Computer Science Principles

Web Programming

Web-Presentation Design with CSS

CHAPTER 18: CSS PROJECT INTEGRATION

DR. ERIC CHOU

IEEE SENIOR MEMBER

CSS3

Chapter 11: CSS Hierarchy and Selectors

Chapter 12: Text, Image and Foreground (Contents)

Chapter 13: Color and Background (Contents)

Chapter 14: Box Model (Padding, Border, and Margin)

Chapter 15: Layout Management (Floating and Positioning: where should the Element go)

Chapter 16: Layout Management (Page Level Planning)

Chapter 17: Layout Management (Transition, Transforms, and Animation: space and time domain transformation)

Chapter 18: CSS Techniques (Put Everything Together)

CSS Techniques

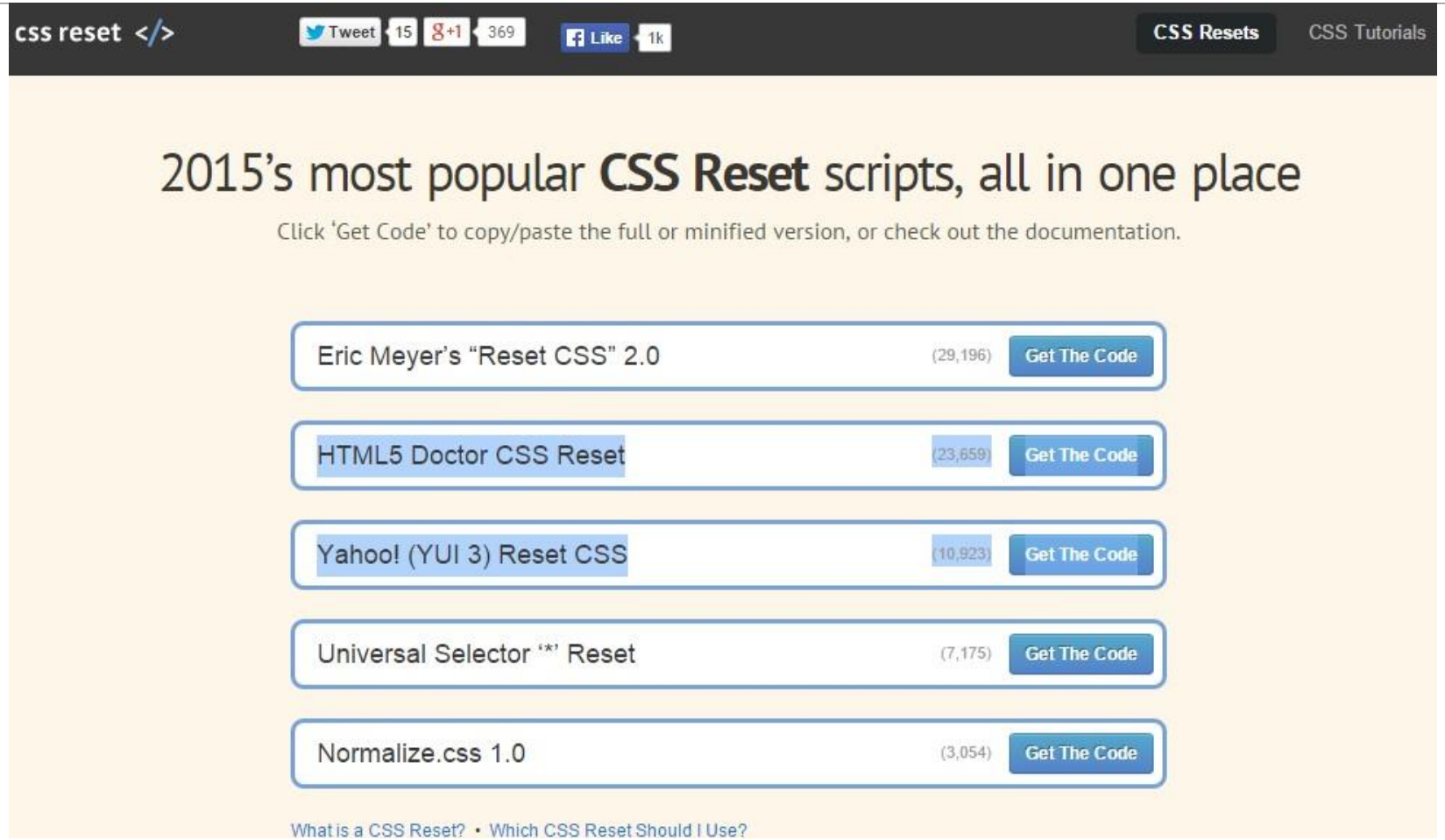
LECTURE 1

CSS Techniques

- A Clean Slate (CSS reset)
- Image Replacement Techniques
- CSS Sprites
- Styling Forms
- Styling Tables
- Basic Responsive Web Design
- Wrapping Up Style Sheets

CSS Reset

http://cssreset.com/

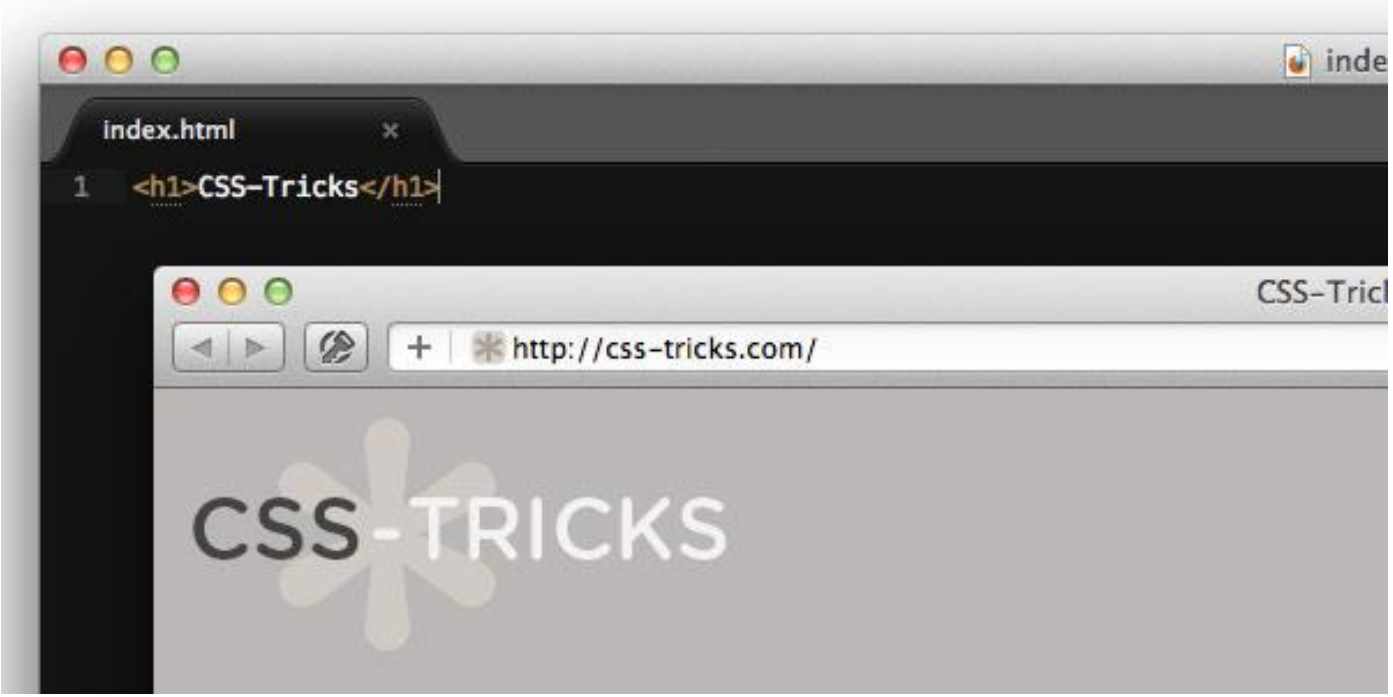


A Clean Slate (CSS Reset)

- Browsers have their own built-in style. If you don't like them, you have to override them by your own CSS style sheet. If there are too much of it, it may be a tedious task. If you do not override them with your own design, your elements will inherit them from the built-in styles.
- CSS reset provide a method to reset all CSS setting by putting your own favorite reset condition at the very beginning of your own HTML page. You may get a fresh start and then, you may put your own style sheet later.
- The most popular reset was written by Eric Meyer. http://meyer.com/eric/tools/css/reset
- To use the reset, place these styles at the top of your own style sheet.

Image Replacement Techniques

CSS image replacement is a technique of **replacing a text element (usually a header tag) with an image.** An example of this would be including a logo on a page. You may want to use a <h1> tag and text for this for the accessibility and SEO benefits, but ideally you'd like to show your logo, not text.



Note that some of these techniques are very old. Web design in the early 2000's was a lot different than it is now, but there was still much thought being put into accessibility. Because of that, most of these techniques still hold up today. It is still very interesting to see all the thought and problem solving that was put into this. Also note that I tested all these FF2, Opera 9, Safari 3, and IE 6 and they all behaved identically (hard to believe, I know).

Image Replacement Techniques

9 Categories

- The report card consists of five major categories:

- **CSS ON / Images ON**
  Represents browsers in their normal states. All techniques should pass this test, since that's the whole point.

- **CSS ON / Images OFF**
  Represents browsing with regular stylesheets applied but images turned off. This is rare but a possibility (folks with bandwidth concerns...) This is the most difficult test. Since most of these techniques go to various lengths to hide text, when the images are turned off that means that nothing is displayed which ain't good. Displaying text only here is considered a pass.

Image Replacement Techniques

9 Categories

- **CSS OFF / Images ON**
  Represents browsing with no stylesheets being applied. Most techniques default to regular web text here which isn't exactly a fail, but since images may still be turned on, I don't consider it a pass either.

- **CSS OFF / Images OFF**
  Represents browsing with both images turned off and no stylesheets applied. Defaulting to text here is considered a pass.

- **Extra Unnecessary Markup**
  Having to add markup for the sole purpose of image replacement is not ideal. Does not achieve true separation from content and design.

Technique #1

Background as foreground, display: none (hide the span)

HTML

```
<h1 id="technique-one">
  <span>CSS-Tricks</span>
</h1>
```

CSS

```
h1#technique-one {
  width: 250px;
  height: 25px;
  background-image: url(logo.gif);
}
h1#technique-one span {
  display: none;
}
```

REPORT CARD			
CSS ON IMAGES ON	CSS ON IMAGES OFF	CSS OFF IMAGES ON	CSS OFF IMAGES OFF
✓ PASS	✗ FAIL	TEXT ONLY	✓ PASS
EXTRA UNNECESSARY MARKUP?		ISSUES	
YES, SPAN		DISPLAY: NONE IS BAD FOR SCREEN READERS	

Technique #2 (move content out of viewport) Giant Box

HTML

```
<h1 class="technique-two">
  CSS-Tricks
</h1>
```

CSS

```
h1.technique-two {
  width: 2350px; height: 75px;
  background: url("images/header-image.jpg")
  margin: 0 0 0 -2000px;
}
```

REPORT CARD			
CSS ON IMAGES ON	CSS ON IMAGES OFF	CSS OFF IMAGES ON	CSS OFF IMAGES OFF
✓ PASS	✗ FAIL	TEXT ONLY	✓ PASS
EXTRA UNNECESSARY MARKUP?		ISSUES	
NO		REQUIRES GIANT BOX, LESS EFFICIENT THAN OTHERS	

Technique #3 (Move text to outer space)

HTML

```
<h1 class="technique-three">
  CSS-Tricks
</h1>
```

CSS

```
h1.technique-three {
  width: 350px;
  height: 75px;
  background: url("images/header-image.jpg")
  text-indent: -9999px;
}
```

