

Web-Informationssysteme, WS 2009/10

Seminar-Übung 1: HTML, CSS, Topologie des Webs

Besprechung am Mi 28.10.2009

— SÜ-1.1 —

Was Ihr wissen müßt über “HTML & CSS”

Wiederholung

SÜ-1.1.1 HTML

- Was ist der Unterschied zwischen **spezifischem**, **generalisiertem** und **generischem** Markup?
- Was sind die wesentlichen Unterschiede zwischen den **HTML Versionen** HTML 4.x, XHTML 1.x, HTML5 und XHTML5.

HTML 4.x SGML-Anwendung, daher volle SGML-Markup-Minimierung, case-insensitive,
XHTML 1.x XML-Anwendung, also keine Markup-Minimierung, case-sensitive, ansonsten gleich wie HTML 4.x.
HTML5 Runderneuerung von HTML 4.x: mehr generalisierter, weniger spezifischer Markup, Standardisierung des Event-Modells und des dynamischen Verhaltens von HTML-Dokumenten, Einbindung von Video & Audio, Canvas-Element zum Freiform-Zeichnen; wie HTML 4.x case-insensitive, erlaubt eingeschränkte Markup-Minimierung; weder SGML noch XML.
XHTML5 XML-Anwendung, ansonsten gleich wie HTML5.

- Nennen Sie **Beispiele** für spezifisches und generalisiertes Markup in HTML!

b, tt vs. p, table

- Wie wird **generisches Markup in HTML** “nachgebildet”?

Mit Hilfe von class-Attributen kann man eine Form von generischen Markup in HTML erreichen. Ein Schema kann man jedoch für solches Markup nicht angeben. Ein weiterer Unterschied zu echten generischen Markup-Sprachen wie XML ist das dasselbe Element Teil verschiedener Klassen sein kann, z.B. `<p class="navigation content header"></p>`

sÜ-1.1.2 CSS

- Was sind die **Bestandteile einer CSS Regel**?

- Selektor { Deklaration }.
- Deklaration ist ein Paar aus Eigenschaftsnamen und -wert.
- Gruppierung von Selektoren (mit gleicher Deklaration) durch “,“, Gruppierung von Deklarationen (mit gleichem Selektor/Selektoren) durch “;”.

EXAMPLE CSS REGEL:

```
h1, h2 > em { color: green; font-weight: bold }
```

- Wie ist ein **CSS Selektor** aufgebaut?

- Sequenz von einfachen Selektoren durch Kombinatoren verbunden, möglicherweise durch Pseudo-Element abgeschlossen.

EXAMPLE CSS SELEKTOR:

```
h1#nav.menu ol + *[rel="up"]
```

Illegal: h1::first-line em

– **Kombinatoren:**

- * (default, “ ”): *descendant*-Kombinator

```
h1 em matches <h1><em>match</em></h1> und <h1><span><em>match</em></span></h1>
```

- * “>”: *child*-Kombinator

```
h1 em matches <h1><em>match</em></h1> aber nicht <h1><span><em>match</em></span></h1>
```

- * “+”: *adjacent sibling*-Kombinator

```
h1 + p matches <h1></h1><p></p> aber nicht <h1></h1><h2></h2><p></p>
```

- * “~”: *general sibling*-Kombinator

```
h1 ~ p matches <h1></h1><p></p> und <h1></h1><h2></h2><p></p>
```

- Was sind die **Typen von CSS Selektoren**?

– **Einfache Selektoren:**

Typ-Selektoren: h1, html | h1, universal *

```
h1 matches <h1></h1>
```

Attribut-Selektoren: [rel="up"], rel

```
a[rel="up"] matches <a rel="up"></a>
```

Klassen-Selektoren: .content

```
.content matches <div class="content"></div> und <div class="content nav"></div>
```

ID-Selektoren: #nav

```
#nav matches <p id="nav"/>
```

Pseudo-Klassen-Selektoren: :first-child, :nth-child(2n+3), :not(a)

Zwischen zwei Kombinatoren in einem CSS Selektor kann nur ein (äußerer) Typ-Selektor, aber eine beliebige Sequenz von Attribut, Klassen, ID, und Pseudo-Klassen-Selektoren vorkommen.

