

CS 2033

Multimedia & Communications II

LECTURE 6 – ADVANCED CSS

1

Announcements

2

- ▶ Assignment 2 is posted on OWL.
 - ▶ Create a website using code.
 - ▶ Do NOT use a program like BlueGriffon or Kompozer.
 - ▶ Do NOT use online templates.
- ▶ Next week is reading week – no classes, no labs, no office hours.

CSS

3

- ▶ CSS is used for adding colours, borders, and other aesthetics to websites. But there's more to it...
- ▶ It is also used for creating layouts. There are several styles that control the size, position, or structure of the HTML elements.
- ▶ They allow us to create a layout without tables.

CSS

4

- ▶ Remember that divs and many other HTML elements can be nested within one another.
- ▶ Important for creating layouts.
- ▶ This relationship is known as parent-child, where the parent is the container / outer element and the child is the inner element.

CSS Layouts

5

- ▶ Divs are the best HTML elements (in my opinion anyway!) Don't be skimping with them!
- ▶ Assign classes to each of the divs.
- ▶ Note that elements can be assigned multiple classes.
 - ▶ i.e. `<div class="bigbox nav"></div>`

CSS Layouts

6

- ▶ Common layout style properties:
 - ▶ position
 - ▶ TRBL
 - ▶ display
 - ▶ margin
 - ▶ float
 - ▶ width / height

CSS Layouts

7

- ▶ There are many ways to create a layout with divs and CSS.
- ▶ Within a page, some sections may be laid out different than others.
- ▶ Some of the common strategies will be explained here, but there are endless possibilities by combining these ideas and your creativity!

Wrapper

8

- ▶ It is usually a good idea to include all or most of the content in a giant div. It is common to call this the **wrapper** or **container**.
- ▶ The wrapper is often centered to keep the content in the middle.
- ▶ This does NOT mean the content is centered!



Wrapper

9

- ▶ The wrapper may be given a width in pixels or %. Either is fine.
 - ▶ Pixels keep a rigid structure.
 - ▶ % is flexible for any window size.
- ▶ Either way, we should make it look good on different sized screens.
- ▶ We will talk more about **responsive** websites later in the term.

Sections

10

- ▶ Sections are HTML elements similar to divs but intended for keeping the different areas separated.
- ▶ It's good practice to use sections to contain each distinct portion of a page, and then use divs within them for more layout control.

Sections

11

```

<div class="wrapper">
  <section>
    <h1>About Us</h1>
    <div class="content">
      <p>ABCDEFGHIJKLMN</p>
    </div>
  </section>
  <section>
    <h1>Services</h1>
    <div class="content">
      <p>ABCDEFGHIJKLMN</p>
    </div>
  </section>
  <section>
    <h1>Contact</h1>
    <div class="content">
      <p>ABCDEFGHIJKLMN</p>
    </div>
  </section>
</div>

```

Columns

12

- ▶ Within sections, there are many options for placing text, images, and other multimedia.
- ▶ The most basic option is to have a single element across the entire width of the section / wrapper.
- ▶ Many sites have multiple columns of side-by-side content.

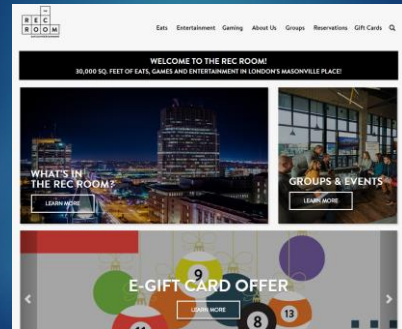
Columns

13



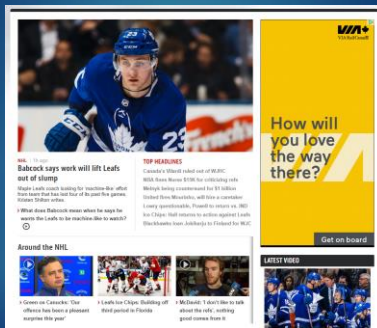
Columns

14



Columns

15



Columns

16

- How do we place elements side by side?
 - Display: block, inline-block, table-cell
 - Float: left or right
 - Position: absolute
 - Within a relative position element
 - Usually not the best option

