WEB DEVELOPMENT ROADMAPS

Download file and follow along!

Lecture 1, Week 1

TODAY

- ❏ Front-End Roadmaps
- ❏ Back-End Roadmaps
- ❏ HTTP/ WWW
- ❏ HTTP Requests & HTTP Responses

FRONT-END ROADMAPS

STEP 1: LEARN THE BASICS

To be a web developer, you must possess a basic understanding of HMTL, CSS and JavaScript.

HTML(Hyper Text Markup Language): HTML is the standard markup language for web pages.

- ❏ HMTL consists of series of elements and these elements tell the browser how to display the content.

- ❏ An HMTL element is defined by a start tag, some content in between, and an end tag:
- ❏ <tagname>Content goes here...</tagname>
- ❏ The HTML element is everything from the start tag to the end tag.

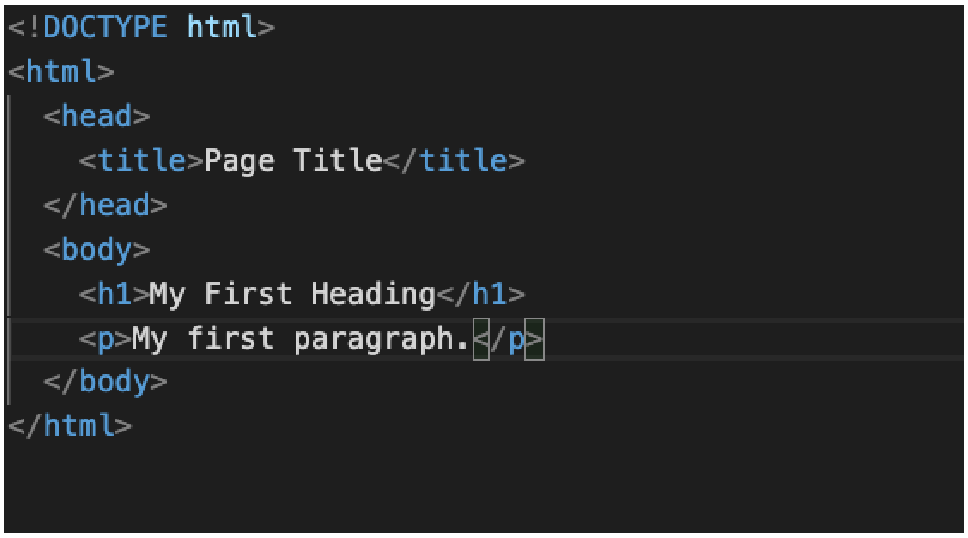
Examples:

<h1>My First Heading</h1>

<p>My first paragraph.</p>

- ❏ HTML describes the structure of a Web page.
- ❏ Web pages can be created and modified using simple text editors like Notepad (on windows PC) or TextEdit (on Mac) or professional HTML editors.
- ❏ You can learn more about HMTL editors and how to create your first web page on this resource HMTL Editors.

A Simple HTML Document Example



```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1>My First Heading</h1>
    <p>My first paragraph.</p>
  </body>
</html>
```

Example Explained

- All HMTL documents begin with the **<!DOCTYPE html>** declaration. This tells the browser that this document is an HTML5 document.

- After the <!DOCTYPE html> declaration, the HTML document itself begins with the opening tag of the root element and ends with the closing tag of the root element. In the example above, can you identify the root element of the HMTL document?

- The **<html>** element is the root element of an HTML page.

- The **<head>** element in an HTML document is the part that is **not** displayed in the web browser when the page is loaded. It contains information such as the page <title>, links to CSS, links to custom favicons, and other metadata (data about the HMTL, such as the author, and important keywords that describe the document).

- The **<title>** element is one of the elements that are enclosed in the <head> element. It specifies a title for the document.

- Another element that can be added to the <head> element is the <meta> element. The <meta> element specifies metadata i.e the data that describes the web page in a meaningful way than can be understood by web crawlers and browsers.

- The <meta> element also specifies the document’s character encoding:

  ```
  <meta charset="utf-8">

  Utf-8 is a universal character set that includes any
  character from any human language. Adding the line of code
  above to your HTML document means that your web page will
  be able to handle displaying any language.
  ```

- Just about all websites you’ll see will employ CSS to make them look visually appealing. You can add CSS to an HTML by inserting the <link> element in the <head> tag.

