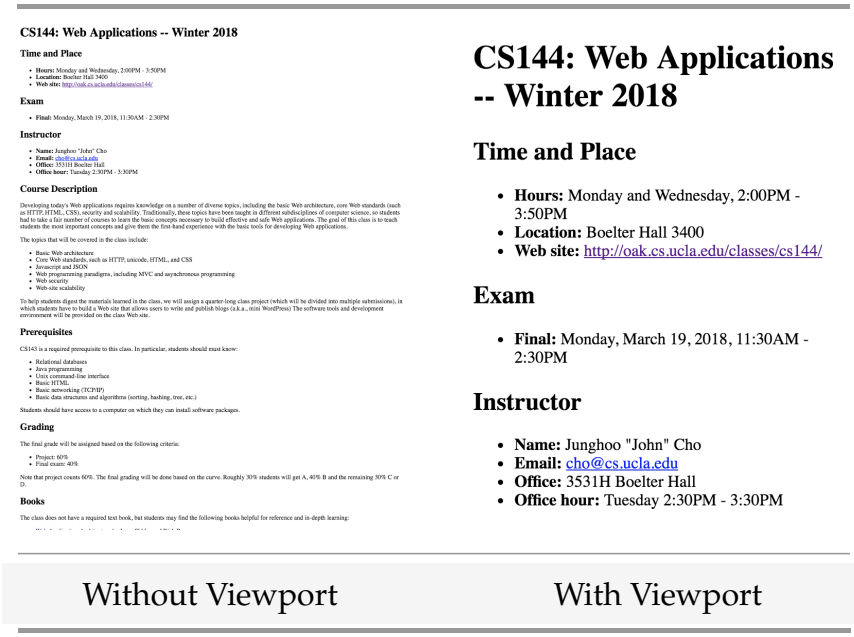
Responsive Web Design (RWD)

- Responsive Web Design: design Web pages, so that it is easy to see on a wide range of of devices
  - phone, tablet, desktop, ...
- Fixed vs Fluid layout
  - *Fixed*: elements have fixed width. Resizing the window does not change the appearance of the page
  - *Fluid*: elements uses "percentage" of page width. Elements dynamically resize to fit window width
- General rules
  - Do NOT force users to scroll horizontally (Why?)
  - Do NOT use fixed-width elements (Why?)
  - Use CSS media queries to apply different styling depending on the screen size

Viewport

- `<meta name="viewport"content="width=device-width, initial-scale=1">`
- `viewport`: user's visible area of a web page
  - width: viewport width
  - initial-scale: initial "zoom level"
- Always add a viewport meta tag
  - Otherwise, default viewport width (~ 980px) is used, which can make text too small



CS144: Web Applications -- Winter 2018

Time and Place

- Hours: Monday and Wednesday, 2:00PM - 3:50PM
- Location: Boelter Hall 3400
- Web site: http://oak.cs.ucla.edu/classes/cs144/

Exam

- Final: Monday, March 19, 2018, 11:30AM - 2:30PM

Instructor

- Name: Junghoo "John" Cho
- Email: cho@cs.ucla.edu
- Office: 3531H Boelter Hall
- Office hour: Tuesday 2:30PM - 3:30PM

Course Description

Developing today's Web applications requires knowledge on a number of diverse topics, including the basic Web architecture, core Web standards (such as HTTP, HTML, CSS), security and scalability. Traditionally, these topics have been taught in different subdisciplines of computer science, so students had to take a fair number of courses to learn the basic concepts necessary to build effective and safe Web applications. The goal of this class is to teach students the most important concepts and give them the first-hand experience with the basic tools for developing Web applications.

The topics that will be covered in the class include:

- Basic Web architecture
- Core Web standards, such as HTTP, unicode, HTML, and CSS
- Javascript and JSON
- Web programming paradigms, including MVC and asynchronous programming
- Web security
- Web-site scalability

To help students digest the materials learned in the class, we will assign a quarter-long class project (which will be divided into multiple submissions), in which students have to build a Web site that allows users to write and publish blogs (a.k.a., mini WordPress) The software tools and development environment will be provided on the class Web site.

Prerequisites

CS143 is a required prerequisite to this class. In particular, students should must know:

- Relational databases
- Java programming
- Unix command-line interface
- Basic HTML
- Basic networking (TCP/IP)
- Basic data structures and algorithms (sorting, hashing, tree, etc.)

Students should have access to a computer on which they can install software packages.

Grading

The final grade will be assigned based on the following criteria:

- Project: 60%
- Final exam: 40%

Note that project counts 60%. The final grading will be done based on the curve. Roughly 30% students will get A, 40% B and the remaining 30% C or D.

