STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

CS 146: Intro to Web Programming and Project Development

Instructor: Iraklis Tsekourakis

Lieb 213

Email: itsekour@stevens.edu

Introduction to CSS

CSS Align

- You can center an item by setting auto margins left & right. Auto will split them evenly

```
.center {
  margin-left: auto;
  margin-right: auto;
  width: 60%;
}
```

  - Does this work for all elements?*
- To align things to the right, you can do it with position:absolute; right: 0px;
- You can also use float to place items to the right

CSS Align

- How would you center an image? (hint: It's an inline element)
- **Tip**: When aligning elements with position, always define margin and padding for the <body> element; this is to avoid visual differences in different browsers

CSS Pseudo-Class

- A pseudo-class is used to define a special state of an element
- For example, it can be used to:
  - Style an element when a user places the mouse over it
  - Style visited and unvisited links differently
  - Style an element when it gets focus
- Pseudo-classes allow you to change the behavior of most elements, or target specific ones

CSS Pseudo-Class

- Syntax is
  - selector:pseudoclass
  - selector.class:pseudoclass
  - selector#id:pseudoclass

```
selector:pseudo-class {
    property:value;
}
```

Existing Pseudo-Classes

:link :visited :active
- Used on links, we already saw these

:hover
- Selects whatever the mouse is on. Works on more than just links!

:focus
- Selects element that has focus. Usually used as input:focus

:first-letter :first-line
- Pretty self-explanatory

:first-child
- This selects an item which is the first child of another tag. For instance p:first-child does not get the first child of a paragraph but the first paragraph which is a child

:before :after
- Allows you to add content with the *content* attribute

:lang(*language*)
- Selects anything that has a language starting with what's written. p:lang(en) would get any paragraph in English

CSS Pseudo-Elements

- A CSS pseudo-element is used to style specified parts of an element
- For example, it can be used to:
  - Style the first letter, or line, of an element
  - Insert content before, or after, the content of an element

```
selector::pseudo-element {
    property:value;
}
```

CSS Pseudo-Elements

- **Notice the double colon notation -** ::first-line versus :first-line
- The double colon replaced the single-colon notation for pseudo-elements in CSS3
- This was an attempt from W3C to distinguish between **pseudo-classes** and **pseudo-elements**
- The single-colon syntax was used for both pseudo-classes and pseudo-elements in CSS2 and CSS1
- For backward compatibility, the single-colon syntax is acceptable for CSS2 and CSS1 pseudo-elements

Pseudo-Element Examples

- The ::first-line pseudo-element is used to add a special style to the first line of a text

```
p::first-line {
    color: #ff0000;
    font-variant: small-caps;
}
```

- The ::first-letter pseudo-element is used to add a special style to the first letter of a text

```
p::first-letter {
    color: #ff0000;
    font-size: xx-large;
}
```

CSS The Cursor Property

- The cursor property specifies the type of cursor to be displayed when pointing on an element

```
span.crosshair {
    cursor: crosshair;
}

span.help {
    cursor: help;
}

span.wait {
    cursor: wait;
}
```

CSS3 2D Transformations

- CSS3 transforms allow you to translate, rotate, scale, and skew elements
- A transformation is an effect that lets an element change shape, size and position
  - translate()
  - rotate()
  - scale()
  - skewX()
  - skewY()
  - matrix()

CSS3 2D Transformations


```
div {
    -ms-transform: rotate(20deg); /* IE 9 */
    -webkit-transform: rotate(20deg); /* Safari */
    transform: rotate(20deg);
}
```

CSS3 3D Transformations

- CSS3 allows you to format your elements using 3D transformations
  - rotateX()
  - rotateY()
  - rotateZ()

CSS3 Multiple Columns

- The CSS3 multi-column layout allows easy definition of multiple columns of text
  - column-count
  - column-gap
  - column-rule-style
  - column-rule-width
  - column-rule-color
  - column-rule
  - column-span
  - column-width

```
div {
    -webkit-column-count: 3; /* Chrome, Safari, Opera */
    -moz-column-count: 3; /* Firefox */
    column-count: 3;
}
```

CSS3 User Interfaces

- The resize property specifies whether or not an element should be resizable by the user

```
div {
    resize: horizontal;
    overflow: auto;
}
```

CSS3 User Interfaces

- The outline-offset property adds space between an outline and the edge or border of an element
- Outlines differ from borders in three ways:
  - An outline is a line drawn around elements, outside the border edge
  - An outline does not take up space
  - An outline may be non-rectangular

```
div {
    border: 1px solid black;
    outline: 1px solid red;
    outline-offset: 15px;
}
```

This div has an outline 15px outside the border edge.

