Grundlagen der Smartform-Erstellung

HTML

HTML steht für **Hyper Text Markup Languge** und ist die Standard-Markup-Sprache zur Erstellung von Webseiten. In Kombination mit weiteren Sprachen wie **Cascading Style Sheets (CSS)** und **JavaScript** bildet HTML die Basis der Web-Entwicklung. Dabei wird HTML eingesetzt, um die Struktur von Webseiten festzulegen. HTML gilt demnach nicht als Programmiersprache sondern als Markup-Sprache und ist folglich leicht zu erlernen.
Im Folgenden stellen wir einige für die Smartform-Erstellung nützliche HTML-Elemente vor. Eine ausführliche Dokumentation sowie zahlreiche Tutorials finden Sie unter W3Schools.

CSS

CSS oder Cascading Style Sheets werden eingesetzt um das **Design bzw. die Präsentation** mit HTML erstellter Webseiten zu beeinflussen. CSS ist darauf ausgelegt Struktur und Präsentation einer Webseite zu trennen. In der Regel werden die Styles in einer seperaten .css-Datei definiert.
Die User Experience einer Smartform wird neben der Funktionalität entscheidend durch ein ansprechendes Design beeinflusst. Die visuelle Bearbeitung mit CSS stellt einen wichtigen Bestandteil des Smartform-Designs dar.
Eine ausführliche Dokumentation sowie zahlreiche Tutorials finden Sie unter W3Schools .

JavaScript

Javascript ist eine client-seitige Programmiersprache, die im Browser ausgeführt wird. Während HTML und CSS den statischen Inhalt einer Webseiten darstellen, kann mit JavaScript auf **Benutzerinteraktion** reagiert werden. Des Weiteren Funktionalitäten wie Berechnungen oder Auto-Completes umgesetzt werden.
Eine ausführliche Dokumentation sowie zahlreiche Tutorials finden Sie unter W3Schools .

Planung

Bevor mit der Umsetzung einer Smartform begonnen wird, sollte dessen Struktur, Design und Funktionalitäten sorgfältig geplant werden. Smartforms versorgen einen Prozess mit notwendigen Informationen. Ein guter Ausgangspunkt für die Planung ist, wann welche Informationen im Prozessablauf benötigt werden. Hierbei handelt sich um funktionale Anforderungen an die Smartform. Sind die funktionalen Anforderungen geklärt, müssen diese in einem benutzerfreundlichen Design realisiert werden. Dabei ist zu beachten, welche Benutzer letztendlich mit der Smartform arbeiten müssen. Eine ansprechende Benutzeroberfläche erleichtert die Arbeit mit Smartforms merklich. Allerdings sollte die Funktionalität immer vor ästhetischen Aspekten stehen.
Bei der Planung einer Smartform bietet sich die Umsetzung eines Paper-Prototypen oder Mockups an. Entwürfe können in einem iterativen Prozess mit Endnutzern besprochen und bei Bedarf modifiziert werden. Für die Umsetzung von Mockups und Prototypen gibt es einige hilfreiche Tools wie Balsamiq

Mockup oder Sketch.

Verwendung von Frameworks

Um die Web-Entwicklung komfortabler zu gestalten, wurden in den letzten Jahren zahlreiche HTML, CSS und JavaScript Frameworks entwickelt. In der Smartform-Entwicklung setzen wir bei TIM regelmäßig Bootstrap ein. **Boostrap** stellt vorgefertigte CSS-Klassen zur Verfügung, die die Entwicklung eines ansprechenden sowie responsiven Smartform-Design ermöglichen. Wie Bootstrap in ein Formular eingebunden wird, zeigen wir anhand unseres ausführlichen Enwicklungsbeispiel. Neben **Bootstrap** erfreuen sich Bibliotheken mit Web-Komponenten wie Polymer zunehmender Verwendung.

Neben Bootstrap und Polymer ist die JavaScript-Bibliothek jQuery wichtiger Bestandteil der Smartform-Entwicklung. Die umfangreichen API stellt zahlreiche Funktionen zur Verfügung, z.B. zum Abfangen von User-Input.
Für sehr aufwendige Smartforms mit komplizierten Berechnungen bzw. Anfragen hat sich das von Google entwickelte Framework AngularJS als äußerst nützlich erwiesen.