- The **<body>** element defines the document's body, and is a container for all the visible contents you see in web pages.

- The **<h1>** element defines a large heading. HTML headings are defined with the <h1> to <h6> tags. <h1> defines the largest and <h6> defined the smallest heading.

Here are some resources you can use to read more on HTML5 to have a solid understanding of how it works:

W3Schools
Mozilla Developer Network

CSS(Cascading Style Sheets): CSS describes how HTML elements are to be displayed.

It is the next technology you should start learning after HTML.

While HTML is used to define the structure of your content, CSS is used to style it and lay it out.

For instance, you can use CSS to:

- Change the color, size, spacing, font of your content.

- Split your content into multiple columns.

- Tell the browser to display your website with a different layout if the user is using a mobile phone, to add animations, and so many other decorative features.

Here are some resources you can use to read more on CSS3 to have a solid understanding of how it works:

W3Schools
Mozilla Developer Network

JAVASCRIPT: The Programming Language for the web. It is a programming language that allows you to implement complex things on web pages.

Every time a web page does more than just sit there and display static information for you to look at, animated 2D/3D graphics, or more, just know that JavaScript is probably involved.

JavaScript can:

- ❏ Change HTML Content.
- ❏ Change HTML Attribute Values.
- ❏ Change HTML Styles (CSS)
- ❏ Hide HTML Elements
- ❏ Show HTML Elements

Here are some resources you can use to read more on JavaScript to have a solid understanding of how it works:

W3Schools
Mozilla Developer Network

STEP 2: DIG DEEPER

When you feel comfortable with HTML and CSS, it is time to dig deeper.

- ❏ Learn how to use Maps, Fonts and Icons in HTML.
- ❏ Master how to access the HMTL DOM.
- ❏ Learn how to use AJAX and JSON for making server requests.

STEP 3: CHOOSE FRAMEWORKS

- ❏ For CSS, you should choose a framework for responsive web design: Here is a list of CSS frameworks you can choose from:
    - ❏ Bootstrap
    - ❏ Material Design
    - ❏ W3.css
- ❏ For JavaScript, you should learn at least one modern framework:
    - ❏ React.js
    - ❏ Angular.js
    - ❏ Vue.js
    - ❏ W3.js

BACK-END ROADMAPS

The Back-End involves writing servers which can handle things like communication with the database, application logic and provision of data requested by the front end.

A server can be written in different languages, you can find some of them below:

- ❏ PHP
- ❏ ASP
- ❏ Python
- ❏ Ruby on Rails

- ❏ Node.js

The server is not visible on the front-end, it is working on the backside. However, it provides responses for every data query from the user. A typical example is user authentication.

The user is presented with a login form, after entering the username and password, the verification is done on the back end to check if the login credentials the user provided are valid.

The Back-End Roadmap is Listed below:

STEP 1: Learn a Language

STEP 2: Version Control Sytems

STEP 3: Relational Databases

STEP 4: Non-relational Databases

STEP 5: Learn about APIs

STEP 6: Web Security Knowledge and Caching

STEP 7: Testing

STEP 8: CI/ CD (Continuous Integration/ Continuous Deployment)

STEP 9: Design & Development Principles

STEP 10: SEOs

STEP 11: Containerization

STEP 12: Web Servers

STEP 13: Building for Scale

STEP 14: Keep Learning

HTTP/ WWW

- HTTP stands for Hyper Text Transfer Protocol
- WWW is about communication between web clients and servers
- Communication between client computers and web servers is done by sending HTTP Requests and receiving HTTP Responses
- The World Wide Web is about communication between web clients and web servers.
- Clients are often browsers (Chrome, Edge, Safari), but they can be any type of program or device.
- Servers are most often computers in the cloud.

HTTP REQUEST/ RESPONSE

- Communication between clients and servers is done by requests and responses
- A client (a browser) sends an HTTP request to the web
- A web server receives the request
- The server runs an application to process the request
- The server returns an HTTP response (output) to the browser
- The client (the browser) receives the response