CSS - Responsive Web Design

- Responsive web design makes your web page look good on all devices
- Responsive web design uses only HTML and CSS
- Responsive web design is not a program or a JavaScript

- It is called responsive web design when you use CSS and HTML to resize, hide, shrink, enlarge, or move the content to make it look good on any screen

What is the Viewport?

- The viewport is the user's visible area of a web page
- The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen
- Before tablets, and smart phones-> desktop webpages: static and fixed size design
- With tablets, phones-> first approach: scale down the whole page
  - Quick fix, but not optimal

Setting the Viewport

- HTML5 introduced a method to let web designers take control over the viewport, through the <meta> tag
- You should include the following <meta> viewport element in all your web pages:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- The width=device-width part sets the width of the page to follow the screen-width of the device
- The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser

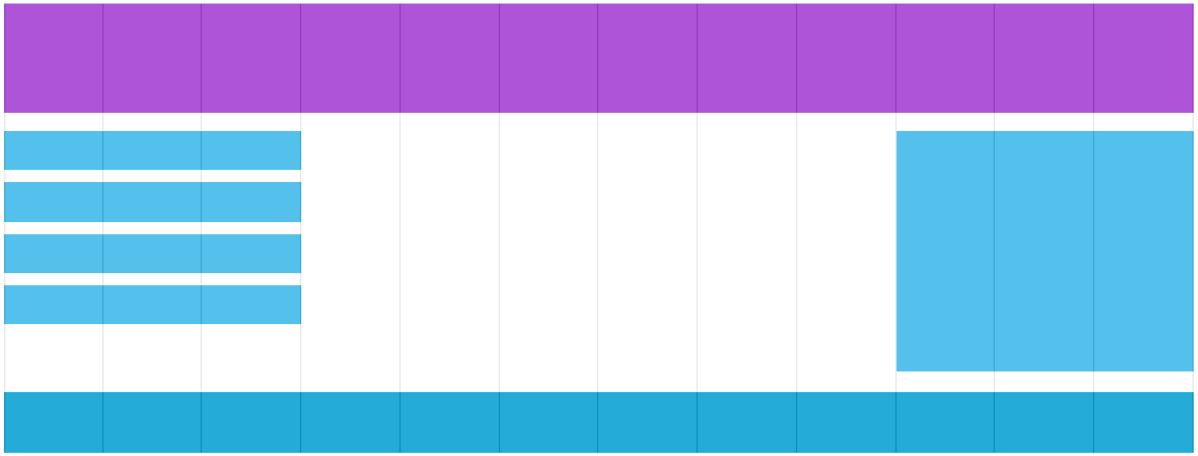
What if we don’t set the Viewport?

Size Content to the Viewport

- Users are used to scroll websites vertically on both desktop and mobile devices - but not horizontally!
- So, if the user is forced to scroll horizontally, or zoom out, to see the whole web page it results in a poor user experience.
- Some additional rules to follow:
  - **Do NOT use large fixed width elements**
  - **Do NOT let the content rely on a particular viewport width to render well**
  - **Use CSS media queries to apply different styling for small and large screens**

Responsive Design – Grid View

- Many web pages are based on a grid-view, which means that the page is divided into columns
- Using a grid-view is very helpful when designing web pages. It makes it easier to place elements on the page



- A responsive grid-view often has 12 columns, and has a total width of 100%, and will shrink and expand as you resize the browser window

Building a Responsive Design – Grid View

- First ensure that all HTML elements have the box-sizing property set to border-box
- This makes sure that the padding and border are included in the total width and height of the elements

```
* {
    box-sizing: border-box;
}
```

Building a Responsive Design – Grid View

```
[class*="col-"] {
    float: left;
    padding: 15px;
    border: 1px solid red;
}

.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```

```
<div class="col-3">

<div class="col-9">
```

CSS Media-Query

- Media query is a CSS technique introduced in CSS3
- It uses the @media rule to include a block of CSS properties only if a certain condition is true

```
/* For mobile phones: */
[class*="col-"] {
    width: 100%;
}
```

```
@media only screen and (min-width: 600px) {
    /* For tablets: */
    .col-m-1 {width: 8.33%;}
    .col-m-2 {width: 16.66%;}
    .col-m-3 {width: 25%;}
    .col-m-4 {width: 33.33%;}
    .col-m-5 {width: 41.66%;}
    .col-m-6 {width: 50%;}
    .col-m-7 {width: 58.33%;}
    .col-m-8 {width: 66.66%;}
```

```
@media only screen and (min-width: 768px) {
    /* For desktop: */
    .col-1 {width: 8.33%;}
    .col-2 {width: 16.66%;}
    .col-3 {width: 25%;}
    .col-4 {width: 33.33%;}
    .col-5 {width: 41.66%;}
    .col-6 {width: 50%;}
    .col-7 {width: 58.33%;}
    .col-8 {width: 66.66%;}
```