1. **Welcome**

This class is covered in four sessions of two hours each. The classes are all held at Fuse SPC Coworking.

To participate in these classes you need several prerequisites:

* A laptop with either Windows or OS X operating system.
* Either [Chrome](https://www.google.com/intl/en/chrome/browser/) and [Web Developer Extension](https://chrome.google.com/webstore/detail/chrome-apps-extensions-de/ohmmkhmmmpcnpikjeljgnaoabkaalbgc?hl=en-US) or [Firefox](https://www.mozilla.org/en-US/firefox/new/) and [Firebug](https://addons.mozilla.org/en-US/firefox/addon/firebug/).
* A text editor - We recommend [Sublime Text 2](http://www.sublimetext.com/) (It's free and available for both Mac and PC).

Erin Brown will be the instructor for this class; Tanya and Brendan will be the assistants,

You will learn HTML (with some CSS) from the ground up. In doing so, you will fill in the blanks you may have missed by going it alone. We are emphasizing the features that designers and developers most frequently use in their daily work.

The slides for this course are at <http://gdi-tri-cities.github.io/gdi-core-html-css/>. Click on the title for this class for the slides for this class.

1. **Welcome**

To get to know a bit about who you are and who you are taking the class with, let’s each answer a few questions. Who wants to go first?

…

To get the most out of this class:

* Come to the class with all the tools needed already installed
* Learn or review some “best practices” and key principles so everyone will have a common starting point
* Have questions in mind to be answered during the workshop.

Topics for this workshop:

* Creating, saving, and editing HTML and CSS files.
* How to separate your page's content (HTML) from its presentation (CSS)-a key aspect of progressive enhancement.
* Structuring your content in a meaningful way by using tried and true HTML elements
* Adding images to your pages and optimizing them for the Web.
* Linking from one Web page to another page, or from one part of a page to another part.
* Styling text (size, color, bold, italics, and more); adding background colors and images; and implementing a fluid, multi-column layout that can shrink and expand to accommodate different screen sizes.

1. **Tools**

**Creating a New Web Page**

You don't need any special tools to create or edit a Web page. You can use any text editor, even Notepad, which is included with Windows, or TextWrangler, which is a free download for OS X (www.barebones.com/products/textwrangler). (Macs include an editor called TextEdit, but **it** has a bug in older versions of OS X that makes **it** too difficult to work with **HTML** files.)

**Browsers and Developer Extensions**

Browsers all have developer extensions that let you inspect the web page and make changes real time to the page to see what effect the changes would make to the page. These changes are not “savable” since you are manipulating a rendered page in a browser, not the source code.

Firefox is slightly different. Although it has developer extensions built in, the add-on Firebug is by far more powerful. So if you are developing for Firefox, make sure you add on Firebug.

**Search Engines**

Probably the most useful tool you have is the search engine; Google is by far the best of the search engines. Whatever you are curious about, you usually can find it through Google. It is an art to be able to form the right query but you will get better at it as you go along.

1. **Get Started: Sublime Text 2**

A Web page is text. The pages are saved in text-only format and can be viewed on practically any browser on any platform, whether desktop, mobile, tablet, or otherwise. A page may look differently viewed on one device versus another, but that's OK. The important thing as a first step is to make content accessible to all users, and HTML does that.

In addition to the content the viewer sees, a Web page includes **HTML** that provides information about the page itself, most of which the viewers don't see and is primarily intended for browsers and search engines. This can include information about the content's primary language (English, French, and so on), character encoding (typically UTF-8), and more.

HTML and CSS files are written in plain text without the hidden codes found in documents like Word and Pages. Because of this, we suggest text editors for you to use to create these files. The plain text is necessary because plain text is all any browser can “interpret” (parse).

1. **Terms**

**Web Design**

Although you can just jump in and start writing Web pages right away, it's a good idea to first think about and plan your site. That way, you'll give yourself direction, and you'll need to do less reorganizing later.

**To plan your site:**

* Figure out why you're creating this site. What do you want to convey?
* Think about your audience. How can you tailor your content to appeal to this audience?
* How many pages will it need? What sort of structure would you like it to have? Do you want visitors to go through your site in a particular sequence, or do you want to make it easy for them to explore in any direction?
* Sketch out your site on paper.
* Devise a simple, consistent naming convention for your pages, images, and other external files).

Sketching out your site and thinking about what it might contain can help you decide what sort of structure it needs.