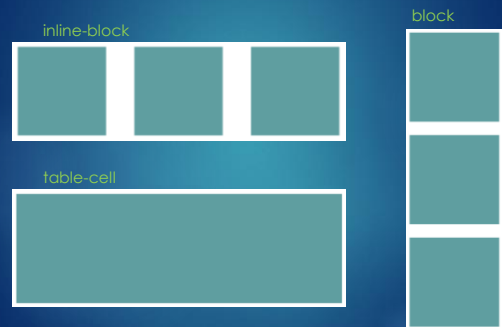
Display

17

- The display style affects how elements are shown sequentially.
- There are several display options. The most common ones are:
 - **Block**: takes up the entire row
 - **Inline-block**: placed side by side if fits
 - **Table-cell**: placed side by side and resizes to fit in a row

Display

18



Display

19

- ▶ Inline-block and table-cell are great for creating columns.
- ▶ Which one is best? It depends...
 - ▶ Should they be spaced out?
 - ▶ Should they fall to the next line if the window/screen is small?

Display

20

- ▶ Margins work with inline-block. Not so much with table-cell, although you can nest divs within and add margins that way.
- ▶ Inline-block elements will only stay together if they can fit. Table-cell elements remain together and just squish smaller.

Display

21

- ▶ See the difference between table-cell, inline-block, and block.
- ▶ <http://www.csd.uwo.ca/~bsarlo/cs2033b/samples/lec6/display.html>

Float

22

- ▶ Float is another layout style.
- ▶ Often used to let text wrap around an image on the left or right.
- ▶ Can also help with column layouts.
- ▶ Floating may break the natural flow and cause overlapping.
- ▶ Not the best option unless you know what you're doing!

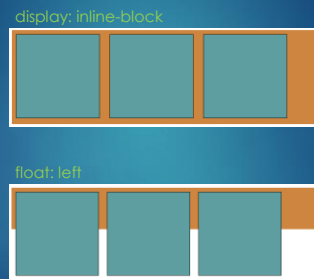
Float

23

- ▶ See the difference between left float, right float, and no float.
- ▶ <http://www.csd.uwo.ca/~bsarlo/cs2033b/samples/lec6/float.html>

Float

24



Position

25

- ▶ The last main layout style is position (often used with TRBL)
- ▶ Static and relative position follow natural flow of elements (blocks).
- ▶ Fixed position remains when scrolling.
 - ▶ Useful for navigation bar, footer, feedback link, etc.

Position

26

- ▶ For column-based layouts, absolute positioning can be used.
- ▶ They must be contained in a relative positioned element first!
- ▶ Absolute is a precise location, but it is (ironically) relative to its parent.
- ▶ Never use absolute if it's not inside a relative parent element.

Position

27

position: relative

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam eget magna porta, volutate tunc sit amet, volutpat enim. Aliquam nec lacus, tempus at nisi quis, mollis enim aliquam nisi. Class aptent tacit sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Vivamus porttitor dui id et ultrices nunc. Proin elementum nisi in magna dapibus pharetra. Pellentesque at nunc tristique, condimentum nisi id, tempus ex. In non risus dolor. Sed ultrices odio eget nunc accumsan efficitur sit amet sit amet nisi. Morbi dui nisi sit et nunc lobortis pellentesque. Nunc consequat magna vitae nunc imperdiet nulla. Pellentesque gravida lacus eget arcu accumsan, at posuere enim maximus. Donec mi nunc, lacinia ac tempus efficitur mollis pretium. Maecenas pharetra nibh sed viverra tempus. Sed ornare pretium ipsum, non ullamcorper ante efficitur ligula. Nunc faucibus ante nec nisi varius sagittis. Cras donec cursus ligula sit enim facilisis sagittis.



