EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Mathematisch-Naturwissenschaftliche Fakultät
Fachbereich Informatik
Informationsdienste – Prof. Dr. Thomas Walter

Einführung Internet-Technologien

Sommersemester 2020

Übungsblatt 2

Theorie

Die Lösungen zu den Theorieaufgaben werden in den Tutorien besprochen und im Moodle veröffentlicht. Sie brauchen *keine* Lösungen zu den Theorieaufgaben abgeben!

1. Welches Problem bestand bei CSS2 bezüglich mehrspaltigem Layout? Wie löst der CSS3-Standard dies?

   Lösung: Eines der Hauptprobleme mehrspaltiger Layouts bestand in der Höhenpositionierung mehrerer Spalten. Wenn die Spalten unterschiedliche Höhen hatten hat der Browser das Layout der kleineren Spalte nur auf deren Höhe angewandt und die größere Spalte noch weiter gerendert. Somit wurde die Anzeige in teils unansehlich zerstückelt. Die häufig verwendeten aber unsauberen Lösungen waren die Nutzung von HTML-Tabellen oder sich wiederholende Hintergrundbilder, die den leeren Raum auffüllten.

   CSS3 löst das Problem über das *column-count*-Attribut. Über dieses kann einem Element mitgegeben werden, wie viele Spalten es für seine Kind-Elemente zur Verfügung hat. Die Render-Engine des Browsers kümmert sich dann um die korrekte Höhenpositionierung der Spalten und der Inhalte. Ein Beispiel hierfür ist in Abbildung **??** angegeben.

```
<style>
.mcol{ column-count: 3; }
</style>

<div class="mcol">
  Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam.
</div>
```

Abbildung 1: Beispiel für 3-Spaltiges Layout über das column-count-Attribut

   Ebenfalls kann das display-Attribut mit den neuen Werten *table*, *table-row* und *table-cell* verwendet werden um Tabellen-, Grid- und Rasterlayouts zu erzeugen. Dies erlaubt es gezielter zu Bestimmen welche Inhalte in welcher Spalte angezeigt

werden, die Render-Engine kann dennoch die gemeinsame Höhe ermitteln und der Entwickler kann sich an den Standard halten ohne unsaubere Workarounds zu verwenden. Ein Beispiel hierfür ist in Abbildung **??** dargestellt.

```
<style>
.row{ display: table-row; }
.cell{ display: table-cell; }
</style>

<div class="row">
 <div class="cell"><h1>First</h1><p>Lorem ipsum dolor sit amet </p></div>
 <div class="cell"><h2>Second</h2><p>euismod tincidunt ut </p></div>
 <div class="cell"><h3>Third</h3><p>minim veniam, quis nostrud </p></div>
</div>
```

Abbildung 2: Beispiel für 3-Spaltiges Layout über das display-Attribut

2. Erklären Sie die CSS Keyframe-Technologie.

Lösung: Die Keyframe-Technik erlaubt es ausschließlich auf CSS basierende Animationen in seinen Webseiten einzubetten. Die Syntax für ein Keyframe ist folgende: keyframes animationname keyframes-selector css-styles; Der Animationsname muss dabei eindeutig für die Webseite sein um die Animationsbefehle einem Element zuordnen zu können. Der Keyframe-Selektor gibt an, nach welcher Zeit die angegebenen CSS-Stylings im Browser angezeigt werden sollen. Es kann dabei beliebig viele Keyframe-Selektoren in einer Animation geben. Die Angabe der Zeitgrenze erfolt in % zur gesamten Animationszeit. Letztere wird jedoch erst als Wert des CSS-Animationsattributs im animierten Element angegeben. Die CSS-Styles entspricht dem Layout des Elements zur Ausführungszeit. Die Änderung des Layouts zwischen zwei Keyframe-Selektoren wird automatisiert von der Render-Engine gleichverteilt über den Animationszeitraum angepasst. Auf diese Weise entstehen gleichmäßig animierte Bewegungen auf dem Bildschirm. Ein Beispiel ist in Abbildung **??** dargestellt.

Um ein HTML-Element über CSS zu animieren muss diesem Element das CSS-Attribut *animation* zugeordnet werden. Dieses hat folgenden Syntax: animation: name duration timing-function delay iteration-count direction fill-mode play-state; Der name entspricht dabei dem animationname. Die Anzeigedauer wird über den duration-Wert eingestellt (z.B 5s oder 100ms). Die timing-function gibt eine Funktion der Animationsgeschwindigkeit an (z.B. linear, cubic-bezier(0,0.1,0.8,1)). Die Animationsausführung kann über den delay-Wert verzögert werden. Wie oft die Animation wiederholt werden soll, wird über den iteration-count angegeben (beispielsweise infinite für unendlich). Die Reihenfolge in der die Animation ausgeführt wird, kann über den direction-Wert definiert werden (beispielsweise normal oder alternate). Der fill-mode-Wert gibt an, was mit einem Element geschehen soll,

```
<style>
div {
    width: 100px; height: 100px; position: relative;
    animation: mymove 5s cubic-bezier(0,0.1,0.8,1) infinite alternate;
}

/* Standard syntax */
@keyframes mymove {
    0%   {top: 0px; background-color: red;}
    25%  {top: 200px; background-color: green;}
    75%  {top: 50px; background-color: blue;}
    100% {top: 100px; background-color: purple;}
}
</style>
<div></div>
```

Abbildung 3: Beispiel für eine CSS-Keyframe-Animation

nachdem die Animation erfolgt ist (beispielsweise forwards um es an seiner Position zu halten oder backwards um es zu seiner Ausgangsposition zurückzusetzen). Abschließend kann über den play-state-Wert die Animationsausführung manipuliert werden (beispielsweise wird sie über den Wert running abgespielt und über den Wert paused angehalten). Ein Beispiel ist in Abbildung **??** dargestellt. Dieses CSS-Attribut muss im Styling des zu animierenden HTML-Elements enthalten sein; beispielsweise über eine Klasse oder ID zugeordnet.