– **Pseudo-Elemente:** ::first-line, ::first-letter

Warum sind Pseudo-Elemente unterschieden von "Einfachen Selektoren"? Die aktuelle CSS3 Spezifikation erlaubt nur **ein** Pseudo-Element pro Selektor, das nur am Ende der Folge von einfachen Selektoren auftreten darf.

- Ihr solltet für jeden CSS Selektor angeben können, **welche Elemente er selektiert**.

siehe oben

- Wie wird, z.B. in der folgenden CSS Regelmenge, entschieden, welche der beiden Regeln **Präzedenz** hat (also ob das p grau oder blau wird)?

```
<p id="abstract" class="content nav">This paper ...</p>
```

```
p.content { color: gray }
2 p#abstract { color: darkblue }
```

Die Präzedenz wird durch eine Selektivitäts-Abschätzung des CSS Selektors der jeweiligen Regel erreicht. Die Regel mit dem Selektor der höheren geschätzten Selektivität hat Vorrang.

Die Spezifität von CSS3 Selektoren wird durch die folgende Formel definiert (nach <http://www.w3.org/TR/css3-selectors/#specificity>):

$$\text{spec}(s) = |\text{ID}(s)| \cdot b^2 + (|\text{class}(s)| + |\text{attr}(s)| + |\text{pseudocl}(s)|) \cdot b + |\text{type}(s)| + |\text{pseudoe}(s)|$$

$\text{ID}(s)$ = ID Selektoren in s (z.B. #nav)

$\text{class}(s)$ = Class-Selektoren in s (z.B. .menu)

$\text{attr}(s)$ = Attribut-Selektoren in s (z.B. [rel])

$\text{type}(s)$ = Typ-Selektoren in s (z.B. ul) ohne *

$\text{pseudocl}(s)$ = Pseudo-Klassen in s (z.B. :root) ohne not

$\text{pseudoe}(s)$ = Pseudo-Elemente in s (z.B. ::first_line)

$$b = \max\{|\text{ID}(s)|, |\text{class}(s)| + |\text{attr}(s)| + |\text{pseudocl}(s)|, |\text{type}(s)| + |\text{pseudoe}(s)|\}$$

- Ihr solltet wichtige **Eigenschaften** wie color zumindest passiv beherrschen!

— SÜ-1.2 —

HTML5 Canvas—GUI-Toolkit für das Web

Programmierung

In Anlage `canvas.html` findet Ihr das Gerüst für eine Web-Seite mit einem HTML5 Canvas Element (`#progressCanvas`).

Ergänzen Sie diese HTML5 Dokument, so dass

1. Das Progress Objekt einen Fortschrittsbalken mit Wert `value` und Maximum `max` repräsentiert, der sich, beim Aufruf der Funktion `Draw()`, in dem gegebenen Canvas zeichnet. Beim Klicken auf den Canvas soll der Fortschrittsbalken vom 25 Werte vorgestellt werden (rotierend).
2. Dazu sollte es ausreichen an den mit `@TODO` markierten Stellen Euren eigenen Code ergänzen. Insbesondere also die Funktionen `DrawBar()` zum Zeichnen des eigentlichen Fortschrittsbalken und `DrawTickMarks()` zum Zeichnen der Skalenmarken.

Ihr finden Tipps zum Umgang mit Canvas u.a. unter:

- Mihai Sucan. *HTML 5 Canvas—The Basics*. Jan 2009.
<http://dev.opera.com/articles/view/html-5-canvas-the-basics/>
- Mihai Sucan. *Creating an HTML 5 canvas painting application*. Mar 2009.
<http://dev.opera.com/articles/view/html5-canvas-painting/>
- Mozilla Developer Center. *Canvas Tutorial*. Aug 2009.
https://developer.mozilla.org/en/Canvas_tutorial

Hinweis: Zum Testen müßt Ihr einen Browser mit HTML5-Canvas Unterstützung verwenden, also eine aktuelle Version von Firefox, Safari, Chrome, Opera, etc.

