zusammen. Sie können auch unabhängig voneinander ein- oder ausgeklappt sein.

**Übung:**

Erstelle noch vor dem Ende des <content> mit den Texten von Seite 9 (Bootstrap ist…) folgende zwei Elemente:

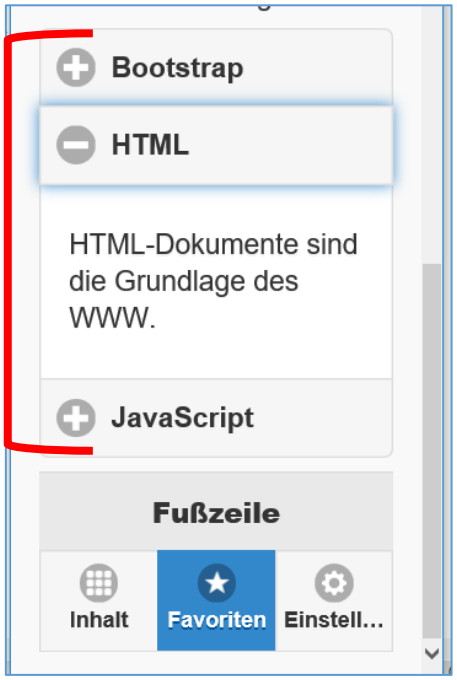
```
    </div>
    <!--Start Collapsible-->
    <div data-role="collapsible" data-collapsed="true">
        <h3>Bootstrap</h3>
        <p>Bootstrap ist ein freies CSS-Framework. Es enthält
          Gestaltungsvorlagen.</p>
     </div>
     <div data-role="collapsible" data-collapsed="true">
        <h3>HTML</h3>
        <p>HTML-Dokumente sind die Grundlage des WWW.</p>
     </div>
    <!--ende collapsible-->
  </div>
  <!-- Ende content-->
```

## 6b)Collapsible-Sets

Um Collapsibles zum Teamwork zu überreden, d.h. zum Gruppieren, muss man sie in einem weiteren Container zusammenfassen, der folgendes Attribut enthält:

**data-role="collapsible-set"**

Damit bilden die Collapsibles ein „Set", von dem stets nur ein Collapsible ausgeklappt sein darf. Öffnet man ein zweites, schließt sich das aktuell geöffnete. Geschlossen sein dürfen jedoch alle Elemente des Sets.




Übung:
Verwende das Collapsible von oben und verändere es:

a)Erstelle einen dritten Eintrag: „JavaScript" und als Inhalt „JavaScript (kurz JS) ist eine Skriptsprache."

b)Ganz oben erfolgt das 

c)Außer dem ersten Eintrag erhalten alle anderen das „data-collapsed="true"

```
50    <!--Start Collapsible-->
51    <div data-role="collapsible-set">
52    <div data-role="collapsible" data-collapsed="true">
53        <h3>Bootstrap</h3>
54        <p>Bootstrap ist ein freies CSS-Framework. Es enthält
55          Gestaltungsvorlagen.</p>
56     </div>
57     <div data-role="collapsible" data-collapsed="true">
58        <h3>HTML</h3>
59        <p>HTML-Dokumente sind die Grundlage des WWW.</p>
60     </div>
61     <div data-role="collapsible" data-collapsed="true">
62        <h3>JavaScript</h3>
63        <p>JavaScript (kurz JS) ist eine Skriptsprache.</p>
64     </div>
65     </div>
66    <!--ende collapsible-->
```

**Tipp**: Ein Collapsible kann auch ein Layout-Grid enthalten (siehe Seite 7-8).

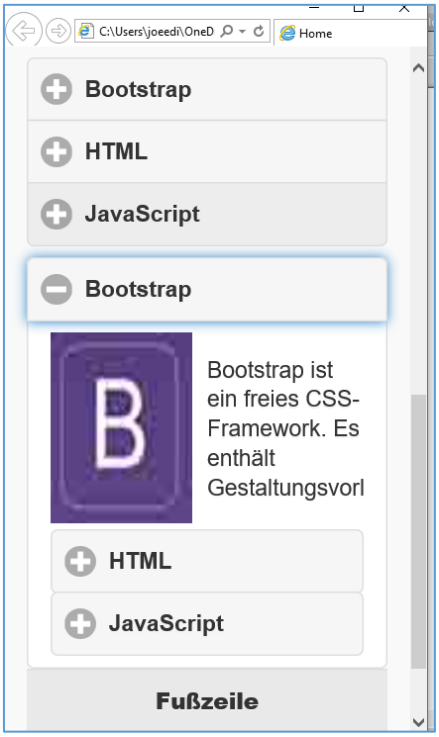
**6c)**Damit in ein **Collapsible ein Layout-Grid eingefügt** werden kann muss folgendes beachtet werden:

Der auf die Überschrift folgende Inhalt des Collapsibles muss ein Grid-Container sein:

**Übung**:

a)Kopiere das „collapsible-sets“, das auf der Seite vorher beschrieben wurde, und füge es darunter ein. Noch bevor der <content> endet.

b)Lösche im Element „Bootstrap“ nach dem <h3> den <p>Bereich, da hier nun das Layout-Grid eingefügt werden soll.
Füge von oben den Bereich ein nämlich Zeile 29-35:




```
58      <!--ende collapsible-->
59      <!--Start beide genmeinsam-->
60      <div data-role="collapsible-set">
61      <div data-role="collapsible" data-collapsed="true">
62          <h3>Bootstrap</h3>
63          <div class="ui-grid-a">
64              <div class="ui-block-a">
65                  <img src="img/boot.jpg" width="100" height="135">
66              </div>
67              <div class="ui-block-b">
68          <p>Bootstrap ist ein freies CSS-Framework. Es enthält
69            Gestaltungsvorlagen.</p>
70       </div>
71      </div>
72      <div data-role="collapsible" data-collapsed="true">
73          <h3>HTML</h3>
74          <p>HTML-Dokumente sind die Grundlage des WWW.</p>
75       </div>
76       <div data-role="collapsible" data-collapsed="true">
77          <h3>JavaScript</h3>
78          <p>JavaScript (kurz JS) ist eine Skriptsprache.</p>
79       </div>
80      <!--ende beide gemeinsam-->
81    </div>
82    <!-- Ende content-->
```

Quellen:

Philipp Friberg, in: Web-Apps mit jQuery Mobile, Mobile Multiplattform-Entwicklung mit HTML5 und JavaScript, dpunkt.Verlag, Heidelberg, 2013