Natürlich können Sie jederzeit eigene Frameworks entweder über den HTML-Code der Smartform oder

über die custom.js-Datei (wird im Entwicklungsbeispiel erläutert) einbinden.

Ein geeigneter Editor

Ein Smartform lässt sich mit einem einfachen Texteditor erstellen. Allerdings bieten speziell für die Programmierung ausgelegte Editoren zahlreiche Vorteile, wie Zeileneinrückung, Code-Highlighting oder Auto-Vervollständigung.
Beliebte Editoren sind z.B. Notepad++, Brackets und Sublime Text.

Formular erstellen

Um ein Formular zu erstellen sind ein öffnendes <form> und ein schließendes </form> - Tag notwendig. Diese beiden Tags kennzeichnen den Beginn und das Ende des Dokuments. Der darzustellende Inhalt befindet sich zwischen diesen beiden Tags.

```
<form>
</form>
```

Klasse und Name für das Formular

Ein Formular muss eine Klasse und einen Namen beinhalten. Diese werden innerhalb des öffnenden <form> - tags definiert.

```
<form class="klassenname" name="formularname">
</form>
```

Formular für alle TIM user sichtbar machen

Mit dem Attribut security=„all“ wird jedem TIM-User erlaubt das Formular einzusehen.

```
<form class="klassenname" name="formularname" security="all">
</form>
```

Javascript einbinden

Die zu der HTML gehörige Javascript Datei befindet sich innerhalb des JBoss in dem Verzeichnis :

```
.....server\default\deploy\loom.ear\loom-portal.war\custom\
```

In dem Ordner welcher zu dem benutzen Mandanten gehört befindet sich eine Datei mit dem Name **custom.js**. Dies ist eine Javascript Datei, die automatisch in das HTML Formular eingebunden wird und somit für die Smarform benutzbar ist. Sollten für das Formular Javascript Funktionen benötigt werden, so können diese in der Datei formuliert werden.

Mit dem Attribut initMethod=„“ wird angegeben welche JavaScript-Funktion beim Laden der Smartform ausgeführt wird.

```
<form class="klassenname" name="formularname" security="all"
initMethod="FUNKTION DIE BEIM Laden DER SF AUSGEFÜHRT WIRD">
</form>
```

Mit dem Attribut validationMethod=„“ wird angegeben welche JavaScript-Funktion vor dem Speichern der Smartform ausgeführt wird. Eine sinnvolle Anwendung wäre hier zum Beispiel ein Pflichtfeldcheck.

```
<form class="klassenname" name="formularname" security="all"
validationMethod="FUNKTION DIE VORM SPEICHERN AUSGEFÜHRT WIRD">
</form>
```

Die validationMethod muss entweder „true“ oder „false“ zurückgeben. Bei „true“ wird gespeichert, bei „false“ wird nicht gespeichert. Hier ein veranschaulichendes Beispiel:

```
gadget.functions.initValidation=function(){
       if(BEDINGUNG EINFÜGEN){
      // tu was
          alert("Daten wurden gespeichert!");
      return true;
       }
       else{
      // tu was
      alert("Speichern nicht möglich!");
      return false;
       }
}
```

