Module 1 Lesson Plans

GitHub is encouraged in this course. Students can create repositories when assignments are first introduced and submit links to those repositories. Students will still have to zip up a folder submission to Moodle, however the repository can be their backup if they miss the due date. Instructors can check the commits made to the repository which should help with academic dishonesty issues. To keep the student repositories private (so that assignments don’t get shared) but viewable to instructors use GitHub classroom.

Day 1 - Introduction & Hello Build Activity

**What to cover:**

Spend the first half of class going over the course outline and marking distribution with the students. Set expectations for the course and review and online guidelines and academic dishonest policy.

The second half of class is for the 'first build' activity. Allow students to attempt to complete this on their own for next class. This activity allows some insight to current skill level of the students and it gets them to think about and practice the outcomes from Web Fundamentals 1. Since we can't walk around the classroom to observe students working on this activity, have students upload what they have completed by the beginning of next class for your review.

This activity is not worth any marks.

**What to do:**

For homework have students complete the Hello Build.

Days 2 & 3 - Hello Build as a Mentored Build

**What to cover:**

Go through the hello build with the students, prompting them for input and discussing best practices and techniques as you go. As a mentor build, expect this demo to require two classes to complete.

Press the importance of properly indented code, using meaningful class names, title tags and alt attributes.

Other things to discuss and touch upon:

- start by creating the HTML structure, use the outliner tool, validate the HTMl, then add CSS styling

- naming practices for best URL's (all lowercase, no spaces, use dashes, cannot start with a number, use semantic naming)

- use semantic HTML & HTML5 sectioning elements

- write consistently formatted code

- what is included in a head tag

- inline vs embedded vs external CSS (separate content from presentation)

- element selectors, class names, & ids

- organize an top-down structure in CSS

- block vs inline elements

Optional: Have students create collaborative notes (google docs) on best practice techniques.

Students should have some base knowledge of Flexbox from Web 1 but this is a good opportunity to discuss flexbox vs floats layouts. Flexbox responds to the content, ie it will layout the content first, then the space. Review the concepts of a flex container and flex items.

**What to do:**

Build - Hello Build

Demo - When to use floats

Day 4 - CSS Resets & Dev Tools

**What to cover:**

Review basic folder structures for a starting framework. Discuss the options for CSS reset and pros and cons of each. Normalize, a Modern CSS Reset, and Bootstraps Reboot.

Emmet HTML Boilerplate https://medium.com/@ole.ersoy/instant-html-boilerplate-in-vscode-with-emmet-3176c0c9ff7e

Instructors are free to choose their own workflow setup for the course. Students should be encouraged to research and understand the code they bring into their projects.

Adding a description meta tag to head. <https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/The_head_metadata_in_HTML>

Show and discuss CSS Box Sizing as it is included in the Modern CSS Reset & Reboot.

<https://developer.mozilla.org/en-US/docs/Web/CSS/box-sizing>

**What to do:**

Demos - CSS Resets and Box Sizing

Can include an activity for working with Dev Tools

Day 5 - Image Optimization for Web

**What to cover:**

"The foundation of the whole internet is data compression"

Using the correct image format matters for both performance and appearance. Review the following types and when to use them.

- JPEG

- PNG

- GIF

https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/image-optimization#selecting\_the\_right\_image\_format

Introduce SVG and WebP discuss the benefits of both of these formats.

Three primary ways to use SVGs:

- SVG as HTML img

- SVG as a CSS background image

- Inline SVG (SVG is markup language too)

Note: Students will learn more about the SVG elements and attributes in Web Usability 2.

WebP browser support is getting better, it is a format that was made for the web and compresses ~30% more than JPG & PNGs. The HTML5 <picture> element can be used to create a fall back for browsers that do not offer support. We will cover how to do that later in the course, but for now students should be aware that this format exists.

On top of choosing the correct image format, we also need to optimize images before using them on the web. There are general guidelines but no hard and fast rules for image file size. Developers are often experimenting to balance image quality and small file sizes.

Images will mainly be JPG, PNG, or SVG for this part of the course.

Steps to prepare images for web using Photoshop:

1. Resize images to the largest size they will be rendered as

2. Save images with the proper format and reduced/optimized quality. In Photoshop this is best done by -> Export -> Save for Web (Legacy)

3. Lastly, run the image through a third party compression tool

- Imagemin (run as a dependancy)

- ImageOptim (Mac app)

- https://squoosh.app/ (Online drop and drag tool)

Note: Squoosh can resize images within the tool and as a side by side comparison it did a better job than Photoshop for compression.

