1360 Web Design

Lab 1: HTML Practice

Submit to Canvas assignment

LAB 1: HTML Practice

- HTML
- File structure
- HTML Validator

HTML is all about **STRUCTURING CONTENT**.

Your page should be **readable** when you're finished with markup, but it <u>won't be pretty</u>.

Do NOT try to adjust the look and feel today beyond what we say in the directions... for example by using a lot of

(PLEASE, NO!!!!)

NEXT WEEK -- we will start with the HTML you created today and add CSS styling.

<u>THEN</u> it will look better.

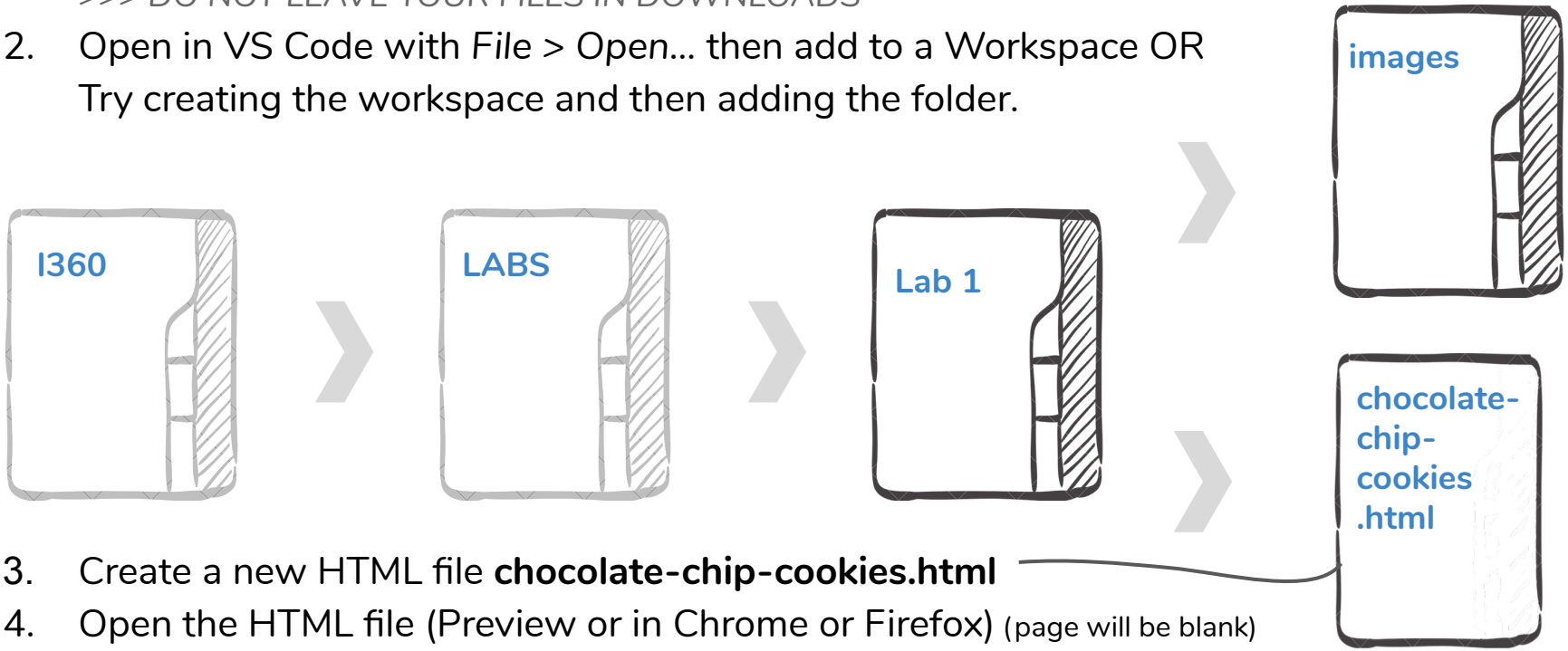
Be patient.

File structure

1. Download the **Lab 1** files for today and place in your I360 folder setup

   >>> DO NOT LEAVE YOUR FILES IN DOWNLOADS

2. Open in VS Code with *File* > *Open…* then add to a Workspace OR
   Try creating the workspace and then adding the folder.



3. Create a new HTML file **chocolate-chip-cookies.html**
4. Open the HTML file (Preview or in Chrome or Firefox) (page will be blank)

Look back at the slides for Lessons 1 & 2

Remind yourself of the steps needed
to (1) **finish preparations**
and (2) **markup the HTML**

*NOTE: For Project 1, much of the HTML you will need can be found in our **P1 Demo** or in **02 Essential HTML** on Canvas*

LAB 1 Details

- Place one image at the top of the page using **IMG**
- Place the other image between “Recipe courtesy of…” and “Total: 1 hr 50 min” using the **FIGURE** / **IMG** / **FIGCAPTION**

- The title should be your **H1**. It is also a link (**A**) to the recipe.
- The section starting with “Total: 1 hr 50 min” is a small **TABLE**
- Use both an **unordered** and an **ordered list**
- Use **SMALL** for the copyright line at the bottom
- Use an HTML entity for the **copyright** symbol at the bottom of the content

- Copy/paste ***lab-1-styles.txt*** into the HEAD of your HTML document (under TITLE)
  - *INCLUDED IN YOUR RESOURCES FOR TODAY - NO NEED TO TYPE IN*
  - You’ll need to add ** ** that the styles attach to in the HTML — *follow the directions in the next few slides*

1) Add a container around your content

Problem: Our content stretches too far -- it's too wide!

