Copenhagen 426

You’re here!

- Welcome to Denmark!!!
- Let’s do more group introductions!

Course Goals

- You will exit this course understanding the 3-tier architecture of web applications.

    1. Client-side concerns
        A. Document Object Model classes, interfaces, and events
        B. Application Architectures using Frameworks (such as React, Vue, Angular)
    2. Server-side concerns
        A. HTTP’s stateless request and response protocol
        B. Application Architectures using Frameworks (such as Express.js)
    3. Data layer concerns*
        A. Document/Object Stores vs Relational Stores
        B. Modeling application’s needs for data storage and querying
    4. Connections
        A. How do the tiers communicate with one another?
        B. How are user-specific concerns addressed? (Authentication, Authorization, Logic)

Reference Material

- Pro-tip:

  Use MDN (Mozilla Developer Network)

  Avoid W3Schools

- When searching for HTML/CSS/JS concept, prefix search with ‘mdn’

  - Example: ‘mdn getElementById’

Web Tech Moves Fast

- The web is a living, dying, evolving, expanding technology platform
    - Only mobile development, machine learning, and devops have a similar velocity of change as the web
    - The scale, expanse, and diversity of web technology exceeds these as a larger population of developers actively innovates on the web
- Some of the specific technologies and techniques you learn today will be obsolete in a few years.
- That’s ok and expected!

My Goal for 426

- Go on a magical study tour of the big, important, high-level concepts.
- Leave understanding:
    1. The architecture of a 3-tier web app
    2. The kinds of technology at play in each tier
    3. The general concerns, strategies, and best practices of each tier
- Leave with new and improved skills:
    1. HTML, CSS, JavaScript/TypeScript (Front and Back-end)
    2. How to Google, read, learn, and apply techniques on the fly.

The Learning Experience

- We face a different challenge from courses like 110 or 401!

- In 110 and 401, the surface area of knowledge is narrow (programming!) and the expected level of mastery is deep.

- In 426, the surface area of knowledge is wide and the expected level of mastery is shallow.

- On-line resources for any individual topic (such as Internetting is Hard) are abundant and often high quality

- We will *intentionally* lean on freely available on-line resources where possible, as opposed to controlling a precise narrative in slides, because this is how real web development is done.

HTML & CSS

Readings

- The chapters you read in the pre-arrival material included:

    1. HTML & CSS
    2. Basic Web Pages
    3. Links and Images
    4. CSS
    5. CSS Box Model
    6. CSS Selectors
    7. Floats
    8. Flexbox

Web Design Notes

On Taste

- Ira Glass: https://vimeo.com/24715531

Wireframe Layout

- When designing a web site or a desktop/ mobile app, starting from a wireframe layout is recommended.

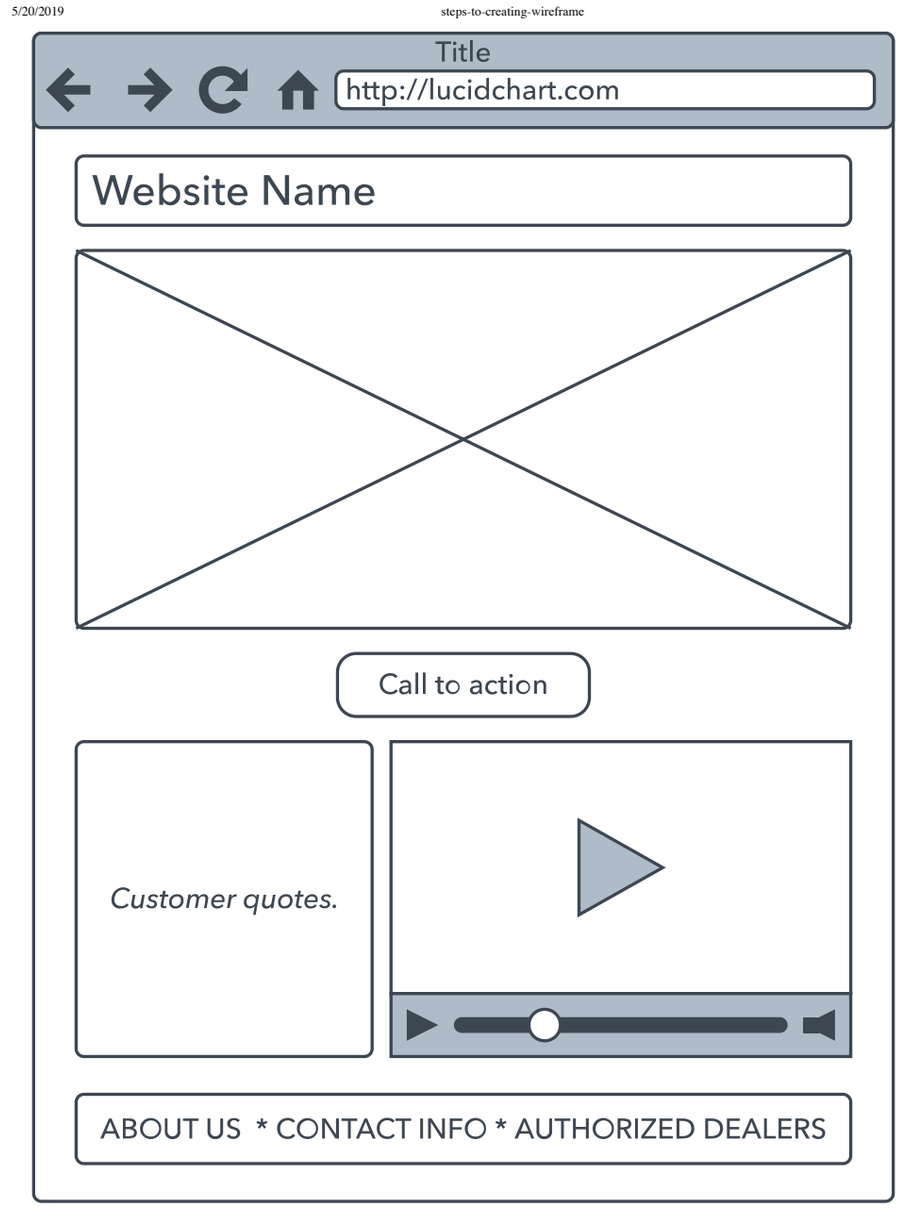
- A wireframe simply places boxes comprising the high level components of a screen.