Further reading: https://web.dev/fast/#optimize-your-images

**What to do:**

Demos - Image Optimization. Have students practice resizing and compressing large stock images. Scalable Vector Graphics. Demonstration how a SVG scales and resizes to fit the container it is in without losing quality, and the different ways of including a SVG in a project.

Day 6 - Introduction to Git & GitHub

**What to cover:**

Goal - By the end of class students will be able understand what Git is and the benefits of using it in our projects. Students will also have created their own GitHub account and have uploaded "Hello Build" as their first repository.

Instructors can start off the students with a local repository using either terminal or GitHub Desktop.

Walk students through the process of initializing a local repository, checking the status, adding files, making commits and pushing them to the remote repository. Branches and merging can be covered later, after students have some practice with the basics.

Note: Students might have to set up their name and email configuration to make a commit.

Git config --global user.name “John Doe”

Git config --global user.email “johndoe@example.com”

Students can easily push to a server from GitHub using Netlify.

<https://www.netlify.com/blog/2016/09/29/a-step-by-step-guide-deploying-on-netlify/>

**What to do:**

Have students create their first repo with the Hello build.

Quiz - Best Practice Quiz

Day 7 - Flexbox Layouts

**What to cover:**

The two axis of the flex box row and how they both work. Practice both justify-content and align-items.

**What to do:**

Demos - Flexbox Content to demonstrate how flexbox responds to the content inside a flex container and the spacing within the row on both the main and cross axis. Build a Flexbox Nav.

Day 8 - Flexbox Layouts

**What to cover:**

Flex direction and flexflow.

**What to do:**

Demos -Flex Flow as a shorthand to flex-direction and flex-wrap. Flexbox Columns, build a 3 column layout that adjusts to different screen widths using flex-wrap. (Stages of squishiness)

Day 9 - Introduction to Media Queries

**What to cover:**

Introduce the proper structure of a media query and demonstrate how they work. Enforce the importance of a mobile first workflow and structure of the CSS.

**What to do:**

Demo - media queries. Review GitHub and help students upload the demos to their GitHub account.

Day 10 - RWB

**What to cover:**

What is RWD?

https://alistapart.com/article/responsive-web-design/

Note: This is now 10 years old. Since then we have added Flexbox and CSS Grid to our toolbox and have nowgrown beyond this start.

The three requirements of a fully responsive web design are:

1. fluid/liquid layout (moved into mixed fluid & fixed era)

2. flexible images (images max-width:100%)

3. media queries

Introduce relative units and why we want to use them. The differences between Ems and rems. Rems have full browser support and are based on the default font size on the root (html) element, it’s not recommended to use them for media queries.

<https://css-tricks.com/is-it-better-to-use-ems-rems-than-px-for-font-size/>

**What to do:**

Demo - Fluid & Fixed Layout.

Demo - How Ems, Rems, and Px work when the browser default font size is changed. Chrome -> Settings, -> Font size. Demo advanced selectors and the order property.

Homework - The student Exercise is for them to do on their own as practice. They will work on their flex layout properties and % widths.

Day 11 - Relative Units & CSS Selectors

**What to cover:**

Advance CSS pseudo selectors such as:

* first child
* only child
* last child
* nth child
* nth last child
* first of type
* nth of type
* last of type

Introduce the flexbox order property.

**What to do:**

Demos - CSS Pseudo Selectors and Flex Grow

Day 12 - In-Class Lab

**What to do:**

In-Class Lab

A mentor build for the in-class lab which includes a flex box layout that is fully responsive. It’s a completely fluid layout, however, you may add in a fixed width container if time allows. Students should still be using semantic and structured HTML when creating flex containers. They shouldn’t be removing headings outside of the sections simply to make the sections as flex containers.

Students have to be present for the in-class assessments to receive a mark. The solution should not be posted on Moodle.

This is a formative assessment to evaluate where the class is in terms of understanding the material covered before going into the assignment.

Day 13 - Responsive Backgrounds

**What to cover:**

Swapping out different sized image files for different devices when using full width images.

Have the students research and discuss what are the current recommendations for image file sizes? Overall page sizes?

There is a lot of different opinions and discussions about this topic on the web, but overall you want to stay between 50 - 150KB per picture and under 250KB for a large full screen image.

Have the students resize, save and further compress images to see how small they can get the file to without loosing quality.

Use Google Lighthouse in the browser dev tools to check your overall image size and score.

**What to do:**

Demo - Responsive Background Images

Day 14 - Assignment 1

**What to cover:**

Assignment 1

Overview, expectations, and marking guide for assignment 1.

**What to do:**

Have students create initial repos for assignment 1 and share them with the instructor (or use GitHub classroom).