REPORT CARD

CSS ON IMAGES ON	CSS ON IMAGES OFF	CSS OFF IMAGES ON	CSS OFF IMAGES OFF
PASS	FAIL	TEXT ONLY	PASS

EXTRA UNNECESSARY MARKUP?	ISSUES
NO	NONE

Technique #4 (Text as alt)

HTML

```
<h1 class="technique-four">
  <a href="#">
    <img src="images/header-image.jpg" alt="
  </a>
</h1>
```

CSS

```
h1.technique-four {
  width: 350px; height: 75px;
  background: url("images/header-image.jpg")
  text-indent: -9999px;
}
```

REPORT CARD

CSS ON IMAGES ON	CSS ON IMAGES OFF	CSS OFF IMAGES ON	CSS OFF IMAGES OFF
PASS	FAIL	PASS	PASS

EXTRA UNNECESSARY MARKUP?	ISSUES
ARGUABLY NO	HEAVYWEIGHT, REQUIRES TWICE THE IMAGES LOADED

Technique #5

(Text as alt and hiding span)

HTML

```
<h1 class="technique-five">
  <img src="images/blank.gif" alt="CSS-Tricks
  <span>CSS-Tricks</span>
</h1>
```

CSS

```
h1.technique-five {
  width: 350px; height: 75px;
  background: url("images/header-image.jpg")
  }
  h1.technique-five span {
    display: none;
  }
```

Technique #6

HTML

```
<h1 class="technique-six">
  CSS-Tricks
</h1>
```

CSS

```
h1.technique-six {
  width: 350px;
  padding: 75px 0 0 0;
  height: 0;
  background: url("images/header-image.jpg")
  overflow: hidden;
}
```

(Move Text out of box and set overflow hidden)
Same trick for textbook

Technique #7

(set 0 size and set overflow hidden)

HTML

```
<h1 class="technique-seven">
  <span>CSS-Tricks</span>
</h1>
```

CSS

```
h1.technique-seven {
  width: 350px; height: 75px;
  background: url("images/header-image.jpg")
  }
  h1.technique-seven span {
    display: block;
    width: 0;
    height: 0;
    overflow: hidden;
  }
```

Technique #8

HTML

```
<h1 class="technique-eight">
  <span></span>CSS-Tricks
</h1>
```

CSS

```
h1.technique-eight {
  width: 350px; height: 75px;
  position: relative;
  }
  h1.technique-eight span {
    background: url("images/header-image.jpg
    position: absolute;
    width: 100%;
    height: 100%;
  }
```

Technique #9

Meaningless font size

(Currently, most browsers are supporting alt property, these techniques are not as important as before.)

HTML

```
<h1 class="technique-nine">
  CSS-Tricks
</h1>
```

CSS

```
h1.technique-nine {
  width: 350px; height: 75px;
  background: url("images/header-image.jpg")
  font-size: 1px;
  color: white;
}
```

CSS Sprites

LECTURE 2

CSS Sprites

- CSS is a technique to reduce the number of image requests in order to improve the site performance. The large image that contains multiple images is know as a sprite, a term coined by the early computer graphic and video game industry. The image gets positioned in the element using the **background-position** property in such a way that only the relevant of it is visible.

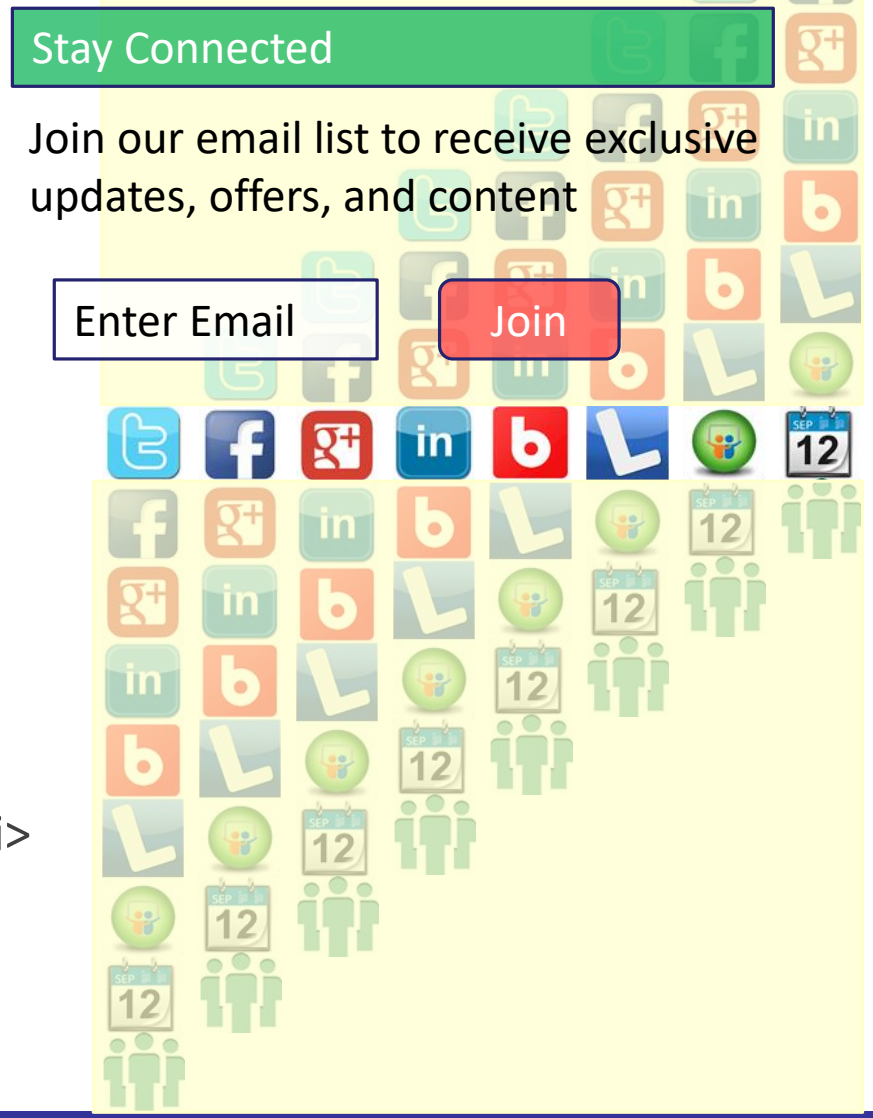
- The strategy:
  1) a list of elements used the same background of same sprite image.
  2) the elements has a common class and a individual class
  3) the common class share the same image file as background. The individual class assign different **background-position**.

