Advanced CSS: Transitions, Animations, & Responsive Web Design

PGCertInfoTech

Programming with Web Technologies

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

Auckland ICT Graduate School

CSS 3

Widespread browser support for CSS3 becoming the norm:

- Many New features
  - (see following slides)
- Module based
  - So no "standard" per se
  - Published snapshots

https://developer.mozilla.org/en/docs/Web/CSS/CSS3
http://www.w3.org/Style/CSS/current-work.en.html

- Leads to some variance in browser support

http://www.w3schools.com/cssref/css3_browsersupport.asp

CSS vendor prefixes

Taking advantage of these newer CSS features can require special vendor prefixes (while they are still in development in the browser):

```
box-sizing: border-box;
-webkit-box-sizing: border-box;
-moz-box-sizing: border-box;

transform: rotate(15deg);
-ms-transform: rotate(15deg);
-webkit-transform: rotate(15deg);
-moz-transform: rotate(15deg);
```

`-webkit:`	Safari, Chrome
`-moz`	Firefox
`-ms:`	IE (Microsoft)

http://peter.sh/experiments/vendor-prefixed-css-property-overview/

New Features

Prominent new features:

- CSS Filters
- Transformations
- Transitions
- Animations

CSS Filters

Graphic effects:

"The easiest way to think of a filter is as a post processing step that does something magical after all your page content has been laid out and drawn."

http://www.html5rocks.com/en/tutorials/filters/understanding-css/

For example:

- `blur(), grayscale(), contrast() …`

```
#brightness-test:hover {    filter: brightness(125%);   }
```

Hover pseudo-class applied to an id selector

Filter name

Can combine multiple filters in one rule

https://developer.mozilla.org/en-US/docs/Web/CSS/filter

CSS 3 Transformations

Transform content in the browser:

- translate()
- rotate()
- scale()
- skew()
- matrix()

Example syntax:

```
#test {
    transform: scale(2);
  }
```

There are also 3D transforms
- more complex
- less well supported

http://caniuse.com/#feat=transforms2d

CSS 3 Transitions

- Change CSS properties over a time period
- Browser calculates intermediate points

- Two inputs:
  - CSS property you want to add an effect to
    - e.g., `margin-left`
  - duration of the effect
    - e.g., `3s`

- Triggered by a change in value of the property

`http://caniuse.com/#feat=css-transitions`

CSS 3 Transitions

Effect + duration:

```
style="transition: margin-left 3s ease 0.1s;"
```

Trigger:

```
#test:hover {
      margin-left: 5em;
}
```

- When the element with an `id` of `test` is hovered over by the user's cursor then change the left margin from its current value to a new value of `5em`
- Often initiated by :
  - user interaction such as `:hover`
  - JavaScript programs

CSS 3 Animations

CSS Animations better choice then JavaScript for simple forms of animation:

- No programming required
- Let browsers deal with efficiency issues

http://daneden.github.io/animate.css/

- Can be integrated with JavaScript for more complex animations

http://caniuse.com/css-animation

Example

```
h1 {
  animation-duration: 3s;
  animation-name: slide-right;
}

@keyframes slide-right {
  from {
    margin-left: 100%;
    width: 300%
  }

  to {
    margin-left: 0%;
    width: 100%;
  }
}
```

You can add intermediate frames by specifying a percentage done:

from = 0%

```
25%  {background:yellow; left:200px; top:0px;}
50%  {background:blue; left:200px; top:200px;}
```

to = 100%

Think about all the different CSS properties you could animate

Further example

```
#moving-text {
     animation-duration: 3s;
     animation-name: slide-right;
     animation-iteration-count: infinite;
     animation-direction: alternate;
}
```

Reusing the previous `@keyframes` rule:

- With added iteration and direction switching

http://daneden.github.io/animate.css/

Used in combination

Transforms, Transitions and Animations

- Allow some simple effects

- Combining 2 or 3 them gives you even more options
- Linking with JavaScript allows for even more intricate interactive options
- Still evolving …

Beyond CSS

An interesting development to keep an eye on, is:

- SASS: Syntactically Awesome Stylesheets

It:

- Introduces more programming like features to specifying styling information
- Supports variables and maths calculations
- Ultimately turned into CSS syntax

Example Sassy CSS File (.scss)

```
$serif-font-stack: "Georgia", "Times New Roman", serif;
$monospace-font-stack: "Cousin", "Courier";

body {
    font: normal 18px/22px $serif-font-stack;
}

pre, code {
    font: 600 bold 18px/22px $monospace-font-stack;
}
```