Erzeugung des Progress Objekts und Event-Handler Set-Up

```
<script type="text/javascript">
2   window.onload = function ()
    {
4       // * It's good to always check, that the browser actually supports HTML canvas:
        // Get a reference to the element.
6       var elem = document.getElementById('progressCanvas');

8       // Always check for properties and methods, to make sure your code doesn't break
        // in other browsers.
10      if (elem && elem.getContext) {

12          var myProgress = new Progress('progressCanvas', 187, 500);

14

16      myProgress.Set('chart.color', 'lightblue');
        myProgress.Draw();

18      /**
        * Install the onclick event handler to increase progress bar.
20      */
        myProgress.canvas.onclick = function (e)
22      {
24          var canvas = document.getElementById(this.id);
            var myProgress = canvas.__object__;

26          myProgress.Set('chart.value', myProgress.Get('chart.value') + 25);
            if (myProgress.Get('chart.value') == myProgress.Get('chart.max') + 25) {
28                myProgress.Set('chart.value', 0);
            } else if (myProgress.Get('chart.value') > myProgress.Get('chart.max')) {
30                myProgress.Set('chart.value', myProgress.Get('chart.max'));
            }

32          myProgress.context.clearRect(0,0, canvas.width, canvas.height);
            myProgress.Draw();

34

36          /**
            * Stop the event bubbling
38          */
            e.stopPropagation = true;
40            e.cancelBubble = true;
        }
42    }
}
```

Progress Konstruktor

```
/**
2  * The progress bar constructor
   *
4  * @param int id    The ID of the canvas tag
   * @param int value The indicated value of the meter.
6  * @param int max   The end value (the upper most) of the meter
   */
8  Progress = function (id, value, max)
   {
10     this.id      = id;
       this.canvas = document.getElementById(id);
12     this.context = this.canvas.getContext('2d');
       this.canvas.__object__ = this;
14
       this.properties = new Array();
16     this.properties['chart.max']      = max;
       this.properties['chart.value']   = value;
18     this.properties['chart.color']    = '#0c0';
       this.properties['chart.tickmarks'] = true;
20     this.properties['chart.tickmarks.color'] = 'black';
       this.properties['chart.gutter']   = 25;
22     this.properties['chart.numticks']  = 10;
       this.properties['chart.orientation'] = 'horizontal';
24     this.properties['chart.background.color'] = '#eee';
       this.properties['chart.shadow']    = true;
26     this.properties['chart.shadow.color'] = 'rgba(0,0,0,0.5)';
       this.properties['chart.shadow.blur'] = 3;
28     this.properties['chart.shadow.offsetx'] = 3;
       this.properties['chart.shadow.offsety'] = 3;
30
       // Check for support
32     if (!this.canvas) {
         alert('[PROGRESS] No canvas support');
34         return;
       }
36   }
```

Progress.Draw-Methode

```
1  /**
   * Draws the progress bar
   */
3  Progress.prototype.Draw = function ()
5  {
   // Figure out the width and height
7  this.width = this.canvas.width - (2 * this.Get('chart.gutter'));
   this.height = this.canvas.height - (2 * this.Get('chart.gutter'));
9
   this.Drawbar();
11  this.DrawTickMarks();
13 }
```

Progress.DrawBar-Methode

```
1  /**
2   *
3   */
4  Progress.prototype.Drawbar = function ()
5  {
6      // Set a shadow if requested
7      if (this.Get('chart.shadow')) {
8          this.context.shadowColor = this.Get('chart.shadow.color');
9          this.context.shadowBlur = this.Get('chart.shadow.blur');
10         this.context.shadowOffsetX = this.Get('chart.shadow.offsetx');
11         this.context.shadowOffsetY = this.Get('chart.shadow.offsety');
12     }
13
14     // Draw the outline
15     this.context.fillStyle = this.Get('chart.background.color');
16     this.context.strokeStyle = 'black';
17     this.context.strokeRect(this.Get('chart.gutter'), this.Get('chart.gutter'), this.width, this.height);
18     this.context.fillRect(this.Get('chart.gutter'), this.Get('chart.gutter'), this.width, this.height);
19
20     // Turn off any shadow
21     this.context.shadowColor = 'rgba(0,0,0,0)';
22     this.context.shadowBlur = 0;
23     this.context.shadowOffsetX = 0;
24     this.context.shadowOffsetY = 0;
25
26     this.context.fillStyle = this.Get('chart.color');
27     this.context.strokeStyle = 'black';
28
29     // Draw the actual bar itself
30     if (this.Get('chart.orientation') == 'horizontal') {
31         var barWidth = Math.min(this.width, (this.Get('chart.value') / this.Get('chart.max')) * this.width);
32         this.context.strokeRect(this.Get('chart.gutter'), this.Get('chart.gutter'), barWidth, this.height);
33         this.context.fillRect(this.Get('chart.gutter'), this.Get('chart.gutter'), barWidth, this.height);
34     } else {
35         var barHeight = Math.min(this.height, (this.Get('chart.value') / this.Get('chart.max')) *
36             this.height);
37         this.context.strokeRect(this.Get('chart.gutter'), this.Get('chart.gutter') + this.height -
38             barHeight, this.width, barHeight);
39         this.context.fillRect(this.Get('chart.gutter'), this.Get('chart.gutter') + this.height - barHeight,
40             this.width, barHeight);
41     }
42 }
```


Progress.DrawTickMarks-Methode

```

1  /**
   * The function that draws the tick marks. Apt name...
3  */
   Progress.prototype.DrawTickMarks = function ()
5  {
       this.context.strokeStyle = this.Get('chart.tickmarks.color');
7
       if (this.Get('chart.tickmarks') && this.Get('chart.orientation') == 'horizontal') {
9
           // This is used by the label function below
           this.tickInterval = this.width / this.Get('chart.numticks');
11
           for (var i=this.Get('chart.gutter') + this.tickInterval; i<=(this.width +
               this.Get('chart.gutter')); i+=this.tickInterval) {
13
               this.context.moveTo(i, this.Get('chart.gutter') + this.height);
               this.context.lineTo(i, this.Get('chart.gutter') + this.height + 4);
15
           }
17
       } else if (this.Get('chart.tickmarks') && this.Get('chart.orientation') == 'vertical') {
19
           this.tickInterval = this.height / this.Get('chart.numticks');
21
           for (var i=this.Get('chart.gutter'); i<=(this.canvas.height - this.Get('chart.gutter') -
               this.tickInterval); i+=this.tickInterval) {
23
               this.context.moveTo(this.canvas.width - this.Get('chart.gutter'), i);
               this.context.lineTo(this.canvas.width - this.Get('chart.gutter') + 4, i);
25
           }
27
       }
       this.context.stroke();
29
   }
31 </script>
   </body> </html>

```

Wie bei anderen GUIs auch wird man in der Praxis wohl meist auf eine Bibliothek zurückgreifen. Zum Erstellen von Graphen in Canvas Objekten gibt es diese auch bereits zahlreich:

- RGraph: A canvas graph library based on the HTML5 canvas tag
<http://www.rgraph.net/>. Davon haben wir den größten Teil des Codes “abgeschaut”.
- PlotKit - Javascript Chart Plotting.
<http://www.liquidx.net/plotkit/>
- Scott Jehl. *JQuery Visualize Plugin*. Sep 2009.
http://www.filamentgroup.com/lab/jquery_visualize_plugin_accessible_charts_graphs_from_tables_html5_canvas/