Don't overdo the planning phase of your site. At some point, you've got to dig in and start writing content and code.

If you're not very familiar with the Web, do some surfing first to get an idea of the possibilities. You might start with the sites of some of your competitors.

It's common, but not required, to map your site's folder structure to how it's organized on paper.

See Erin Kissane's article "A Checklist for Content Work"

(http://alistapart.com/article/a-checklist-for-content-work) for ideas about how you might approach crafting your site's content. It's a taste of her book, which elaborates on the subject of content strategy.

Jason Beaird's *The Principles* of *Beautiful Web Design* (Sitepoint, 2010) may interest you if you're a non-designer or novice designer looking for guidance on how to design an attractive, effective site.

**Web Development**

During this class we will be developing a static web page – just including HTML and CSS. Dynamic web development includes a programming language like JavaScript, Ruby, .Net.

There are two parts to web development – front end – or presentation layer and back end - or business layer. Someone who is proficient in both front end and back end is referred to as a“ full stack” developer.

1. **Clients and Servers**

**Internet** –You can think of the Internet as a fully connected set of **paths** and **nodes** – looking somewhat like a spider web. At the source, the message is broken up into small packets of information with headers that say where it is going and how to reassemble the message. These packets travel through the paths-fiber optic cable, copper twisted pair telephone cable or air (wireless). The nodes are routers that read the headers on the packets and send them on to the next available node in the direction of the final destination. At the final destination, the packets are reassembled into the original message – either the request sent to the server or the webpage at the client.

**Clients** – We view websites using a software application called a *browser.* Browsers are programs on the client (for example, your laptop) that interpret HTML, CSS and scripting code (e.g. JavaScript) and display webpages as you see them. Popular browsers are Chrome, Firefox, Internet Explorer, Safari and Opera. There are others but they are not used much. One of the reasons that these are the popular browsers is that they make an effort to follow web standards as they are developed.

Although browser updates are free and readily available, many people do not use the latest version of these browsers. Therefore, web developers have to make their pages somewhat “backward compatible”.

Each browser implements the standards differently and adopts the standards at various rates. CSS normalizers help in creating a unified look across browsers and JavaScript frameworks like jQuery do the same for dynamic behavior. Using the principle of *progressive enhancement*, web developers can build websites that structurally make sure the content is always available across all browsers.

**Web Servers** – When you view a webpage, your computer (the client) sends a message to another computer (the server) to send back the webpage. Your message consists of the URL (Uniform Resource Locator) and information on where to send back the webpage. The URL tells where to send the message and what to send back (more on URLs in a future email).

Almost any computer can be configured as a web server, including your laptop. Your laptop could not handle all the traffic of a publically accessed web server, but it could be configured as a web server called “localhost” so you can view your website as if you were an outside viewer.

For publically available websites, you will probably use the services of a **webhosting** company unless you work for a company that maintains its own web servers.

**Devices** – Since more and more devices can be used to view websites, web designers are taking into account the various screen configurations and speeds of display to provide a good experience regardless of the device (platform).

**Screen Readers** – When users can’t view a webpage, a *screen reader* program can read the webpage to them –as long as the developer has used HTML correctly. Some countries have specific accessibility laws that require web pages are readable.

1. **Components of A Web Page**

While Web pages have become increasingly complex, their underlying structure remains remarkably simple. The first thing you should know is that it's impossible to create a Web page without **HTML.** As you will learn, **HTML** houses your content and describes its meaning. Inturn, Web browsers render your HTML-encased content for users.

A Web page is primarily made up of three components:

* *Text content:* The bare text that appears on the page to inform visitors about your business, family vacation, products, or whatever the focus of your page may be.
* *References to other files:* These load items such as images, audio, video, and SVG files, and they link to other **HTML** pages and assets, as well as to style sheets (which control your page's layout) and JavaScript files (which add behavior to your page).
* *Markup:* The **HTML** elements that describe your text content and make the references work. (The m in **HTML** stands for *markup.)*

1. **Web Page Technologies**

HTML is a clever system of including information about the content in a text document. This information, called markup, describes the *meaning* of the content, that is, the *semantics.*

I know the word semantics is a mouthful - it just means "giving something meaning”. You can use “meaning" or “structure" instead.

Browsers do a better job when the programmer gives the browser help in interpreting html documents by using the best tag for the job. This is a central difference between poorly functioning webpages and effective webpages.