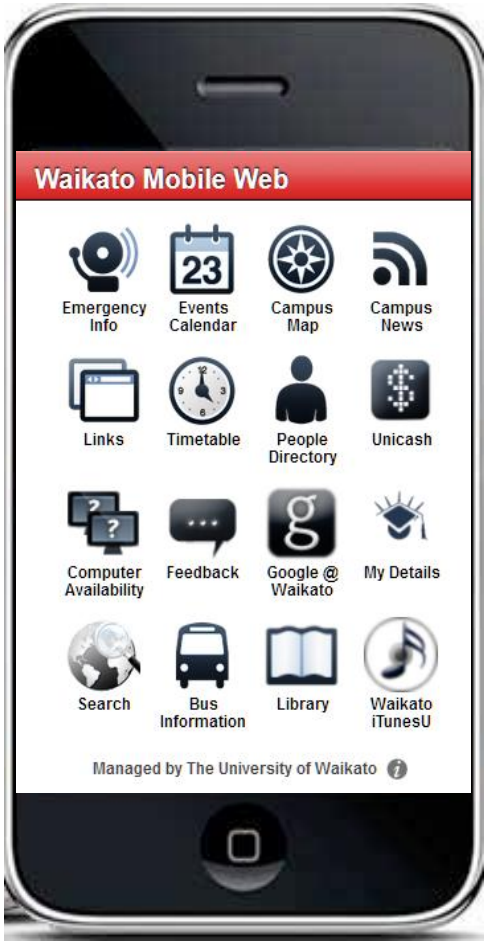
SCSS *variable*

http://www.sitepoint.com/6-current-options-css-preprocessors/

Responsive design

- Problem: lots of devices with different displays
- Mobile devices: phones, tablets
- Projectors, printers, audio screenreaders, …

- One solution:
  - Develop a separate website for different classes of device:
  - `http://m.waikato.ac.nz/`
- Better solution:
  - Use CSS to adapt display for different situations



@media rules

```
@media screen {
    p {
        font-family: Calibri, sans-serif;
    }
}

@media print {
    p {
      font-family: Gotham Narrow, sans-serif;
    }
}
```

Different presentation for different media

Allows choice to fine-tune presentation choices for a particular medium
e.g. some fonts are better for screen display, others for printing

@media types

- `all` — all media type devices
- `aural` — speech and sound synthesizers
- `braille` — braille tactile feedback devices
- `embossed` — paged braille printers
- `handheld` — small or handheld devices
- `print` — printers
- `projection` — projected presentations, like slides
- `screen` — computer screens
- `tty` — media using a fixed-pitch character grid, like teletypes and terminals
- `tv` — television-type devices

CSS 3 Media Queries

CSS 3 extends media-dependent style sheets with:

- Media Queries
  http://www.w3.org/TR/css3-mediaqueries/
- As either `<link>` elements, `@media` or `@import` rules

```
<link rel="stylesheet"   media="screen and (min-width: 641px)
and (max-width: 800px)" href="ipad.css">


@media screen and (color), projection and (color) { … }


@import url(color.css) screen and (color);
```

Customises content to different display sizes and types

CSS 3 Media Queries

- height, width
  - Often used to target various mobile devices
- aspect-ratio
- orientation

only keyword hides the rule from older browsers

```
<style>
  @media only all and (orientation: portrait) { ... }
</style>
```

- resolution
  - `@media print and (min-resolution: 300dpi) { … }`
  - Can use to choose different quality images

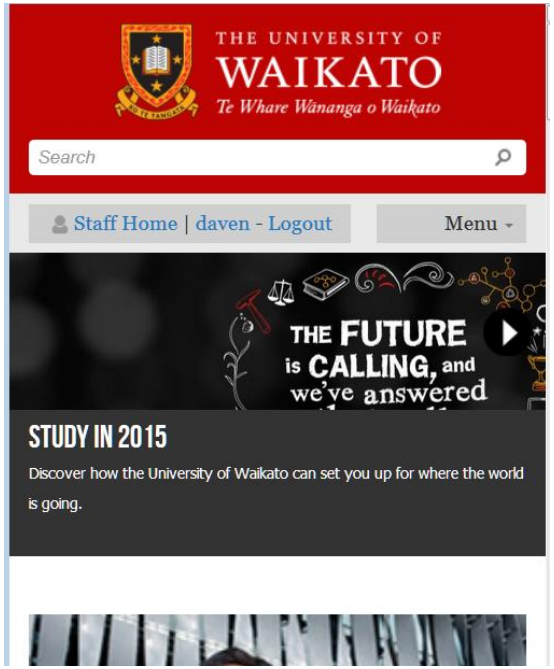
http://caniuse.com/#feat=css-mediaqueries

Responsive Design

- Lots of related terms: Fluid, Flexible, Adaptive, Elastic
- All share idea of creating designs that work on different devices

- Fixed-width designs are a simple solution
  - But can lead to annoying presence of horizontal scroll bars

Responsive Designs *change* at different widths




```
@media screen  and (min-width: 500px) and (max-width: 699px) { … }
```

Widths at which they switch designs are called *Breakpoints*