We need an HTML element to serve as a **container** for our content — a way to *control the width* of the content displayed the browser.

```
<body>
    <div class="container">
        Content goes here
    </div>
</body>
```

The ** ** tag is the most generic container within HTML. It stands for "division" and basically means **"make a box here".**

NOTE: This code has been provided for you in the Lab 1 resources for today.

2) Add styles to container & images

```
<style>
    .container {
        width: 720px;
        margin: 35px auto; /* centers the container */
        border: 1px solid black;
        padding: 50px;
        background-color: #FFFFFF;
    }
    img {
        /* forces images to pay attention to parent */
        width: 100%;
        display: block;
    }
</style>
```

we'll go through CSS in detail on Day 3 & 4, but we want to go ahead and add this bit today in order to **control our page.**

BTW, CSS in the HEAD (in a STYLE tag anywhere AFTER META... that location is important) is called an internal style sheet.

NOTE: This code has been provided for you in the Lab 1 resources for today.

2A) Add basic styling to your container

```
<head>
    ...
    <style>
    .container {
        width: 720px;
        margin: 35px auto;
        border: 1px solid black;
        padding: 50px;
        background-color: #FFFFFF;
    }
    </style>
</head>
```

The STYLE tag is used to include CSS inside an HTML page (within the HEAD)

Arbitrary choice dependent on project

Centers a block-level element, but ONLY IF it has a **width** narrower than 100%.

Adds a border. Adds space between the border and the content.

Changes the background to white.

NOTE: This code has been provided for you in the Lab 1 resources for today.

2B) Get your images to behave

Problem: Now our images don't fit!

```
<head>
    …
    <style>
    …
    img {
        width: 100%;
        display: block;
    }
    </style>
</head>
```

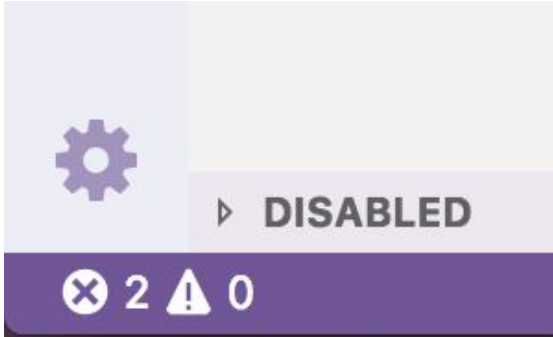
Tells the image to pay attention to the width of its parent (container).

The image will now behave like we expect it to and will act like a block-level element pushing other content before and after it instead of trying to appear inline.

3) Final step: Validate your code

Make sure you have no errors in VS Code

- No added extension needed
- Errors show up as red squiggles in the code
- Details can be found by clicking on the error notifications in the lower left corner of the VS Code interface.
- The VS Code error checker WILL NOT catch unclosed HTML tags and some other basic issues. It will be up to you to make sure all that is in place.



Indicates an error

Error messages and errors will sometimes not be that helpful. Know that the issue may be with where the error is indicated or it might be ABOVE that line and it just took the checker a bit to catch that it wasn't an expected value that followed.

Click the notifications for more details. Try fixing issues one at a time from top to bottom of your code.

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width,
        initial-scale=1.0">
6       <meta http-equiv="X-UA-Compatible"
        content="ie=edge">
7       <title>Example</title>
8   <body>
9   </head>
10      <h1><p>I360 Web Design</p></h1>
11      <p>In this course, we will be learning the
        foundations of web development.
12      <ol>
13          <li>HTML</li>
14          <li>CSS</li>
15      </ol>
16  </body>
17  </html>
```

3) Final step: Validate your code

At several points in our development stage, we will want to more thoroughly check the validation. Run code through the https://validator.w3.org/ validator as many times as needed:

- Paste code into “direct input” tab
- Click “check” to see if any errors are caught
- **This is the first of many ways to debug a web page**
- **Use AS NEEDED to debug your code, and to check if it’s valid**

HINT: We’ll run your project code through a validator… you probably should too!

Using the W3C validator:

- If the validator shows a **red bar** and says there are errors, go back to your editor and go line by line in your code. Try to fix the issues one at a time, then revalidate.
  TIP: The error will likely be ON or ABOVE the line indicated.

- Keep fixing / testing until the validator shows a **green bar** saying no errors.

HOW TO TEST

Select all of your code with COMMAND-A (CTRL-A)

Copy the code with COMMAND-C (CTRL-C)

Paste the code into the box under "Direct Input" tab in the validator. COMMAND-V (CTRL-V)

Click the **CHECK** button.

Did you...

Hang on to this code. We will style the page you marked up today in our next lab.

- **Replace special characters** found in the content with the corresponding HTML entities to improve your typography? (i.e. double quote marks)
- Add all the required HTML code needed for a **valid, blank web page**?
- Markup ALL of the text content with HTML tags — nothing should remain as just plain text
- Add a **div** with class **container** then apply the structural **CSS** we gave you?
- **Validate** your HTML and attempt to fix any errors?

Did you learn today's skills?

If you are not sure on any of these, please attend a help session:

Can you...

- Complete a basic **FIND AND REPLACE** in VS Code
- Use **keyboard shortcuts** to indent / unindent, comment / uncomment a block of code
- Explain how your **images** show up - what does '**images/**' do for us?
- Complete the lab using **web standard HTML**
- Understand and be able to set up a basic **file structure** for your project
- **Validate** your code using **keyboard shortcuts**, e.g. using **SELECT ALL**, **CUT**, **COPY**, **PASTE**... and the amazing **UNDO**

Save this project for use in lab next week.