HTML5 & CSS3 WEB DESIGN FUNDAMENTALS

with JavaScript

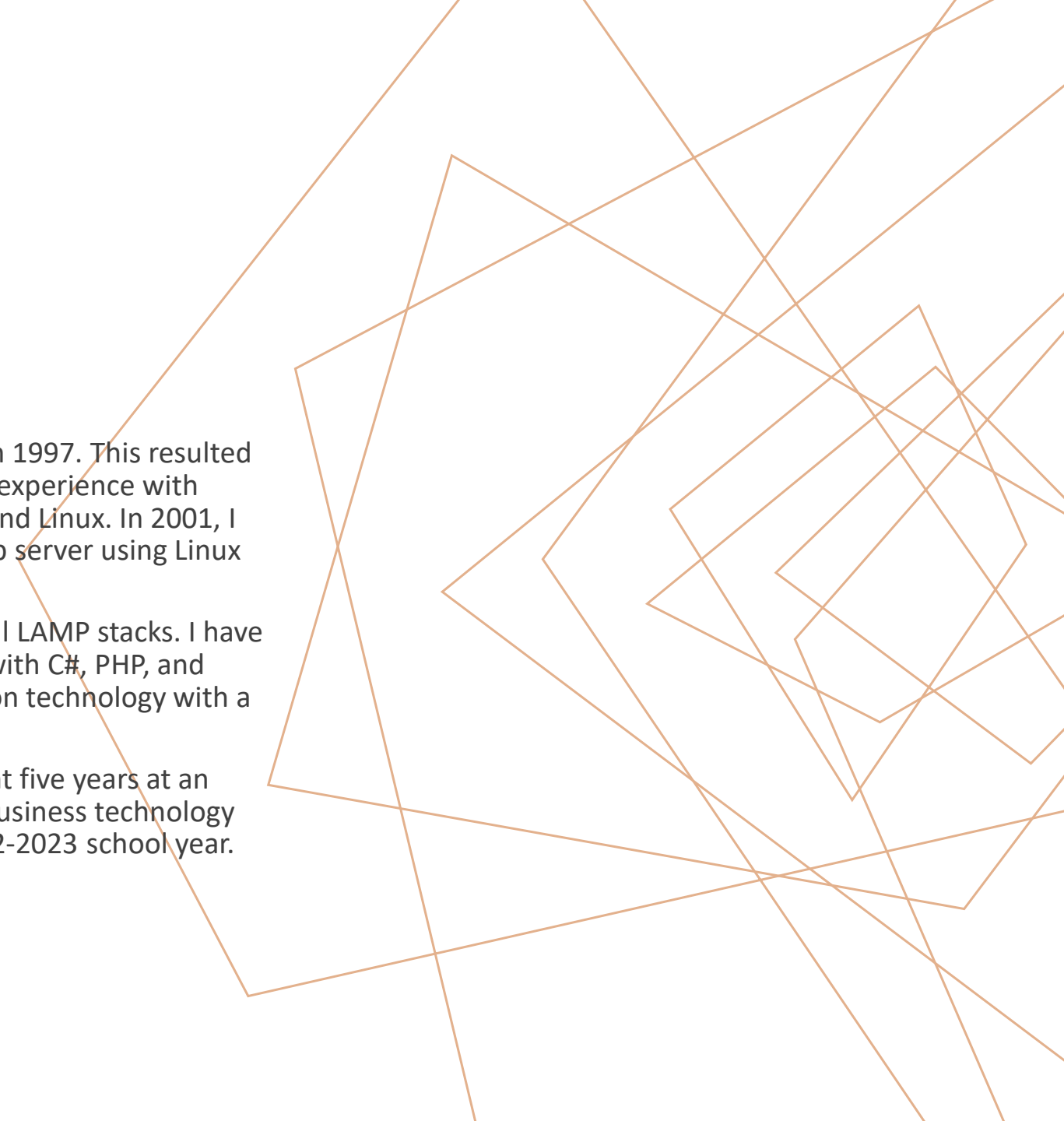
Neil Ball

ABOUT ME

My experience as a computer enthusiast began in 1997. This resulted in the acquisition of pc technician skills. By 1999, I had experience with multiple operating systems, including DOS, Windows, and Linux. In 2001, I successfully installed and operated my first Apache web server using Linux and published a web page online.

Throughout the years, I have administered several LAMP stacks. I have hosted OpenSimulator servers; spending a lot of time with C#, PHP, and MySQL. I have a Master of Science degree in information technology with a concentration in software development.

My career path has led me into education. I spent five years at an elementary school in central Florida. I begin teaching business technology education at a central Florida middle school in the 2022-2023 school year.



COURSE OUTCOMES

ESSENTIAL — Learn HTML5 & CSS3 web design fundamentals.

PROFESSIONAL — Explore web development best practices.

DESIGN — Build responsive & mobile-first webpages.

STANDARD — Implement current website technologies.

REFERENCE MANUAL

***All good* coders refer to documentation.**

There are many online sources.

https://www.w3schools.com/

https://stackoverflow.com/

https://www.w3.org/standards/webdesign/htmlcss

A reference manual (in physical book form) is also very useful. It is highly recommended to find one you like.
There are many to choose from. Ensure that it was published recently and check the reviews.

Module 2

The Internet & The World Wide Web

THE INTERNET & THE WORLD WIDE WEB

INTERNET

This is the worldwide connection of linked computers that facilitates communication of various media.

WORLD WIDE WEB

This is the services provided by the collection of websites serving webpages to clients.

NETWORK

This is two or more computers linked together to share information and resources

World Wide Web Consortium (W3C)

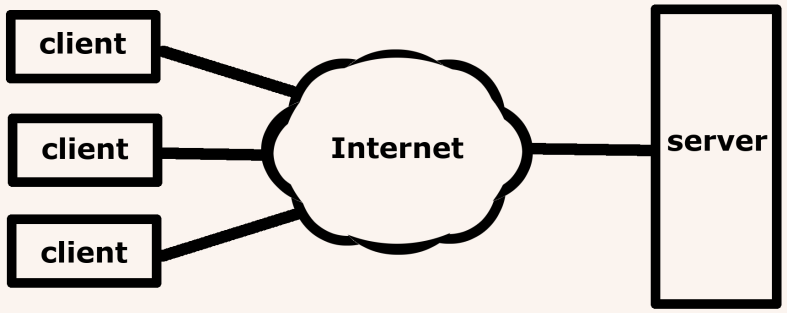
This is the organization responsible for creating web standards and guidelines.

Network Model Overview

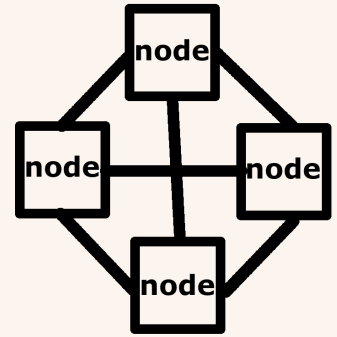
- Client-Server
  - Clients are computers that request and receive data from servers.
  - Servers are computers that accept requests from and deliver data to clients.
  - This is a common model related to web design.

- Peer-to-Peer (P2P)
  - Nodes are networked computers that engage in both client and server functions.
  - A network of nodes decentralizes the data. If one node fails, others continue service.

Client-Server Model

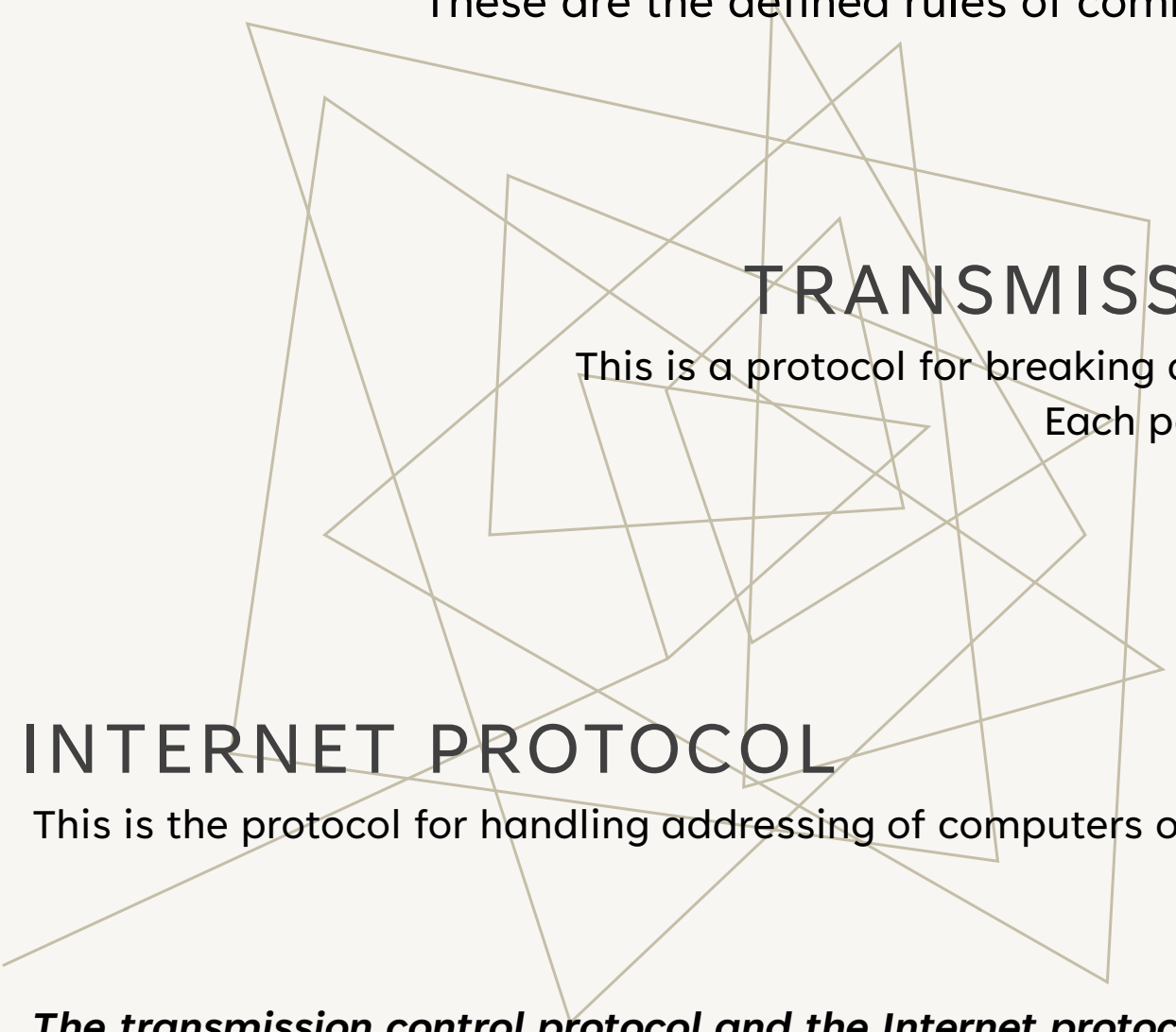


Peer-to-Peer Model



PROTOCOLS

These are the defined rules of communication between server and client.



TRANSMISSION CONTROL PROTOCOL (TCP)

This is a protocol for breaking a file into parts called packets for delivery and reassembling it.
Each packet contains necessary addressing and ordering information.

INTERNET PROTOCOL

This is the protocol for handling addressing of computers on the Internet.

The transmission control protocol and the Internet protocol work together. They are commonly referred to as TCP/IP.

PROTOCOLS

These are the defined rules of communication between server and client.

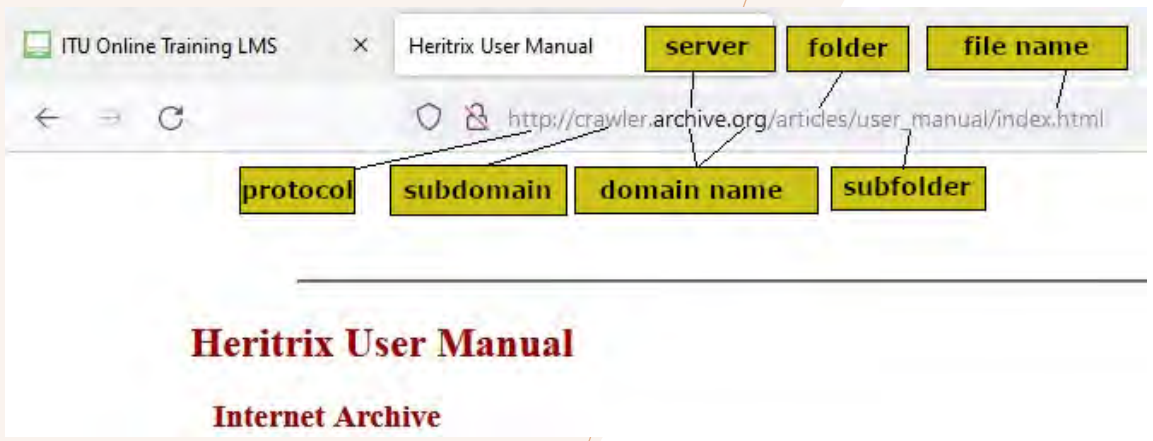
HYPERTEXT TRANSFER PROTOCOL (HTTP)

The essential protocol for webpage transfer on the World Wide Web.
HTTPS refers to HTTP Secure. HTTPS is used in securely encrypting websites.

FILE TRANSFER PROTOCOL (FTP)

This is a protocol that facilitates the transfer of files over the internet.

BROWSING THE WEB



WEB BROWSER

This is program that allows users to interact with webpages.

DOMAIN NAME SYSTEM (DNS)

This system consists of servers that associate web addresses with IP addresses.

UNIFORM RESOURCE LOCATOR (URL)

This is the human-readable address of webpages or files found on a web server; also known as a web address.

ADDRESS BAR

This is the location in the web browser where web addresses are entered or located.

MODERN WEB BROWSERS

Mozilla Firefox is the oldest of the most common modern web browsers.

Google Chrome has become very popular in recent years.

Microsoft Edge has replaced Internet Explorer.

GOOD WEB BROWSERS

LibreWolf is my daily driver. It is based on Firefox.

LibreWolf includes added privacy and security features.

Ungoogled **Chromium** is for when I have a required Chrome-compatible use case.

SeaMonkey is a full-featured web browser. It supports many protocols that most modern browsers do not.

It is essentially an up-to-date *Netscape Communicator*.

MARKUP LANGUAGE

This is a language used to define how content is displayed.

HYPERTEXT MARKUP LANGUAGE (HTML)

This is the standard authoring language for creating documents to be viewed in web browsers.

EXTENSIBLE HYPERTEXT MARKUP LANGUAGE (XHTML)

- This authoring language is a rewritten integration of HTML and XML.
- Mobile devices more widely accept it.

EXTENSIBLE MARKUP LANGUAGE (XML)

- This markup language is used for describing and exchanging data.
- It can work with HTML for transporting data to and from webpages.

DYNAMIC HYPERTEXT MARKUP LANGUAGE (DHTML)

- This is a term used to refer to web technology combinations, including JavaScript, CSS, and HTML.

ACCESSIBILITY REQUIREMENTS

Accessibility		Non-discrimination	
Public Sector	*Private Sector*	*Public Sector*	*Private Sector*
Section 255 of the Telecommunications Act of 1996		Americans with Disabilities Act of 1990 (ADA), as amended (2009)	
Section 508 of the US Rehabilitation Act of 1973, as amended (1998)	21st Century Communications and Video Accessibility Act of 2010 (CVAA)		Air Carrier Access Act of 1986 (2013)

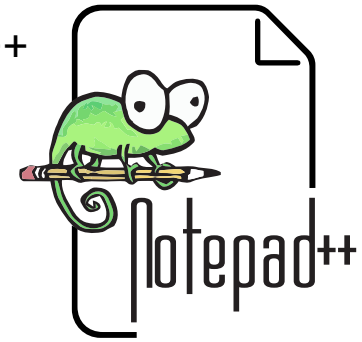
Organizations are required to make websites accessible to those with disabilities.
Failure to comply may result in a forced redesign of the website, fines, and lawsuits.

Module 3

HTML Fundamentals

WEB DEVELOPMENT TOOLS

Notepad++



Brackets

Windows Notepad

Visual Studio Code

WYSIWYG editors (What You See Is What You Get) [whizzy wig] are used to develop web pages without requiring specific knowledge of how to write a webpage by hand. **Text editors** are programs that allows users print, edit and save text to a file.

ENVIRONMENT

0. Install Notepad++ or Brackets.
   - It is always useful to have a separate application to open files.

1. Install Visual Studio Code.
   - It includes numerous features to help with development.

2. Install Live Server
   - In Visual Studio Code, find Extensions (Ctrl+Shift+X).
   - Type “Live Server” in the search bar.

3. Install Live Server Web Extension
   - It is available for Chrome and Firefox.
   - Locate it in Chrome extensions or Firefox add-ons.

HTML5 SYNTAX

TAGS

These are commands that define webpage appearance.

ELEMENTS

These include starting and ending tags and the content within them.

ATTRIBUTES

These are characteristics that can be added to tags to define element properties.

DOCUMENT TYPE DECLARATION (DOCTYPE)

<!DOCTYPE html>

This is the DOCTYPE declaration for HTML5.

This is the first line of an HTML document.
The DOCTYPE tells the web browser which markup language and version to use.

HTML5 SYNTAX

ELEMENTS (TAGS)

HTML **<html></html>**

This contains all other elements besides DOCTYPE and begins an HTML document.

HEAD **<head></head>**

This contains elements for the webpage title and other information.

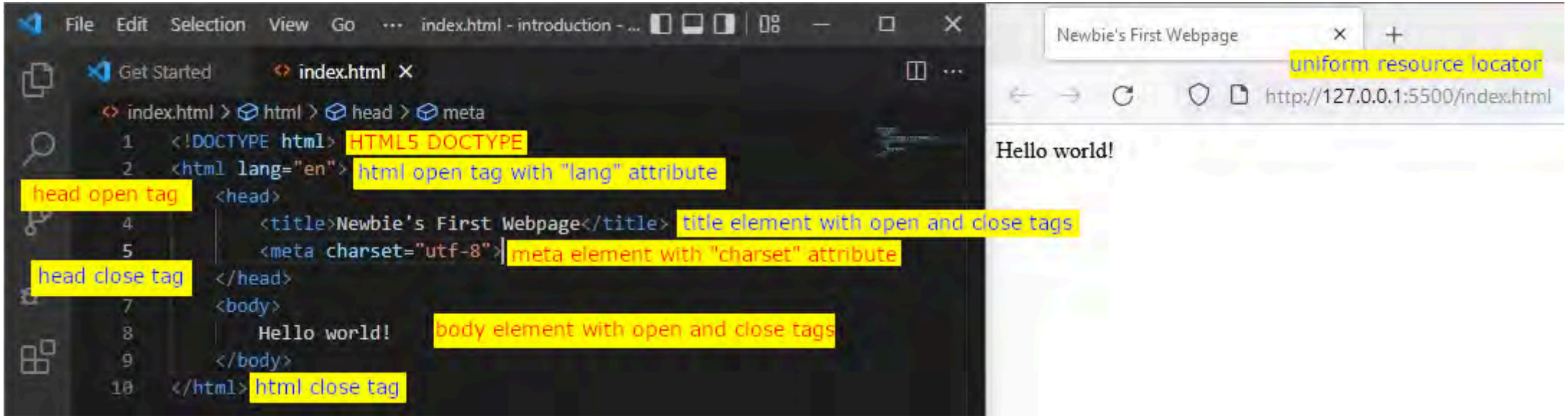
META **<meta charset="utf-8">**

This contains webpage data information and is found within the head element.

BODY **<body></body>**

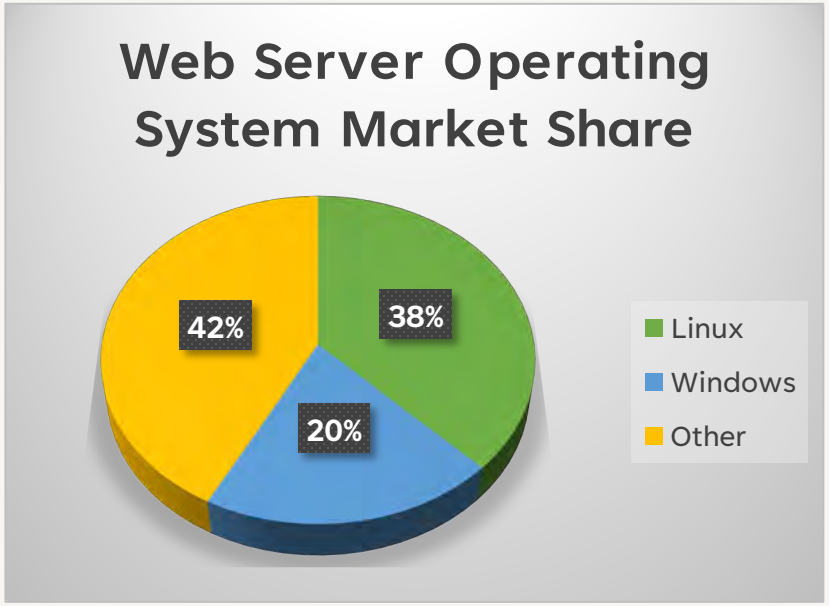
This contains essentially all elements for displaying webpage content.

HTML5 SYNTAX



NAMING SCHEME NOTES

The operating system you use for web page development likely will not be the operating system that hosts the website.



Linux, BSD, and Unix operating systems have *case-sensitive* naming rules. This means that they treat file names that would otherwise be identical as different if the letter casing differs. For example, “file.txt” is a different file from “File.txt”.

Windows operating systems are case-insensitive. However, Windows requires a file type, such as “.exe” at the end of each file.

In order to ensure cross-platform compatibility, it is good practice to keep file names lowercase (for Linux) and end each file with the correct file type, such as “.html” or “.css” (for Windows).

Underscores “_” or hyphens “-” may be used to separate words within file names. (Avoid spaces in file and folder names.) However, do not begin or end file names with them. Protip: pick **one** and stick with it.

HTML SKELETON

Skeletons provide the scaffolding for producing a website.

Like the scaffolding of a building, it should be built beforehand.

Unlike scaffolding, copies from it become a permanent part of the site.

These are developed *prior to* **and** *throughout* the development process.

DEFAULT WEB PAGE

The default web page exists so that the web browser has a page to display when no file is specified.

- Users normally visit a website by typing in a domain name or clicking a hyperlink.

- When no file is specified, web browsers locate **index.html** (the default web page) if it exists on the web server.

ANCHOR ELEMENTS

Hyperlinks and hyperlinked images are displayed using anchor elements.

Hyperlinks (links) are webpage contents that can be clicked to obtain content from other locations. They typically look like blue, underlined text, but can also be a part of displayed images or other forms of media.

Absolute links point to resources with a URL.

Relative links point to resources *relative* to the location of the webpage which hosts them.

- If the link is on a page that is in a subfolder, it may be necessary navigate the file system using two periods and a forward slash to move to the parent folder. ../

The **href** attribute must always be defined in anchor tags.

```
<a href="anchor.html">Anchor Page</a>
```

HEADING ELEMENTS

Heading elements are used before paragraphs to provide titles to them.

There are six heading tags to choose from.

The h1 tag provides the largest font. The h6 tag provides the smallest.

Line breaks after heading closing tags are automatic.

```
<h1>Title</h1>
```

PARAGRAPH ELEMENTS

Paragraph elements group sentences or paragraphs together. They add space above and below each element.

** **Socrates is human. Humans are mortal. Socrates is mortal.** **

LINE BREAK ELEMENTS

These separate sentences and elements onto new lines, including space between them.

They do not require a closing tag.

A is B.
 B is C.
 A is C.

BLOCKQUOTE ELEMENTS

Blockquotes isolate the text with spaces and indents to distinguish a quote from other text.

<blockquote>I think. Therefore, I am.**</blockquote>**

PHRASE ELEMENTS

These add structural information to fragments of text.

****emphasis****

****strong emphasis****

<code>code fragment**</code>**

<var>variable**</var>**

LIST ELEMENTS

There are unordered, ordered, and description lists.

ORDERED LIST AND UNORDERED LIST ELEMENTS

Unordered lists default to a list of bullets.
Ordered lists default to a list identified by numbers.
List items are added with the <li> tag.

```
<ol>
    <li>First</li>
    <li>Second</li>
</ol>
```

```
<ul>
    <li>Something</li>
    <li>Else</li>
</ul>
```

DESCRIPTION LIST ELEMENTS

Description list elements display information as a list of terms and their indented descriptions (<dt> & <dd>).

```
<dl>
    <dt>Term</dt>
    <dd>Description</dd>
</dl>
```

HTML COMMENTS

Comments are notes to developers that are not displayed when the site is viewed in a web browser.

<!--Insert comments here.-->

RESERVED CHARACTERS

HTML requires the use of certain characters as a part of its language. These characters are known as reserved characters. You instead type in *entities* to obtain the desired character.

"	'	&	<	>
&quot;	&apos;	&amp;	&lt;	&gt;
&#34;	&#39;	&#38;	&#60;	&#62;

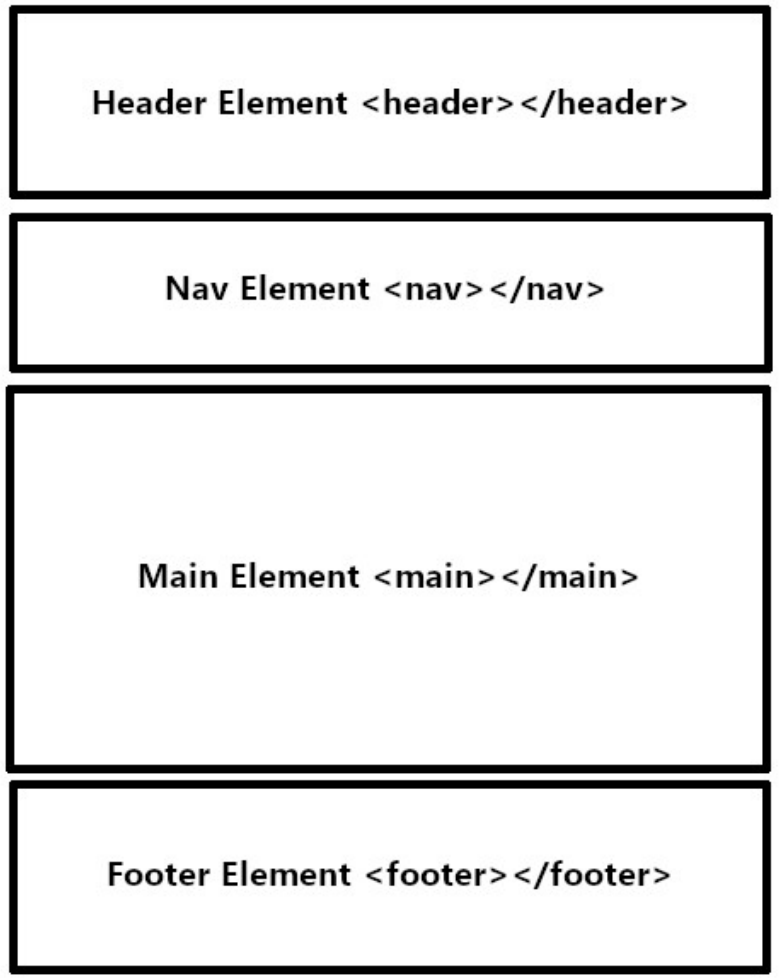
ENTITIES

HTML Entities are codes typed into an HTML document to display certain characters. These entities are used
For reserved characters and other special characters.

(non-breaking space)	****	**/**	**©**	**®**
	&bsol;	&sol;	&copy;	&reg;
	&#92;	&#47;	&#169;	&#174;

SEMANTIC ELEMENTS

A **wireframe** is a sketch of a webpage layout that provides the primary element locations. Wireframes serve as visual guides for customers, other stakeholders, and developers alike.



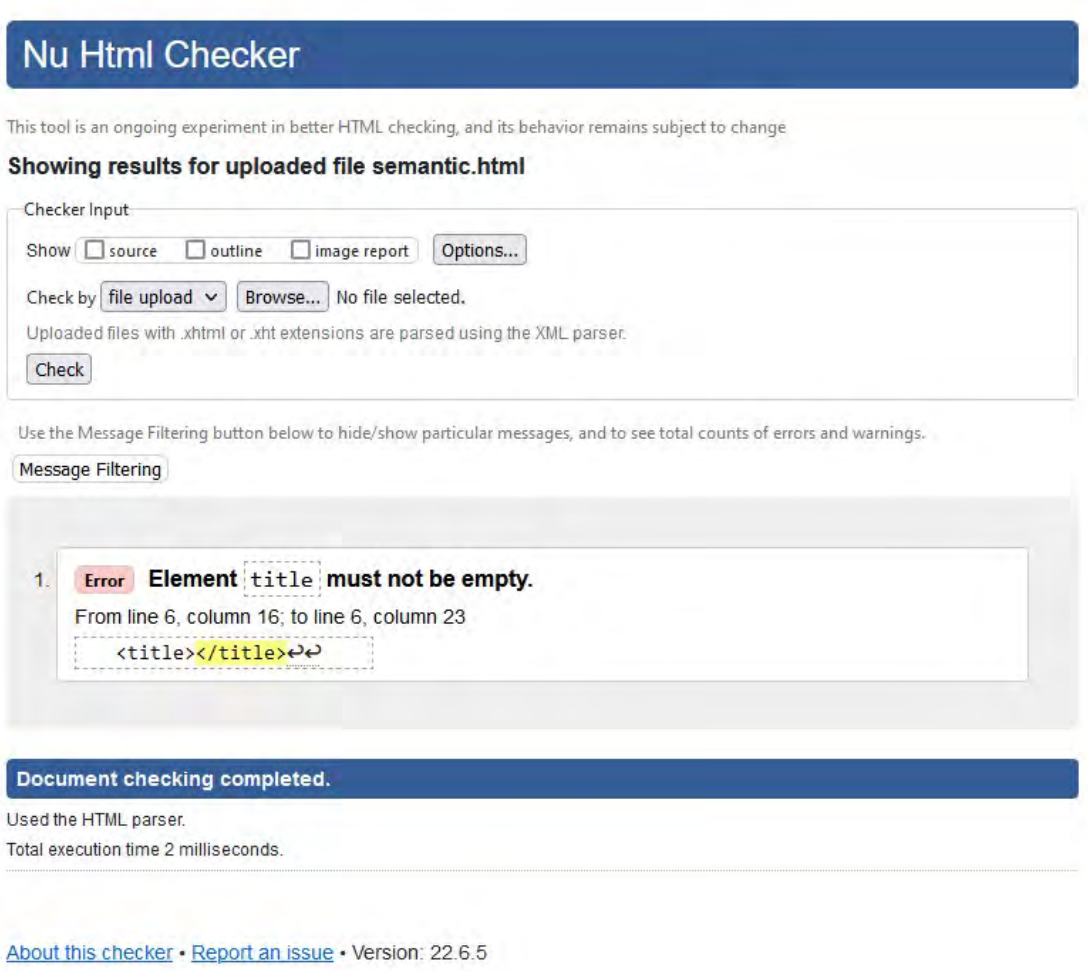
SEMANTIC ELEMENTS

<header> Header </header>	Header elements contain logos or names and the like. This is typically what you would expect at the top of a page.
<nav> Navigation </nav>	Navigation elements provide links to various webpages on the site.
<main> Main </main>	The intended webpage content is contained within the main element.
<section> Section </section>	The section element specifically groups webpage content.
<article> Article </article>	Article elements contain posted content.
<aside> Aside </aside>	Aside elements group sidebar information.
<footer> Footer </footer>	Footer elements contain the information appropriate at the bottom of a page, such as contact or copyright information.

HTML VALIDATION

https://validator.w3.org/

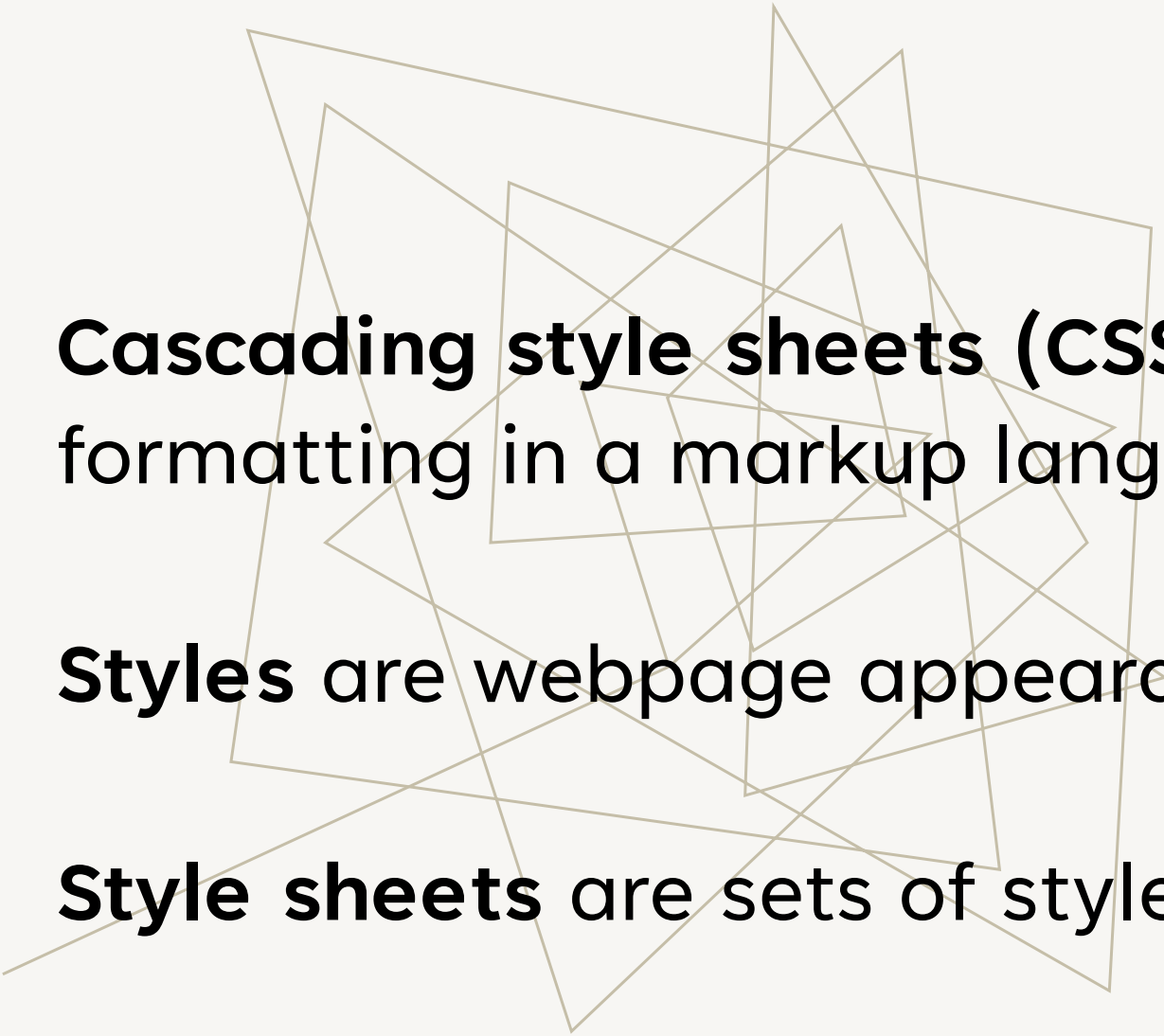
Validating your webpage facilitates error correction and correct syntax.



Module 4

CSS Fundamentals

CASCADING STYLE SHEETS (CSS)



Cascading style sheets (CSS) is a language for document formatting in a markup language.

Styles are webpage appearance rules.

Style sheets are sets of styles.

STYLE LOCATIONS

- **Inline styles** are inserted into the tag of an element.
  - <h1 style="font-color: red">Title</h1>

- **Embedded (internal) style sheets** are located within the head (<head>) element.
  - Like inline styles, these work within the given webpage.

- **External (linked) style sheets** are .css files that are linked to webpages.
  - This has the benefit of using styles across entire websites.

CSS ESSENTIALS

Selectors identify the elements to style.

- Headers, id selectors, class selectors, and descendant selectors can be used.
- Declarations are included within selectors.

Declarations are style formatting definitions.

- These consist of *properties* and *values*.

CSS DECLARATIONS

Declarations are found within selectors and consist of properties and values.

Properties are the applied characteristic identifications, such as *font-style*.

Values are what is entered to apply the desired characteristic, such as *italic*.

```
body {

        font-style: italic;

}
```

INLINE STYLES

The **style attribute** is inserted into the opening tag of an element in order to add a style. They *override* all other style sheets.

Semicolons are used to separate declarations.

<h1 **style=“background-color: yellow”**>Here is a heading.</h1>

<h1 **style=“background-color: yellow; color: blue”**>
Here is a another.
</h1>

EMBEDDED (INTERNAL) STYLE SHEETS

Embedded style sheets use the same basic format as external style sheets. They use the **style element** from within the head element. They *override* external style sheets if they are used.

```
<head>
    <style>
        body {
            background-color: black;
        }
    </style>
</head>
```

EXTERNAL STYLE SHEETS

Since a separate .css file is used, the webpage must use a link element within the head element to reference the style sheet.
Link elements require the **rel** and **href** attributes for this to work.

There is no specific CSS naming convention. Use what helps with site development (while following previously mentioned suggestions).

```
<head>
    <link rel="stylesheet" href="css/default.css">
</head>
```

EXTERNAL STYLE SHEETS

Selector types

- **Element selectors** apply styles to selected element using the element name.
- **Id selectors** uses an element’s id attribute value to identify where to apply styles.
- **Class selectors** uses an element’s class attribute value to identify where to apply styles.
- **Descendant selectors** can select elements within elements for style application.

ELEMENT SELECTORS

Index.html	**css/default.css**

<head>
  <link rel="stylesheet" href="css/default.css">
</head>
 	
<**body**>
  Hello world!
</body>
	body {
	  color: red;
	}

ID SELECTORS

Index.html	**css/default.css**

<head>
  <link rel="stylesheet" href="css/default.css">
</head>
 	
<body>
  <div **id="essential"**>Hello world!
</body>
	#essential {
	  color: green;
	}

CLASS SELECTORS

Index.html	**css/default.css**

<head>
  <link rel="stylesheet" href="css/default.css">
</head>
 	
<body>
  <div **class="vital"**>Hello world!
</body>
	.vital {
	  color: green;
	}

DESCENDANT SELECTORS

Index.html	**css/default.css**

<head>
  <link rel="stylesheet" href="css/default.css">
</head>
 	
<**body**>
  <**div**>Hello world!
</body>
	body div {
	  color: green;
	}

ELEMENTS (TAGS) *CONTINUED*

DIV

** **

This divides the webpage into identifiable subsections.

The id attribute is essentially required.

SPAN

****

This allows for CSS formatting distinct from surrounding text.

Adding a class attribute allows for styling.

CSS COLORS

Color values are expressed in three different ways, by color name, in hexadecimal, and RGB.

Hexadecimal values begin with the pound sign (#) and consist of six hexadecimal digits, such as #ABCDEF.

RGB values are entered using three integers ranging from 0 to 255, such as rgb(171,205,239). Each number in this format is contained within a **channel**.

CSS COLORS

RGB

rgb(171,205,239)

Hexadecimal

#ABCDEF

AB	CD	EF
10*16 +11	12*16 +13	14*16 +15
171	205	239

0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

CSS COLORS

Color	Hexadecimal	RGB
black	#000000	rgb(0,0,0)
white	#FFFFFF	rgb(255,255,255)
red	#FF0000	rgb(255,0,0)
green	#008000	rgb(0,128,0)
blue	#0000FF	rgb(0,0,255)

ID VERSUS CLASS

Id is used to identify **exactly one** element.

Class is used to identify **one or more** elements.

CSS TEXT PROPERTIES

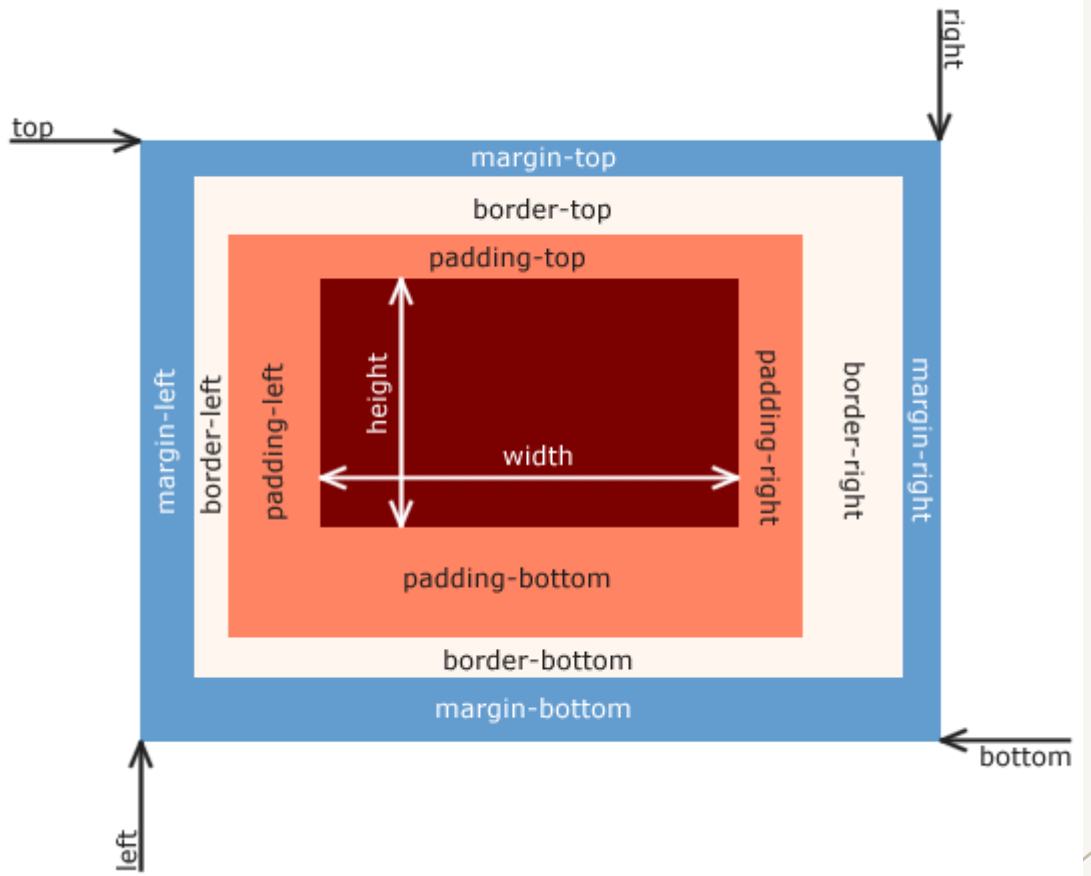
Property	Description	Example
font-family	This provides a list of suggested fonts for browser use.	font-family: Aerial, “Comic Sans”, Calibri;
font-size	Use font size measurement units.	font-size: 14pt; font-size: 1.5em;
font-weight		font-weight: bold;
font-style		font-style: italic;
text-align		text-align: center;
color	This is the color of the text.	color: blue;

Font Size Measurement Units		
em	Relative to element default font size	font-size: 1.5em;
%	Relative to element default font size	font-size: 75%;
px	pixel count	font-size: 20px;
pt	points (think word processing)	font-size: 14pt;
keyword	Sizes chosen by limited worded list	font-size: large;

CSS BOX MODEL

Block elements refer to the resulting logical “blocks” that result from elements beginning and ending with new lines.

- **Margins** refer to the spacing between block elements.
- **Padding** is the space between a block element’s border and its content.
- **Borders** separate the margins from the padding of a block element.



COMMENTS & DEBUGGING

CSS comments use the following syntax:

/*Place your comment here.*/

Comments are not only good for telling others how your code works. They are a great tool for debugging.

https://jigsaw.w3.org/css-validator/

Module 5

Images and Links

IMAGES

Pixels are the smallest portions of color or light on an image display.

Optimization involves reducing image file size using various means.

- Image cropping
- Modifying dimensions
- Quality adjustment
- Changing image file format

IMAGES

GIMP is an image manipulation program.

Formats

- Joint Photographic Experts Group (.jpg)
- Portable Network Graphics (.png)
- Graphics Interchange Format (.gif)
- Scalable Vector Graphics (.svg)

IMAGES

Raster (pixel) graphics map image data into a grid of rectangles.

Vector graphics create images using geometric shapes.

Raster graphics	Vector graphics
.jpg	.svg
.png	
.gif	

IMAGES

Joint Photographic Experts Group (.jpg)

- ***Lossy compression*** (best for still images)
- Over 16 million colors
- No transparency

IMAGES

Portable Network Graphics (.png)

- Lossless compression (less than ideal)
- Over 16 million colors
- ***Transparency***

IMAGES

Graphics Interchange Format (.gif)

- Lossless compression
- Still images or ***animated frames***
- 256 colors
- Transparency

IMAGES

Inkscape is a vector graphics editor.

Scalable Vector Graphics (.svg)

- ***Maintain quality when resized***
- Can be created purely with code

```
<svg height="400" width="400">
        <circle cx="200" cy="200" r="160" stroke="#000000" stroke-width="2">
</svg>
```

IMAGE ELEMENTS

The *<img>* tag and its *src* attribute are used to display an image on the screen.

<img src="images/test.jpg">

The *alt* attribute is used to describe the image when it cannot be seen.

- Text-based web browsers benefit from this.
- More importantly, this is used for ***accessibility*** purposes.

<img src="images/test.jpg" alt="A pencil and paper test">

The *height* and *width* attributes provide the area in which the image is displayed.

- The input values are in **pixels**.
- If these attributes are not defined, the native resolution will be displayed.
- *Changing these attributes does not change the size of the displayed file.*

<img src="images/test.jpg" height="1100" width="850">

IMAGES WITH HYPERLINKS

Inserting an image element within an anchor element will result in a clickable image that links to another webpage or website.

```
<a href="https://www.w3.org/">
    <img src="images/pic.jpg">
</a>
```

OPEN LINKS IN NEW TAB

The **target attribute** is used to open hyperlinks in new windows or tabs.

Users then do not have to press the “back” button in order to navigate to the previous page.

```
<a href=“https://www.w3.org”
target=“_blank”>w3.org</a>
```

TELEPHONE & EMAIL LINKS

Telephone links place “tel:” before the digits of the telephone number within the href attribute of the anchor element.

```
<a href="tel:5555551212">(555) 555-1212</a>
```

Email links place “mailto:” before the email address within the href attribute of the anchor element.

```
<a href="mailto:owner@anybusiness.abc">owner@anybusiness.abc</a>
```

Validate & Debug!

HTML: https://validator.w3.org/

CSS: https://jigsaw.w3.org/css-validator/

Module 6

Responsive Design

RESPONSIVE DESIGN

Responsive design is the strategy of accounting for as many device types as possible throughout the development process.

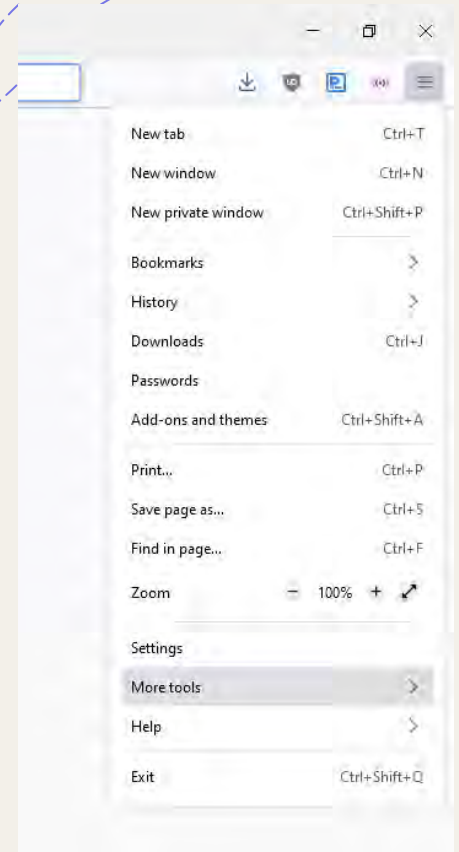
The **viewport** is the webpage viewing area.

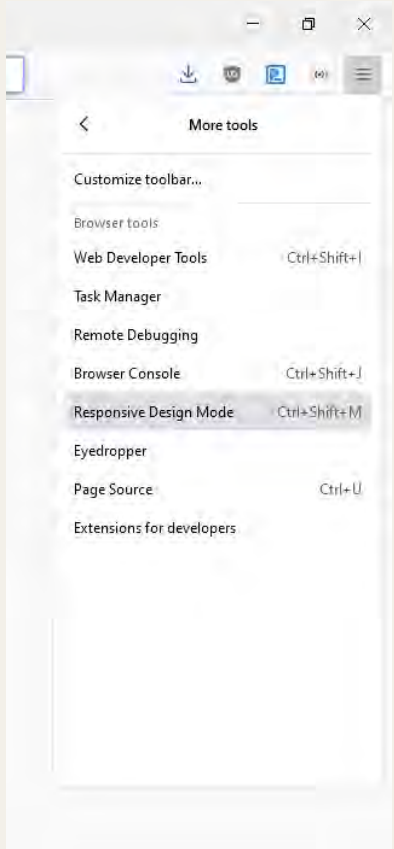
The **mobile-first strategy** involves site design that prioritizes the smallest viewports.

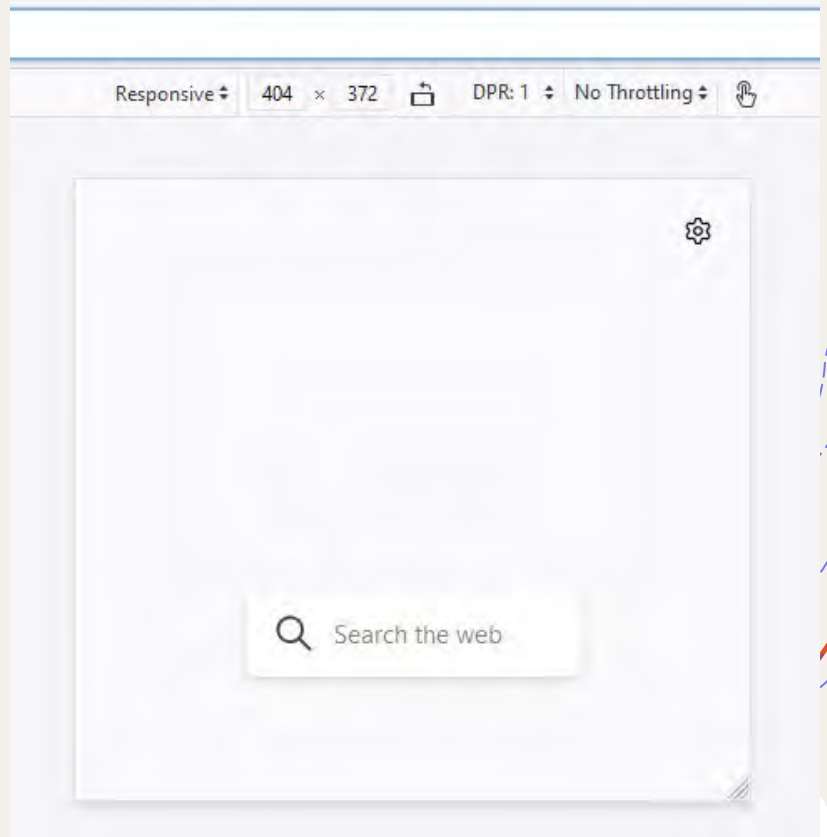
RESPONSIVE DESIGN

Use your browser to locate its ***responsive design mode***. This will allow you to test countless viewports.

Here is the example for reaching responsive design mode in *LibreWolf* (Firefox).







RESPONSIVE DESIGN

Fixed layouts retain their look despite the viewport size.

Fluid layouts grow and shrink site content based on the size of the viewport.

Flexible images use CSS rules in fluid layouts to resize.

RESPONSIVE DESIGN

The **viewport meta element** is used to deliver websites that are optimized to the client device.

<meta name="viewport" content="width=device-width, initial-scale=1.0">

FLEXIBLE IMAGES

The viewport meta element allows for the styling of image elements in a responsive way.

```
img {
    width: 100%;
    height:
auto;
  }
```

USEFUL CSS STYLES

margin-top: 0.5em;	This gives some room above the element.
border-radius: 1.0em;	This rounds the corners of the header.
max-width: 800px;	This prevents the image from expanding beyond 800 pixels wide.

USEFUL CSS STYLES

list-style-type: none;	This removes bullet points.
border-top: solid 0.3em black;	This provides a border at the top of the element.
background-image: linear-gradient(white, black);	Instead of a picture, this blends two or more colors together gradually from top to bottom.

Module 7

Media Queries

MEDIA QUERIES

Media queries allow different CSS styles on the same page based on the size of the viewport and the media upon which it is viewed.

They are utilized with the *media attribute* within *link elements*.

```
<link rel="stylesheet" href="css/styles.css" media="screen">
```

MEDIA QUERIES

Screen media typically include smartphones, tablets, and computer monitors.

<link rel="stylesheet" href="css/**styles.css**" media="**screen**">

Print media is used for printing a webpage.

<link rel="stylesheet" href="css/**stylesprint.css**" media="**print**">

Speech media is used for reading the page out loud using screen readers.

<link rel="stylesheet" href="css/**stylesspeech.css**" media="**speech**">

MEDIA QUERIES

Media queries can also be specified using the CSS files themselves instead. As pages become more complex, this method becomes preferred.

```
@media screen {

}
```

```
@media print {

}
```

```
@media speech {

}
```

MEDIA QUERIES

Media queries have numerous **features** that can specify behavior changes based on the size (or orientation) of the screen or viewport.

```
@media screen and (min-width: 481px) {
    nav li { display: inline; }
}
```

MEDIA QUERIES

max-device-height **min-device-height**	Changes are made based on the pixel height of the ***screen***.
max-device-width **min-device-width**	Changes are made based on the pixel width of the ***screen***.
max-height **min-height**	***Viewport*** height accounts for changes.
max-width **min-width**	***Viewport*** width accounts for changes.
orientation	Landscape or portrait

Module 8

Layouts

FIGURE ELEMENT & HIDING CLASSES

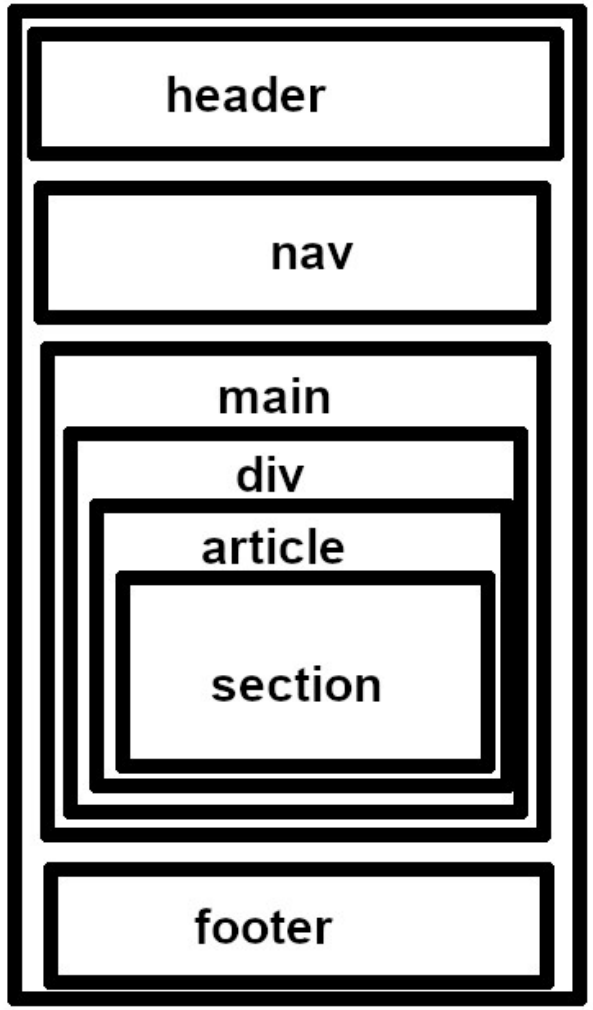
The **figure element** is a distinct section for image elements. They can include **figcaption element** to entitle a group of images in a figure element.

Sometimes elements such as these are too large for a mobile viewport. Our example provides for a figure element to be displayed for tablets and desktops.

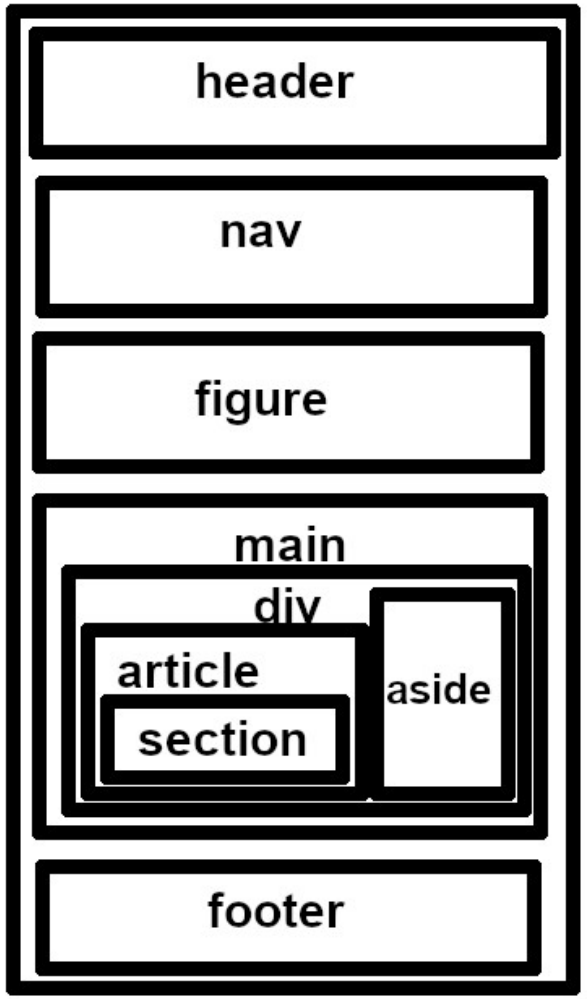
In this example, we will use the ***display: none*** style with the help of classes in CSS to hide items from certain viewports.

.mobile { display: none; }

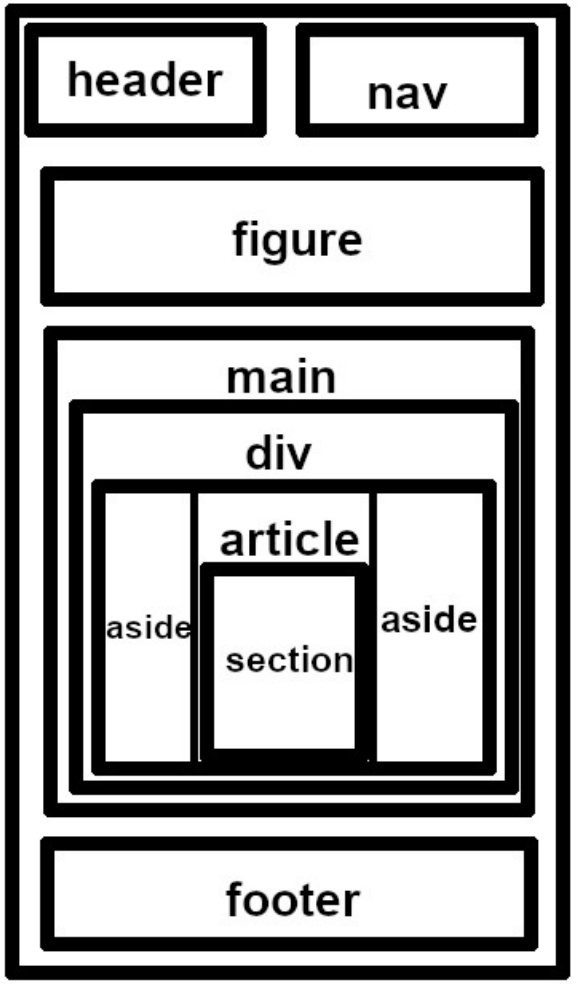
LAYOUT WIREFRAMES



mobile



tablet



desktop

ONE, TWO, AND THREE-COLUMN LAYOUTS

The mobile-first strategy involves keeping the page simple. Sometimes this means leaving out content, such as two of the columns.

We gradually add in content for tablets. Here we will use a two-column layout.

Desktops might have a three-column layout. We will use one here.

ONE, TWO, AND THREE-COLUMN LAYOUTS

We can more easily control the columns using ***multiple classes*** for each element as shown.

```
<div class="row">
        <div class="column left desktop">
            <aside>This is the left column.</aside>
        </div>

        <div class="column center">
            <article>This is the home page.<section></section></article>
        </div>

        <div class="column right tablet">
            <aside>This is the right column.</aside>
        </div>
    </div>
```

ONE, TWO, AND THREE-COLUMN LAYOUTS

It is important to note that the website has a three-column layout in every viewport.

However, in two of the viewports, columns are hidden from view as is done with the *figure element*.

The same classes are ***reused*** to hide multiple elements.

ONE, TWO, AND THREE-COLUMN LAYOUTS

Scroll boxes can organize columns in a clean way. Instead of allowing the column to control the size of the element, limiting the columns to the desired size is possible. Then the overflow can be placed in a scroll box.

```
main div .center {
        height: 400px; /*This limits the size of the box.*/
        overflow: scroll; /*If the box overflows, make it scroll*/
}
```

IMPORT STYLE SHEETS

The style.css is quite long now. Yet it does not have to be.

We can break up the CSS rules into multiple files and import them into style.css.

The **@import rule** must be placed at the top of the style.css page in order to fetch what was placed in the other pages.

@import url(“mobilesytle.css”);

SKELETAL NOTES

Notice how we have not added *any* major content yet!

Just like regular skeletons and scaffolding, they are complex and require a lot of work to get right.

This will give us a strong foundation upon which to place our content and to make modifications as we **code, debug, and repeat.**

Module 9

Tables

TABLES

Table elements present HTML data in a formatted table.

Caption elements entitle table elements.

```
<table>
    <caption>Table Title</caption>
    <tr><th></th></tr>
    <tr><th></th></tr>
</table>
```

TABLES

Table row elements enclose table headers or table data.

Table header elements entitle table columns.

Table data elements provide the data to fill the table.

```
<table>
    <caption></caption>
    <tr><th>Table header</th></tr>
    <tr><td>Table data</td></tr>
</table>
```

TABLE BORDERS

In order to view tables, the **border** style must be set.

border: 0.1em solid black;

They must be set for the table element, the table header elements, and the table data elements.

CSS makes this easy:

```
table {
    border: 0.1em solid black;
}
```

```
th, td {
    border: 0.1em solid black;
}
```

TABLE BORDERS

These are how the default borders look. Notice the inside lines, the space between them, and the border surrounding them.

Class Schedule

Class	Days	Time
Adult	M W F	1900-2100
Teen	M W F	1700-1900
Child	T Th	1600-1800

TABLE BORDERS

These are collapsed borders. It is more like what you were thinking when you thought of tables. This requires a style rule.

```
border-collapse: collapse;
```

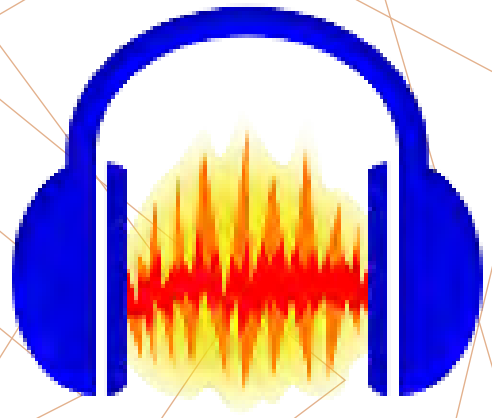
Class Schedule

Class	Days	Time
Adult	M W F	1900-2100
Teen	M W F	1700-1900
Child	T Th	1600-1800

Module 10

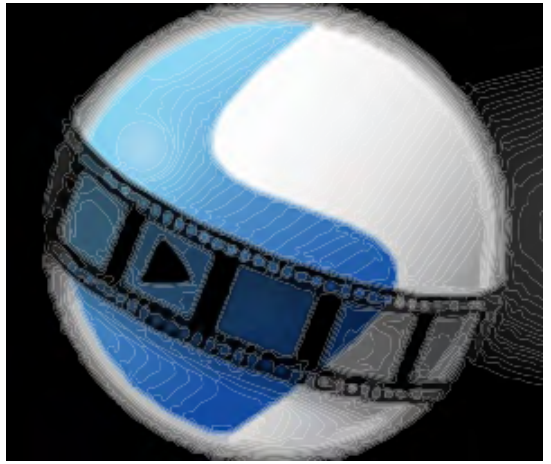
Multimedia

CREATING MULTIMEDIA



Audacity is an audio recording and editing program that exports to numerous audio formats.

OBS Studio is video capturing and streaming software.



OpenShot Video Editor is video editing software that transcodes many video formats.

MULTIMEDIA FORMATS

HTML5 natively supports the following audio and video formats:

Video: Theora Ogg (.ogg), WebM (.webm), and MPEG-4 (.mp4)

Audio: Ogg Vorbis (.ogg), WAV (.wav), and MP3 (.mp3)

Codecs are technologies essential to compress and decompress audio, video, and image files.

MULTIMEDIA ELEMENTS

The **video element** allows a video to be embedded on a web page.

```
<video width=“320" height="240" controls>
        <source src=“video.mp4" type="video/mp4">
Your browser does not support the video tag.
</video>
```

The **controls attribute** adds volume, play, and pause features to video or audio elements.

MULTIMEDIA ELEMENTS

The **audio element** allows audio to be embedded on a web page.

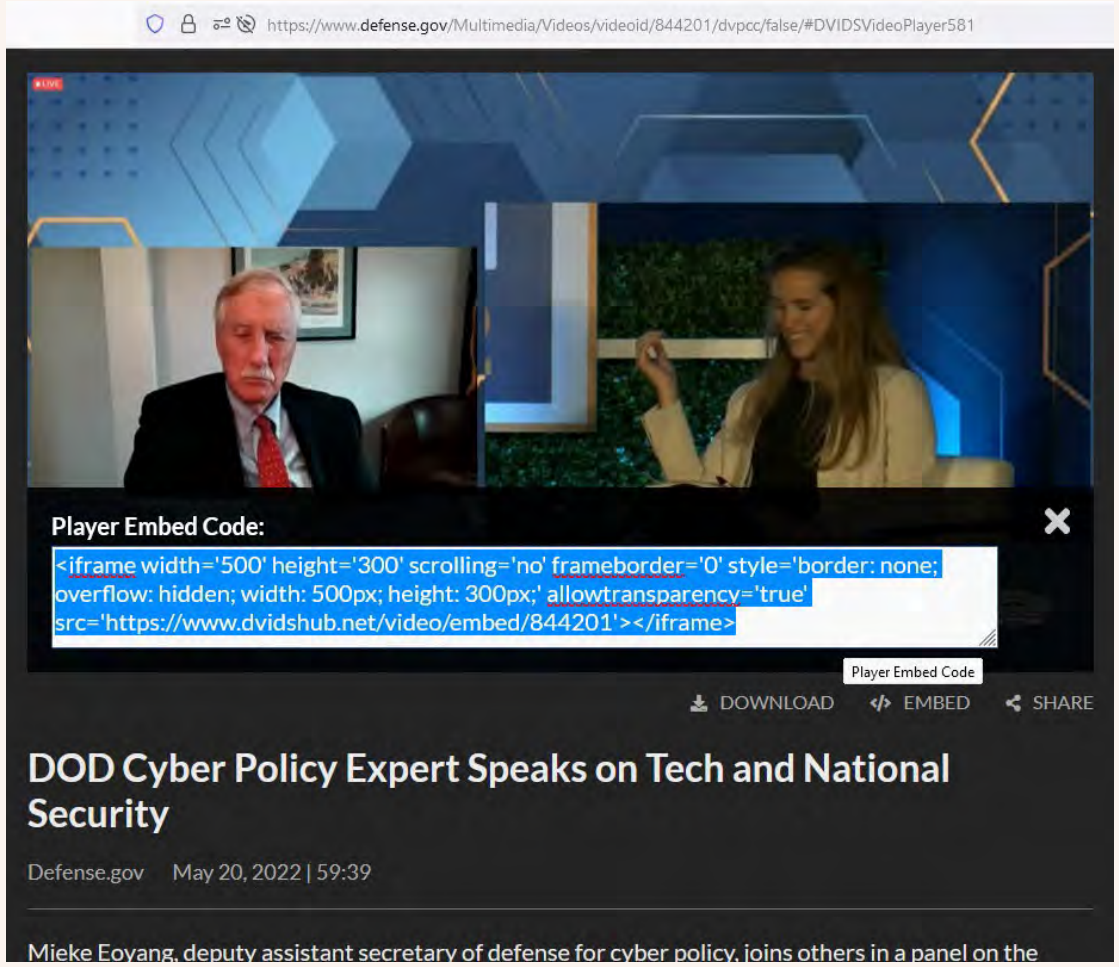
```
<audio controls>
        <source src="audio.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

The **autoplay attribute** is supposed to automatically play media files of audio or video elements, but modern web browsers have placed some restrictions that feature.

EXTERNAL MULTIMEDIA

HTML5 can serve multimedia originating from the web server or external sources.

External multimedia typically provides an embed code that can be saved on the web page.



<iframe width='500' height='300' scrolling='no' frameborder='0' style='border: none; overflow: hidden; width: 500px; height: 300px;' allowtransparency='true' src='https://www.dvidshub.net/video/embed/844201'></iframe>

Module 11

Forms & JavaScript

FORMS

Forms allow users to input information for retrieval from the website.
Input controls provide the means to enter text on the webpage.

```
<input type="text" id="name" name="name">
```

Here are some common input controls:

button	This provides a button on a webpage.
File	This allows a user to input a file for upload.
radio	These make a list that allow for one option from it to be made.
submit	This processes a webpage form.
text	This gives a single line for text input.

FORMS

The **textarea control** provides a large box on a web page for the input of multiple lines of text.

<textarea id=“textarea” name=“textarea” rows=“10” cols=“50”>**</textarea>**

FORMS

The **select control** provides a drop-down list that provide users a choice.

```
<select name="course">
        <option>Adult</option>
        <option>Teen</option>
        <option>Child</option>
</select>
```

FORMS

Label elements are paired with input controls using their id values using the **for attribute to provide** labels for them.

<label for="course">Course**</label>**

FORMS

The **action attribute** is used by form elements to perform an operation specified by it.

Common Gateway Interface (CGI) scripts can interact with web pages to provide information using languages such as PHP.

<form method="GET" **action=**"getForm.php">

JAVASCRIPT

The **onclick attribute** allows a button to take an action such as executing a JavaScript function.

<input type="submit" id="submit" value="Submit" **onclick="calculate()"**>

JAVASCRIPT

JavaScript is a scripting language that is used to provide extra functionality to web sites. The execution takes place within the web browser, which is why it is called a **client-side scripting** language.

JavaScript scripts can be on the web page internally or stored in an external location. Calls to internal or external scripts should be made within the head element.

External:

```
<script src="your/scriptlocation/script.js" type="text/javascript"></script>
```

Internal:

```
<script type="text/javascript">//code goes here</script>
```

JAVASCRIPT

Like CSS, JavaScript uses C-style multi-line comments.

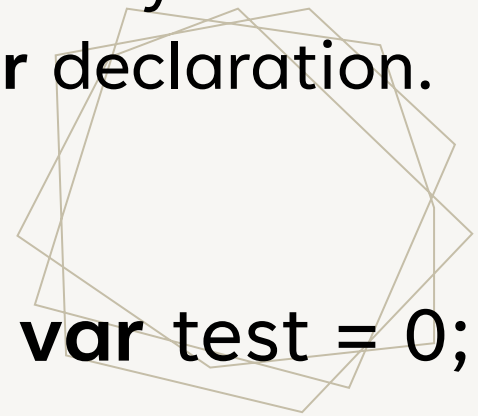
```
/*This
        is
    multiline!*/
```

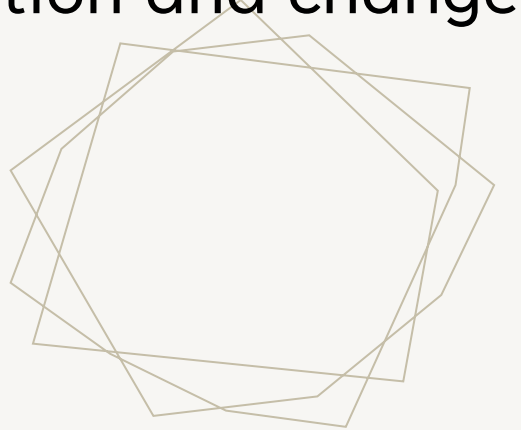
Yet JavaScript also includes C-style single line comments:

```
//This is single line!
```

JAVASCRIPT

Variables are a programmatic way to store information and changes to it. They are created using the **var** declaration.





```
var test = 0;
```

Semicolons must terminate ***every line*** of JavaScript. (Whitespace does not count.)

JAVASCRIPT

The **document.GetElementById()** method allows JavaScript to obtain data from HTML elements according to an element's id.

In the following example, the test variable is assigned the value of the element identified by "something".

```
test = document.getElementById("something").value;
```

JAVASCRIPT

Functions allow for the developer to create a sequence of operations to perform within a single call.

```
<script type=“text/javascript”>

        function calculate() {
                var test = 0;
                test = document.getElementById(“something").value;
        }
</script>


<input type="submit" id="submit" value="Submit" onclick="calculate()">
```

JAVASCRIPT

Math operations can be performed on variables and stored within them or other variables.

```
test = test + 1;

//The following also does the same thing.

test += 1;
```

JAVASCRIPT

The **alert()** method provides the user a message in an alert box.

alert(“This is the value of test” + test);

JQUERY

jQuery is a JavaScript library that provides prewritten functions that you can implement in your own projects.

Module 12

The Web Server

WEB SERVERS

There are many applications that provide web service.

IIS – This is Microsoft's web server that comes with Windows.

Apache – This is one of web server programs that receives the heaviest use.

Nginx (pronounced "engine x") – This is becoming very popular since it is faster than Apache and natively supports extra features.

WEB SERVERS

While Apache and Nginx can run on Windows, it is primarily used on Linux or BSD.

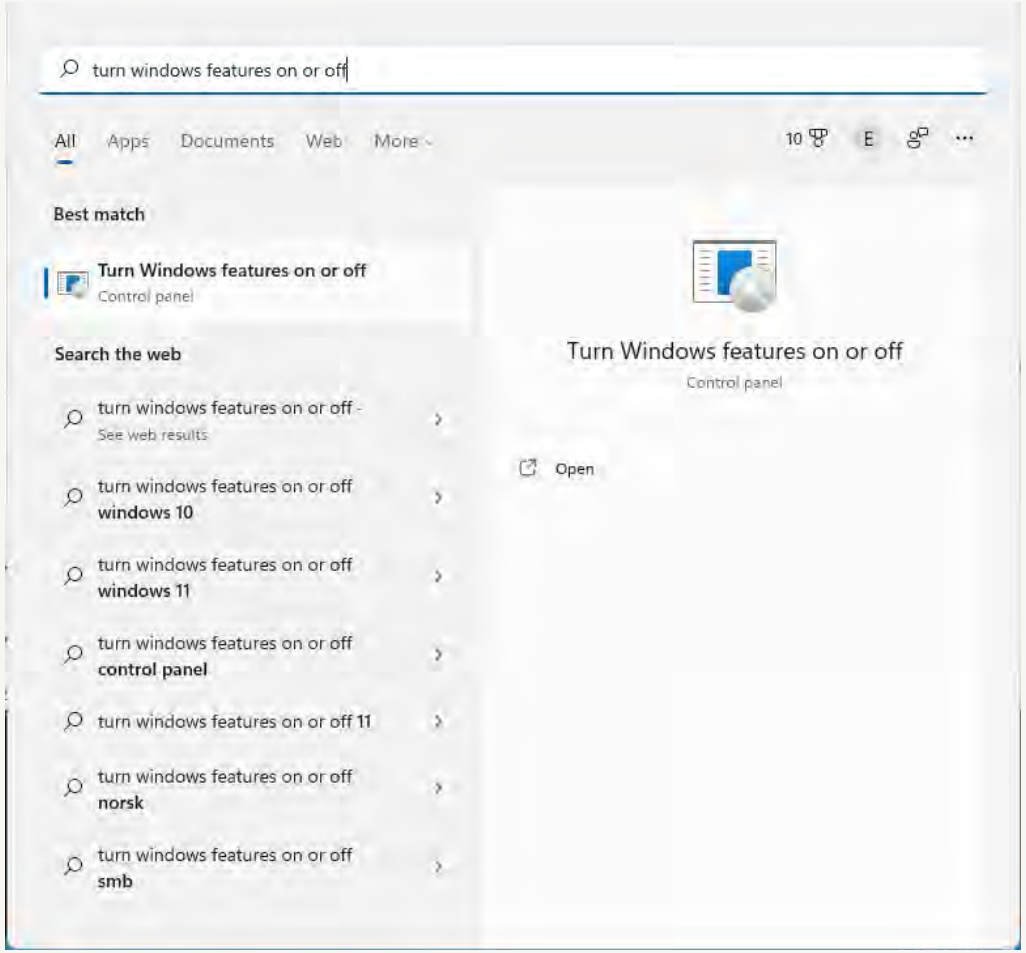
This is how to install and run the IIS web server in Windows 11 and make it serve our site. A similar process should work for previous Windows versions.

WEB SERVERS

Press the Start button.

Type “turn windows features on or off” in Windows Search.

Press “Turn Windows features on or off”.

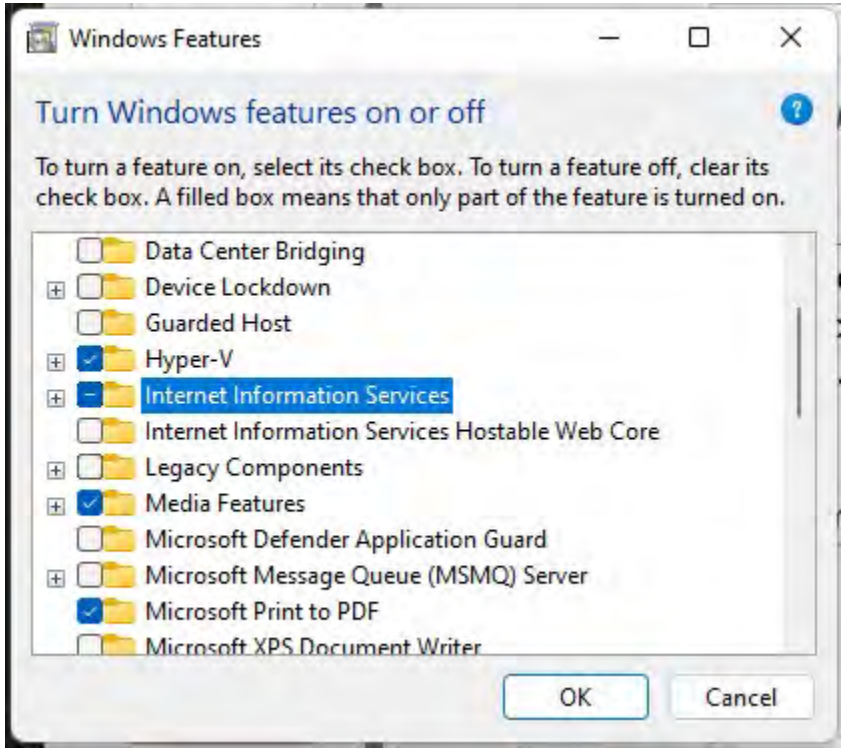


WEB SERVERS

Locate “Internet Information Services” in the Windows Features menu.

Press the check box next to “Internet Information Services”.

Press “OK”.

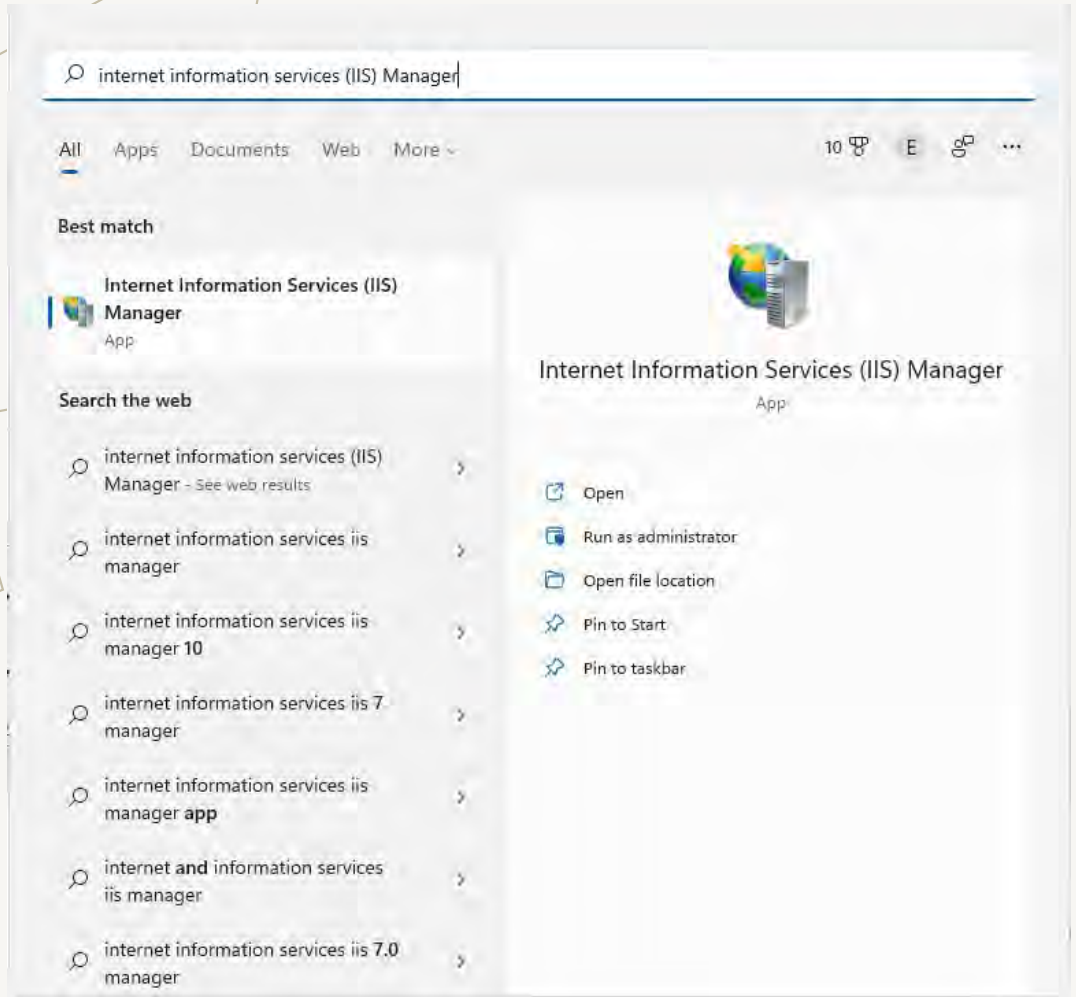


WEB SERVERS

Press the Start button.

Type “internet information services (IIS) Manager” into Windows Search.

Press “Internet Information Services (IIS) Manager”.

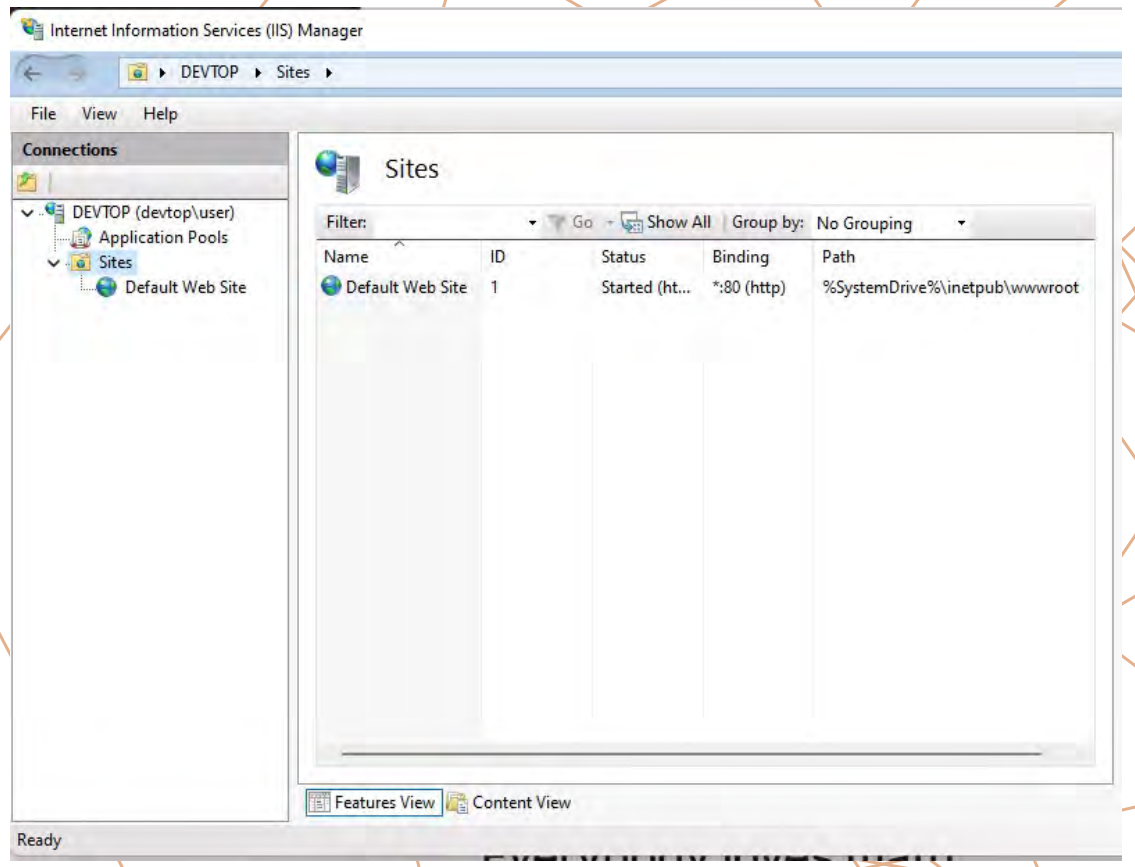


WEB SERVERS

On the left, select the dropdown menu next to your computer name.

Select the “Sites” folder.

The “Path” column on the right gives you the folder location.



%SystemDrive% is usually C:

WEB SERVERS

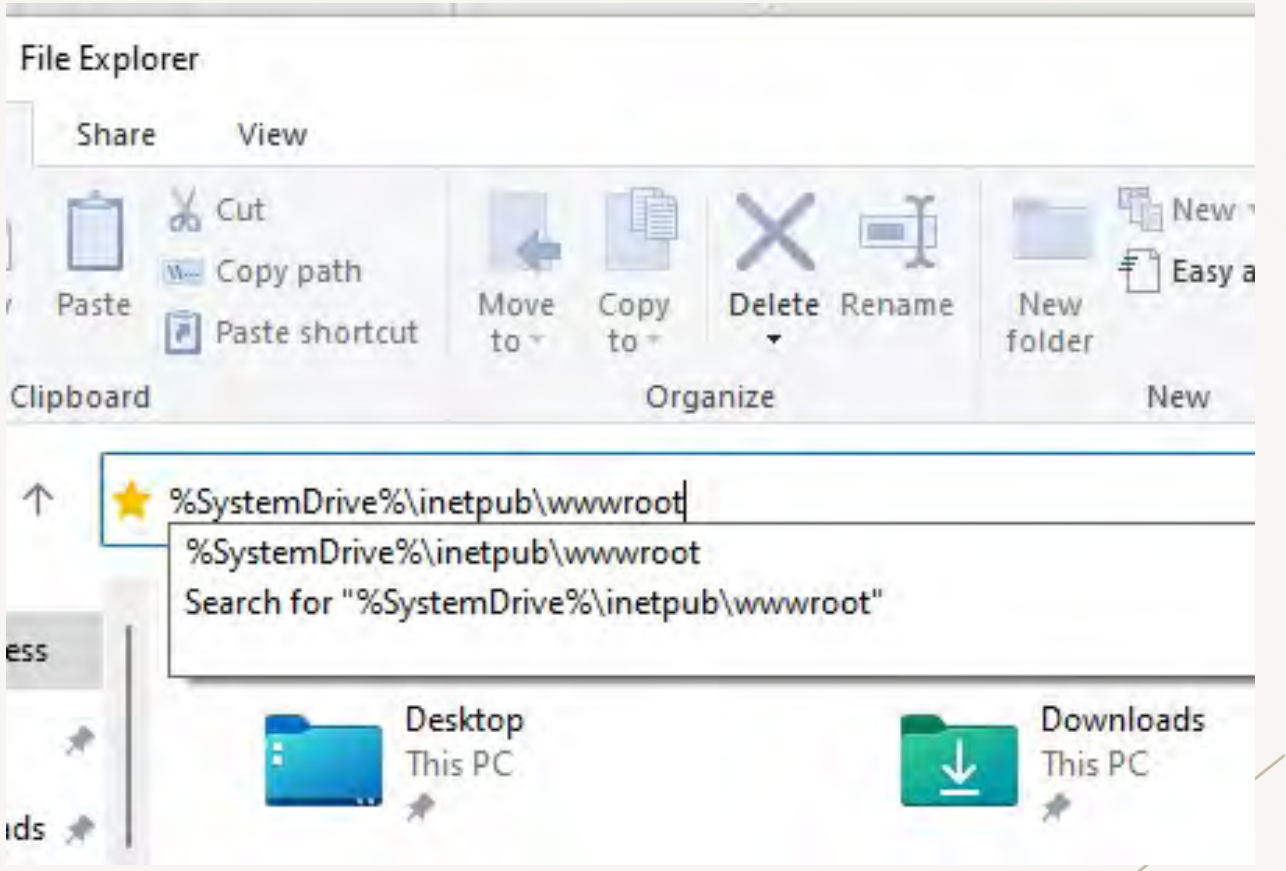
Open File Explorer.

Type

%SystemDrive%\inetpub\wwwroot

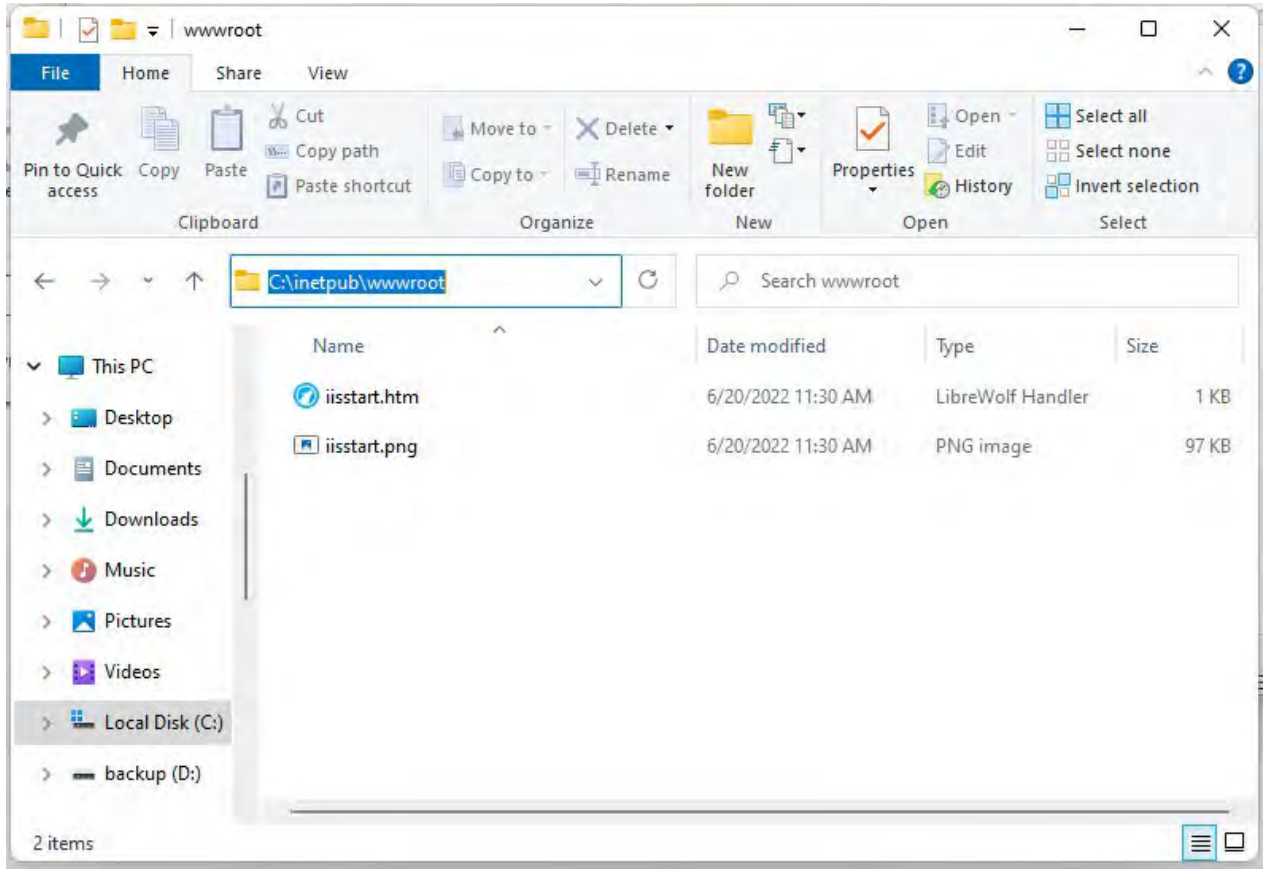
into the address bar.

Press enter.



WEB SERVERS

This is the main folder of the IIS web server.



Now open your web browser and type

localhost

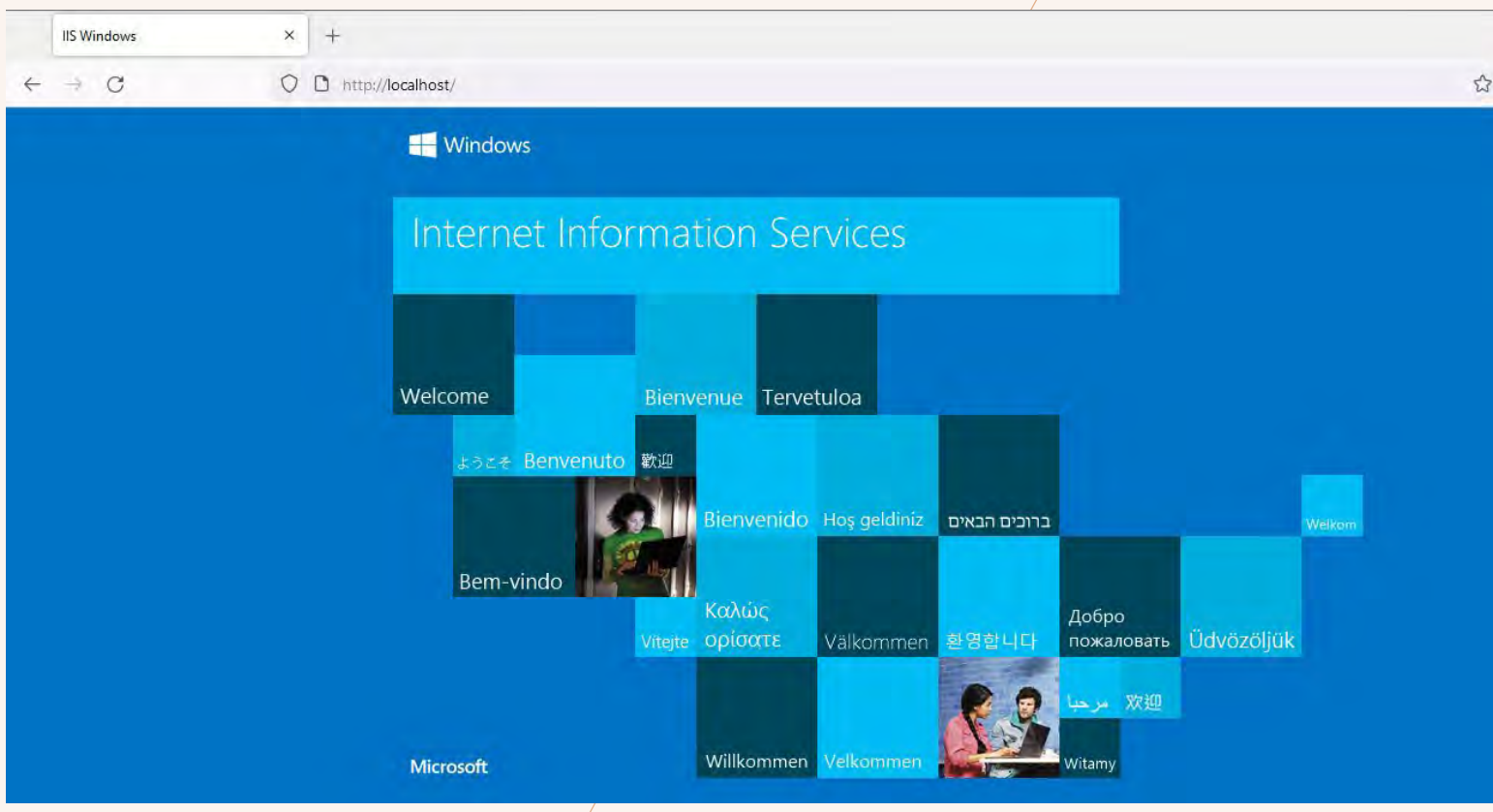
into your address bar.

Press enter.

WEB SERVERS

This is how the contents of your web server folder are displayed by your web server.

Let’s change that!



WEB SERVERS

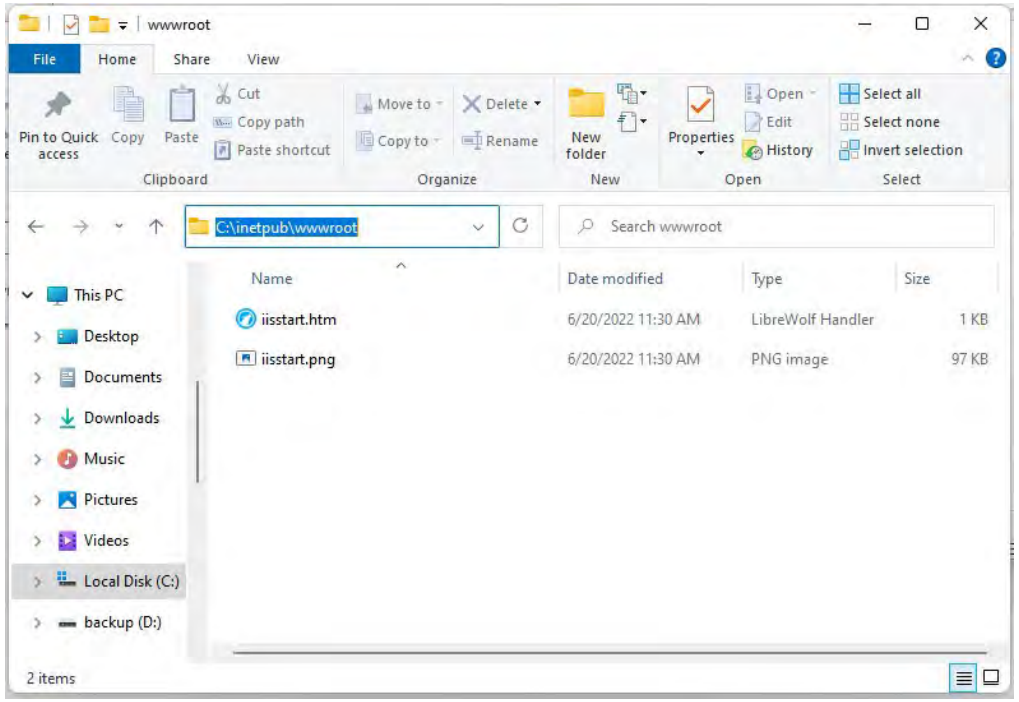
Select every file ***within*** your wwwroot folder.

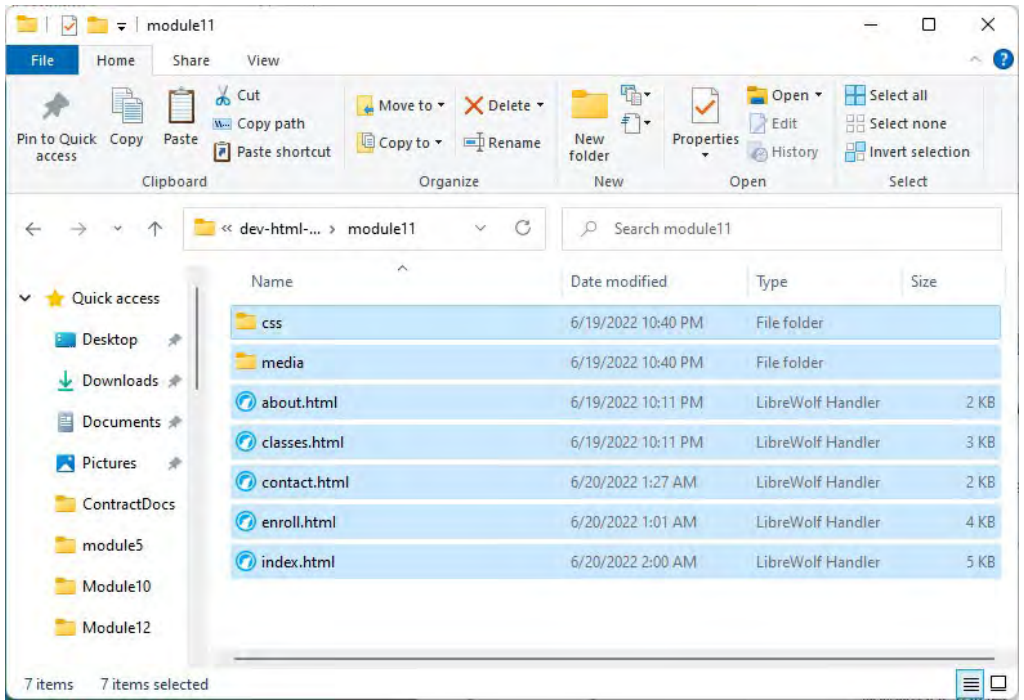
Delete these files.

Open your HTML project folder.

Copy every file and folder ***within*** your HTML project folder.

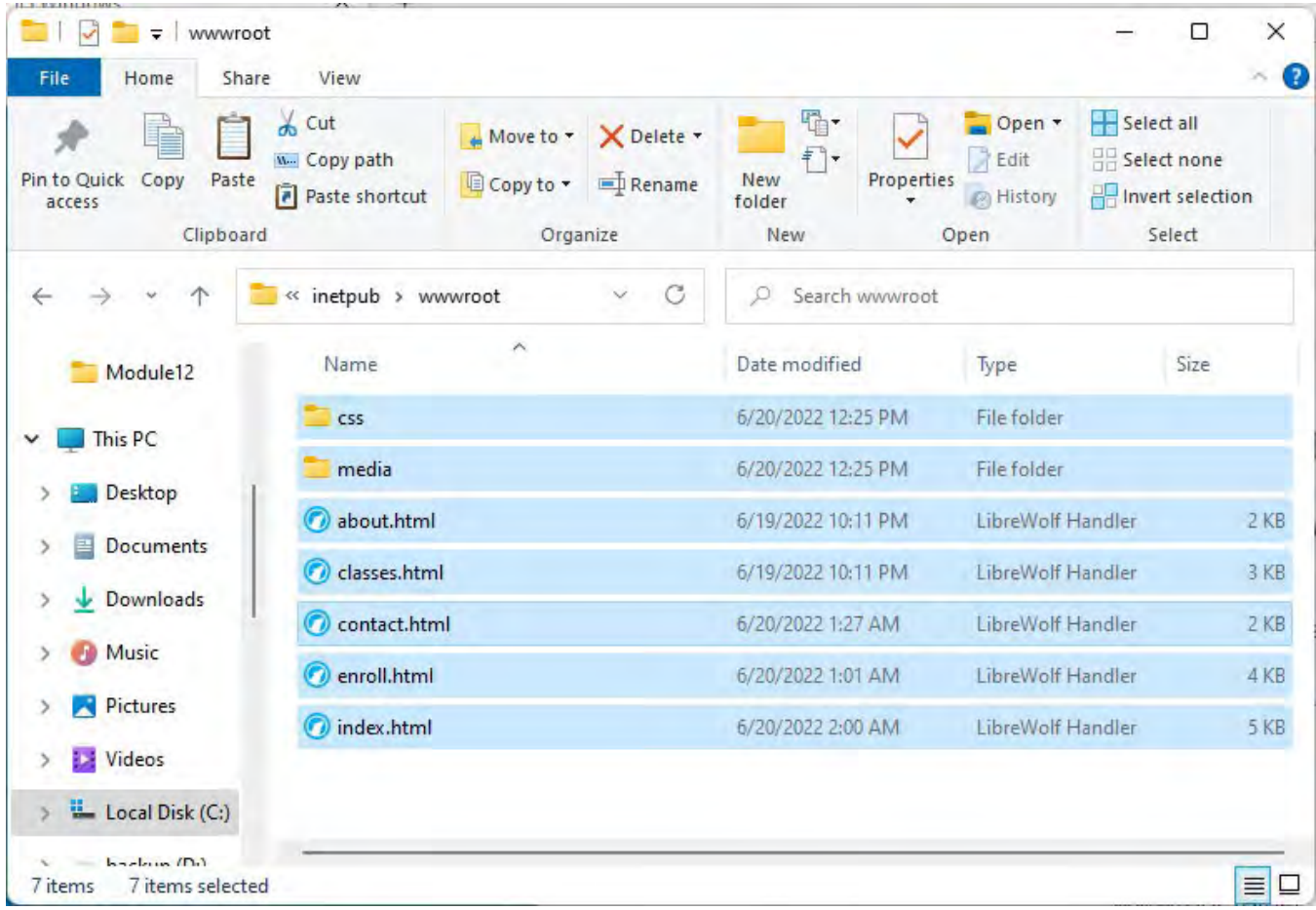
Paste those files and folder into your wwwroot folder.





WEB SERVERS

Refresh the localhost web page in your web browser.



WEB SERVERS

Voila! Congratulations on hosting your own web server!

It is important to note that you would not run a production website this way.

Always keep a production web server and a development web server (such as this one) separate!

Yet they play by the same fundamental rules. So, you can configure this development web server to host a web site on the Internet with a few extra configurations.

Locate the local IP address of your web server and you can test your web site on other devices in your network!

Module 13

Programming in JavaScript

DATA TYPES

Data does not simply exist without a certain form. Fundamental to programming languages, including JavaScript, is the notion of **data types.**

Variables in JavaScript can store many data types. Three of these data types include:

- **Numbers**
- **Strings**
- **Booleans**

DATA TYPES

When declaring variables in JavaScript, it is not necessary to declare the data type. Instead, JavaScript is **dynamically typed**. You simply declare a variable and (perhaps) assign a value. This can be both a blessing and a curse.

The feature allows data types to change in a variable for different situations. Yet you must remain vigilant of the data type under each given circumstance.

DATA TYPES

Numbers can represent both integers and floating-point numbers. That is (technically) because all Numbers are floating-point numbers in JavaScript.

Depending on the situation, JavaScript can sometimes treat what you think is a Number as a String (and vice versa).

DATA TYPES

Strings consist of text characters. String “math” typically consists of taking two strings and joining them together. This is called **concatenation**.

When you try to add two numbers and, for some reason, they are next to each other, it is likely because they are being treated like Strings.

Understanding the relationships between data types will allow you to exploit the behavior to your advantage.

Booleans can either be *true* or *false*.

true and *false* are keywords.

You can assign variables the values of *true* or *false*, but you cannot name your variable *true* or *false* as these keywords are reserved to JavaScript.

Keep in mind that if you assign to a variable “true” or “false” *in quotation marks*, it will be treated as a String.

DATA TYPES

You can exert some control over the data types like this:

var thisNumber = 25; //Number (lack of quotation marks)

var thisString = “25”; //String (2 & 5 are *characters*)

var thisBoolean = false; //Boolean (*true* or *false*)

GLOBAL METHODS

JavaScript comes equipped with its own methods. There are many of these methods, including those that simplify converting data types. Here are two useful examples:

- **parseInt()** – This converts a string into an integer.
- **parseFloat()** – This converts a string into a floating-point.

Keep in mind that the string should be numbers to work properly.

VARIABLES

Variables store data in memory for later use. They are called variables because their values may change.

There are two ways to declare variables.

The “third way” is declared like a variable, but the value **cannot** change and therefore is not a variable.

VARIABLES

- **var** – This is the original way to declare variables in JavaScript.
- **let** – This is new and works to declare variables.
- **const** – While grouped with variables and declared in a similar way, these are used for *constants*. The value always stays the same.

CONDITIONAL STATEMENTS

Conditional statements control program flow by requiring certain conditions to be met for execution. The condition which must be met is found in parentheses. The condition must evaluate as *true*.

There are four types:

- if
- else
- else if
- switch case

CONDITIONAL STATEMENTS

If statements use a condition found in parentheses to check whether the code within its braces should execute.

```
If (true) {
        //This always works because the statement is true.
}
```

CONDITIONAL STATEMENTS

Else statements are used in conjunction with if statements and execute when the if statement is false.

```
If (false) {
        //This never works because the statement is false.
}
else {
        //This always works because the statement is false.
}
```

CONDITIONAL STATEMENTS

Else If statements connect a new condition to be met in the event that the previous if statement returns false.

```
If (false) {
        //This never works because the statement is false.
}
else if (true) {
        //This always works because the previous
        //if statement is false and
        //the current if statement is true.
}
```

CONDITIONAL STATEMENTS

The switch statement offers a set of cases that can be triggered by a variable of a certain value.

If variable equals 1, then doThis() will execute.

If variable equals 2, then doThat() will execute.

The break keyword break allows for breaking out of the switch statement without executing any code below it.

```
Switch (variable) {
        case 1:
            doThis();
            break;
        case 2:
            doThat();
            break;
        default:
            doSomething()
}
```

LOOPS

Loops allow for automatic repetition of execution given certain conditions.

There are three types of JavaScript loops:

- while loop
- do while loop
- for loop

LOOPS

While loops check a condition before attempting execution.

```
while (true) {
        /*This is an infinite loop.  It can never escape because the
        condition is always true.*/
}
```

LOOPS

Do while loops will always execute once before checking the condition.

```
do {
        /*This will execute only once because the condition is
        false.*/
}
while (false);
```

LOOPS

For loops execute a certain number of times based on an incrementing variable as compared to a given condition.

```
for (var x = 0; x < 5; x++) {
        /*Generally, this will loop five times.*/
}
```

OPERATORS

Various operators in JavaScript perform many different operations on data. Below are some common operators.

Arithmetic operators	
+	a = 2 + 4;
-	a = 3 – 2;
*	a = 4 * 3;
/	a = 4 / 2;
++	a++;
--	a--;

Assignment Operator	
let a = 0;	
+=	a += 4;
-=	a -= 4;
*=	a *= 4;
/=	a /= 4;

Comparison Operators	
==	equal to
!=	not equal to
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to