Tabelle erstellen

Spalte1	Spalte2	Spalte3
Spalte1	Spalte2	Spalte3
Spalte1	Spalte2	Spalte3

Der sich öffnenden **table** Tag zeigt den Beginn einer Tabelle an. Innerhalb der darauffolgenden **colgroup** wird für jede Spalte angegeben welche Breite diese besitzen soll. In diesem Fall soll eine Tabelle mit 3 Spalten mit einer Breite von jeweils 100 Pixel erstellt werden.
Der **tr** Tag ist der Startpunkt für eine neue Zeile innerhalb einer Tabelle, wohingegen der **td** Tag einen neue Spalte einleitet. Innerhalb des **td** Tags ist also genau definiert in welcher Zeile und Spalte man sich befindet und es kann hier der gewünschte Inhalt eingegeben werden.

```
<table>
  <colgroup>
    <col width="100px"/>
    <col width="100px" />
    <col width="100px" />
  </colgroup>
  <tr>
    <td>Spalte1 Zeile1</td>
    <td>Spalte2 Zeile1</td>
```

```
    <td>Spalte3 Zeile1</td>
  </tr>
  <tr>
    <td>Spalte1 Zeile2</td>
    <td>Spalte2 Zeile2</td>
    <td>Spalte3 Zeile2</td>
  </tr>
  <tr>
    <td>Spalte1 Zeile3</td>
    <td>Spalte2 Zeile3</td>
    <td>Spalte3 Zeile3</td>
  </tr>
</table>
```

Bild (Logo) einfügen

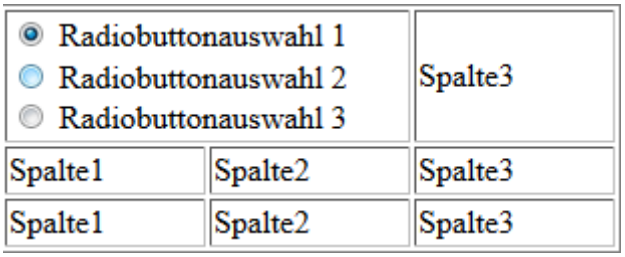
TIM task in motion	Spalte2	Spalte3
Spalte1	Spalte2	Spalte3
Spalte1	Spalte2	Spalte3

Um ein Bild einzufügen wird das Tag **img** benötigt. Diesem müssen die Attribut src=„..“ (hier wird der Pfad zu dem gewünschten Bild angegeben) und das Attribut alt=„..“ (dies wird angezeigt falls das Bild nicht geladen werden kann) mitgegeben werden. In diesem Beispiel soll sich das Bild innerhalb der ersten Spalte befinden. Da das Bild zu groß wäre, kann dieses runterskaliert werden indem zusätzlich das Attribut **width** mitgegeben wird. Diese gibt die Breite des Bildes in Pixel vor.

```
<table>
  <colgroup>
    <col width="100px"/>
    <col width="100px" />
    <col width="100px" />
  </colgroup>
  <tr>
    <td><img src="tim_800.png" width="100" alt="logo"></td>
    <td>Spalte2</td>
    <td>Spalte3</td>
  </tr>
  <tr>
    <td>Spalte1</td>
    <td>Spalte2</td>
    <td>Spalte3</td>
  </tr>
  <tr>
   <td>Spalte1</td>
   <td>Spalte2</td>
   <td>Spalte3</td>
  </tr>
```

```
</table>
```

Auswahlbox (Radiobuttons)



Radiobuttons sind Auswahlfelder die immer zu einer Gruppe zusammengefasst werden. Aus dieser Gruppe kann nur eine Auswahl getroffen werden. Hierzu wird dem „in-sich-schließenden“ <input> - Element der Typ 'radio' zugewiesen. Um mehrere Radiobuttons zu einer Gruppe zusammen zu führen, muss diesem das selbe 'name'-Attribut zugewiesen werden. Falls beim öffnen der Smartform ein Radiobutton bereits gecheckt sein soll, muss diesem das Attribut 'checked=„checked' zugewiesen sein. ACHTUNG! Dieses Attribut darf nur ein Radiobutton je Radiobuttongruppe enthalten. Bei allen <input> - Elementen werden die Prozessvariablen auf die 'id' gesetzt, lediglich bei Radiobuttons wird die Prozessvariable auf das 'name' Attribut gesetzt. Mit dem Attribut colspan=„2“ werden eine Tabellenzelle mit deren nachfolgenden (rechtsliegenden) Tabellenzelle verbunden.

