CMPT 165

INTRODUCTION TO THE INTERNET AND THE WORLD WIDE WEB

By Hassan S. Shavarani

UNIT8: MORE CSS

TOPICS

1. Positioning in CSS
2. Responsive Design
3. CSS Frameworks

WHAT WE ARE EXPECTED TO KNOW FROM CSS?

we know how to modify
fonts, **colors**, **spacing**, and **borders**

useful and necessary changes,
but not the hard part of CSS !

things become more interesting
when you start moving stuff around the page !

POSITIONING EXAMPLE

we will demonstrate different positioning tags using this example

THE `float` PROPERTY

- it will take content and move it to the left or right sides of the page
- when floated content moves, whatever content follows it will move up, and flow around it
- let's look at the demo at https://developer.mozilla.org/en-US/docs/Web/CSS/float

THE `clear` PROPERTY

- it tells the browser that you want to move a piece of content down until there is nothing floating beside it, either to the left, right, or both sides
- let's look at the demo at
  https://developer.mozilla.org/en-US/docs/Web/CSS/clear

THE `position` PROPERTY

- it is both more **powerful** and more **dangerous** than `float` and `clear`
- different values:
  - `position: static` default value
  - `position: relative` works like static but you can adjust the position of the element relative to where it would have naturally been
  - `position: absolute` on an element lets you use the CSS properties `top`, `bottom`, `left`, and `right`
  - `position: fixed` is like absolute, but relative to the window (fixed things won't move as the user scrolls)
- let's look at the demo at https://developer.mozilla.org/en-US/docs/Web/CSS/position

THE `position` PROPERTY PROBLEM

as soon as you start positioning content,
it is taken **out of the normal flow** of content on the page
and will overlap other content

---

you need to think of all of the possible ways your content can overlap on different devices and window sizes !

THE `opacity` PROPERTY

- specifies the level of transparency of elements
- let's look at the demo at https://developer.mozilla.org/en-US/docs/Web/CSS/opacity

THE `z-index` PROPERTY

- moves the figure behind (in front of) the other content
- let's look at the demo at
  https://developer.mozilla.org/en-US/docs/Web/CSS/z-index

THE `display` PROPERTY

- specifies the type of rendering box used for an element
- let's look at the demo at https://developer.mozilla.org/en-US/docs/Web/CSS/display

RESPONSIVE DESIGN

one of the difficult things about designing for the web
is that we don't get to really control
the way our content is viewed

the user's web browser is responsible for processing
our HTML, CSS, and JavaScript

when designing web pages, you should try to create

responsive designs

that is, make sure your pages are nicely readable on a wide variety of devices

DEVICE SIZES

many beginners to web design make the mistake of only looking at their pages on their own screen, and never test on larger or smaller screens

be careful not to fall into his trap

let's check out the positioning example with different device sizes

TESTING TIPS FOR DIFFERENT DEVICE SIZES

- resize your browser window: not everybody has the same screen, or keeps their browser maximized
- shrink your browser window to a phone-like size to see how the elements of your page fit with smaller space
  even better: actually try the page on a smartphone
- try the largest browser window you can (and scale the page with ctrl-minus/⌘-minus and ctrl-plus/⌘-plus to get an even wider range), to make sure things are still readable

LONG LINE READABILITY PROBLEM

one common problem on larger screens is long lines
lines of text that extend all the way across
a large display are very difficult to read
It may be wise to start with CSS like this,
to help keep things readable:

```
body {
    max-width: 40em;
    line-height: 1.4;
    margin: 1em auto;
    color: #333;
}
```

BEING MOBILE-FRIENDLY

when it comes to mobile browsers,
they might get a beautiful mobile-friendly design,
or they might have to display
a desktop-only site as best they can,
and have to detect which situation they're in

BEING MOBILE-FRIENDLY: `viewport` META TAG

often, phone browsers will decide that
your page isn't mobile-friendly and display it with a desktop-like width
you can stop the mobile browser form doing that using:

```
<meta name="viewport" content="width=device-width,initial-scale=1" />
```

MEDIA QUERIES

assigning CSS properties based on screen size, screen type, screen orientation, ...

as an example, they give us a way to express "on a smaller screen, do this ..." in CSS

* for more examples lets look at here
* and here is the guide to use media queries

CSS FRAMEWORKS

the more page layout you do with CSS,
the more you're likely to run into things that
are either **difficult** to do with CSS, or just **tricky**

e.g.sometimes it is really hard to predict how different parts of your style will interact with each other

THE PROBLEM

if the content of the pages changes,
it can also be **challenging to get a layout right** on a wide variety of devices

THE SOLUTION

some pre-made code that solves common problems and can help us get things working more easily

CSS frameworks are pre-prepared stylesheet codes to serve our solution !

BOOTSTRAP

the most popular CSS framework is Bootstrap, which was originally created at Twitter for their site

```
<link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.6/css/bootstrap.min.css"
/>
<link rel="stylesheet" href="bootstrap-changes.css" />
```

BOOTSTRAP: COLUMNS AND THE GRID

in bootstrap, the page is automatically divided up into 12 columns, you can create content that takes up any number of these columns

```
<section class="container">
    <div class="row">
        <nav class="col-xs-3"> [the menu] </nav>
        <main class="col-xs-9"> [main content] </main>
    </div>
</section>
```

BOOTSTRAP GRID WINDOW SIZES

- extra-small (xs, roughly phone-sized)
- small (sm, small tablet-sized)
- medium (md, large tablet or small desktop)
- large (lg, desktop screen)
- extra-large (xl, huge desktop screen)

BOOTSTRAP GRID WINDOW SIZES: EXAMPLE

```
<div class="row">
    <div class="col-xs-12 col-sm-6 col-md-4"> … </div>
    <div class="col-xs-12 col-sm-6 col-md-4"> … </div>
    <div class="col-xs-12 col-sm-6 col-md-4"> …
        <figure class="hidden-xs-down">…</figure>
    </div>
</div>
```

* here is an example using these classes

CSS FRAMEWORKS: ADVANTAGES

they **make positioning much easier** than fighting with float and position yourself, they are used on many modern web sites for exactly this reason

CSS FRAMEWORKS: DRAWBACKS

many web sites use them!

and so its **default styles** start to **look very generic and familiar**

remember that the CSS framework
is just a stylesheet like any other;
you don't have to forget everything
you know about CSS when you start using it

e.g. you can override some of tags to customize the look to your own taste

Any Questions?