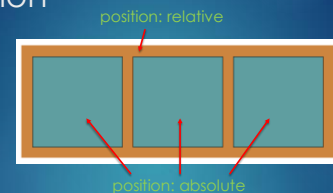
position: absolute

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam eget magna porta, volutate tunc sit amet, volutpat enim. Aliquam nec lacus, tempus at nisi quis, mollis enim aliquam nisi. Class aptent tacit sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Vivamus porttitor dui id et ultrices nunc. Proin elementum nisi in magna dapibus pharetra. Pellentesque at nunc tristique, condimentum nisi id, tempus ex. In non risus dolor. Sed ultrices odio eget nunc accumsan efficitur sit amet sit amet nisi. Morbi dui nisi sit et nunc lobortis pellentesque. Nunc consequat magna vitae nunc imperdiet nulla. Pellentesque gravida lacus eget arcu accumsan, at posuere enim maximus. Donec mi nunc, lacinia ac tempus efficitur mollis pretium. Maecenas pharetra nibh sed viverra tempus. Sed ornare pretium ipsum, non ullamcorper ante efficitur ligula. Nunc faucibus ante nec nisi varius sagittis. Cras donec cursus ligula sit enim facilisis sagittis.

Absolute positioning doesn't care what's underneath. It's a precise location.

Position

28



- ▶ See the difference between relative and absolute positions.
- ▶ <http://www.csd.uwo.ca/~bsarlo/cs2033b/samples/lec6/position.html>

Position

29

- ▶ Absolute within relative position can create side-by-side layouts.
- ▶ Disadvantages?
 - ▶ Sticks out of page on small windows.
 - ▶ Each element's location has to be calculated and specified.
 - ▶ Behaves like floats if height of parent is not known.

Layouts

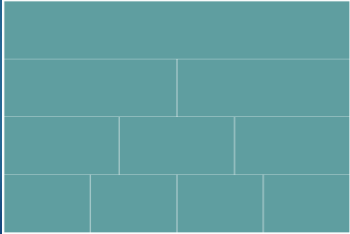
30

- ▶ Overall, better to avoid absolute positioning unless you know what you're doing.
- ▶ Floats should generally be avoided for layouts as well.
- ▶ Verdict: inline-block or table-cell display are usually the best options!

Layouts

31

- ▶ This was created with just sections and divs, and CSS styles.. No tables!



Layouts

32

- ▶ We can use groups of the different sized elements as we want.
- ▶ Here's an example of a website structured using these blocks.

▶ <http://www.csd.uwo.ca/~bsarlo/cs2033b/samples/lec6/layout-example.html>

Child selectors

33

- ▶ We have seen several ways to select elements but there are more!
- ▶ The asterisk * is used to select everything at once.
 - ▶ i.e. `* { margin: 2px; }`
- ▶ Another approach is selecting all the children of parent elements.
 - ▶ i.e. all paragraphs within divs.

Child selectors

34

- ▶ It is easy to select specific children.
- ▶ Write the parent selector, then a space, and the child selector.
 - ▶ Selectors can be any combination of types (tag, class, or ID).
- ▶ `#wrapper p { }`
- ▶ `div .red { }`
- ▶ `ul li.link { }`

Child selectors

35

- ▶ They can continue with multiple levels of children.
 - ▶ i.e. `#main div ul.nav li p { }`
- ▶ We can also use the comma grouping method combined with these child selectors.
 - ▶ i.e. `ul.nav li, div p, #buttons { }`

Pseudo-classes

36

- ▶ Pseudo-classes are advanced CSS selectors that apply to certain states or actions on elements.
- ▶ i.e. hovering the cursor over an element
- ▶ They are specified with a colon : followed by a pseudo-class type.
 - ▶ i.e. `:hover`

Pseudo-classes

37

- ▶ Apply pseudo-classes to elements by tag, class, ID, or any combination.
- ▶ `a:hover { }`
- ▶ `.link:hover { }`
- ▶ `#title:hover { }`
- ▶ `ul li:hover { }`

Pseudo-classes

38

- ▶ Works great for simple link rollovers.
- ▶ Also used in more complex navigation menus with dropdown menus.
- ▶ <http://www.csd.uwo.ca/~bsarlo/cs2033b/samples/lec6/dropdown.html>

Pseudo-classes

39

- ▶ The idea is lists within lists.
- ▶ Main links are visible and sub-links are hidden by default.
- ▶ Sub-links become visible when the main link is hovered by a cursor.
- ▶ Primary CSS for this:
 - ▶ Display style (none ↔ block)
 - ▶ `:hover` pseudo-class (to trigger display change)