3. Wozu dient *Responsive Design* im Webumfeld?

Lösung: Im Webumfeld kann serverseitig nicht verlässlich geprüft werden, auf welchem Gerät eine Darstellung angezeigt wird. Um das Problem zumindest zu reduzieren existiert der Responsive Design-Ansatz. Bei diesem wird die Seite in atomar darstellbare Elemente unterteilt, welche in Abhängigkeit von den verfügbaren Abmaßen des anzeigenden Endgeräts positioniert werden. Dabei werden die Design-Angaben direkt zum Client gesendet und dieser kann auf Basis seines Wissens das Rendering übernehmen. Dabei kann der Client zudem bei Änderungen der Abmaße (z.B. beim kleinziehen des Browsers) direkt auf die Änderungen reagieren und die Inhalte neu rendern.

4. Erklären Sie das *Content-is-like-water*-Paradigma in Bezug auf Responsive Design.

Lösung:

Das Content-is-like-water Paradigma (siehe Abbildung **??**) wird zur Veranschaulichung des Inhalts-Präsentations-Problems im Web genutzt: Ein Design wird für ein bestimmtes Ausgabemedium erstellt und ist für dieses optimiert. Auf anderen Ausgabemedien kann dieses Design Probleme bereiten, wenn das Konzeptmedium beispielsweise größer als das anzeigende ist. Das Paradigma argumentiert das

Abbildung 4: Content-is-like-water Paradigma

Inhalt sich wie Wasser verhalten sollte: Er muss sich an die ihn umgebenden Bedingungen anpassen und nicht die ihn umgebenden Bedingungen an sich. Dieses Paradigma liegt dem Responsive-Design Ansatz zugrunde: Inhalt wird auf atomare Elemente runtergebrochen. Wie Wasserstoff-Atome sind dies die kleinsten, logisch zusammenhängenden Elemente, die nicht weiter unterteilt werden können. Durch stückweises Einfügen dieser atomaren Elemente in eine Anzeige kann die Render-Engine die Elemente optimiert auf das Anzeige positionieren. Somit passt sich der Inhalt an die Möglichkeiten der Anzeige an - wie Wasser sich an das Gefäß anpasst, in dem es ist.

5. Was müssen Sie in HTML und CSS tun, um eine Seite responsiv zu gestalten?

Lösung: In HTML muss der Viewport gesetzt werden, der angibt wie groß die sichtbaren Dimensionen auf dem Endgerät sind. Die Syntax ist dabei folgende: `<meta name="viewport" content="width=device-width, initial-scale=1.0">`. Das Beispiel setzt die Seitendimensionen auf die Anzeige-Dimensionen und das Zoomlevel auf 1. CSS ermöglicht Response Design über Media Queries.

6. Wie unterscheidet sich *Responsive Design* zu *Adaptive Design*?

Lösung: Responsive Design ist flexibler als Adaptive Design. Letzteres basiert auf statisch erstellten Designs, die nach deren Initialisierung nicht mehr verändert werden können. Somit rendert der Client den Inhalt in einem bestimmten adaptiven Design, welches für die Anzeige optimiert ist, dann jedoch nicht mehr angepasst werden kann. Bei Responsive Design hingegen kann sich das Design sogar während

der Anzeige ändern; beispielsweise beim Verändern der Browserdimensionen. Hierfür werden so genannte Breakpoints genutzt, die definieren wann die responsive gestylten Elemente einen Zeilenumbruch durchführen müssen.

7. Was versteht man unter dem *Mobile First* Ansatz?

Lösung: Mit dem Begriff Mobile First wird ein Konzept für das Webdesign sowie die Konzeption von Websites bezeichnet. Dieses Konzept sieht vor, dass die für mobile Endgeräte optimierte Version zuerst entsteht und sukzessive Erweiterungen stattfinden. Somit werden alle Elemente weitest möglich logisch entkoppelt um die inhaltlich zusammengehörigen Elemente auf das Ausgabegerät optiomiert zu clustern. Dies erleichtert die Erstellung mobiler Ansichten und durch die Atomarisierung der Inhalte ist eine Erweiterung auf größere Ausgabegeräte deutlich einfacher möglich als wenn man mit einem Design beginnt, welches für große Displays erdacht wurde und das auf kleinere runtergebrochen werden soll. Damit folgt die Strategie „Mobile First“ dem Trend, dass immer mehr Nutzer mit dem Smartphone oder Tablet im Internet surfen.

8. Was ist Bootstrap?

Lösung: „Bootstrap ist ein Frontend-Framework, mit dem du Websites gestalten kannst. Es werden HTML- und CSS-Vorlagen bereitgestellt um unterschiedlichste Website-Elemente darzustellen. Dazu gehören Formulare, Buttons, Tabellen, Navigation sowie ein Grid-System für Layouts. Darüber hinaus ist es durch JavaScript-Module möglich, Interaktionen (z. B. eine Bilder-Slideshow, Tabs und Dialogboxen) in die Website einzubinden. Zudem bietet Bootstrap alle Voraussetzungen um responsive Webdesigns zu gestalten, die dann auch auf Smartphones oder Tablets optimal dargestellt werden.“ [Übernommen von `https://www.bootstrapworld.de/was-ist-bootstrap.html`]