Making Websites Mobile Friendly

Why mobile friendly, or mobile first?

- To have a mobile-friendly design is to satisfy all of the different use cases for your website or web application
- Some users may log on from a desktop, others from a phone, still others from a tablet
- Providing an optimized experience for their device of choice will ensure a lower **bounce rate**[1] and greater user satisfaction
- Some platforms even have the mobile version done first: mobile-first

[1] The percentage of users who leave your website immediately.

Achieving Mobile-friendly-ness

- To get started with making your website mobile friendly, you can keep these points in mind:
  - Your website works well regardless of the screen *width* it is being viewed at
  - Interactions that rely on a :hover event or the presence of a mouse need to also work in a touch-only environment

Different Widths, Different CSS

- A **media query** allows you to specify CSS to use *only* in the case where a certain screen size is being used
- The use of media queries to accommodate different *breakpoints*, or screen resolutions has been popularized due to the increased use of mobile computing devices

Anatomy of a Media Query

Apply this query to all device types:

```
@media all
```

Apply these styles to screens 960px and below:

```
(max-width: 960px)

@media all and (max-width: 960px){
  body{
    background-color: orange;
  }
}
```

`max-width`, `min-width`, etc

- max-width and min-width are used to set the boundaries of a media query
- The following rule would apply to anybody viewing your page between 481px and 960px window width:

```
@media all and (max-width: 960px) and (min-width: 481px)
```

`max-width`, `min-width`, etc

- The following rule would apply to anybody viewing your page with a browser width below 480px:

```
@media all and (max-width: 480px)
```

- The following rule would apply to anybody viewing your page with a browser width above 1300px:

```
@media all and (min-width: 1300px)
```