CSS Sprite

(Modified from Textbook)

The Markup

```
<ul>
   <li><a href=“” class=“hide twitter”>Twitter</a></li>
   <li><a href=“” class=“hide fb”>Facebook</a></li>
   <li><a href=“” class=“hide gplus”>Google+</a></li>
   <li><a href=“” class=“hide linkedin”>LinkedIn</a></li>
   <li><a href=“” class=“hide blip”>Blip TV</a></li>
   <li><a href=“” class=“hide lanyrd”>Lanyrd</a></li>
   <li><a href=“” class=“hide slides”>Slideshare</a></li>
   <li><a href=“” class=“hide sched”>Schedule</a></li>
   <li><a href=“” class=“hide attendees”>Attendee List</a></li>
</ul>
```



CSS Sprites

The Styles

```
.hide { text-indent: 100%;  white-space: nowrap;  overflow: hidden;  }
li a { display: block;  width: 29px;  height: 18px;  background-image: url(social.png); }
li a.twitter        { background-position:  0  0; }
li a.fb             { background-position:  0  -20px; }
li a.gplus          { background-position:  0  -40px; }
li a.linkedin       { background-position:  0  -60px; }
li a.blip           { background-position:  0  -80px; }
li a.lanyrd         { background-position:  0  -100px; }
li a.slides         { background-position:  0  -120px; }
li a.sched          { background-position:  0  -140px; }
li a.attendees      { background-position:  0  -160px; }
```

CSS Styling Forms

LECTURE 3

Styling Forms

Step 0: Build the basic form

Step 1: Adding basic styles

Step 2: Aligning labels and inputs

Step 3: Fixing fieldsets and minor labels

Step 4: Adjusting the buttons

Step 1: Adding Basic Styles

ul, il: list layout

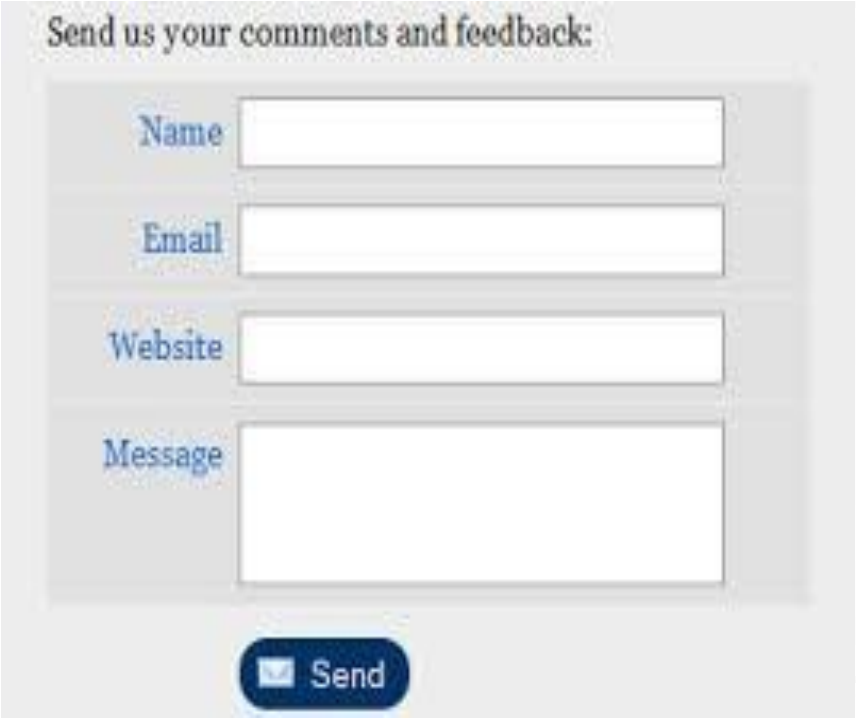
clear: next Element location

overflow: hideen

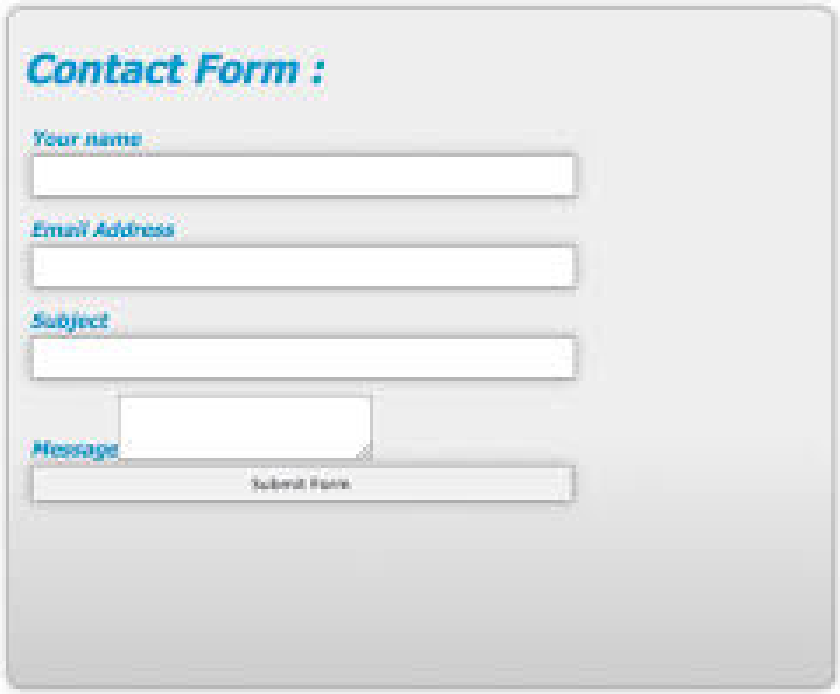
color:

font size:

Step 2: Align labels and inputs



(Aligned Right)



(Stay On Top)

(Aligned Left)

Step 3: Fixing fieldsets and minor labels

Issues:

- **fieldset** border
- **radio display** (inline or block)
- **checkbox** line up or run-down (use clear property)
- **text-align**: text alignment
- Margins, Colors

Step 4: Adjusting the Buttons

Type Selector:

```
input[type="submit"] { …; }   /* select only a certain type of input */
```

Possible Types: submit, reset, radio, checkbox, password, text, file, date, time, datetime-local, date-label, month, week, …

CSS Styling Tables

LECTURE 4

Styling Tables

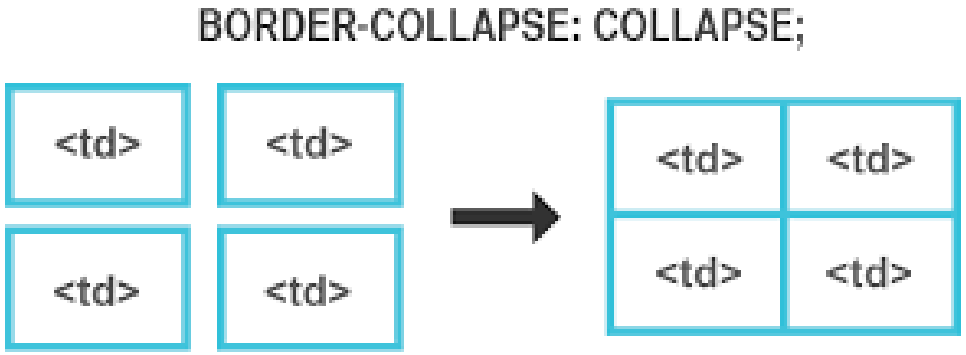
Separated and Collapsed Borders

border-collapse: separate;

/* default value, Refer to box model */

border-collapse: collapse;

/* see example on the right */



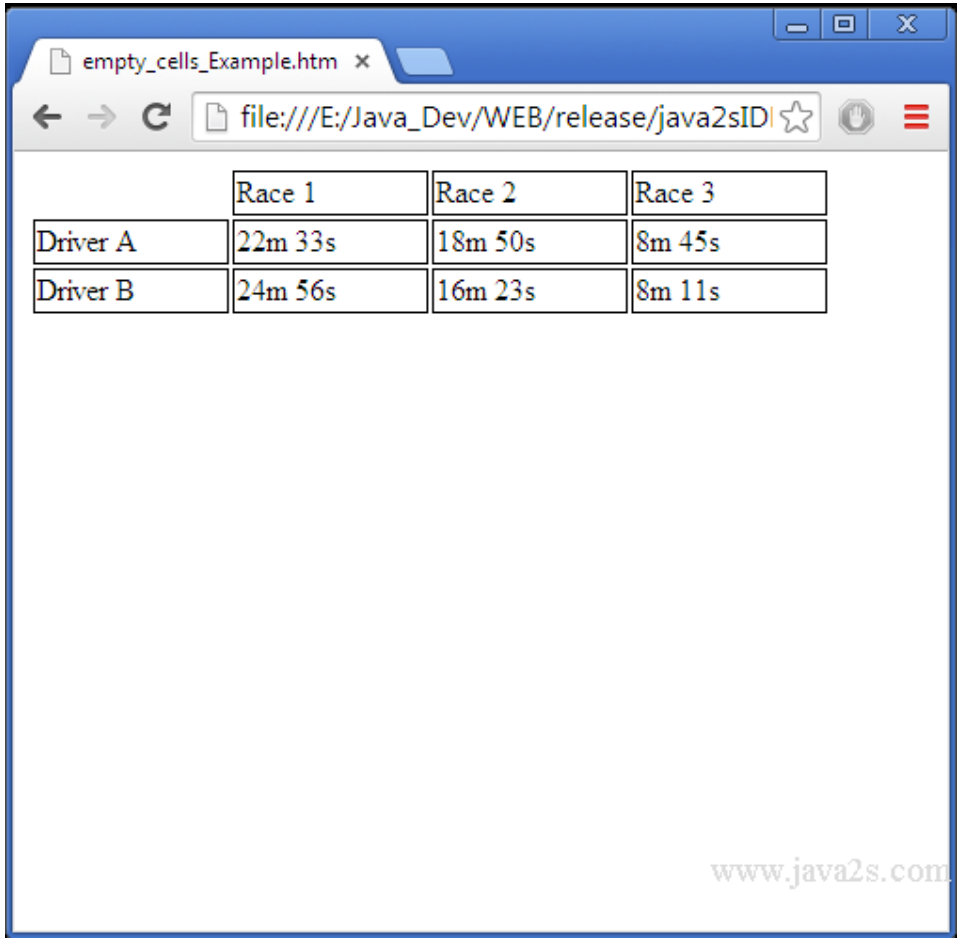
Border-style: hidden;

/* box model but does show border */

Styling Tables

Empty Cells

empty-cells: show | hide | inherit



Responsive Design

LECTURE 5

Basic Responsive Web Design

- **A Fluid Layout:** Fluid layout as Chapter 16
- **Flexible Images:** When the layout scales down, the images and other embedded media need to scale with it; otherwise, they would hang out of view. We'll make sure the Jenware images scale down to fit.
- **CSS Media Queries:** Media queries are a method for applying styles based on the medium via which the document is displayed. Queries start with questions, such as "Is the document being printed?" The use these print-appropriate styles." Or, "Is the document on a screen, and is that screen at least 1,024 pixels wide and in landscape mode? Then use these styles."