Pseudo-classes

40

- ▶ There are lots of pseudo-classes besides `:hover`.
 - ▶ `:required`
 - ▶ `:focus`
 - ▶ `:visited`
 - ▶ `:active`
 - ▶ `:enabled`
 - ▶ and many more

CSS transitions

41

- ▶ By default, style changes like rollovers are instantaneous.
- ▶ This is fine but it can look too static.
- ▶ CSS allows us to animate style changes using gradual transitions.
- ▶ <http://www.csd.uwo.ca/~bsarlo/cs2033b/samples/lec6/transitions.html>

CSS transitions

42

- ▶ Make a style change gradual in a selector with the `transition` style rule.
- ▶ Within this rule, we can specify the parameters like duration, speed curve (i.e. ease or linear), and time delay (i.e. start after 5s).
- ▶ These can be specified for individual style changes or for all styles at once.

CSS transitions

43

- ▶ Suppose we have:
- ▶

```
div {
  background-color:red;
}
div:hover {
  background-color:black;
}
```
- ▶ Hovering the div will turn it black but by default it will be immediate.

CSS transitions

44

- ▶ Add the transition line in the div rule-set. Apply a 2s transition.
- ▶

```
div {
  background-color:red;
  transition: background-color 2s;
}
div:hover {
  background-color:black;
}
```

CSS transitions

45

- ▶ You can specify multiple style properties to be animated by separating them with commas.
- ▶ Suppose you want the width and height to change.
- ▶ Add the following line:
 - ▶ `transition: width 2s, height 2s;`

CSS transitions

46

- ▶ To apply the transition to all style properties at once, use the word `all` in place of a property name.
- ▶ i.e. `transition: all 2s;`
- ▶ This helps simplify the code in cases where many changed styles are intended to have a transition.

CSS animations

47

- ▶ Transitions are for style changes, usually triggered by `:hover`, `:focus`, or another pseudo-class.
- ▶ Sometimes we want an animation that is independent of user input.
 - ▶ i.e. a shape growing and shrinking back and forth continuously.

CSS animations

48

- ▶ This is called an `animation` in CSS.
- ▶ Note the difference between transitions and animations.
- ▶ CSS animations use keyframes
 - ▶ Remember these from CS1033?
 - ▶ In CSS we can also have intermediate keyframes for multiple segments.

CSS animations

49

- ▶ `@keyframes growAnim {`
`from { width:200px; }`
`to { width:300px; }`
`}`
- ▶ This animation, named `growAnim`, has 2 keyframes.
 - ▶ Start (from): width of 200px
 - ▶ End (to): width of 300px

CSS animations

50

- ▶ If we want it to grow in different segments, we can use a %-based keyframe structure instead.
- ▶ `@keyframes growAnim {`
`0% { width:200px; }`
`75% { width:250px; }`
`100% { width:300px; }`
`}`
- ▶ This is similar to `ease-in`.

CSS animations

51

- ▶ Once a keyframe structure is created, it can be applied to a rule-set easily.
- ▶ `animation: growAnim 3s infinite;`
- ▶ The main parameters are:
 - ▶ Animation name
 - ▶ Duration
 - ▶ Iteration count (looping)

CSS animations

52

- ▶ Two other common animation parameters are:
 - ▶ Direction: `normal` (start to finish), `reverse` (finish to start), `alternate`, etc.
 - ▶ Timing function: `ease`, `ease-in`, `ease-out`, `linear`, etc.
- ▶ <http://www.csd.uwo.ca/~bsarlo/cs203/3b/samples/lec6/animations.html>

Transitions and animations

53

- ▶ Notes on transitions/animations:
 - ▶ Make looped animations seamless.
 - ▶ Use the `alternate` direction parameter.
 - ▶ Use the %-based keyframe structure and set the same styles for 100% as 0%.
 - ▶ Don't make your page overly flashy with animations.
 - ▶ Not all properties are animatable.

Conclusion

54

- ▶ That's all for today.
- ▶ Have a great reading week!