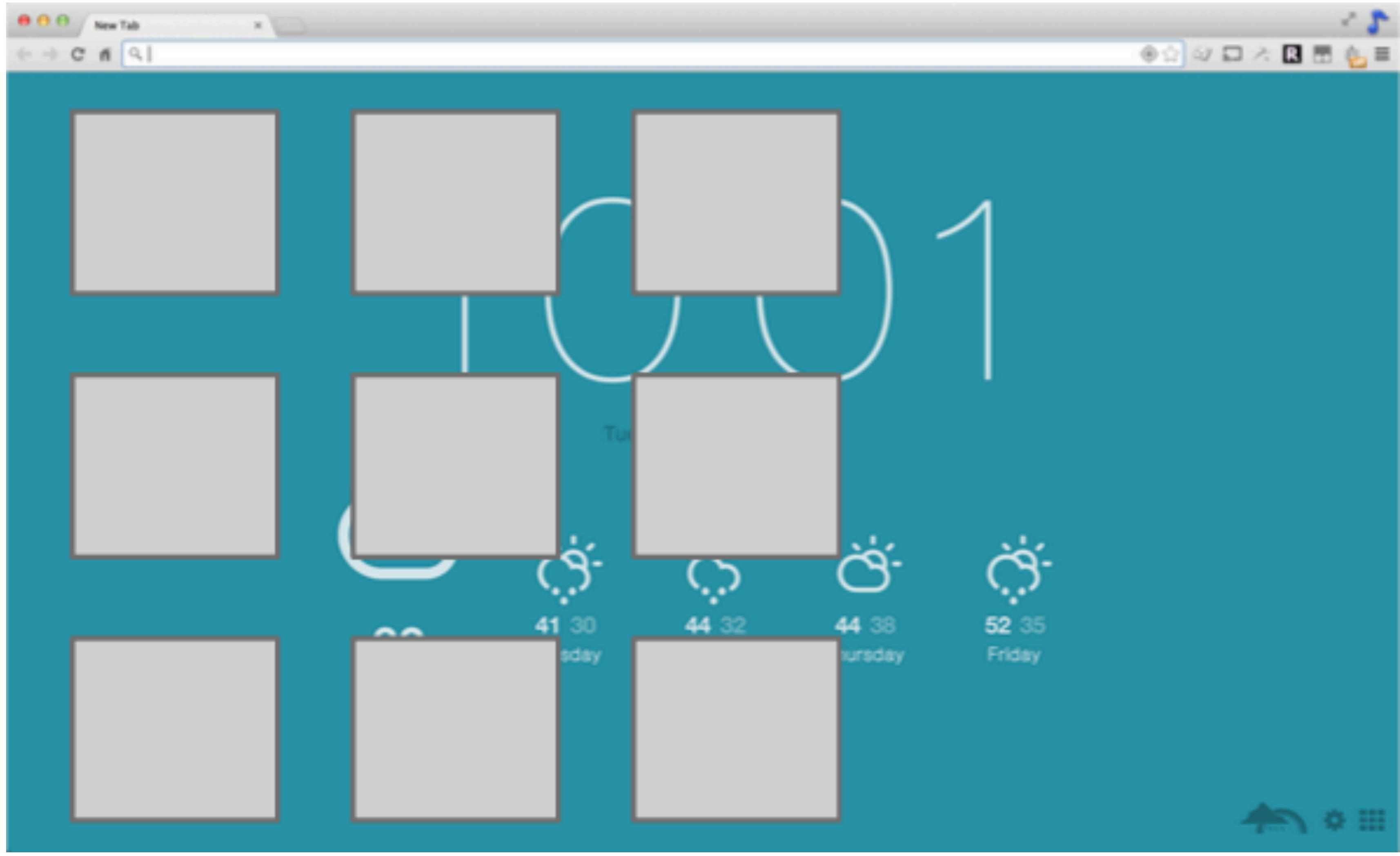
Example Use Cases

- A media query can be used to show or hide certain elements on the page based on the page's width
- For instance, perhaps you have an "additional information" section on an event that would be a bit of information overload on a mobile device
- Use `display: none;` to hide this section, on mobile devices only, with a media query

Example Use Cases

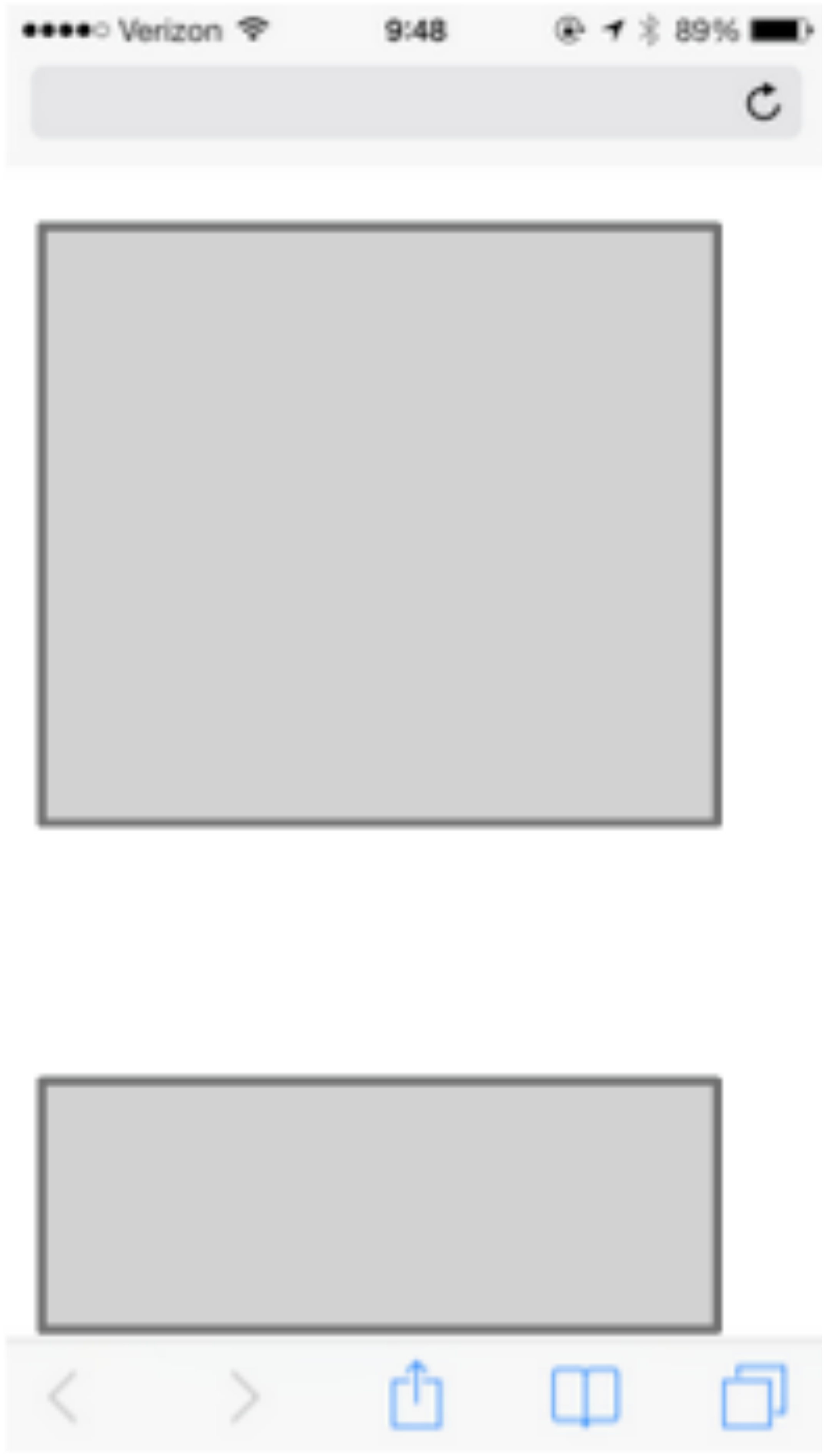
- A page with items aligned next to each other might need each of them to have a margin on the left in order to put some space between the items