The point is that although on a simple static page two tags may seem equivalent but as soon as you get past either simple or static, the meaning of tags becomes important.

For example, the **p** element marks up paragraph content. Many people don’t know when to use <p> and when to use <div> because they don’t see a difference on the page. But there are some different attributes for <div> and <p>. And you may what to add different styling for a paragraph than just a generic block.

HTML does *not* define how the content should appear in a browser; that's the role of CSS (Cascading Style Sheets). HTML5 stresses this distinction more than any prior version of HTML. It's at the core of the language.

You might be wondering why, if that's the case, some text in a basic HTML page looks larger than other text, or is bold or italicized.

Great question. The reason is that every Web browser has a built-in CSS file (a *cascading* *style sheet)* that dictates how each HTML element displays by default, unless you create your own that overwrites it. The default presentation varies slightly from browser to browser, but on the whole it is fairly consistent. More importantly, the content's underlying structure and meaning as defined by your HTML remain the same

You will create your own CSS in class 2. It will override the built in CSS for every element that you define CSS for.

1. **What Is HTML?**

Every webpage has some HTML (Hyper Text Markup Language). HTML defines the meaning of your content. CSS (Cascading Style Sheets) define how you content looks as a webpage. Every browser interprets HTML and CSS differently. Each browser has a default barebones CSS that styles the page. Your CSS will replace and enhance this default style.

HTML was around before CSS was invented so at first HTML tried to provide the presentation as well as the structure of the web page. When you look at older webpages you will see some convoluted uses of HTML to hack together something that HTML5 and CSS3 are able to do more cleanly today.

Every web page is interpreted by your browser and displayed, or rendered on your computer. If you want to see what is rendered, you can “view source” on any page.

1. **What Is HTML?**

Let’s do that now. Go to <http://gdi-tri-cities.github.io/gdi-core-html-css/example.html>

Right click (Windows) or double finger press (Mac) and select “view source” from the menu.

The browser interprets both HTML and CSS (and scripting languages like JavaScript, for that matter) line by line from the top of the file to the bottom. Multiple CSS files and JavaScript files can be added into the HTML file and they are interpreted in the order they listed in the HTML file.

Many webpages include content other than text such as images, audio, video or animation. These are usually files linked into the HTML file using appropriate HTML elements that tell the browser “add this file’s content and interpret it as an image”(or video, audio, or animation)

Small websites made up of static webpages are usually written with just HTML and CSS. Larger websites, particularly those that are updated frequently, will use a content management system (CMS), blogging tools or e-commerce software. The server uses these technologies to produce the final HTML and CSS file sent to the browser.

Dynamic webpages and web applications may also use a database and a server side programming language such as Java, .NET, or Ruby.

Even though things can rapidly get complex in web development, it always starts simply and adds complexity baby step by baby step. A thorough understanding of HTML and CSS will help you determine when a more complex solution is needed and when the seemingly complex problem can really be solved by a slick use of HTML and CSS.

Information on the Web should be accessible by all regardless of the sophistication of their browser or device. This is known as the principle of *universality*. *Progressive enhancement* (<http://en.wikipedia.org/wiki/Progressive_enhancement>) helps you build sites with universality in mind.

Progressive enhancement is simple but powerful:

* Start your webpage with **HTML** content and behavior that is accessible to all viewers
* Then add presentation with **CSS**
* Add additional dynamic behavior with **JavaScript** (or some other client side programming language)

The result is that devices and browsers capable of accessing basic pages will get the simplified, default experience; devices and browsers capable of viewing more robust sites will see the enhanced version. The experience on your site doesn't have to be the same for everyone, as long as your content is accessible. In essence, the idea behind progressive enhancement is that everyone gets a page they can use.

Another principle you may have heard about is a complementary best practice, Responsive Web Design (<http://en.wikipedia.org/wiki/Responsive_design>). RWD is using CSS to create an optimal viewing experience across a wide range of screen formats (cell phone, computer, or tablet).

Both of these principles are derived from the idea of separating document structure, contents, and semantics (HTML) from presentation (CSS), and behavior (JavaScript).

1. **History of HTML**

In 1990, Tim Berners-Lee, then a scientist at CERN (the Swiss organization originally called *Conseil Européen pour la Recherche Nucléaire* or European Council for Nuclear Research), proposed and later prototyped a hypertext project in response to a request for a simple, immediate, information-sharing facility, to be used among physicists working at CERN and other academic institutions. He called the project "WorldWideWeb".

The first publicly available description of HTML was a document called "HTML Tags", first mentioned on the Internet by Berners-Lee in late 1991. Eleven of these elements still exist in HTML 4.

Netscape and Microsoft developed competing HTML standards and the browser incompatibilities were born. In January 1997, the W3 Consortium formally endorsed HTML 3.2 as an HTML cross-industry specification. HTML 3.2 had been reviewed by all member organizations, including major browser vendors such as Netscape and Microsoft. By providing a neutral forum, the W3 Consortium had successfully obtained agreement upon a standard version of HTML.

Immediately after HTML 3.2, The “Cougar” project was proposed as a new version of HTML Late in 1997, the W3 Consortium adopted it as HTML. This is the first time CSS appears. Three years later, XML and HTML are combined as XHTML as an attempt to enforce cleaner code. Browsers, however, allow backward compatibility and apply a looser validation.

In 2002, IE has pretty much taken the lion share of the browser market. The term “table-less presentation” is used to imply using CSS to format the web page rather than using tables to provide the placement of text. Tables are relegated to actual tabular data on a web page.

In 2005, AJAX – Asynchronous JavaScript And XMLHttpRequest – is used to communicate request between client (browser) and server enabling more responsive dynamic web pages.

In 2008, Google releases Chrome and the first draft of HTML 5 is released. Although, HTML 5 is slated for release in 2022, IE and Firefox browsers begin implementing HTML 5 features. In 2010, when Apple declares that it will no longer support Flash, and pushes the audio and video features of HTML 5 into the forefront. YouTube uses a combination of Flash and HTML 5 to serve up 2 billion videos per day.

1. **What We Are Building Today**

No notes

1. **Organizing Files**

Before you have too many files, it's a good idea to figure out where you're going to put them. It's customary (but not required) to create a folder for each main section within your site, allowing you to group related HTML pages together.

**To organize your files:**

1. Create a central folder or directory to hold all the material that will be available on your Web site. On the Mac, choose **File> New Folder** in the Finder ®. In Windows, from the desktop (or within a folder of your choosing), right click and choose **New> Folder**. Give the folder a name.
2. Create sub-folders in a way that reflects the organization of your website. For instance, you may decide to create a separate folder for each section of your site, along with individual sub-folders within those as necessary.
3. It is common to create a top-level folder for your site's images and optionally add sub-folders to help you organize your images by section or other criteria. Another approach is to create a top level folder named Assets (or something similar) and put your images folder in that, along with folders for other assets, such as video, style sheets, and so on.

Use short, descriptive names for your files and folders, preferably separating words in a name with a dash *(not* a space). Use all lowercase letters so that your URLs are easier to type and thus your pages are easier to reach.

**File Names**

Like any other text document, a Web page has a file name that identifies itself to you, your visitors, and your visitors' Web browsers. There are a few tips to keep in mind when assigning file names to your Web pages that will help you organize your files, make it easier for your visitors to find and access your pages, ensure that their browsers view the pages correctly, and improve SEO (Search Engine Optimization)

**Use Lowercase File Names**

Linux servers are case sensitive and Windows servers are not. It’s better to keep all you file names as lower case so that you don’t have to think about case when using filenames on your pages.

**Separate Words with a Dash**

Never include spaces between words in your file names. Instead, use a dash, for example, company-history.html and my-favorite-movies.html. Dashes are preferred by search engines over underscores.

**Use the Proper Extension**

The principal way a browser knows that it should read a text document as a Web page is by looking at its extension. Although .htm also works, .html is customary, so I recommend you use that as your extension. If the page has some other extension, such as “.txt”, the browser will treat it as text and show all your nice code to the visitor.

**TIP:** Be aware that neither Mac OS nor Windows always reveals a document's real extension. Change your folder options, if necessary, so you can *see* extensions.

**To create a new Web page:**

1. Open any text editor.
2. Choose **File> New** to create a new, blank document
3. Create the **HTML** content.
4. Be sure to save your file

Don't use word processors, like Microsoft Word, to code your **HTML** pages. They may add unnecessary or invalid code to your files.

**Editing Web Pages**

Editing webpages is similar to creating a new page. Just choose **File> Open** from your text editor of choice and open the file.

Because Web pages are most often viewed with a Web browser, when you double-click them on the desktop the default browser cheerily opens up and displays them. If you want to edit the Web page, you'll have to manually open it in your text editor.

Once you've made changes to an already saved document, you can usually simply choose **File> Save** to save the changes, without having to worry about the format.

In Windows, you can also right-click the document's icon or file name and then choose **Edit** or **Open With** in the pop-up menu that appears. On a Mac, 2 finger-click the icon; select **Open With** in the pop-up menu; and then choose the desired text editor.

1. **DOCTYPE**

Let's take a look at a basic HTML page to give you context for the class. This is just to give you a taste of HTML.

Download and open example.html. Look at it in the browser by clicking on the file and in the text editor by opening it from the text editor

You will have a default browser set in Windows Explorer for Windows and Finder preferences on Mac.



This simple webpage is the rendering of the following HTML. If you copy this text into a text editor like Notepad (PC) or TextEdit (Mac) make sure plain text selected in Preferences) and save the file as **example.html**, you can click on that file and it will display in your default browser.

**<**!DOCTYPE html**>**

**<html lang=**"en">

**<head>**

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

**<title>**Girl Develop It -- Learn to Code**</title>**

**</head>**

**<body>**

**<h1>**Girl Develop It -- Learn to Code**</h1>**

**<img** *src*="images/homepage-view.png" *alt*="GDI home page" **/>**

**<p>**Empowering women of diverse backgrounds from around the world to learn how to develop software. Learn more about **<a** *href*="http://girldevelopit.com" *rel*="external" *title*="Learn more about GDI">Girl Develop It**</a>** around the country and here in the TriCities**.</p>**

**<p><small>&copy;** Girl Develop It**</small></p>**

**</body>**

**</html>**

Each page begins with the DOCTYPE declaration, which tells the browser the HTML version of the page.

Always include a DOCTYPE tag at the top of your web pages. HTML5's tag, **<!DOCTYPE html>,** is simple and appears to be backwardly compatible with previous versions of HTML**.** Previous version

The browsers will accept unquoted contents of any tag whether it’s upper or lower case. However, the convention is to use lower case. The exception is the word DOCTYPE. It’s more common to see DOCTYPE in all uppercase.

1. **HTML Tag**

The lang attribute is optional. In this case, we see the language for this page is English. Since the lang attribute is valid on all HTML elements. In web pages with multiple languages, you can wrap your language specific code in a different tag to indicate its language.

The **</html>** end tag signals the end of the page.

1. **Head and Body Tags**

Everything above the **<body>** start tag is the instructional information for browsers and search engines.

The head element contains the title of the page and meta information bout the page. Meta information is not visible to the user. Meta information tells search engines about your page and who created it.

All this content above the **<body>** is invisible to the user except the text between **<title>** and **</title>. “**Girl Develop It -- Learn to Code” appears as the title at the very top of the browser window and on a browser tab. It's typically the default name of a browser bookmark or favorite and is valuable information for search engines. Later on we will explain what the other parts of the top segment of a page do.

Your page's visible content goes between **<body>** and **</body>.**

1. **Head and Body Tags: Example**

No text

1. **Let’s Develop It**

Help the class create an empty html page.

The code's indentation has absolutely no bearing on whether the code is valid HTML. It also doesn't affect how the content displays in the browser. However, it's customary to indent code that's nested in a parent element to make it easier to glean the hierarchy of elements as you read through the code.

1. **Types of HTML Elements**

Now that you've seen some HTML, let's take a closer look at what constitutes markup.

HTML has three principal markup components: *elements, attributes,* and *values.* You've seen examples of each in our basic page.

**Elements**

Elements are like little labels that describe the different parts of a Web page: "This is a heading, that thing over there is a paragraph, and that group of links is navigation." Some elements have one or more attributes, which further describe the purpose and content (if any) of the element.

A start tag, optional attributes, optional content, optional nested tags, and an end tag indicate an HTML element.

Some elements, like the image element don’t follow this scheme – there’s no end tag.

Empty elements, like **img** shown here, do not surround any text content. They are a single tag that serves both to open and close the element. The space and forward slash at the end are optional in HTML, but it's common to include them. This was introduced for for compatibility with XHTML and has persisted even though no one uses XHTML any more. However, the **>** that completes the element is required.

It's customary to type your element names in all lowercase. Although HTML allows uppercase, it's looked upon as a dated practice.

**Attributes and Values**

*Attributes* contain information about the content in the document. In HTML, it is optional but customary for an attribute’s value *to be* enclosed in quotation marks and in lowercase letters.

Some attributes can accept any value; others have a set of *enumerated* (predefined) values. Be sure to write enumerated values in all lowercase letters.

Some elements, can take one or more attributes, each with its own value. The order is not important. Separate each attribute-value pair from the next with a space.

Many attributes require a number for their value, particularly those describing size and length. A numeric value never includes units, just the number. Where units are applicable, as in the width and height of an image or video, they are understood to be pixels.

Some attributes, like **href** and **src,** reference other files and thus must contain values in the form of a URL, or Uniform Resource Locator, a file's unique address on the Web.

1. **Anatomy of an HTML Element**

Technically, HTML documents contain tags and not elements. The elements are only generated *after* the parsing step when the page is rendered in the browser.

Still, for convenience everyone calls them “elements” rather than “element mark ups”.

1. **Tag Breakdown**

A start tag consists of a left angle bracket, an element label, optional attributes and a right angle bracket. In the following paragraph element start tag the p is the element label.

An end tag consists of a left angle bracket, a forward slash, an element label, and a right angle bracket. In the following example of a paragraph element end tag the only difference is the forward slash.

1. **Attribute Breakdown**

Attributes appear in the start tag and are made up of:

* a name (red here),
* an equal sign (sometimes a colon)
* a value (usually quoted).

The attributes can be listed in any order but usually the element label comes first after the left angle bracket.

1. **Nesting**

This example shows another important characteristic of HTML elements – they can nest like Russian *matreyshka* dolls. The innermost element end tag must come before the next element end tag and so on up to the final end tag **</html>.**

Some elements, like <div> can nest inside elements of the same type. Others, like the <p> cannot.

1. **Nesting: Example**

**Parents and Children**

If one element contains another, it is considered to be the parent of the enclosed, or child, element. Any elements contained in the child element are considered descendants of the outer, parent element. You can actually create a family tree of a Web page that shows the hierarchical relationships between each element on the page and that uniquely identifies each element.

This underlying, family tree-like structure is a key feature of HTML code. It facilitates both styling elements and applying JavaScript behavior to them.

It's important to note that when elements contain other elements, each element must be properly nested, that is, fully contained within its parent. Whenever you use an end tag, it should correspond to the last unclosed start tag. In other words, first open element 1, then open element 2, then close element 2, and then close element 1.

1. **Element: Paragraph**

Some HTML elements (for example, the **h1**, and **p**) cause content to display in separate blocks like paragraphs in a book, while other elements (for example, the **a** and **em**) don’t break the flow of a line.

Before HTML5, most elements were categorized as either *block-level* (the ones that displayed in their own block or line) or *inline* (the ones that displayed within a line of text). In HTML5, *inline* elements are now called *phrasing content* elements. The old *block-level* elements are now divided among new HTML5 categories focused on their semantic value.

In our intermediate HTML and CSS course, we focus on HTML5 and CSS3. This course is focused on the basics that are common in both HTML4 and HTML5.

However, whether you use an HTML element that existed since the dawn of the language or one new in HTML5, your goal should be the same: Choose the elements that best describe the meaning of your content without regard for their presentation.

1. **Example: Paragraphs**

The paragraph is marked up as a **p** (paragraph) element Just as in printed materials, a paragraph can contain a single sentence or several sentences. If our page needed another paragraph, you'd simply add another **p** element after the first one.

1. **Element: Header**

HTML provides you six heading levels, **h1**-**h6**, with h1 being the most important. An **h2** is a subheading of an **h1**, an **h3** is a subheading of an **h2**. and so on, just like when you type a document with various headings in a word processor.

Every HTML page should have an **h1** (or more, depending on your content), so marking up our heading with **h1** was the obvious choice.

1. **Example: Headers**

Regardless of the heading level, styling headers is the domain of the style sheet.

1. **Formatted Text**

There are two elements nested within our paragraph that define the meaning of pieces of text: emphasis (**em**) and importance (**strong**). These are examples of the *phrasing* *content* elements that HTML provides, the majority of which improve the meaning of paragraph text.

The **em** element means "emphasize.” Remember that because HTML describes the meaning of content, **em** dictates meaning, not visual, emphasis even though it's common to render text in an **em** element in italics.

Some old HTML will have <i> tags and <b> tags that render the same as <em> and <strong> respectively. These tags are deprecated and shouldn’t be used.

1. **Let’s Develop It**

No Text

1. **Element: Image**

Of course, part of what makes the Web so vibrant are the links from one page to another and the images, videos, music, animations, and more. These resources (such as videos) are not actually included in the HTML file. Instead they are external files saved independently and are simply referenced from within the page through links. Since the references are nothing more than text, the HTML file remains nearly universally accessible.

Except for text-only ones, browsers can handle links and images without skipping a beat.

The **img** element is the primary choice for displaying an image. The **src** attribute is in the form of an URL, or Uniform Resource Locator, a file's unique address on the Web where the image resides. It can be an absolute or relative URL. The **alt** attribute provides text that displays if the image doesn't load or if the page is viewed in a text-only browser.

1. **Element: Line Break**

The line break tag is another empty tag like img but it usually has no attributes. The **HTML <br> element**(or HTMLLine Break Element) produces a line break (carriage-return). In text.

It is useful for writing a poem or an address, where the division of lines is significant.

Do not use <br> to increase the gap between lines of text; use the CSS margin property or the <p> element. In XHTML documents, write this element as <br />.

1. **Let’s Develop It!**

Help the class add images and line breaks in their HTML page.

1. **Element: Link**

Define a link to another page with the **a** (anchor) element. The anchor element has a number of uses depending on the attributes in the tag. In this case the attribute is **href (** “hypertext reference”) making this type of anchor element the most powerful element in all of HTML because it makes the Web - the Web: It links one page to another page or resource, or links one part of a page to another part of a page (either the same page or a different one).

This **href** attribute of the anchor element defines a link to the Girl Develop it home page. The optional **title** attribute enhances the meaning of the anchor element by providing information about the linked page. It appears in the browser when a user hovers over the link.

1. **Link attributes**

The optional **target** attribute indicateshow to open the link.

Links are so common that the element and tag is most often referred to as an “a href” (pronounced “A H ref”)

Once you've learned more about the HTML elements available to you, choosing the right ones for your content is usually a straightforward task. Occasionally, you'll come across a piece of content that reasonably could be marked up in more than one way. There isn't always a single way.

HTML doesn't try to provide an element for every type of content imaginable, because the language would become ungainly. Instead, it takes a practical, real- world stance, defining elements that cover the vast majority of cases.

Part of HTML's beauty is that it's simple for anyone to learn the basics, build some pages, and grow their knowledge from there. So, although there are approximately 100 HTML elements, there's a core handful you'll find yourself using time and again, while the remaining ones are reserved for less common cases. You've already learned the basics of several common elements, so you're well on your way.

1. **URLS**

Uniform Resource Locator, or URL, is a fancy name for address. It contains information about where a file is and what a browser should do with it. Each file on the Internet has a unique URL.

The first part of the address is called the *scheme.* It tells the browser how to deal with the file that it is about to open. The most common scheme you will see is HTTP, or Hypertext Transfer Protocol. It is used to access Web pages.

The second part of the address is the *name* of the server where the file is located, followed by the *path* that leads to the file, and the *filename* itself. Sometimes, a URL omits a filename and ends with a path, which may or may not include a trailing forward slash. In this case, the address refers to the default file in the last directory in the path, typically called index.html.

**Relative URLs**

A relative address describes the location of the desired file in relationship to the location of the file that contains the link. So, the link might be to a page in the same folder as the page with the link or the link may be to a page in a folder under the one with the link.

The relative URL for a file that is in the same directory as the current page (that is, the one containing the URL in question) is simply the *filename* (including its extension).

You create the URL for a file in a subdirectory of the current directory by typing the name of the subdirectory followed by a forward slash and then the name and extension of the desired file.

To reference a file in a directory at a higher level of the file hierarchy, use two periods and a forward slash. You can combine and repeat the two periods and forward slash to reference any file on the same server or drive as the current file.

On a webserver, all links within the website are relative to the root directory of the website. So far, we have been working with one webpage. When you add more webpages to your website, you can link from one to another with relative links.

If you aren't developing your site locally on a server, then generally you'll want to use relative. They'll make it easy to move your pages from a local system to a server. As long as the relative position of each file remains constant, you won't have to change any of the paths, so the links will work correctly.