Falls die Auswahl eines Radiobuttons auch über Klicken auf die zugehörige Beschriftung möglich sein soll, muss die Beschriftung von einem öffnenden <label> - und einem schließenden </label> - Tag umschlossen werden. Dieses wird mit dem Attribut 'for=“„' über die 'id' des jeweiligen <input>-Elements angesprochen. Mit einem
 - Element (break) wird ein Zeilenumbruch erzeugt.

```
<table>
  <colgroup>
    <col width="100px"/>
    <col width="100px" />
    <col width="100px" />
  </colgroup>
  <tr>
    <td colspan=2>
      <input type="radio" value="1" name="radiobuttongroup" id="radio_1"
checked="checked" /> <label for="radio_1">Radiobuttonauswahl 1</label><br/>
      <input type="radio" value="2" name="radiobuttongroup" id="radio_2"/>
<label for="radio_2">Radiobuttonauswahl 2</label><br/>
      <input type="radio" value="3" name="radiobuttongroup" id="radio_3"/>
<label for="radio_3">Radiobuttonauswahl 3</label>
    </td>
    <td>Spalte3</td>
  </tr>
  <tr>
    <td>Spalte1</td>
    <td>Spalte2</td>
    <td>Spalte3</td>
```

```
  </tr>
  <tr>
    <td>Spalte1</td>
    <td>Spalte2</td>
    <td>Spalte3</td>
  </tr>
</table>
```

Auswahlbox (Checkboxes)

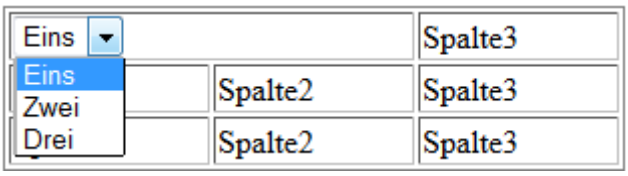
☑ Checkbox 1 ☐ Checkbox 2 ☑ Checkbox 3		Spalte3
Spalte1	Spalte2	Spalte3
Spalte1	Spalte2	Spalte3

Checkboxen sind ebenfalls Auswahlfelder, können jedoch einzeln und unabhängig voneinander angeklickt werden. Ansonsten gelten für diese die gleichen Regeln wie für <radio> Buttons.

```
<table>
  <colgroup>
    <col width="100px"/>
    <col width="100px" />
    <col width="100px" />
  </colgroup>
  <tr>
    <td colspan=2>
      <input type="checkbox" name="check1" id="check1" value="true"/> <label
for="check1">Checkbox 1</label><br/>
      <input type="checkbox" name="check2" id="check2" value="true"/> <label
for="check2">Checkbox 2</label><br/>
      <input type="checkbox" name="check3" id="check3" value="true"/> <label
for="check3">Checkbox 3</label>
    </td>
    <td>Spalte3</td>
  </tr>
  <tr>
    <td>Spalte1</td>
    <td>Spalte2</td>
    <td>Spalte3</td>
  </tr>
  <tr>
    <td>Spalte1</td>
    <td>Spalte2</td>
    <td>Spalte3</td>
  </tr>
```

```
</table>
```

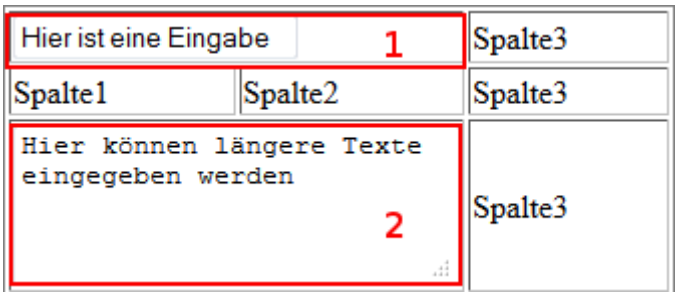
Auswahlfeld (Selectbox)



Um eine Selectbox zu erstellen, sind ein öffnendes <select> und ein schließendes </select> - Tag notwendig. Um diesem Optionen zur Auswahl hinzuzufügen, können zwischen den <select> - Tags beliebig viele **options** hinzugefügt werden. Jede Option benötigt das Attribut 'value=“„' dessen Inhalt unterschiedlich zum Text zwischen den <option> - Tags ist und nicht angezeigt wird. Über das 'value=“„' - Attribut werden in TIM Prozessvariablen gesetzt und abgefragt.

```
<table>
  <colgroup>
    <col width="100px"/>
    <col width="100px" />
    <col width="100px" />
  </colgroup>
  <tr>
    <td colspan=2>
      <select id="Auswahl" name="Auswahl">
         <option value="1">Eins</option>
         <option value="2">Zwei</option>
         <option value="3">Drei</option>
      </select>
    </td>
    <td>Spalte3</td>
  </tr>
  <tr>
    <td>Spalte1</td>
    <td>Spalte2</td>
    <td>Spalte3</td>
  </tr>
  <tr>
    <td>Spalte1</td>
    <td>Spalte2</td>
    <td>Spalte3</td>
  </tr>
</table>
```