Setting the Viewport

- To fit standard websites onto small screens, mobile browsers render the page on a canvas called the **viewport** and then shrink that viewport down to fit the width of the screen (device width).
- Mobile Safari introduced the viewport **<meta>** tag that allows developers to control the size of that initial viewport. Soon other mobile browsers followed suit and this is an essential first step to a responsive design. Simply add the following meta element to the head of the HTML document:
- **<meta name="viewport" content="width=device-width, initial-scale=1">**
- This line tells the browser to set the width of the viewport equal to the width of the device screen (width=device-width), whatever that happens to be. The initial-scale sets the zoom level to 1 (100%).

Adaptive Layout

- **Adaptive web design** is just another term for ‘progressive enhancement’ of which responsive web design can (and often should) be an integral part, but is a more holistic approach to web design in that it also takes into account varying levels of markup, CSS, JavaScript and assistive technology support.
- Some examples of an adaptive strategy:
  - Programming touch event functionality, such as swiping through a gallery, for devices that understand touch events.
  - Adding geolocation functionality for devices that can track the user’s location (e.g., “Use my current location”).
  - Programming a canvas element to replace an image if the user’s browser supports HTML5.

Making Image Flexible

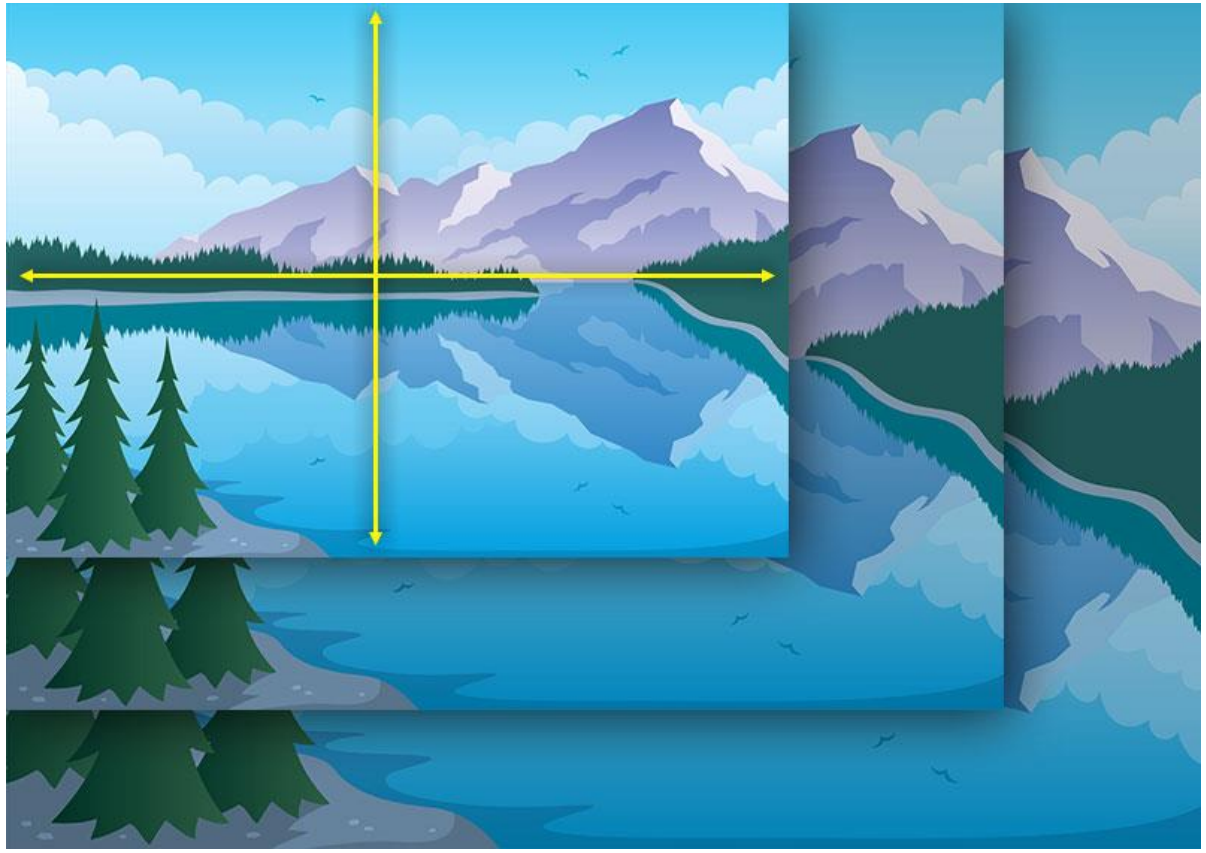
img { max-width: 100%}

Scale freely up to 100% of the image.

Issue:

(1) Raster-based image may lose information

(2) use SVG (vector-based) image would be better.



Media Query

LECTURE 6

Media Query Magic

Media Type:

print: print preview mode

speech: speech synthesizer

handheld: small-screen and limited bandwidth

braille: Intended for braille tactile feedback devices

projection: projector

screen: computer screen

tty: terminal

tv: TV

embossed: paged braille printers

all

Media Query Magic

Media features mostly evaluate by Media queries:
device-width, orientation, resolution.
(usually with min- or max- as the prefix for the testing condition)

Example:

```
@media screen and (min-width: 480px;) {
    /* put the style setting for this device */
}
```

Media Features you may test with @media queries

Feature	Description
width	The width of the display area (viewport).
height	The height of the display area (viewport).
device-width	The width of the devices rendering surface (the whole screen).
device-height	The height of the devices rendering surface (the whole screen).
orientation	Whether the device is in portrait or landscape orientation. (Does not accept in-/max- prefixes.)
aspect-ratio	Ratio of the viewport's width divided by height (width/height).
device-aspect-ratio	Ratio of the whole screen's (rendering surface) width to height
color	The bit depth of the display; for example, color: 8 tests for whether the device has at least 8-bit color.
color-index	The number of colors in the color lookup table.
monochrome	The number of bits per pixel in a monochrome device.
resolution	The density of pixels in the device. This is increasing relevant for detecting high-resolution displays.
scan	Whether a tv media type uses progressive or interlace scanning. (does not accept min-/max- prefixes.)
Grid	Whether the device uses a grid display, such as a fixed-width font.(Does not accept min-/max- prefixes.)