(next slide for mobile use case)



Example Uses

- In a one-column mobile layout, it might make sense to set `margin-left: 0;`, since the items are not next to each other past a certain width



One more thing

In order to ensure proper rendering and touch zooming, add the viewport meta tag to the `<head>` section of your website:

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

Exercise

- Build an image gallery using floats and the CSS `background-image` property on `<div>`s instead of `<img>` tags that:
  - also looks good when the browser is resized to 400px
  - uses a different, smaller background-image for each image when the browser is resized
  - has a frame around each image in full-size width and no frame at 400px
- To get a readout of the browser width, make sure the Chrome Developer tools are open (Command + Option + I)

Where do I put my breakpoints?

- There are differing philosophies on the best way to lay out your break points
- Mobile-first: you don't have a breakpoint for mobile devices but instead, your design begins at the mobile level and extends from there
- Desktop-first: you build your website for the desktop, then make it mobile and larger-screen friendly from there

Mobile-first

```
/* smaller than a tablet is the default design*/

/* Small devices (tablets, 768px and up) */
@media (min-width: 768px) { ... }

/* Medium devices (desktops, 992px and up) */
@media (min-width: 992px) { ... }

/* Large devices (large desktops, 1200px and up) */
@media (min-width: 1200px) { ... }
```

"Desktop-first"

```
/* Large desktop */
@media (min-width: 1200px) { ... }

/* Portrait tablet to landscape and desktop */
@media (min-width: 768px) and (max-width: 979px) { ... }

/* Landscape phone to portrait tablet */
@media (max-width: 767px) { ... }

/* Landscape phones and down */
@media (max-width: 480px) { ... }
```

Example Source

- These media queries were taken from versions 2 and 3 of Bootstrap, a popular responsive web design framework
- Version 2 of this framework was "desktop-first", while version 3 was "mobile-first"
- You can use whatever breakpoints you'd like, but this is a good starting point

Testing your responsive designs

- The first way a responsive design is usually tested is to just resize your browser window
- The *best* way a responsive design is tested is by actually using it on a target device
- Of course, we can't all afford every device and device version ever
- Put your website on a password protected server and take a walk over to the Apple or Sony Store for some peace of mind...
- At the very least, test your design in the major browsers: Chrome, Firefox, Internet Explorer, Safari

Read this article

To understand responsive web design better:
alistapart.com/article/responsive-web-design

Everyone will read this and then we'll move on!

Exercise

- Using the provided `acme-desktop` and `acme-mobile` png files as a reference, create a desktop-friendly and mobile-friendly version of the ACME homepage using breakpoints

Extended Exercise

- Take an existing project (or three) and make them mobile friendly with the help of your instructor!