Eingabefelder



Es gibt verschiedene Arten von Eingabefelder in die freier Text geschrieben werden kann. Für ein normales Textfeld wird der Tag **input** in Kombination mit dem Attribut **type=„text“** benutzt. Dies bildet ein einfaches Textfeld mit einer Zeile. (siehe Element 1)
Um ein großes Eingabefeld (z.B. für Beschreibungen) zu erstellen, sind ein öffnendes <textarea> - und ein schließendes </textarea> - Tag notwendig. Diese kann in der Größe vom Benutzer angepasst werden und es können mehrere Zeilen hineingeschrieben werden. (Siehe Element 2)

```
<table id="information" class="irgendwas" border="1">
  <colgroup>
   <col width="100px"/>
   <col width="100px" />
   <col width="100px" />
  </colgroup>
  <tr>
    <td colspan=2>
     <input type="text" id="eingabeFeld" name="eingabeFeld"></input>
    </td>
    <td>Spalte3</td>
  </tr>
  <tr>
    <td>Spalte1</td>
    <td>Spalte2</td>
    <td>Spalte3</td>
  </tr>
  <tr>
    <td colspan=2>
      <textarea id="textarea" name="textarea"></textarea>
    </td>
    <td>Spalte3</td>
  </tr>
</table>
```

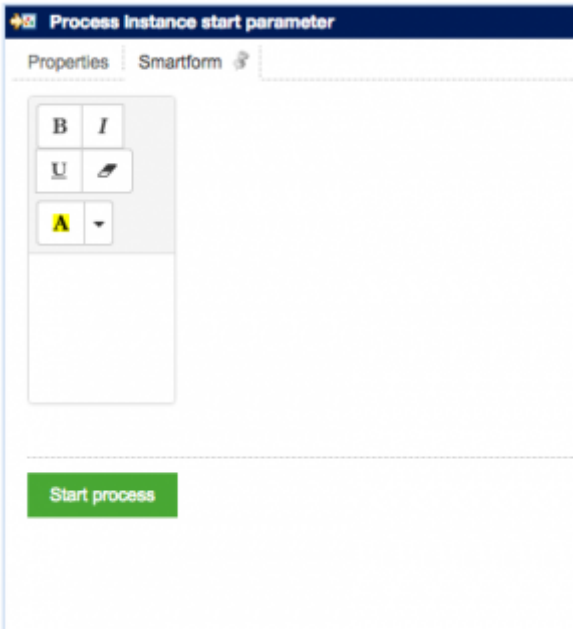
Formatierungen in Textareas

Um die Formatierungselemente kursiv, fettdruck, unterstrichen, blaue Schriftfarbe und rote Schriftfarbe in einer Textarea bereitszustellen muss die Klasse **richtext** mitgegeben werden.

Achtung! Auch der Name der Variable muss richtext enthalten und Bootstrap muss mit geladen

werden (data-bootstrap=„true“ im <form> mitgeben).

```
<textarea class="richtext" id="richtext_name_der_variable"></textarea>
```

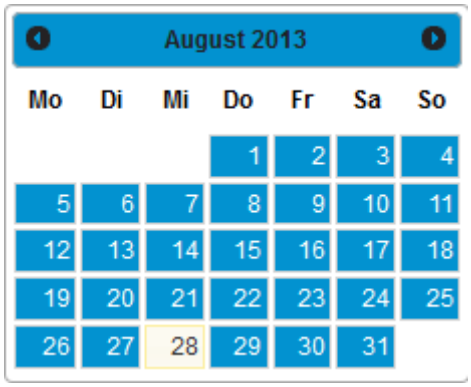


Datepicker (Kalender)

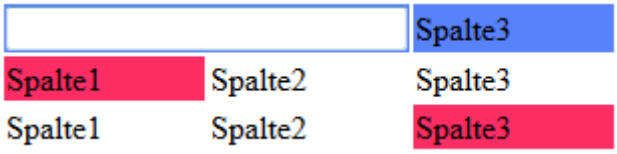
Um einen sogenannten Datepicker auf ein Eingabefeld zu legen muss lediglich diesem die Klasse **datepicker** mitgegeben werden.

```
 <input type="text" id="datepicker_field" name="datepicker_field"
class="datepicker"></input>
```

Die Javascript Bibliothek **JQuery** welche automatisch in jede Smartform eingebunden wird, öffnet nun bei Klick in das Feld einen Kalender aus welchem ein Datum ausgewählt werden kann, welches anschließend in das Eingabefeld übernommen wird.



Stylesheets hinzufügen (CSS)



Zwischen einem öffnenden <style> und einem schließenden </style> - Tag werden die Stylesheets definiert. Dem öffnenden <style> - Tag muss das Attribut 'type=„text/css“' zugewiesen werden.
Um Elemente anzusprechen die eine Klasse besitzen, muss der Klassenname mit einem vorgestellten Punkt hinzugefügt werden. Innerhalb der folgenden geschweiften Klammern kann nun definiert werden wie das Element beeinflusst werden soll. Die Angaben über das Aussehen betreffen alle Elemente die eine Klasse mit diesem Name besitzen.
Um ein Element mit einer bestimmten ID anzusprechen muss die ID mit eine vorgestellten Doppelkreuz(#) hinzugefügt werden. Alle Elemente die diese ID besitzen richten ihr Aussehen nun nach der folgenden Definition.
Sollen alle Elemente eines bestimmten Typs angesprochen werden so muss einfach der Tag des Elements hinzugefügt werden. In diesem Fall **input**.

In dem folgenden Beispiel bekommen die erste Spalte in der zweiten Zeile und die dritte Spalte in der dritten Zeile die Klasse **background** mit. Wie in dem Style Bereich definiert, wird der Hintergrund dieser beiden Zellen rot eingefärbt.
Nachfolgend werden alle **input** Elemente auf eine Breite von 200 Pixel gesetzt.
Alle Elemente mit der ID **ersteZeile** bekommen einen blauen Hintergrund.
Schließlich wird nun der Rahmen der Tabelle auf eine Breite von 0 Pixel gesetzt und verschwindet damit.

```
<style type="text/css">
  .background{
    background-color:#FE2E64;
  }
  input{
    width:200px;
  }
  #ersteZeile{
    background-color:#5882FA;
  }
  #tabelle{
    border-size:0px;
  }
</style>
<table id="tabelle">
  <colgroup>
    <col width="100px"/>
    <col width="100px" />
    <col width="100px" />
  </colgroup>
  <tr id="ersteZeile">
    <td colspan=2>
       <input type="text" id="eingabeFeld" name="eingabeFeld"></input>
```

```
    </td>
    <td>Spalte3</td>
  </tr>
  <tr>
    <td class="background">Spalte1</td>
    <td>Spalte2</td>
    <td>Spalte3</td>
  </tr>
  <tr>
    <td>Spalte1</td>
    <td>Spalte2</td>
    <td class="background">Spalte3</td>
  </tr>
</table>
```

Pflichtfelder beim Instanzstart

Wenn ein Eingabefeld bereits beim Instanzstart als Pflichtfeld definert werden soll, ist dass Attribut „required“ mit dem Wert „true“ notwendig.

```
<input type="text" id"pflichtfeld" name="pflichtfeld" required="true" />
```

Wenn ein Auswahlfeld(Selectbox) zum Pflichtfeld gemacht wird und die Grundoption (z.B. „Bitte auswählen...“) als nicht gültiger Wert definiert werden soll,
muss diese Option das Attribut default=„default“ erhalten.

```
<select>
<option default="default">Bitte auswählen...</option>
<option>Wert 1</option>
...
</select>
```