**Lab objectives**

* Understand how web pages are requested from a server
* Understand how web page requests are handled by server and the role of files and document root
* Understand how web pages are constructed in a browser from HTML, CSS, and the DOM
* Understand HTTP and web server - browser communication

**Instructions**

Try to do as many of the exercises as possible, in the order listed. You may skip an exercise if it is impossible to accomplish the task described (for technical reasons). If you have trouble with an exercise, contact the instructor for help – do not just skip the exercise! You may work with a partner, but each of you must submit your individual work. Work should be done electronically on this document (cut and paste as needed but be careful of incorrect characters). Anything that involves code should be tried out in VS Code before submitting (you may cut and paste directly from VS Code). Submit work before the next class. If you cannot submit by this time, try to do so as soon as possible.

You may wish to open the [presentation slides](https://dport96.github.io/ITM352/morea/030.ui-basics/Web%20UI%20Basics.pptx) for convenence of viewing and copy-paste. When asked by the instructor do the lab exercises indicated. If you get done early, feel free to go on. If you are not done before the instructor indicates the next exercise, raise your hand and explain where you are at. Later parts often depend on completing previous parts so do not let yourself get behind and expect to catch up!

Note: Whenever you see something like and it's not obviously an HTML tag, it means replace this with you own information as indicated inside. e.g. replace with Joe (or whatever your name is).

For exercises that do not ask specific questions but have you perform tasks, copy any code you created and the output as your answer to the exercise. Some exercises ask for explanations. Make notes for your answer and complete these after class before submitting. Do not try to make complete answers if it keeps you from progressing!

Do all your code work in VS Code. Copy and paste from there. Please ask the instructor for help if you get stuck this is **NOT** a test.

**Submission**

When we are done, submit work before the next class. If you cannot submit by this time, try to do so as soon as possible after class. Copy this lab with your answers and paste your into the Laulima assignment submission box to [Laulima Assignment Lab 2: Web page basics, web servers](https://laulima.hawaii.edu). If you do not submit something to Laulima you will not get credit.

**Exercise #1: HTTP and client-server communication**

Task 1: Open your workspace in VS Code and create a folder Lab2. In this folder, create an empty file name it hello-world.text then copy the following code into it:

<h1>Hello World!</h1>

Task 2: Open a terminal in the Lab2 director and start an http-server (installed in Lab 1) http-server . Remember that the . means the “web server document root will be in the current directory”.

Task 3: Open a browser and make a request to localhost port 8080 with the HTTP protocol and no path. What do you see and where did this come from? Now copy and paste the console output (IN VS Code) here. Explain what this is and where it came from.

Task 4: Now click on hello-world.text or type http://localhost:8080/hello-world.text. Explain what this is and where this came from. Is this HTML? Explain why or why not. Again, look at the console output and explain. **The server served up the file but its showing the text including the <h1></h1> tags on the page because the document is a .text file and so the browser and server cannot render the document in html and can only present the plain text document as is.**

Task 5: Exit your hello-HTTP-server process (by hitting ctrl-c usually). Refresh your webpage or retype your request. Explain why the page does not load. Now change the file extension from .text to .html.

Go to the terminal and re-run http-server . and refresh the web-page. Why do you **not** see the HTML code and now the page renders as you expected? **Browser made request to the server and the server takes the contents of the page and sent it back to the browser and now the server is rendering it in html and telling the browser to do the same thing which results in the bigger bolder text instead of the plain text like the .text page showed.**

Task 6: Exit your hello-HTTP-server process again. Make sure it has stopped and you are at the command prompt!

**Exercise #2: HTTP-Server as a web page file server**

Task 1: In your terminal cd to the parent directory of Lab2. Start http-server let . be the root directory.

Task 2: Create a directory SmartPhoneProducts1 and an empty file in it store.html. In the file use the [Emmet “doc” abbreviation](https://code.visualstudio.com/docs/editor/emmet) to add an HTML document boilerplate. Now change the file name to store.text

Taks 3. Add a title Smartphone Store Home and a top-level heading in the body <your name> Used Smart Phone Store. Go to the URL localhost:8080/SmartPhoneProducts1/store.text to view the page.