Books

The class does not have a required text book, but students may find the following books helpful for reference and in-depth learning:

CS144: Web Applications -- Winter 2018

Time and Place

- **Hours:** Monday and Wednesday, 2:00PM - 3:50PM
- **Location:** Boelter Hall 3400
- **Web site:** http://oak.cs.ucla.edu/classes/cs144/

Exam

- **Final:** Monday, March 19, 2018, 11:30AM - 2:30PM

Instructor

- **Name:** Junghoo "John" Cho
- **Email:** cho@cs.ucla.edu
- **Office:** 3531H Boelter Hall
- **Office hour:** Tuesday 2:30PM - 3:30PM

Without Viewport | With Viewport

Media queries

- Media query allows applying different css rules depending on device property
- Example

```
@media screen and (max-width: 800px) {
    /* CSS rules */
}
```

  - Apply the CSS rules only if the page is displayed on screen and viewport width is 800px or less
  - *Breakpoint*: the viewport-width boundary for applying different rules. 800px in this example
- Possible conditions in media queries
  - Media types
    - screen, print, speech, and all (default)
  - Media features
    - orientation, min-width, max-width, min-height, max-height, resolution, ...
  - Boolean operators

    - , =OR, `and`=AND, `not`=NOT
    - Precedence: `not` > `and` > ,

- Q: When does the following rule apply?

```
@media screen, (orientation: portrait) {
    /* ... */
}
```

CSS Flexbox

- "Flexible box"
  - New addition to CSS to help create flexible layout of elements
  - A *flex container* (`display: flex;`) includes many *flex items*.

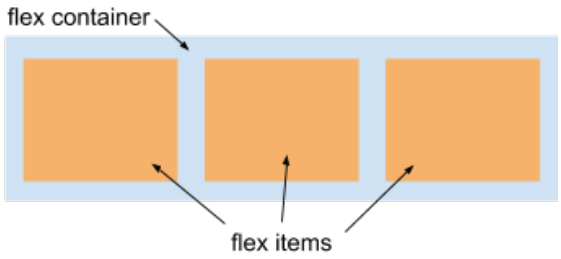


Figure 1: CSS flexbox

  - Flex items are arranged horizontally (`flex-direction: row;`) or vertically (`flex-direction: column;`)
  - Flex items are dynamically resized as the container size changes
    - grow/shrink factors can be specified (`flex-growth`, `flex-shrink`, and `flex-basis`).
  - Example: http://oak.cs.ucla.edu/classes/cs144/examples/css-flexbox.html

```
<!DOCTYPE html>
<html>
<head>
<title>CSS flexbox</title>
<style>
    #container {
```

```
        display: flex;
        flex-direction: row;
    }
    nav   { flex: 1 2 100px; }
            /* flex-growth flex-shrink flex-basis */
    #main { flex: 2 1 200px; }
    aside { flex: 1 2 100px; }

    * { text-align: center; border: 1px solid black; }
</style>
</head>
<body>
    <header>Header</header>
    <div id="container">
        <nav>Navigation</nav>
        <div id="main">Main Text</div>
        <aside>Aside</aside>
    </div>
    <footer>Footer</footer>
</body>
</html>
```

Dynamic image size

- `<img src="a.png"style="max-width: 100%;">`
- `max-width: 100%;` ensures that image width is not larger than 100% of its parent element

"Grid system"

- Popular CSS frameworks partition a page into multiple "columns" or "grids"
  - e.g., Bootstrap uses 12-column grid system
  - grid system makes it easy to design and adjust page layout
- Each HTML element (typically `div`) occupies a certain number of columns
  - Bootstrap example:

```
<div class="container">
    <div class="row">
        <div class="col-4">First column</div>
        <div class="col-8">Second column</div>
    </div>
</div>
```

    * First and second columns occupy 33%, and 66% of content width, respectively
- Column arrangement can be adjusted depending on the screen size
  - Bootstrap

```
<div class="container">
    <div class="row">
        <div class="col-6 col-md-4">First column</div>
        <div class="col-6 col-md-8">Second column</div>
    </div>
</div>
```

    * `md` stands for "medium" (~ desktop).
      · `xs`: x-small (phone), `sm`: small (tablet), `lg`: large (large desktop)
    * By default, first and second columns occupy 50% each
    * For "desktop" (and up), first and second columns occupies 33% and 66%, respectively

References

- More detailed explanation on flexbox: https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- Bootstrap: https://getbootstrap.com/