Media queries in the document head

- The **@media** queries we've been looking at so far in the style sheet itself. Media queries can also be carried out with the media attribute in the link element to conditionally load separate **.css** files when the conditions are met.

```
<head>
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href="2column-styles.css" media="screen and (min-width:780px)">
</head>
```

Mobile first media queries

- Take care of the mobile device first.
- Use the media queries to bring overriding styles that adapt the design as more display real estate and features become available.
- Mobile-first media queries tend to begin with the min- prefix, bring in new styles when the width is at least the specified width or larger.

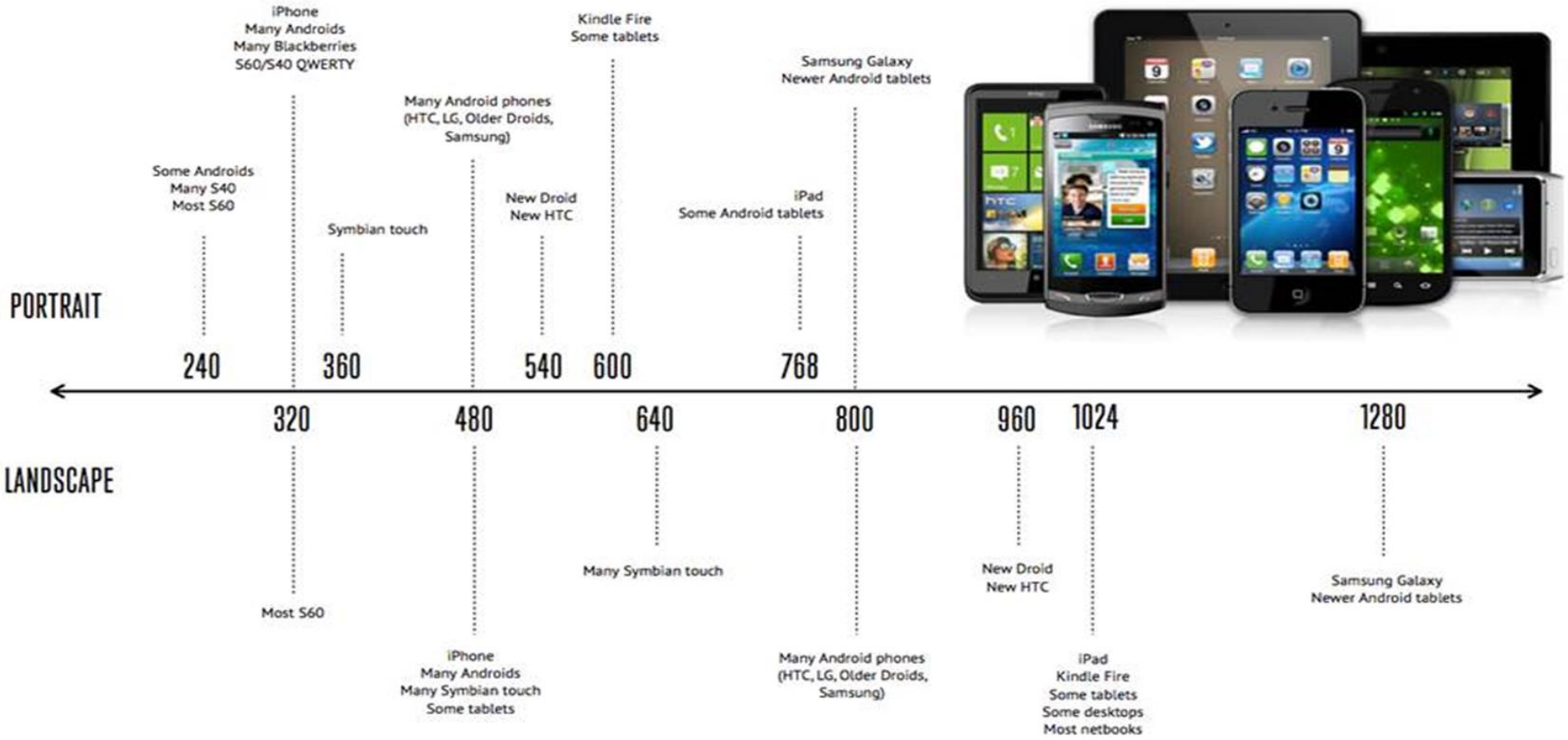
Tricky Bits

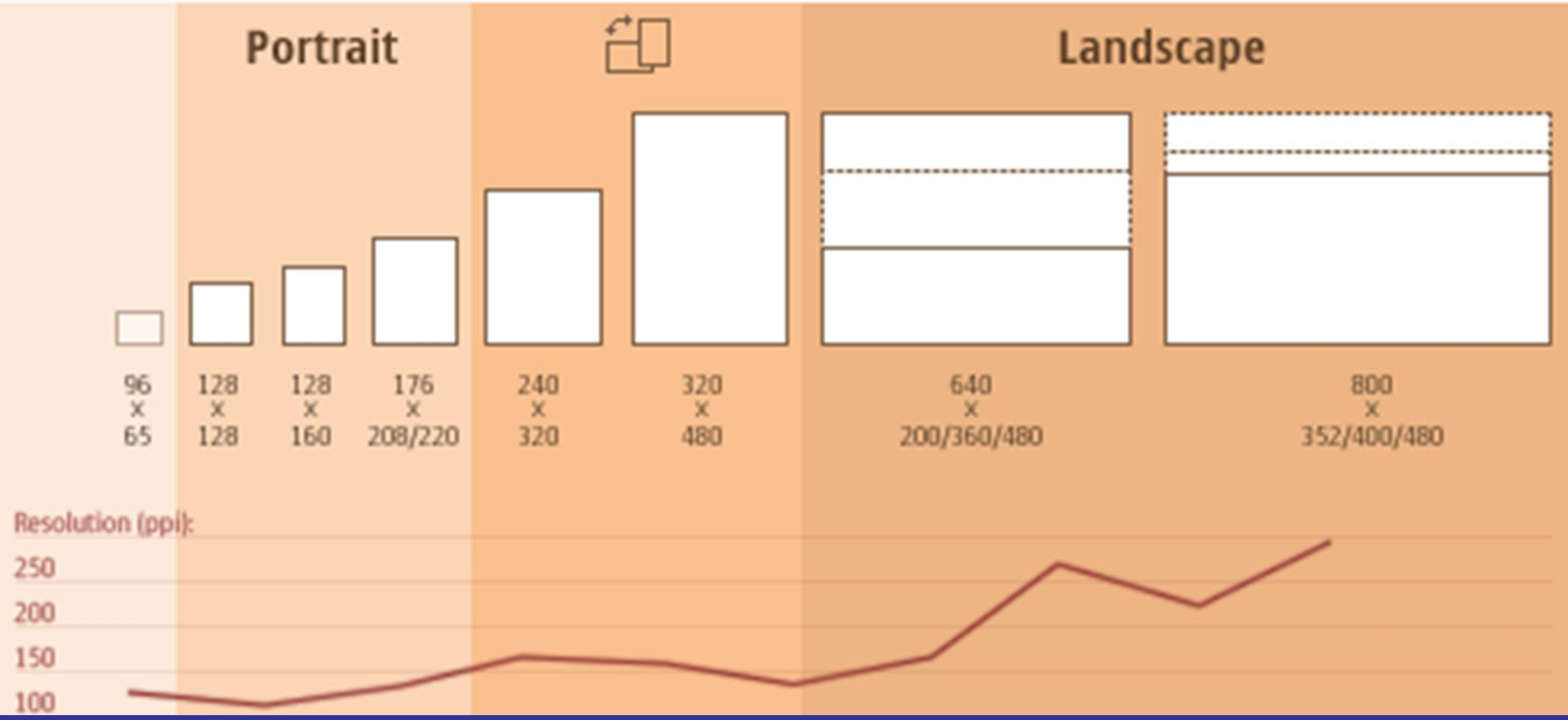
LECTURE 7