**Absolute URLs**

An absolute URL shows the entire path to the file, including the *scheme*, the *server name*, the complete *path*, and the *filename* itself. An absolute URL is analogous to a complete street address,. No matter where a letter is sent from, the post office will be able to find the recipient.

In terms of URLs, when use an absolute URL in a link. it will be the same no matter where the calling file – the file containing the link – is

When you're referencing a file from someone else's server, you'll always use an absolute URL.

1. **Let’s Develop It!**

Help class with links.

1. **Ordered and Unordered Lists**

The **<ol>**and **<ul>**both represent a list of items. They differ only in whether the order is meaningful. if the order of items is meaningful, the **<ol>**element should be used, otherwise use **<ul>.**

Typically, ordered-list items are displayed with a preceding numbering, which can be of any form, like numerals, letters or Romans numerals or even simple bullets.

Typically, unordered-list items are displayed with a bullet, which can be of several forms, like a dot, a circle or a squared.

This numbered style is not defined in the HTML description of the page, but in its associated CSS, using the **list-style-type**property. For most browsers, the default list-style-type is disk for unordered lists and Arabic

There is no limitation to the depth and nesting of lists defined with the **<ol>**and **<ul>**elements.

1. **Comments**

The **comment** tag is used to insert **comments** in the source code. **Comments** are not displayed in the browsers. You can use **comments** to explain your code, which can help you when you edit the source code at a later date.

Incorrect comments can cause parts of your page to become invisible or render incorrectly. Not all browsers handle nested comments well.

The comment tag does not support any attributes.

1. **Tables**

The HTML Table Element (<table>) represents data in two dimensions or more.

Prior to the creation of CSS, HTML <table> elements were often used as a method for page layout. This usage has been discouraged since HTML 4, and the <table> element should not be used for layout purposes.

Several tags are needed to display a table. At minimum are <tr> table row start and end tags and <td> table cell start and end tags.

Table tags must be nested in a specific order to display correctly.

1. **A Web Page's Content**

**Text**

The text contained within elements is perhaps a Web page's most basic ingredient. Text in an HTML page has some important characteristics.

When a browser renders HTML it collapses extra spaces or tabs into a single space and either converts returns and line feeds into a single space or ignores them altogether.

HTML used to be restricted to ASCII characters-basically the letters of the English language, numerals, and a few of the most common symbols. Accented characters (common to many languages of Western Europe) and many everyday symbols had to be created with special character references like **&eacute;** (for ) or **&copy;** (for ©).

See a full list at <http://www.elizabethcastro.com/html/extras/entities.html>.

Unicode mitigates a lot of issues with special characters. It's standard practice to encode pages in UTF-8, as in the basic page and save HTML files with the same encoding. I recommend you do the same.

Because Unicode is a superset of ASCII-it's everything ASCII is, and a lot more-Unicode-encoded documents are compatible with existing browsers and editors, except particularly old ones. Brows ers that don't understand Unicode will interpret the ASCII portion of the document properly, while browsers that do understand Unicode will display the non-ASCII portion as well. Even so, it's still common to use character references at times, such as for the copyright symbol since it's easy to both remember and type &copy;

Note that when you view the document with a browser, the extra returns and spaces are ignored and the character reference is replaced by the corresponding symbol (©).

Specify your document's character encoding directly after the head start tag. The **charset** attribute sets the encoding type.

**Key Takeaways**

The basics of HTML and some key best practices provide the foundation for building effective Web sites. Let's revisit the key takeaways:

* A Web page is primarily made up of three components: text content, references to other files, and markup.
* HTML markup is composed of elements, attributes, and values.
* It's customary to write your HTML in all lowercase (DOCTYPE is an exception), surround your attribute values with quotes, and close empty elements with a space and a forward slash ( */).*
* Always begin your HTML documents with the DOCTYPE declaration:

**<!DOCTYPE html>**

* A page's content goes in the **body** element. Instructions primarily intended for the browser and search engines are before that, mostly in the **head.**
* Mark up your content with HTML and without regard for how it should appear in a browser.
* HTML improves accessibility and can make your site more efficient, and easier to maintain and style.
* CSS controls the presentation of HTML content.
* Each browser's own style sheet dictates the default presentation of HTML. You can overwrite these rules with your own CSS.
* Create file and folder names in all lowercase, and separate words with a dash instead of a space or underscore.