- There are tools for designers to quickly create wireframes.

  - The example left is from LucidChart

  - MarvelApp.com is considered a leading, modern wireframe tool.

- Not working collaboratively/remote Sharpie on Paper or Dry Erase on Whiteboard works great!



Wireframe to Design

- COMP426 does not address graphic design tools & technique.
  A wireframe is the extent of our “design outside of code” tooling.
  - Graphic design is beyond the scope of 426 and gets coverage in School of Media & Journalism and Art Department.
  - However, it’s worth knowing the general process in industry.
  - After wireframes are agreeable, mockups take a high-level pass at filling in the boxes with actual design elements and decisions.
  - Once a mockup is decided upon, it will be further fleshed out into a “final comp”, short for comprehensive layout.
  - Final comps are handed off to a front-end developer to translate from visual graphics into HTML and CSS suitable for the web.

What can you do with only a wireframe?

- Wireframes help you think about the structure of your website's HTML and CSS *before* you start laying down code.

- Having an outline guide where you are going will help you divide-and-conquer the task of translating your wireframe/design from concept to code.

- Each "area" with more than one element likely deserves its own **div**ision tag or semantically appropriate container.

- Each part of the design with elements laid out left-to-right likely needs a **flex** layout in CSS (or float).

- Where there is text in a wireframe, it is common to begin coding with "lorem ipsum" text

    - Demo: https://loremipsum.io/

    - History: https://www.lipsum.com/

Kris' Hot Design Tip #1

- Typography is a subtle art form with a ***hefty*** impact.
  - It's a dimension of a design most people can't put a finger on.
  - Unless it's done in egregiously poor taste!

- Why does the text of a post on Medium or New York Times look so Nice and read so well compared to the average professor's web site?
  The typography styling!

- Tip: Mimic the typographic choices of a site you like.

- Resources:
  - Web Typography: https://internetingishard.com/html-and-css/web-typography/
  - Typography in ten minutes: https://practicaltypography.com/
  - Practical Typography: https://practicaltypography.com/
  - High-quality free fonts: https://practicaltypography.com/free-fonts.html

JavaScript

A letter to future you…

- Take out a piece of paper and write yourself a short letter to you at the end of next week.

- Remind yourself:
    1. How special it is to be abroad in Copenhagen
    2. How excited you are to be here
    3. How little time you have here so make the most of it

- Fold it up and tuck it somewhere in your bag out of the way.

The <script> Tag is the Web's Super Power

- If you add the code to the right to a page, you are running *ECMAScript (JavaScript) code* in the context of your web page!

```
<script type="text/javascript">
    alert("Hello, world.");
</script>
```

- When the browser reaches the script tag, it evaluates it immediately.

- Typically you *should not* write JavaScript "inlined" with HTML.

- Like CSS, scripts are written in .js files and included via a src attribute.

JS Reference

- Great Reference on Modern JS: https://javascript.info/
- Let’s work through some ideas specific to JavaScript not often seen in other languages:
    - https://javascript.info/javascript-specials

Node.js

- Node.js is a platform for running JavaScript outside of a browser
- It is powered by Google Chrome’s V8 Engine!
- Let’s download and install Node 10 on our host machines:
  - https://nodejs.org/en/

npm
Node Package Manager

- Modern JavaScript projects tend to incorporate open-source libraries (such as React, lodash) and build tools (WebPack, Parcel, TypeScript)

- Once node is installed, npm is available at the command-line

- Common commands:

- **npm init** - Starts a project

- **npm install --save <package>** - Install a package to project

- **npm install -g <package>** - Install a package (tool) globally

- **npm run <script>** - Runs a specific script in the project

Running a Simple Server

- Let’s install a useful tool: http-server
- It’s a node.js package that ships with a command to start a simple HTTP server to serve a directory of files
- For now, we’ll use it to serve our travel notes’ public dir
- Commands:
- Install: **npm install -g http-server**
- Run (from project directory): **http-server public**
- Shut down: **Ctrl+C**

Including External JS

- The typical way of including a script is to use the src attribute of the script tag:

```
<script type=“text/javascript” src=“./some-file.js”></script>
```

- Like with CSS, this will load the JavaScript file from the neighboring folder.

- IMPORTANT - This should go just before your closing </body> tag to avoid blocking loading other content.

- We’ll look at organizing multi-file JS projects and TypeScript next.