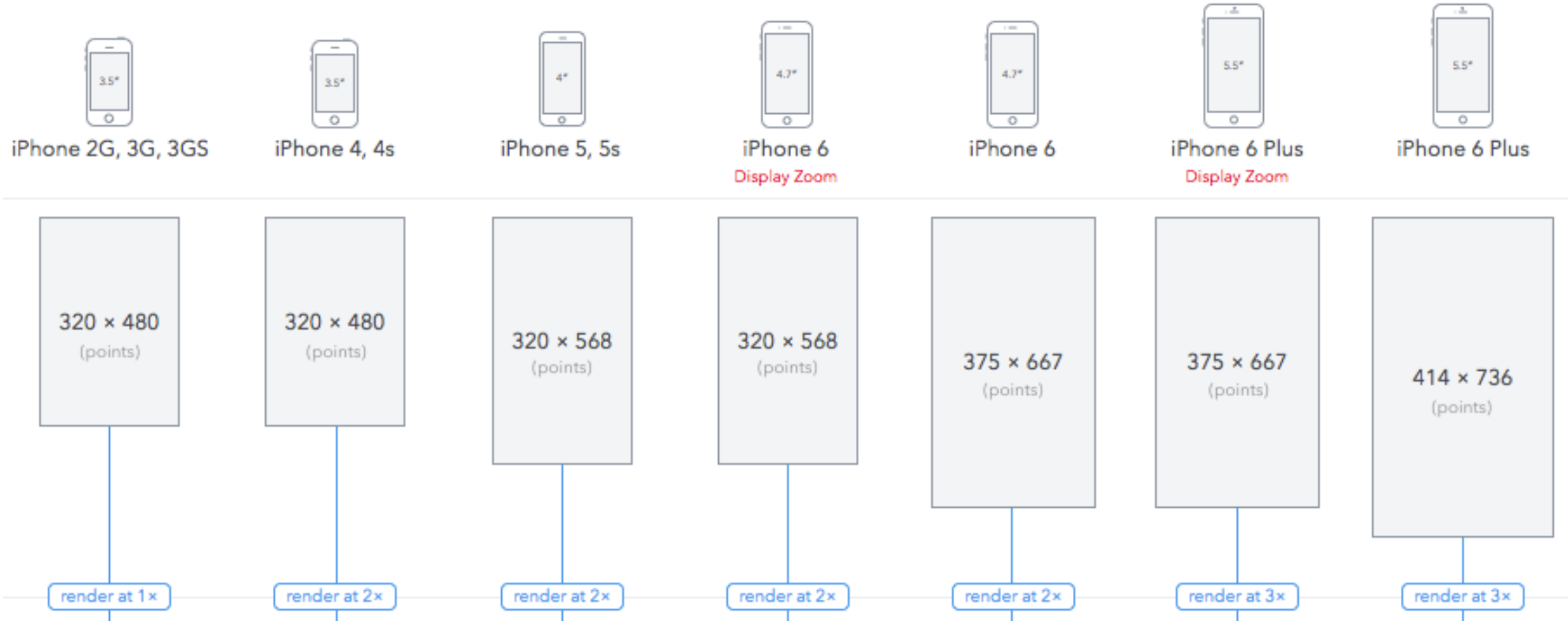
Tricky Bits

- **Choosing breakpoints:** One of the primary design decisions in creating a responsive design is deciding at which widths to introduce a significant design change. The point at which the media query delivers a new set of styles ins known as a breakpoint.
- **Responsive images:** One of the most vexing problems facing mobile web developers is how to get images right. Ideally, a device should download only the image size that is appropriate for its dimensions and network speed. The goal is to avoid downloading unnecessary data, whether that comes in the form of an image that is larger than it needs to be for a small screen or downloading two version s of an image when only one is needed. Check textbook for the responsive image information.

Devices and Screen Resolutions







Tricky bits

- One size doesn’t fit all
- Responsive limitations