What did you see in the console output? **I see the plain text page in the browser with all the html tags.**

What do you see in the browsers address bar? Explain this! **http://localhost:8080/SmartPhoneProducts1/store.text**

Why is the web page not rendering? (i.e. the HTML is shown, not the web page it is supposed to create). **It is not rendering because the server is telling the browser it is a text document type and so that is how the browser is rendering the page.**

Change the filename extension from .text to .html, page-back, refresh the page , click the renamed link and check that the web page looks as expected. Explain this. **It looks as expected because now the server is telling the browser to render this page using the hml tags to display it as expected.**

Task 4. Create a new file in SmartPhoneProducts1 “products\_display.html”, add an HTML document boilerplate (EMMET doc), title it “Smartphone Products”, add a top-level heading “ Used Smart Phone Store".

Task 5. In your browser, page-back and click on the products\_display.html link. Note the GET request in the console output and the address in the browser address box. Check that the page is as expected.

**Exercise #3: A simple home page for a web application**

Task 1. Rename store.html “index.html”, page-back, refresh the page. Explain what happened and why this is expected and useful. **It automatically went to the index.html because it is in the same directory as the page we were on.**

Task 2: Create a folder “Images” in your SmartPhoneProducts1 folder and right-click on the following image and “save image as” into this new folder:

A hand holding a cellphone

Description automatically generated

Task 3: Display this image with height="50%" width="50%" after the top-level heading.

Task 4: Add a hyperlink to the products\_display.html page for this image. Add another top-level heading after this link “Click the image to enter store!”

Task 5: Add CSS to the HEAD element that styles the BODY element with text-align: center and background: red

**Note:** You may find it more convenient to use an auto-refresh server such as live-server, fast-live-reload, or reload. Install this now if you wish.

**Exercise #4: A simple products display page**

Going back to your products\_display.html file. Build page sections as follows:

* Get the JSON data at [products\_info.json](https://dport96.github.io/ITM352/morea/030.ui-basics/products_info.json) and covert each list item into an HTML section by replacing all the { with <section class="item"> and replace all the }, with </section>
* Replace the [ JSON list delimiters with <div><main> and ] with </main></div> (do this manually!)
* In each section, make the name values second-level headers <h2>. Use find RegEx with "name": "(.\*)", and replace with RegEx with <h2>$1</h2> or do this manually!
* In each section, put the price values in paragraphs (i.e. enclose them inside <p></p>). Put a &dollar; in front of the value inside the paragraph tags. Modify the find and replace RegEx you did previously.
* For each image value, covert it to an <img> with the src attribute the value.
* Add a header element above the div with a top-level header “Used Smartphone Store”
* Similarly add a footer below the div with a top-level header “Your One Stop Phone Shop”
* Create a style sheet “products-style.css” and link it in the HEAD. Reload the page and open Chrome dev tool (Inspect the page), click on sources, double click products-style.css. Edit directly in the Chrome tool editor (if you are not using Chrome you can edit in VS Code and reload the file as needed):
  + Style the body element width: 800px; margin: 0 auto;
  + Size the img elements for the item class 200px by 200px
  + Align the text for h1,h2 elements to center
  + Style the main element:
  + display: flex;
  + flex-flow: row wrap;
  + justify-content: space-around;

add the image phone\_globe.jpg as a background with no-repeat center. Change the color for text in this element so the text shows up well with this background image.

* + Style the div children of body:
  + border: 5px solid black;
  + display: flex;
  + Organize your CSS by adding comments and moving the CSS into the following sections:
  + /\* General \*/
  + ...
  + /\* Typography \*/
  + ...
  + /\* block styling \*/
  + ...
  + /\* Form styling \*/
  + ...
  + /\* Styling the products display \*/
  + ...

The ... is where you put your CSS rules. Don’t add the dots!

Here’s an example of what the products display page should look like when you’re done (you can inspect page source):

**Extra stuff:**

In your index.html add the following to STYLE element in the HEAD

img {

border: 1px solid #ddd;

/\* Gray border \*/

border-radius: 4px;

/\* Rounded border \*/

padding: 5px;

/\* Some padding \*/

width: 150px;

/\* Set a small width \*/

}

/\* Add a hover effect (blue shadow) \*/

img:hover {

box-shadow: 0 0 5px 1px black;

}

**Extra credit!**

Create a more elegant and useful products display. Here’s an example (click to enlarge):

[](https://dport96.github.io/ITM352/morea/030.ui-basics/SmartPhoneProducts1-elegant-display.png)

There are many online tools and references that may be helpful in creating web pages by hand. Here are some to try out: