Modern
HTML and CSS

MAX001AC

Audience:

Anyone needing a solid foundation in CSS and HTML.

Prerequisites:

None.

AUTHOR

Michael T Smith
Microsoft Certified Trainer
SharePoint / Office Servers and Services MVP
Senior Instructor, MAX Technical Training

MAX Technical Training, Inc.
4900 Parkway Drive, Suite 160
Mason, OH 45040
513.322.8888

MA-2001 A1 3/27/17 5/28/17

Table of Contents

Module 1: A Brief History of HTML and the Web

Note: If you are viewing this document using SkillPipe, you may need to click some of the images and code examples to enlarge them.

From the w3.org site:

> "*HTML is the World Wide Web's core markup language. Originally, HTML was primarily designed as a language for semantically describing scientific documents. Its general design, however, has enabled it to be adapted, over the subsequent years, to describe a number of other types of documents and even applications.*"

Welcome!

HTML has moved from a markup language for scientific documentation into just about every user interface you use today. Obviously used in internet web pages, HTML is also used in cell phones, help files, email and even as the primary user interface for traditionally desktop based applications like accounts payables.

This class is for:

- Web site developers needing to support a wide range of browsers.
- Application developers creating desktop applications.
- Anyone needing a solid foundation in CSS and HTML.

Prerequisites

While any background in HTML or development is a plus, all you need are basic PC skills and a desire to learn web development.

Which Comes First? HTML, CSS or JavaScript?

Web development requires a large number of skills, and it's often hard to describe one aspect of web development without referring to all of the other skills. Fifteen to twenty years ago you could create useful web pages with just HTML. Today you need at least HTML and CSS to do any useful web development, and you really need JavaScript to make a web page dance and sing! ☺ In this course we will focus on HTML and CSS and leave JavaScript to another course.

Versions?

While the various web browsers still have incompatibilities, most are now reasonably HTML 5 and CSS 3 compliant. In this course when we refer to HTML and CSS, we are referring to those two versions. Recent proposed enhancements, including HTML 5.1 and CSS4 are not largely supported yet and are not covered here. Like most technologies, change is constant, and you will forever be learning new tools and tricks.

History

Who "owns" HTML?

HTML was originally defined by the World Wide Web Consortium, or W3C (w3.org), and was maintained by them until just after HTML 4.1 when the W3C moved towards a new XHTML standard. Apple, Mozilla and Opera were concerned about the future of HTML and created the Web Hypertext Application Technology Working Group (WHATWG) to focus on the future development of HTML. The work of WHATWG led to the development of HTML 5. Today both W3C and WHATWG are working on HTML standards.

Resources:

- https://www.w3.org/
- https://www.w3.org/TR/html5/
- https://whatwg.org/
- https://wiki.whatwg.org/wiki/FAQ
- http://developer.telerik.com/featured/w3c-vs-whatwg-html5-specs-differences-documented/

History?

Rather than repeating any of the many histories of HTML, read section 1.4 from this article from the W3.org site:

http://www.w3.org/TR/html51/introduction.html#introduction-history

Details, Details, Details…

Web development touches so many technologies that it is hard to focus on any one in isolation. In this section we will cover some of the things that you will see while learning HTML that are important to know, but not part of HTML itself.

Your instructor will provide an overview of these topics for now, and will expand on these topics as needed.

Modern Browsers and Compatibility

Web design has a number of challenges for the web developer:

- While HTML 5 is the current "standard" for web design, not all browsers implement all of the features. They don't even implement the core pre-HTML 5 features in a consistent manner.
- Many of the cell phones, tablets, smart TVs and other devices each have their own defaults and unique features.
- Not all users have decided to upgrade to the latest and greatest browsers. ☺

To deal with these issues you could:

- Design for the lowest common dominator, and try to live a very limited set of tools.
- Create adaptive designs that degrade gracefully so that you can do the really cool things with the fully compatible browsers, and still produce a working page for everything else.

To be adaptive we will:

- Use tools like normalize.css as a start to create an even playing field for our HTML work. (Set the default margins and fonts to be the same for all browsers.)
- Use the Modernizr JavaScript library to do feature detection.
- Use "shims" to add HTML5-like features to older or incompatible browsers. These are usually a combination of JavaScript and CSS. Example: Use the HTML5Shiv JavaScript library to support styling of new HTML5 tags (section, header, etc.) in older browsers (IE9 and earlier).
- Test, test and test!

Shims, Shivs, and Polyfills

As all browsers have not implemented HTML and CSS to the same level, or even the same way, we will need tools to "level the playing field". Shims are wedges that you use to level, adjust or tweak things. In the world of HTML, shims are typically CSS and JavaScript that you add to your page to make browsers more compatible, let older browsers support new features, and just make the developers life a bit easier. A "shiv" is the same as a "shim", and the term is most often associated with "HTML5shiv.js".

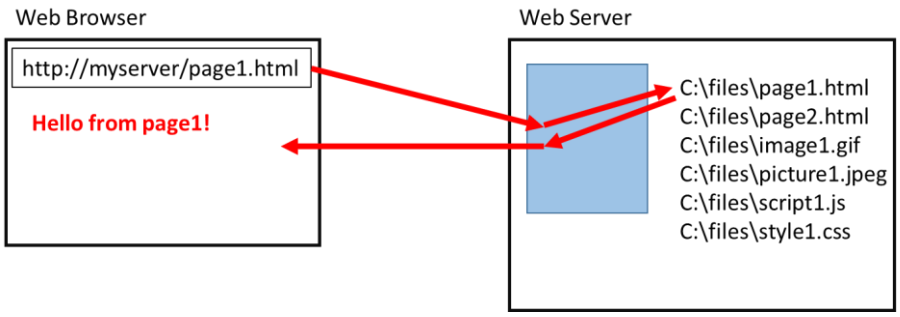
A "polyfill" is a shim that implements a modern API in an older browser without impacting newer browsers that already support the API. If the polyfill is removed from a newer browser that does not need it, it will make no change to functionality. A shim makes changes so all browsers behave the same. As an example, "normalize.css" changes the default margins and removing normalize.css may change the layout of the page.

The three terms are often used interchangeably.

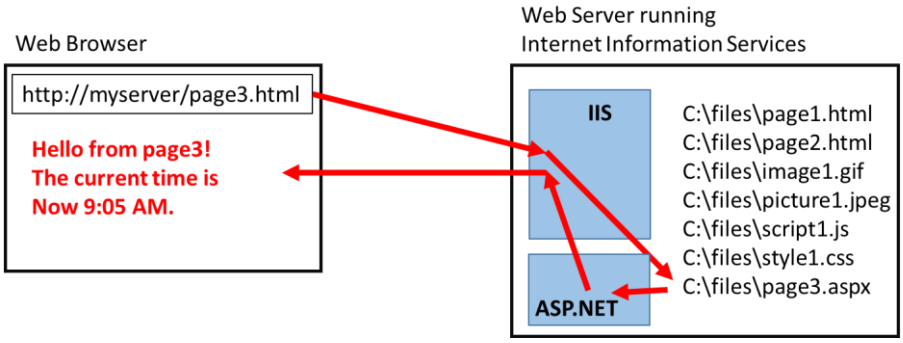
For a list of shims, shivs and polyfills see:
https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-browser-Polyfills

The Web Server

In the simplest form, a web server receives a request from a client in the form of a URL (Uniform Resource Locator) and returns a resource, typically an HTML file.



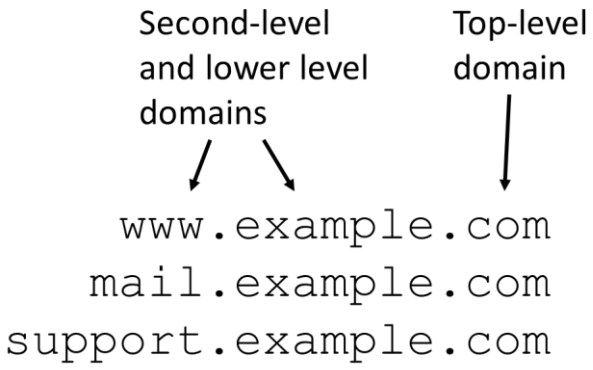
Most servers today will forward the request to a service that processes the request and then returns dynamically generated HTML or other content.



The most common web servers are Apache, Internet Information Services (Microsoft), nginx (pronounced Engine X) and GWS (Google).

Domain Names

Domain names are often called host names as they refer to a host or server.

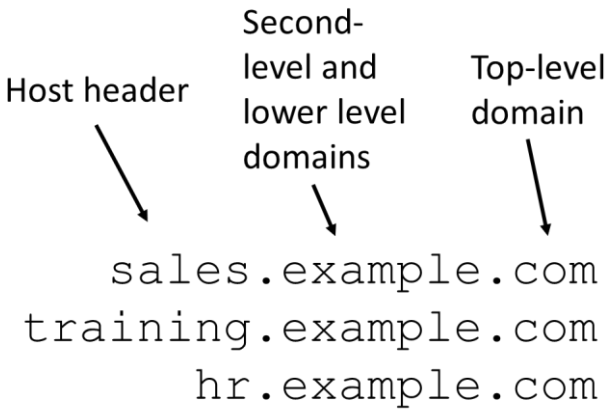


Top Level Domains

In the early days of the web there were only a handful of top level domains: gov, edu, com, mil, org, net, and int. Today there are 1,000. While they have been registered, most are not in common usage yet. Examples: .fun, .ford, .hospital, .hotmail, .microsoft, .apple.

Host Header

The third level of a domain name is often used to uniquely identify a web application on a server hosting multiple applications. This is referred to as a "host header" and is often used for "vanity domains" and as a means of having multiple domains share a single IP address.

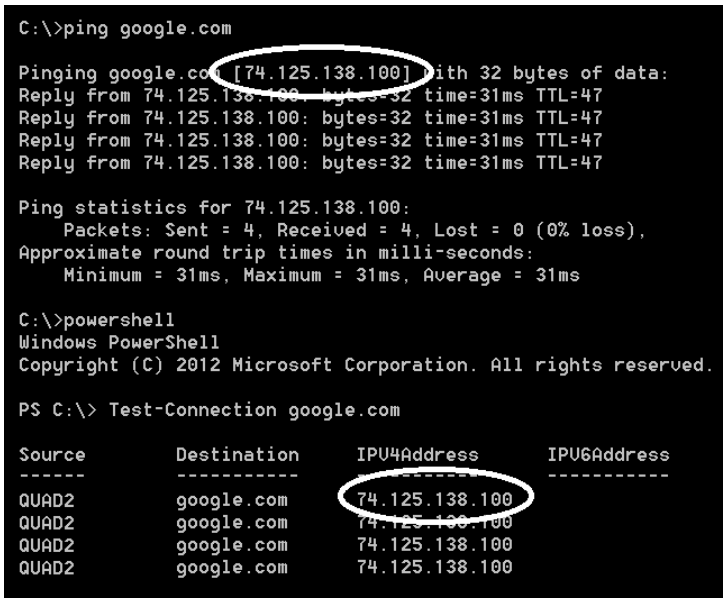


Notes:

- Domain names are not case sensitive, but are occasionally typed in marketing materials with inner capital letters to make them easier to read and remember.
- Domain names are registered with an Internet Domain Registrar such as GoDaddy.

IP Addresses

Each domain is matched to an IP address. The IP address is used to locate the web server on the network or internet. You can find the IP address for most domains by using the DOS PING command or the PowerShell Test-Connection cmdlet.



URL / Uniform Resource Locator

A URL is a full path to a resource on a web server. It will include the domain, or name, for the server, the protocol for the request and the resource being requested.

Default Page

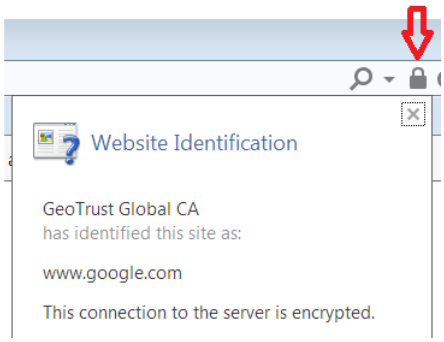
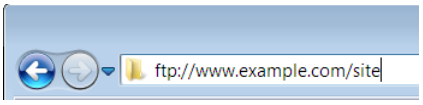
When a resource file name is not supplied, most web servers will return a default page. These pages are typically named "default", "index" or "home" with extensions that may be unique to the server software. You may see default pages with names like index.html, home.aspx, default.php, or even just "home". In this course we will be building basic HTML pages and will use ".html" as the file extension.

Protocols

The web supports multiple protocols, or communication standards, to exchange data between a client application and web server. In your web development work, you will be using HTTP and HTTPS for web pages and maybe FTP/FTPS for uploading your files to a web server.

A sampling of web communications protocols.

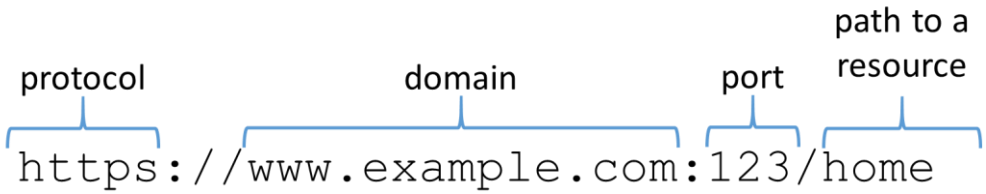
http://	Hypertext Transfer Protocol (HTTP) is the primary protocol for web server requests. The data can either be clear text or compressed. HTTP is used for both the delivery of web pages displayed in browsers and for web services that are consumed by applications. 
https://	HTTPS adds Secure Sockets Layer (SSL) encryption to HTTP and requires the addition of an encryption certificate to the web server. Most web browsers will provide details about the certificate and the encryption.

	
ftp://	File Transfer Protocol (FTP) is used for transferring files to and from a web server. You can use FTP directly from Windows Explorer, from FTP tools and from many HTML editors. 
ftps://	FTPS adds Secure Sockets Layer (SSL) encryption to FTP.
ws:// and wss://	The WebSocket (WS) protocol is used to open a persistent connection between a client application and a server. WS is often used for chat applications. The HTML 5 specification added an API for WebSockets.

For a general list of computer communications protocols see:
https://en.wikipedia.org/wiki/Lists_of_network_protocols

Port Numbers

While not common on public web sites, a port number is often used in the URLs of development, testing and other internal web sites. The default port number is 80 for http requests and 433 for https requests. Non-default port numbers are added to the URL between the domain name and the resource path.



The Life of a Web Page

The general flow of a user's request for a web page will follow this pattern:

1. User types a URL into the address bar of a browser.
    - Behind the scenes the domain name from the URL is sent to a DNS server to retrieve the IP address of the server, which is then used to route the request across the network or internet.
2. The server receives the request and does one of the following:

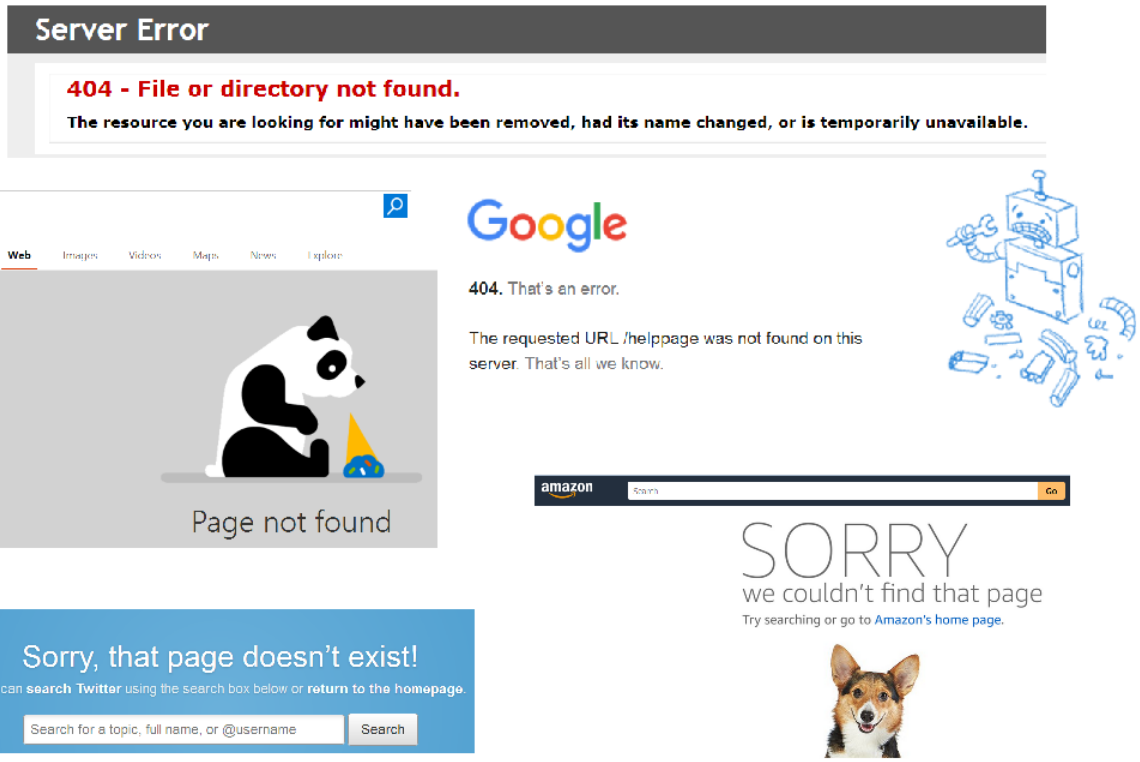
- Returns an error message if the page does not exist. (A 404 error for example.)
- Returns the contents of simple HTM or HTML files.
- Forwards the request to a server side processor that generates the final HTML. (Examples: asp, aspx, php, etc.)

3. The user's browser receives the text of the generated HTML and:
    1. Parses the HTML tags and converts them to elements. These elements are then added to a Document Object Model (DOM).
    2. Requests and processes any linked files. These might include images, JavaScript or CSS files.
    3. Creates a display using the data found in the DOM.
    4. Modifies the DOM using JavaScript.

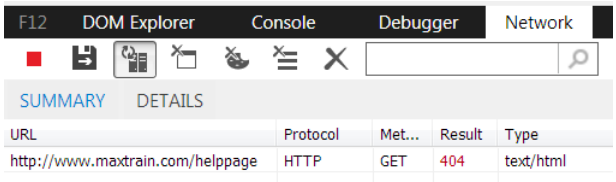
HTTP Status Code Definitions

Each request to a web server returns a status code. These codes are returned in the HTTP Header and can be discovered using the browser's developer tools (F12 in most browsers). Many web sites will also return "user friendly" error pages.

Here is a default 404 status page and four custom error page examples:



An example of an Internet Explorer 11 F12 Developer Tools display of a 404 error:



Below are a few of the more common status codes returned by web servers. For a complete list see: http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

Note that each server technology may support a different subset of the status codes. For those returned by Microsoft Internet Information Services (IIS). (see: https://support.microsoft.com/en-us/kb/943891) Some servers, such as IIS, return "sub-codes" of the codes to report statuses unique to that server. For example. IIS returns multiple 403 statuses including "403.20 - Forbidden: Passport logon failed".

Group 1xx	**Informational**	
Group 2xx	**Successful**	
	200 OK	The client's request was successfully received, understood, and accepted.
Group 3xx	**Redirection**	
	301 Moved Permanently	A page has been moved / URL changed and the client/user should update their links.
	302 Found	A temporary redirect.
Group 4xx	**Client Error**	
	400 Bad Request	The request could not be understood by the server due to malformed syntax.
	401 Unauthorized	The request requires user authentication. IIS returns 5 "sub-codes".
	403 Forbidden	The server understood the request, but is refusing to fulfill it. Authorization will not help and the request SHOULD NOT be repeated. IIS returns over 22 "sub-codes" of status 403. Example: "403.20 - Forbidden: Passport logon failed"
	404 Not Found	IIS returns 20 "sub-codes" of status 404. Example: "404.7 - File extension denied."
Group 5xx	**Server Error**	
	500 Internal Server Error	IIS returns more than 14 "sub-codes" of status 500. Example: "500.13 - Web server is too busy"
	501 Not Implemented	The server does not support the functionality required to fulfill the request.
	503 Service Unavailable	The server is currently unable to handle the request due to a temporary overloading or maintenance of the server.

Most Important Things to Know as a Web Developer

- **HTML 5 with CSS3 is the current standard, but is not standing still.**

  - HTML 5.1 has been proposed (November 1, 2016 https://www.w3.org/TR/HTML51)
  - CSS4 is in the works… CSS is defined in modules, so there is no single integrated CSS4 specification. CSS4 level features will be eventually added by the browser vendors. (http://www.w3.org/Style/CSS/current-work)
  - HTML 5 and CSS3 are not consistently implemented across all modern browsers.
  - The trend is to drop the "5" and to just use "HTML" going forward. In this course we will use the version number only when describing features unique to a version.

- **Not all users on the web are using modern browsers.** You need to allow for these browsers.
- **The HTML standards do not define how to display content.** They only define the contextual purpose of the HTML tags.
- **CSS is used to control how HTML tags are displayed**, but does not define the context or purpose of content.
- **Web development technologies you need to master:**
  - HTML – Hyper Text Markup Language
  - CSS – Cascading Style Sheets
  - JavaScript
  - Images / graphics / videos
  - And many CSS and JavaScript libraries, add-ins and tools.
- **You can never learn it all.** (Google and Bing are your friends!)

Module 2: Core HTML Elements

Note: If you are viewing this document using SkillPipe, you may need to click some of the images and code examples to enlarge them.

HTML and CSS Editors

HTML and CSS can be created in plain text editors like Notepad, HTML aware text editors like Notepad++ and Visual Studio, or WYSIWYG (What You See Is What You Get) graphical editors like Adobe Dreamweaver.

WYSIWYG editors approximate the appearance of a web page. As each browser brand and version displays a page a little differently, you will still need to test the results in each of the browsers that you need to support.

A sampling of editors:

- Notepad (basic plain text editor supplied with Windows)
- Notepad++ https://notepad-plus-plus.org/ (free)
- Aptana Studio http://www.aptana.com (free open source)
- Microsoft Visual Studio
- Visual Studio Code https://code.visualstudio.com/ (free)
    - See here to setup VSCode for HTML work: http://thisdavej.com/build-an-amazing-html-editor-using-visual-studio-code/
- Adobe Dreamweaver http://www.adobe.com/products/dreamweaver.html (free trial available)

Editor Features

Here are some of the features to expect from HTML editors.

- Color coding to highlight tags, attributes and text.

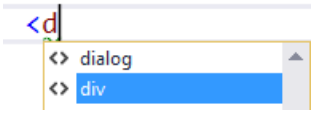
```
<html>
<head>
    <meta charset="utf-8" />
    <title></title>
    <style type="text/css">
        p { color:red; border-st
```

- Color coding or other effects to highlight invalid tags or invalid tag order.

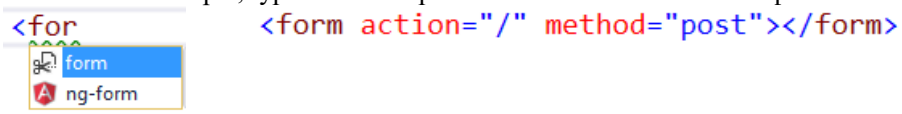
```
<span> <div> test </div> </span>
```

Element 'div' cannot be nested inside element 'span'.

- Auto-complete and lookup features to help select and type tags.



- Snippet and code sample library features to build complete structures in just a few clicks. (In this Visual Studio example, type “for” and press tab twice to write the complete <form> block.)



- Short cuts to open pages in a browser, link to validator sites and publish pages to a web server.

Text, Spaces and Tabs

HTML and CSS can be typed entirely on one line, or neatly formatted across as many lines as needed. Both of the following examples will create the exact same page.

```
<html><head><title>test</title></head><body>Hello world!</body></html>
```

```
<html>
<head>

  <title>test</title>

</head>
<body>

  Hello world!

</body>
</html>
```

You format your HTML and CSS to improve readability for the developers. The browser does not care! As each HTML editor has different default settings, and each developer may have their own preferences, your team may want to establish some guidelines for consistently formatting HTML.

Working with Tags

Terminology

HTML markup uses “tags” to define “elements”. While these two terms are often interchanged, tags are what you type to define what the browser will later create as an element.

- **HTML document** – The text file containing HTML tags and content.
- **Tag** – Tags delimit the start and end of elements.
- **Element** – a component of an HTML document. Elements are defined using tags and typically are composed of a start tag and an end tag. Most elements contain content between the start and end tags in the form of text and/or other elements.

- **Attribute** – Attributes are element modifiers that set element properties.
- **Text** – Any content between start and end tags that is not another tag.
- **Document Object Model (DOM)** – Tags are parsed to create Elements which are then added to the DOM. The DOM is processed to render the displayed output of a page. The DOM is also manipulated by JavaScript to add, remove or change elements.

The markup used in HTML is added as a "tag". Each tag is in the form of:

- **A starting bracket** (less than symbol). "<"
- **A name**. Examples: button, img, a, style
- **Attributes**. Most are optional. Examples: id, style, href, src, height, width
- **A closing bracket** (greater than symbol). ">"
- **Content** in the form of text or other tags. Not all tags have content.
- **A closing tag**. Not all tags have closing tags.
    - A starting bracket (less than symbol). "<"
    - A forward slash "/"
    - A name. The same name as the starting tag.
    - A closing bracket (greater than symbol). ">"

Some tags have content.

Content is the text or tags typed between start and end tags. The start tag's name is enclosed in angle brackets and may include attributes. The end tag includes the tag's name preceded by a slash.

```
<b>this is the content for the bold tag</b>
<div>this is a DIV!</div>
```

Some tags only have text as content.

Any angle brackets typed here will be displayed as brackets and will not be treated as elements.

```
<title>This is a title for the page</title>
```

Some tags have no content.

They render output or serve as a placeholder. For example, the <hr> tag displays a line, or horizontal rule, across the page. The <br> moves the text that follows onto the next line. These tags are self-closing and are sometimes called "void elements" as they have no content. In XHTML these must include an ending slash and that practice still shows up in many HTML pages and editors. While adding the closing slash is not needed, it does no harm.

```
This is above the line<hr>while this is below.
This is not on the same line<br>as this.
This is not on the same line<br />as this.
```

The <hr> tags draws a horizontal rule (line). The <br> adds a line break.

The void elements include: (most often used elements are in bold)

area, base, **br**, col, command, embed, **hr**, **img**, **input**, keygen, **link**, meta, param, source, track, wbr

Some tags work in sets.

Lists and tables are examples of sets of tags that are always used together. The use of any one of the tags by itself is generally invalid and has no practical use.

Lists have an outer tag such as <ol> (ordered (numbered) list) and child tags that represent each item in the list.

```
<ol>
   <li>Cats</li>
   <li>Dogs</li>
   <li>Birds</li>
</ol>
```

1. Cats
2. Dogs
3. Birds

Tables use three tags: <table>, <tr>, <th> and <td> where <tr> represents a row, <th> represents a header cell and <td> represents the data in a cell.

```
<table>
  <tr>
    <th>Fish</th><th>price</th>
  </tr>
  <tr>
    <td>Catfish</td><td>19.95</td>
  </tr>
  <tr>
    <td>Cod</td><td>17.95</td>
  </tr>
  <tr>
    <td>Trout</td><td>24.95</td>
  </tr>
</table>
```

Fish	price
Catfish	19.95
Cod	17.95
Trout	24.95

Block-level vs. Inline Elements

Most HTML elements can be grouped into two big categories, block-level and inline, due to similar characteristics. HTML 5 documentation is moving towards calling these two categories “flow content” and “phrasing content”.

Inline elements:

- Are treated like text and are moved and word wrapped with the surrounding text.

- Do not generate line breaks by default. (But can be styled to do so.) Note that some tags, such as <br>, that start a new line are also considered to be inline tags.
- Generally align with the baseline of the surrounding text. (The bottom of an image will align with the bottom of letters that do not have descenders.)
- Cannot contain block-level elements.

Examples of Inline elements:

…	bold	…	Italic
<u>…</u>	underline	<img>	Image
<a>…	Anchor / hyperlink	…	Emphasized (defaults to italic)
…	Strong (defaults to bold)	<strike>…</strike>	Strike-through
_…	Subscript (H_2O)	[…]	Superscript ($x^2 + y^2$)
…	A container of inline text and tags.		

For more examples of inline elements see:
https://developer.mozilla.org/en-US/docs/Web/HTML/Inline_elements.

Block-level elements:

- Typically generate line breaks both before and after the element. (But can be styled to not do so.)
- Can contain both block-level and inline elements.

Examples of Block-level elements:

…	A generic container.	…	List Item (used in and)
<header> <footer> <article> <section>	HTML 5 context tags that generally behave as a .	<hr>	Horizontal rule (line)
…	Paragraph. Adds additional white space before and after by default.	<blockquote>… </blockquote>	Long quotation (similar to but with both left and right indents by default.

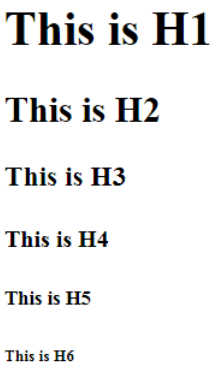
For more examples of Block-level elements see:
https://developer.mozilla.org/en-US/docs/Web/HTML/Block-level_elements

Context vs. Display

HTML was originally designed as a markup language (Hyper Text Markup Language) and the tags were intended to indicated the purpose of the contained content and not how is was to be displayed. The early

browsers added default display characteristics to many of the tags to visually hint as to the purpose of the text. For example, H1 was intended to part of an outline structure and an H1 marked the text that was the top level of the outline. Most browsers styled the H1 tag as large bold text, but each choose their own size and font to use. Notice that the <h6> below is even smaller than the default font size, making it impractical as a header unless it is restyled.

Examples of the default formatting of <h1> through <h6>:



Bold and Italics: <b> vs. and <i> vs.

The <b> and <i> tags are exceptions to the "tags are markup, not presentation" rule. These simply defined the use of **bold** and *italics* styles to display text and add no other importance. and are semantic and imply the context or purpose of the enclosed text. The distinction becomes important when you start thinking about text-to-speech, text-to-brail and other forms of delivery of web content. Also, search engines may choose to ignore <b> and <i> while considering text inside of and to be more important. Browsers render the same as <b> and the same as <i> for default text displays.

Note that both <b> and <i> can be styled using CSS and could be displayed as anything!

For a long article on the differences and proposed best practices see:
http://html5doctor.com/i-b-em-strong-element/

Attributes

As HTML evolved, styling was added by using attributes. The TD tag that represents a single cell in a TABLE has 14 attributes, most of which are now considered obsolete in HTML 5.

```
<td height="20px" width="100px">some text</td>
```

The obsolete attributes have generally been replaced with CSS ("style=" in this example).

```
<td style="height:20px; width:100px">some text</td>
```

Cascading Style Sheets (CSS)

Modern HTML design uses CSS to define the display of a tag. Although CSS can be added directly to the tag, best practice is to store CSS as a block inside of STYLE tags or as a linked file. CSS offers more options for styling HTML than was available from tag attributes.

Example:

```
<p style="color:red; border-style:solid; font-size:12pt">This is a paragraph</p>
<p style="color:red; border-style:solid; font-size:12pt">This is another paragraph</p>
```

Or better, define the CSS once for all <p> tags:

```
<html>
<head>
   <meta charset="utf-8" />
   <title></title>
   <style>
      p { color:red; border-style:solid; font-size:12pt }
   </style>
</head>
<body>
  <p>This is a paragraph</p>
  <p>This is another paragraph</p>
</body
</html>
```

In the example above, note that the styling data is stored separate from the HTML that it modifies. This makes it much easier to change the design of a page without extensive editing, or somewhat risky search and replace.

HTML is not Case Sensitive

HTML tags and their attributes are not case sensitive. The current convention is to type both as lowercase. The tag's content and attribute *values* will be case sensitive when accessed from JavaScript.

What is case sensitive?

- URLs may be case sensitive. While the server's name (www.example.com) is not, the path that follows might be. The server technology controls case sensitivity.
- Query string values may be case sensitive. (http://www.example.com/products?**part=AB123**)
- JavaScript is very case sensitive. "var PartNumber" describes a different variable than "var partnumber".
- XML tags are always case sensitive.
- XML and JSON data is usually case sensitive, or at least case important.
- CSS itself is not case sensitive. Both of the following are equivalent:
  `h1 { font-size:16pt; }  H1 { Font-SIZE:16Pt; }`
- CSS ID and Class selectors ***are case sensitive*** in some browsers. The following are NOT equivalent:
  `.makeItRed { color:red; }  .MakeItRed { color:red; }  .makeitred { color:red; }`
  `<span class="MAKEITRED"…  <span class="makeitRED"…`

Best Practices?

- Current guidance is to type all HTML tags in lowercase.
- Create a naming convention for your team. For example, all IDs and CSS class names should be in camel case. I.e. makeItRed partNumber employeeFirstName
- Know your data! You will need to discover and document any requirements for data in XML or JSON format.

Attributes

Most elements have options that are defined in the start tag as “attributes”. An attribute has a name followed by an equal sign (=) and a text value.

```
<img src="logo.gif" height="100" width="100">
```

While the attribute names are not case sensitive, the **values** of the ID and CLASS attributes are case sensitive when used as CSS selectors.

Attribute Syntax

Attributes typically have a name, an equal sign and a value. Spaces around the equal sign are optional.

- If the attribute’s value is composed of letters and digits, and does not include space characters or the “'”, “"”, “=”, “>”, or “<”characters, then no quotes are required.
  ```
  <p id=para1 style=firstpara>
  ```
- Attribute values can be enclosed in double quotes (") as long as the value does not contain a literal double quote.
  ```
  <img id="pic1" alt="Team logo" src="logo1.gif">
  ```
- Attribute values can be enclosed in single quotes (') as long as the value does not contain a literal single quote.
  ```
  <img id='pic1' alt='Team logo' src='logo1.gif'>
  ```

Single quotes can be used inside of text enclosed inside of double quotes, and vice versa.

```
<img id='pic1' alt='Team logo with "Three circles" ' src='logo1.gif'>
<img id="pic1" alt="Team logo with 'Three circles' " src="logo1.gif">
```

Empty Attribute Syntax

A few attributes are used by simply being present. As an example, an <input> in a <form> can be disabled by just adding the disabled attribute without setting a value.

```
<input type="text" name="city" disabled>
```

You do not need to type “=” and a value. You will often see the following in samples of HTML, all of which are treated by the browser the same as just the attribute alone.

```
disabled=disabled    disabled=""    disabled=true
```

Again… the value is not important! **disabled=false** and **disabled=true** both make the <input> disabled.

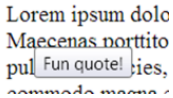
Global Attributes

HTML includes a collection of attributes that are available on every tag. (Or at least on every tag used inside of <body>…</body>.) These include accesskey, class, dir, id, lang, style, tabindex and title. HTML 5 adds contenteditable, contextmenu, data-*, draggable, dropzone, hidden, spellcheck and translate.

The most common global attributes:

- id – An id to uniquely identify an element for JavaScript and CSS. No two elements in the same page should have the same id.
  ```
  <img id="teamlogo" …
  ```
- class – The name or names of CSS classes to be applied to the element.
  ```
  <img id="teamlogo" class="imageWithBorder" …
  ```
- style – An inline style for the element.
  ```
  <img id="teamlogo" style="border-width:1px; border-style:solid;" …
  ```
- title – Additional information about the element. May be displayed as a tool tip and may be indexed by search engines.

```
<blockquote title="Fun quote!">
Lorem ipsum dolor sit amet, con
</blockquote>
```



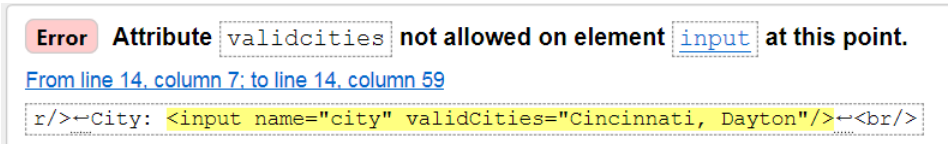
Custom Attributes

You can add your own custom attributes to HTML tags to store data that will be used by JavaScript code.

Guidelines:

- Name your attributes so they will not conflict with future additions to the HTML standards. (Difficult unless you are psychic!)
- Name your attributes using the HTML 5 convention of "data-*somename*".
  ```
  <p data-productid="A123">text</p>
  ```

Custom attributes that are not named using the HTML 5 recommendations will be flagged as errors by most HTML validators and editors.



Comments

HTML commenting is used for two purposes, adding documentation text to a page, and to cause a block of HTML to be ignored by the browser.

```
<!-- uncomment the following code when the site is down for maintenance -->

<!--
    <b>The site is currently down for maintenance. Click <a href="Status.html">here</a>
    for updates.</b>
-->
```

The text within the comment tags:

- must not start with a ">" character or the string "->"
- must not contain the string "--"
- must not end with a "-" character

Note that anyone can read your comments by using the browser's View Source features.

Internet Explorer Conditional Comments

Older versions of Internet Explorer (pre IE10) included their own special comments to detect browser versions and to conditionally include or ignore blocks of HTML. All other browsers ignore these and treat them as ordinary non-conditional comments. While you may still find these in some HTML pages, these are no longer supported in IE 10 and later and are treated as ordinary non-conditional comments. See: https://msdn.microsoft.com/en-us/library/hh801214(v=vs.85).aspx

In the following example, "HTML 5shiv.js" is only being loaded for versions of Internet Explorer before version 9. All other browsers will ignore the script block.

```
<!--[if lt IE 9]>
  <script src="HTML 5shiv.js"></script>
<![endif]-->
```

CSS Comments

CSS comments use "/*" and "*/" to delimit the comment text. CSS comments are used inside of <style> blocks and inside of linked CSS files.

```
/* Add a border around all table elements. */
table tr td { border-width:1px; border-style:solid;  }
```

Non-Standard Tags

Note that invalid tags are not necessarily unsupported tags. Browser vendors have added non-standard tags over the years that are not part of the W3 standards. As an example, Microsoft added <marquee> to display scrolling messages. While not "valid" or a standard, <marquee> is supported by most browsers. A scrolling "marquee" can also be created using CSS3 and as a CSS3 marquee is standards based it is the best practice.

Netscape added <blink> and it was supported in Firefox through version 22. It is no longer supported by any browser.

For a list of "non-conforming" features see: http://www.w3.org/TR/HTML5/obsolete.html#obsolete

Every Page Includes…

You can just type a few HTML tags and text into a file, name it with an .HTM or .HTML extension, and modern browsers will try their best to render it.

Try It!

1. Open Notepad.
2. Type the following text:

```
I'm a <b>web</b> developer!
```

3. Save the file to your lab folder as SimpleHTML.html (C:\HTMLCSS\Labs\Lab1\SimpleHTML.html)
4. Open a browser and type the path to your new file.

But… this is not a valid HTML document!

Every page should include:

DOCTYPE

We will learn more about DOCTYPE shortly, but for now just think of it as a note to the browser (or validator) that specifies the version of HTML that follows. The DOCTYPE for HTML 5 is "<!DOCTYPE html>"

<html></html>

The html tags enclose the HTML of the page. The only thing that should be outside of the html tags is the DOCTYPE tag.

<head></head>

The head tags enclose non-display tags. These include the site title and links to JavaScript and CSS files.

<body></body>

The body tags enclose the displayed portion of the page.

A Simple, but Complete Page

```
<!DOCTYPE html>
<html>
  <head>
     <meta charset="utf-8" />
     <title>Welcome Page</title>
  </head>
  <body>
    Welcome to my site!
  </body
</html>
```

File Extensions

Text files that contain HTML typically have a file extension of .HTM or .HTML. If the HTML file is the result of server side processing, then the file extension may indicate the technology used on the server to generate the page.

Examples:

.htm or .html	These are typically text files that are returned from the server as-is.	The three letter version is from the days of DOS where all files had names with up to eight characters and file types / extensions of up to three characters. MYPAGE.HTM
.asp	Microsoft Active Server Pages ("classic ASP")	Server side scripting.
.aspx	Microsoft .NET Forms Pages	Server side compiled code generated pages using ASP.NET.
.php	Originally PHP was for "Personal Home Page". It is now "PHP: Hypertext Preprocessor".	Server-side scripting language.

Core Page Elements

Note: Many of the elements listed below are included here for completeness and will be covered in detail later in the course. Only the most common elements and attributes will be listed in this course! Please refer to the following resources for more detailed information.

- https://www.w3.org/community/webed/wiki/HTML/Elements
- https://html.spec.whatwg.org/multipage/semantics.html#semantics
- http://www.w3schools.com/tags/default.asp
- https://developer.mozilla.org/en-US/docs/Web/HTML

DOCTYPE

The first line of an HTML file should specify the version of HTML being used by adding a document type declaration, or DOCTYPE.

For HTML 5 and later:

```
<!DOCTYPE html>
```

For older versions of HTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN">
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/REC-html40/strict.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

If the DOCTYPE is not specified, the browser switches to “quirks mode” and “guesses” how to process the page. (See https://en.wikipedia.org/wiki/Quirks_mode)

This course will focus on HTML 5 and the examples will use the HTML 5 DOCTYPE.

Note: While HTML is not case sensitive, XHTML is. For compatibility with XHTML almost all HTML example code will have “DOCTYPE” in all upper case while “html” is in lower case.

Document Structure Elements

The top level of tags used to define an HTML document include:

<html>…</html>

The <html> tag represents the root of the HTML document and, with the exception of the DOCTYPE, encloses all of the content of the document.

<head>…</head>

The <head> tag encloses information about the page, such as Title, metadata about the page, and links to external content such as JavaScript and CSS files. None of the content of the <head> tag is directly displayed in the page output. The <title> tag contents may display in the browser in a tab, a title bar or as a Favorite or Bookmark. While the <head> tag has attributes, they are deprecated in favor of CSS styling.

<body>…</body>

The <body> tag contains all of the tags and text to render the page.

Document Head Elements

The <head> tag has one required child tag, <title>, and several optional child tags. The contents of <head> should only consist of other tags and comments, and not include any direct text. The <head> tag has no attributes.

Most common <head> tags:

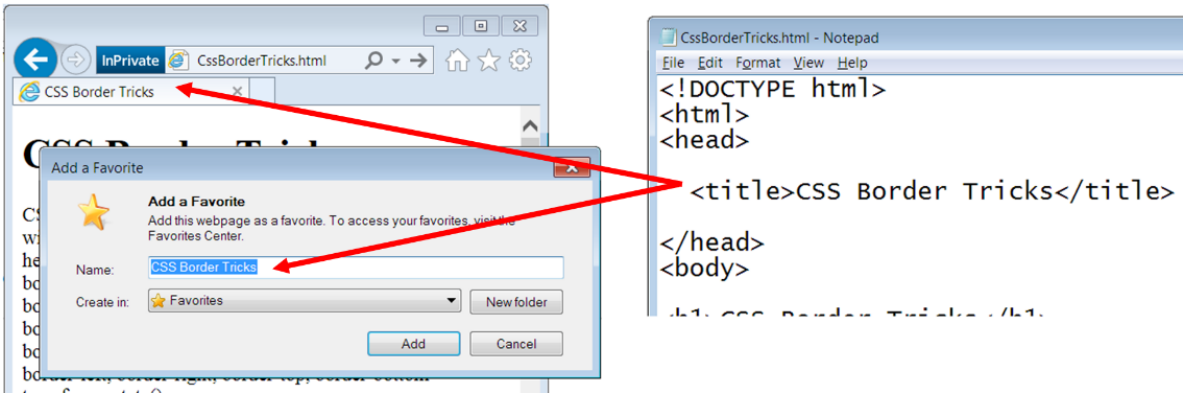
<title>…</title>

The <title> tag is required and can only consist of text characters. <title> cannot contain any other HTML tags.

```
<title>This is a web page!</title>
```

Title text is used for:

- The page name and is displayed in a browser tab or title bar.
- The primary page title displayed by most search engines.
- The default text for browser favorites / bookmarks.



Title is a very important tag!

<style>…</style>

The <style> tag is used to enclose CSS style definitions. (To link to an external CSS file use <link>.)

```
<style>
  h1 { color:green; }
</style>
```

The style tag has a “type” attribute to specify the style language to use. At this time there is only one: CSS. This attribute is optional and defaults to “type="text/css"” if not supplied.

```
<style type="text/css">
  h1 { color:green; }
</style>
```

<link>…</link>

The <link> tag is used to link to external resources such as CSS style sheet files. The <link> tag is a void tag and has no end tag.

```
<link rel="stylesheet" type="text/css" href="MyTheme.css">
```

<meta>

The <meta> tag supplies metadata about the page. This might include keywords to benefit search engines, the character set being used or information for browsers such as "viewport". The <meta> tag is a void tag and has no end tag.

Character Encoding

While most browsers can "sniff" the character encoding being used, a best practice is to identify the encoding. Another best practice to actually encode your file with the same encoding! (Notepad has an option in the SaveAs dialog box to select encoding.) The default encoding for HTML 5 is "UTF-8".

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

Descriptive Metadata

You can supply additional data for search engines by using the author, description and keywords meta tags. The viewer of your page will not see this data.

```
<meta name="author" content="Mike Smith">
<meta name="description" content="Fun toys! Model airplanes and boats.">
<meta name="language" content="en-us">
<meta name="keywords" content="RC,Radio Control,airplane,boat">
```

The use of the "keywords" meta tag for search engine optimization (SEO) is open to debate. Keywords have been so misused that most of the search engines either ignore them or consider them as a negative. If you do use this meta tag, list only a few very relevant keywords, and be honest!

Robots

The "robots" meta name is to request that search engines "index" or "noindex" your site, and to "follow" or "nofollow" links from your site to other sites. Keep in mind that only "nice" robots will honor your request. Also see "robots.txt": http://www.robotstxt.org/

```
<meta name="robots" content="index,nofollow">
```

Viewport

Due to the smaller screens of most mobile devices, their browsers render a page in a virtual screen, the Viewport, that's often larger than the physical screen. Viewport only applies to mobile devices and browsers.

W3schools.com recommends the following as a default viewport meta tag:

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

This will set the default page width to match the screen width, and an initial zoom of 100%.

As an alternative, you can use the CSS viewport rule, and it may be a best practice for now to use both.

```
@viewport{
   zoom: 1.0;
   width: device-width;
}
```

<script>

The <script> tag is used to either embed JavaScript code in the page, or to link to a JavaScript file. The <script> tag can occur in the <head> or <body> sections and can occur more than once.

```
<script type="text/javascript">  JavaScript code here  </script>
<script type="text/javascript" src="scripts/MyJavaScriptLibrary.js"></script>

<script>  JavaScript code here  </script>
<script src="scripts/MyJavaScriptLibrary.js"></script>
```

Attributes:

- The “type” attribute is optional in HTML 5, but required in HTML 4.01 and earlier. (So go ahead and add it.) All browsers will default to “test/javascript” if “type” is not defined.
- “async” and “defer” (only for external scripts. i.e. when the “src” attribute is present.)
  When is the JavaScript executed?
    - If the “async” attribute is present, then the script is executed asynchronously with the rest of the page. (I.e. it may, or may not, be run before all page elements have been loaded.)
    - If the “defer” attribute is present, then the script is executed after the page is finished parsing.
    - If neither “async” or “defer” are present then the script is executed immediately, before the rest of the page is processed.
    - Inline JavaScript is executed immediately.

<base>

The <base> tag is used to specify the default URL for all links on a page that have relative URLs. If used, this tag should be the first tag in the <head> section. The <base> tag is a void tag and has no end tag.

Attributes: The <base> tag has two attributes, href and target.

- href is the URL to be used for any relative links in the page. In the following example, the “logo.gif” file will be loaded from http://www.example.com/images/logo.gif.

- target is the destination for the linked content and includes a context name, _blank, _self, _parent, or _top.

```
<head>
  <base href="http://www.example.com/images">
</head>
<body>
   <img src="logo.gif" >
</body>
```

Nesting

Many elements can be nested inside of other elements. Every HTML element is nested inside of the <html> tag. The following example nests italics tags inside of bold tags.

```
<b>This <i>is</i> a web page.</b>
```

Many of the elements that cannot be nested make sense… you cannot nest <b> inside of <b>. The nesting rules for other elements will make sense once you learn more about their uses. As an example you cannot nest a (division) inside of a <p> (paragraph).

Testing HTML

The best advice to a new web developer is "test, test and test". There's no end to browser incompatibilities!

- Test in every web browser that might be used: Internet Explorer, Firefox, Safari, Chrome, etc.
- Test in every device that might be used: desktops, large monitors, small tablets, phones, etc. (A page that works in "browser X" may look fine on a desktop, not display well on a phone, and be unusable on a touchscreen tablet.)
- How does the page work for a vision impaired user?

Browser Compatibility

There are a number of web sites that try to document each HTML and CSS feature's support in each vendor's browser. "Try" is because the browsers change as often as weekly!

- http://caniuse.com Search for just about any HTML 5 or CSS 3 feature to see compatibility by browser version. (Click the Show All button for a more detailed version history.)
- http://html5test.com Use this site to test your browser or compare browser features. (Click the links or arrows for more details.)

Validators

Many web pages that work are not necessarily well designed or use valid HTML. Browsers are very tolerant of incorrect HTML and may still render a page correctly.

Examples of invalid HTML that works, *sometimes*:

- Pages that do not include a DOCTYPE.
- Unknown (and misspelled) tags are ignored, but their content is still displayed. For example "<flashing>" is not a real tag, any text between the start and end tags will still be rendered. "<flashing>hello</flashing>" will just display "hello".

Tools to validate your HTML:

- Your editor. Most HTML editors will validate as you type.
- Use an HTML validation web site such as:
  - http://validator.w3.org
  - https://html5.validator.nu
- Use a browser add-in such as the this one for Firefox: https://addons.mozilla.org/en-US/firefox/addon/html-validator/

Tools to validate CSS:

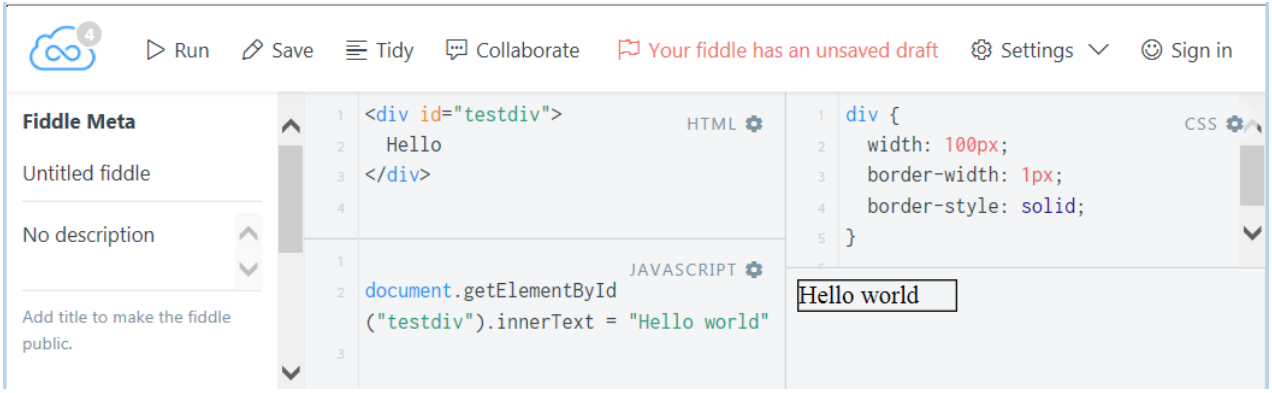
- http://jigsaw.w3.org/css-validator/

Even if you write 100% valid HTML, it still may not work in all browsers, and may not represent good page design.

Testing Tools

JSFiddle

The web site, jsfiddle.net, is an excellent place to both test fragments of HTML, CSS and JavaScript, and to share examples and demos. (If you would like to see and play with the demo below, just go to the saved copy at https://jsfiddle.net/svvozw4a/.



Other sites:

- The W3Schools "Try It" page
  http://www.w3schools.com/html/tryit.asp?filename=tryhtml_default

- The W3Schools CSS Selector Tester
  http://www.w3schools.com/cssref/trysel.asp
- CSSDesk
  http://www.cssdesk.com/
- JavaScript Tester
  http://www.webtoolkitonline.com/javascript-tester.html
- Regular Expression tester
  https://regex101.com/

Browser and Device Testing

The best test for your HTML pages are the real world browsers. Find out who your target audience is, and which browsers they use. If you are working in a corporate environment where the browser and version are set by IT, then life is easy! Otherwise, you will need to get access to as many browsers, browser versions and devices as possible. Your team should define the minimum list of browsers must be supported. For example you might target your work for: IE 10 and 11, Edge, Firefox 48 and later, Chrome 40 and later, Safari 9 and later. Your team may need to own an assortment of the supported devices, or have acquired emulators.

Browser Emulators

There are downloadable and online emulators for most browsers and devices. Do a web search for "web browser emulator" to see what's currently available.

A sampling of Online Emulators:

- Turbo.Net Browser Sandbox https://turbo.net/browsers
- BrowserStack (includes Mobile Browser Emulators) https://www.browserstack.com/
- http://www.mobilephoneemulator.com/

Screen Readers

The blind and vision impaired use screen reader programs to read the text of a screen or a page using a computerized voice. You can use the tools below to ensure text, links, graphics etc. can be "read" by a blind person.

- http://www.webanywhere.cs.washington.edu/
- http://webaim.org/articles/voiceover/ (for Macintoshes)
- http://webaim.org/articles/nvda/ and http://www.nvaccess.org/ (A downloadable screen reader for Windows.)

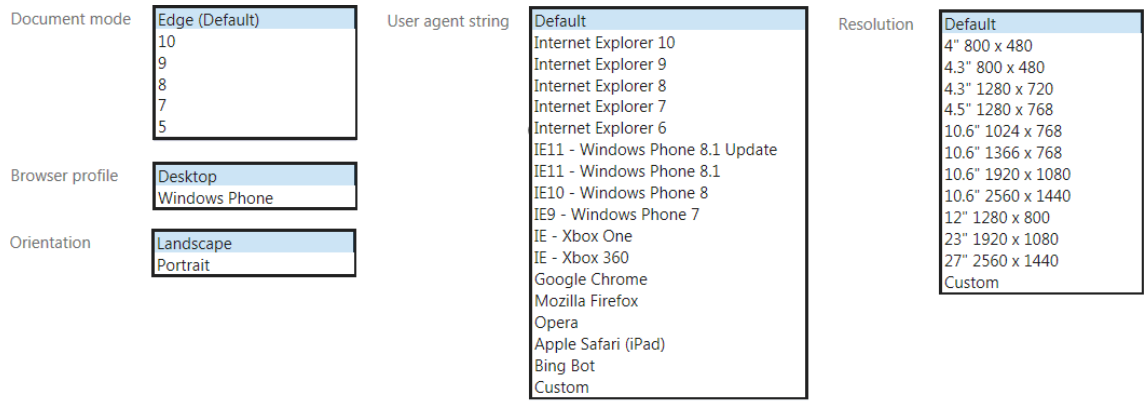
To learn more about designing sites to work with screen readers see:

- http://webaim.org/techniques/screenreader/

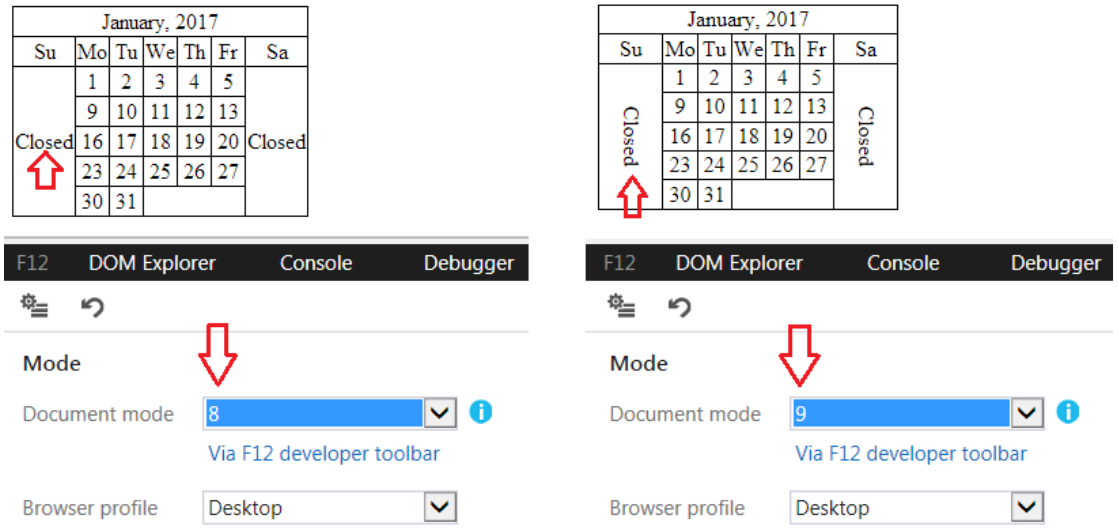
Internet Explorer F12 Tools – Emulation

Microsoft Internet Explorer includes a collection of developer and testing tools that are available from the browser's F12 key or menus. The emulation feature lets you select a browser version, a device type (only desktop and Windows Phone) and a resolution. Firefox and Chrome also have similar F12 developer tools.

A few of the options:

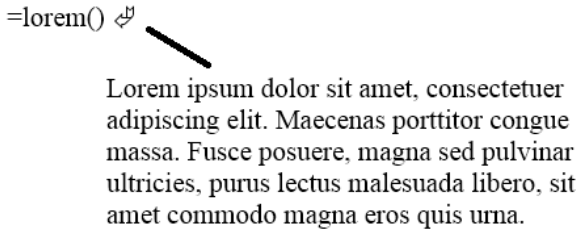


As an example, the CSS rotated text transform below did not work in IE 8 emulation, but did work in IE 9 and later.



Sample Text

When typesetters needed sample text to demonstrate the layout of an ad or flyer they would use seemingly random Latin text that starts out with "Lorem ipsum". You can generate that text by using a little known Microsoft Word function. Open Word and at the start of a blank line just type =lorem() and press enter!



You can supply two numbers, one for the number of paragraphs and one for number of lines, to create any amount of text.

```
=lorem(10,5)
```

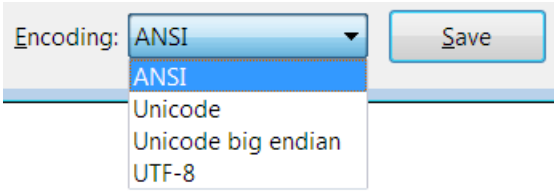
If you would prefer English text then use **=rand()** instead of **=lorem()** to generate text from the Word help files. To learn more about the history of "lorem ipsum" see: https://en.wikipedia.org/wiki/Lorem_ipsum

HTML and Text

Typing text into web pages sounds like the easy part of web development, that is until you create the wrong kind of file, or want enter something like ☺, ™, ©, or a non-breaking space.

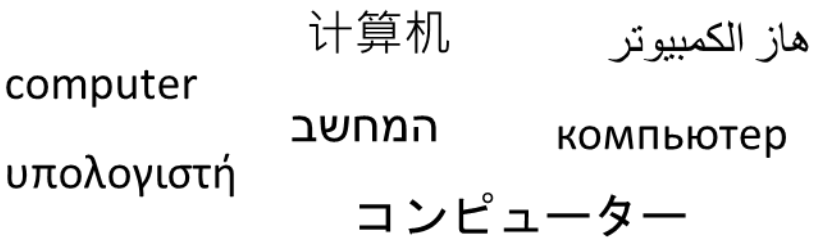
File Encoding

Text files seem pretty simple, just a list of characters. When you save a text file from Notepad you can select an "Encoding" to set how characters are stored in the file.



One of the first character sets was ASCII, which included 127 characters. ASCII basically included A-Z, a-z, 0-9 and some symbols. It did not include the characters needed for "résumé", "¥" or "£" or the characters used by most of the languages in use today. The default encoding used by Notepad is ANSI, which includes 256 characters, and supports "résumé", "¥" or "£".

Modern web development requires the ability to support all of the characters and symbols in the world. Unicode represents more than 128,000 characters. Unicode is mapped to various encodings, with UTF-8 being the most common on the web.



- The default character encoding for HTML 5 is UTF-8.
- The default character encoding for HTML 4 is ISO-8859-1. (Similar to ANSI)

When creating a web page you can alert the browser to which character encoding is being used with a <meta> tag.

- For HTML 5:
  ```
  <meta charset="UTF-8">
  ```
- For HTML 4:
  ```
  <meta http-equiv="Content-Type" content="text/html;charset=ISO-8859-1">
  ```

Tip! If you specify an encoding (via a meta tag or a DOCTYPE), remember to save your files in that format!!!

HTML Entities

How do you type "résumé" or "¥" when your keyboard does not have those characters? How do you display a "<" when HTML thinks that is the start of a tag? You might find a keyboard shortcut or multi-key combination in some editors or operating systems. But for most of the special characters you will need to know how to code those using HTML entities.

HTML entities can be entered using one of the three techniques:

- Decimal numbers &#*nnnn*;
- Hexadecimal numbers: &#x*hhhh*;
- HTML entity name: &*name*;

Decimal and hexadecimal can be used for all characters while entity names can only be used for 252 different characters.

Examples for the copyright symbol (©):

```
&#169;   &#x00A9;   &copy;
```

Some of the more common HTML entities that are used to avoid conflicts with HTML special characters:

Less-than sign	<	<
Greater-than sign	>	>
Ampersand	&	&
Double quote	"	"
Single quote / apostrophe	'	'

Some of the more common HTML entities that are used to select characters not found on many keyboards:

Non-breaking space		
Copyright	©	&copy;
Registered trademark	®	&reg;
Trademark	™	&trade
Cent	¢	&cent;
Pound	£	&pound;
Yen	¥	&yen;
Fraction 1/4	¼	&frac14;
Fraction 1/2	½	&frac12;
Fraction 3/4	¾	&frac34;

Fractions:

You can create any fraction using subscripts, a "fraction slash" and subscripts.
For example to create $^{7}/_{8}$ use:

```
<sup>7</sup>&frasl;<sub>8</sub>
```

While this will create the fraction, it will not be scaled the same as the &fracxx; entities. To get closer to the same size as the current font you will need to "tweak" the size a bit:

```
<span style="font-size:70%"><sup>7</sup>&frasl;<sub>8</sub></span>
```

Or you can get even a bit closer by creating your own subscripts and superscripts using and CSS:

```
<style>
  .fractionsup {font-size: 55%; vertical-align: 50%;}
  .fractionsub {font-size: 55%;}
</style>

<span class="fractionsup">7</span>&frasl;<span class="fractionsub">8</span>
```

This is three quarters (¾) using ¾

This is three quarters ($^{3}/_{4}$) using ⁄ and sup and sub.

This is three quarters ($^{3}⁄_{4}$) using ⁄ with spans and CSS.

See here for symbol entities: http://www.w3schools.com/charsets/ref_utf_geometric.asp

🗁 **Sample file: CssBorderTricks.html** (The above entities are listed at the end of this page.)

White space

In HTML, with only a few exceptions, all white space rolls into a single space. “Hello world” will render the same as “Hello world” and “Hello *tab* world”. White space is considered to be spaces, carriage returns and tabs.

You do have a few options to manage white space: non-breaking space entities, the <br> tag, the <pre> tag, transparent images and of course CSS!

 Line Break

Line breaks in HTML source code are treated as white space and do not generate line breaks in the browser output. Both of the following produce the same output:

```
This is a line of text.

This is a
line of text.
```

The <br> tag adds a line break to the output. Unlike the <p> (paragraph) tag, the <br> does not add any extra white space before or after the <br>.

The <br> tag is a void tag and has no content or closing tag. Prior to HTML 5 the tag was written as a self-closing tag (<br />). Starting with HTML 5 you can just type <br>. All browsers will accept <br>, <br/> or <br />. The Visual Studio HTML editor still replaces the <br> you typed with <br />.

Non-breaking space entity

A space that looks like a space, but is not used for line breaks is a non-breaking space. In Microsoft Word you would press Ctrl+Shift+space to enter a non-breaking space. HTML entities are special characters written using and “&”, a name or number and a “;”. The HTML non-breaking space is “ ”. As an example, “HTML 5” would look better if the “HTML” and the “5” are always on the same line. In a web page we would then type: HTML 5

Using “HTML 5”

When writing HTML 5 we want to keep “HTML” and “5” on the same line.

Using “HTML 5”

When writing HTML 5 we want to keep “HTML” and “5” on the same line.

<pre>…</pre>

You can use the <pre> tag to enclose text where you need to keep white space formatting. <pre> is most often used for displays of source code. (The “<” and “>” symbols will still need to be escaped. (“<” and “>”) <pre> always displays using a fixed space font.

```
<pre>This line      has     extra white space
and a carrage return.</pre>
```

```
This line      has     extra white space
and a carrage return.
```

CSS “white-space”

The CSS “white-space” feature can be used to control how white spaces and carriage returns are treated. “white-space” will not change the font.

```
<p style="white-space:pre">This line     has     extra white space
and a carriage return.</p>
```

Using <pre>

```
This line      has     extra white space
and a carriage return.
```

Using <p> styled with white-space:pre

This line has extra white space
and a carriage return.

“white-space” options:

normal	This is the default. Normal text wrap and space handling.
nowrap	Whitespace will be collapsed into a single space, and text will not wrap to the next line.
pre-line	Whitespace will be collapsed into a single space and text will wrap to the next line as needed and where there are carriage returns.
pre-wrap	Whitespace will not be collapsed, but text will wrap to the next line as needed and where there are carriage returns.

One Pixel images

Where precise spacing is needed, and you cannot achieve it with CSS options, you can create images of one or more pixels to fill the space. For example, our company design standards requires 10 pixels between “Example” and “Corp.”. We can achieve this with a 10px wide image or a 1px image scaled to 10px. While a black pixel was used in the example below, you would either use a pixel with the same color as the background or a transparent pixel.

```
Welcome to Example<img src="1pixel.gif" height="1" width="10">Corp.
```

Welcome to Example_Corp.

Tip: Scaled one pixel images can be used for all kinds of interesting effects. How about a bar chart?

```
Product 1 <img src="1pixel.gif" height="8" width="100"><br>
Product 2 <img src="1pixel.gif" height="8" width="150"><br>
Product 3 <img src="1pixel.gif" height="8" width="75"><br>
Product 4 <img src="1pixel.gif" height="8" width="110"><br>
```

Product 1
Product 2
Product 3
Product 4

Module 3: Cascading Style Sheets

You have already seen CSS in use in the earlier modules of this course. Here we dive in a bit deeper! More CSS will also be found later modules as we look at tables, images and other features.

Note: While we refer to “CSS” here, we are generally discussing “CSS3”.

Before CSS

The first versions of HTML controlled the display of HTML tags two ways:

- The defaults selected by each browser manufacture, and they rarely agreed as to size, font or placement.
- The use of tag attributes such as height, width and color.

Tag attributes had a number of problems:

- Colors, sizes and fonts were hardcoded into each tag. If you wanted to change the size of all of the <h2> tags you would have to edit every <h2> in every page of the size. While search and replace is great, it can do some unexpected things. If you just wanted to change a set of tags, such as all tags with product descriptions, search and replace would not usually be practical and would require a manual review of each change.
- New features could not be added without changing the specification of each tag that used the feature.

With CSS

You can create CSS for:

- Every tag of a certain type. For example, change the size of all H1’s to 30px.
- Every tag that shares a class name. (<h2 class="newProduct">) For example, change all text about new products to italic.
- Every set of tags with the same pattern. For example, make text in every <p> tag that’s in a with a class of “newProduct” a different color.

Cascading Style Sheets (CSS)

CSS is a style sheet language used to define the presentation of HTML elements. While HTML tags are intended to define the context or purpose of their content, there were not intended to define the presentation or appearance of the content. As an example, <h1> defines the content to be the top level heading of an outline.

The default presentation of an <h1> or <h2> will vary by browser, but typically will look like this:

```
<h1>A Toy Proposal</h1>
<h2>Section 1 - Description</h2>
<p>This proposal is for a new kind of robot ...</p>
```

A Toy Proposal

Section 1 - Description

This proposal is for a new kind of robot ...

With a little bit of CSS we can change the appearance of the design. As CSS is typically stored in a <style> block or a linked file, we can easily change the CSS, and therefore the design of the page, without editing all of the HTML.

```
h1 {
  border-width:1px; border-style:solid;
  background-color:lightgray;
  font-family:Arial; font-size:24px;
}
h2 {
  font-style:italic;
  font-family:"Comic Sans MS"; font-size:18px;
  margin-left:24px;
}
p {
  font-family:Arial; font-size:14px;
  margin-left:24px;
}
```

A Toy Proposal

Section 1 - Description

This proposal is for a new kind of robot ...

A search engine will still see the H1 and H2 tags and make some guesses about the importance of each section of text.

A CSS Demonstration

For an example of CSS in action, visit the CSS Zen Garden (http://www.csszengarden.com/).

1. View the source of the page and note that all content is either in DIVs or the new HTML 5 DIV-like tags (header, footer, section, etc.).

2. Now find and click the links for the other designs. Again, view the source of the page and note that the only thing that has changed is the linked CSS file. Major look and feel design changes with just a change of CSS! (The JavaScript in the page is only being used to manage the loading of fonts. All of the cool stuff is from pure CSS!)
3. Resize your browser and note if the design adapts to the display size.
4. Access the CSS Zen Garden from your cell phone or tablet and see if the design still works!

Adding CSS to a Page

CSS is added in one of three ways:

- Inline CSS – Added directly to each element. This is the least flexible approach. The following will change the font size for this one <p> tag only.

```
<p style="font-size:24px">
```

- As a style block in the <head> section of the page. The following will change every <p> tag in this page.

```
<html>
 <head>
  <title>Test Page</title>
  <style>
    p { font-size:24px; }
  </style>
  …
```

- As a linked stylesheet. An edit to mystyles.css will change the <p> tag in every page in the site that links to this file.

```
<html>
 <head>
  <title>Test Page</title>
  <link rel="stylesheet" type="text/css" href="mystyles.css">
  …
```

Order of CSS Processing

CSS is processed in the order that it is found, with the exception of inline CSS which is always applied last.

1. The styles in the <head> section are applied first. If there are multiple <link> or <style> blocks in the <head> section, then they are applied top down. Each overriding the previous as needed.
2. The <style> blocks in the <body> section are applied in the order they are found. They will be applied to a tag even if the tag is listed before the <style> block.
3. Finally, the CSS in a tag's style attribute (an inline style) will be applied.

```
<h1 style="color:red">Hello!</h1>
```

CSS style sheets can force an override of an inline style by adding "!important" to the definition.

```
h1 { color:green !important; }
```

"!important" will also override any styles defined after the current style. In the following sample, the final font size is 24px. Any thing marked as "!important" will not be overwritten by anything that follows, unless it has an "!important" added. (The last "!important" wins!)

```
font-size: 24px !important;
font-size: 26px;
```

Experimental Vendor Prefixes

CSS is constantly evolving. New CSS features are first defined as proposals. Browser vendors often release experimental versions of these new proposed features that are named with a vendor prefix. When they are ready to release the final CSS feature, it is done with the proper CSS feature name, which may have even changed from the original proposal.

As an example, consider the "display:flex" feature. Over time, and across browser vendors, it's had several names: box, flexbox and flex. When Safari first released it they named it "-webkit-box" while Microsoft named it "-ms-flexbox". In order to use this feature in as many browsers as possible you will need to list each known experimental version.

- The "-webkit-box" version works with iOS6 and Safari 3.1-6.
- The "-ms-flexbox" works with IE 10.
- The "-webkit-flex" works with Safari 6.1+ and iOS 7.1+.
- The "flex" version works with the newest versions of most browsers.

How do you deal with all of this? You could just list them all, with the final standard listed last.

```
display: -webkit-box;
display: -ms-flexbox;
display: -webkit-flex;
display: flex;
```

You could check with a compatibility site like http://caniuse.com to see which browsers support which versions and add only the prefixes you need to support.

You could use a tool like Autoprefixer (https://github.com/postcss/autoprefixer) that post-processes your CSS and adds prefixes as needed.

In any case, test, test and test again!

The prefixes:

- -ms-
  Microsoft browsers
- -webkit-
  Browsers based on the Web Kit Open Source Web Browser Engine such as iOS and Safari.
- -moz-
  Browsers based on Mozilla, such as Firefox.
- -o-
  Opera

CSS Units

When CSS is used to change the size of an element it uses one of several predefined length units. These units are either absolute (10 pixels) or relative (150%).

Absolute Units

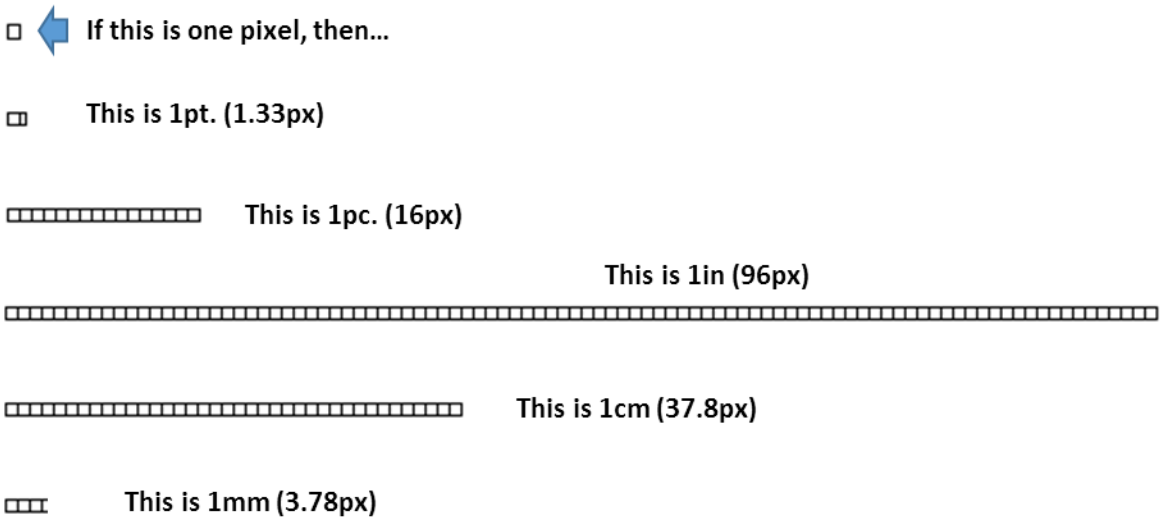
While the units listed below appear to be what you would find on rulers, they are all derived from a pixel. A pixel may match up to a single pixel or dot on a screen, but will vary based on the device and user option to scale or zoom a screen. So… a CSS pixel is not always a screen pixel.

Which unit should you use? Whichever one you like! Inches, centimeters and millimeters are convenient when setting sizes for print media (8.5" x 11" paper) while points are useful for text.

- px pixels (originally a dot on a screen)
- pt points (a typesetting unit)
- pc picas
- in inches
- cm centimeters
- mm millimeters

Here's how they are related:

- 1px = 1/96th of 1in
- 1pt = 1/72 of 1in = 1.33px
- 1pc = 12 pt
- 1in = 96px = 2.54cm
- 1cm = 10 millimeters = 37.8px
- 1mm = 3.78px



Relative Units

Relative units make it easy resize an entire page or section of a page as all you need to change is the "base" object's size and all of the child objects using relative units will adjust to match.

Relative to what?

To understand the relative units you will need to know that each one is relative to.

em	Relative to the font-size of the parent element. Pronounced "m". For example, consider a with CSS to set the font-size to 12pt. Setting the font-size of a <p> tag that's inside of that to 2em would set the <p> tag's font to 24pt.
rem	Relative to the font-size of the "root element" (Technically the <html> element, although the base font size is most often set on <body>.)
%	Percent. Relative to the font-size of the parent element. For example, consider a with CSS to set the font-size to 12pt. Setting the font-size of a <p> tag that's inside of that to 150% would set the <p> tag's font to 18pt.
ex	Relative the x-height of the parent element font. *(rarely used)*
ch	Relative to the width of the font's zero. (0) *(rarely used)*
vw	Relative to 1/100th of the width of the viewport. The viewport is the displayed or visible portion of a page. If you set the width of an object to 100vw then it will fill the width of the browser. (Viewport is especially useful when working with mobile devices.)
vh	Relative to 1/100th of the height of the viewport.
vmin	Relative to 1/100th of viewport's smaller dimension.
vmax	Relative to 1/100th of viewport's larger dimension. (not currently supported in IE11 or Edge)

CSS Selectors

CSS uses "rules" to select the tags and text it will impact. These rules are based on selectors:

- **Element** – Selects all elements with that tag name. In this example, "h1" is the selector and will select all <h1> tags.
```
h1 { color:red; }
```
- **Class** – Classes are named sets of CSS. In this example "productTitle" is the class name and will select all tags with the attribute "style='productTitle"'.
```
.productTitle { color:red; }
```
- **ID** – Select the element with that ID. In this example the one element with the ID of "HelpLink" with be selected.
```
#HelpLink { color:red; }
```
- ***** (asterisk) – Wildcard selects all elements.

Tip: To experiment with CSS selectors, visit the W3Schools "Try CSS Selector" page at http://www.w3schools.com/cssref/trysel.asp

Grouping Selectors

Selectors can be grouped to allow selection of just about anything on the page.

- **element, element** – (with commas) Selects all of the listed elements. The following changes the color of heading tags.
  ```
  h1,h2,h3,h4,h5,h5 { color: red; }
  ```

- **element1 element2** – (no commas) Selects all element2's inside of element1. (Does not select element1.)
  ```
  div p   (selects all paragraphs in all divs.)
  ```

- **element1>element2** – Selects all element2's who's parent is element1. (Does not select element1 or children of element2.) I.e., the element2' between element1's start and end tags
  ```
  div>p
  → <div><p>this is red</p> <p>this is red</p></div> <p>this is not red</p>
  ```

- **element1+element2** – Selects all element2's that are immediate after element1.
  ```
  div+p { color:red; }
  → <div><p>this is not red</p></div> <p>this is red</p> <p>this is not red</p>
  ```

- **element1~element2** – (CSS3) Both elements must have the same parent, but element2 does not have to be immediately preceded by element1. The following example selects <i> tags that follow a <p> as long as they are both inside of the same parent tag.
  ```
  p~i { color:red; }
  → <div><p>some text</p> more text <i>this is red</i> </div> this is <i>not red</i>
  ```

Attribute Selectors

CSS can select tags based on the existence of an attribute or attribute with a specific value. Attribute selectors are always enclosed in square brackets.

- [*attribute*] – Select all tags with that attribute. The following example selects all tags with a "src" attribute.
  ```
  [src]
  → <img src="logo.png">
  ```

- [*attribute*=*value*] – Selects all tags with that attribute and value. The following example selects all image tags that use the help.png image file.
  ```
  [src="help.png"]
  → <img src="help.png">
  ```

- [*attribute*^=*value*] – (CSS3) Selects all tags with that attribute and starts with that value. The following example selects all image tags that use an image file that starts with "help".
  ```
  [src^="help"]
  → <img src="helpindex.png"> and <img src="helpicon.png">
  ```
  [*attribute*$=*value*] – (CSS3) Selects all tags with that attribute and ends with that value. The following example selects

all image tags that use an image file name that ends with ".png".

```
[src$=".png"]
→ <img src="helpindex.png"> and <img src="helpicon.png">
```

- [*attribute**=*value*] – (CSS3) Selects all tags with that attribute and contains that value. The following example selects all image tags that use an image file name that contains "index".

```
[src*="index"]
→ <img src="helpindex.png"> and <img src="documentindexicon.png">
```

Pseudo-class Selectors

These selectors work on elements based on their state instead of their name or values. As there are a large number of these, we will only review a few of them here. More are covered with the elements they are frequently used with.

For a more complete list see http://www.w3schools.com/cssref/css_selectors.asp and especially note the CSS version number column. While some of the Pseudo-class selectors were available in CSS1 and CSS2, most were introduced with CSS3. Before using any of the newer class names check http://caniuse.com/#search=pseudo for browser compatibility!

Pseudo-class Selectors for the Anchor tag

As an example of pseudo-classes, consider the <a> tag. It has one appearance when displayed, another while the mouse is over the link, and another after the link has been visited. To change the appearance of the three <a> states use these Pseudo-classes:

- Unvisited link:

```
a:link { color:blue; }
```

- Visited link:

```
a:visited {color:red; }
```

- Mouse is over the link: (The a:hover definition must follow a:link and a:visited.)

```
a:hover { color:black; }
```

- Active link:

```
a:active { color:green; }
```

The above can be combined with other CSS selectors as needed:

- To style the following link:

```
<a class="productlink" href="123.html">Product 123</a>
```

use:

```
a.productlink:link { color:blue; }
```

Media Queries

CSS can define different options for different devices and screen resolutions. Remembering that CSS "cascades" you will first define your default or most common CSS for the page. You can then "cascade" alternate CSS as needed to target different devices and resolutions.

To define the alternate CSS you will write rules that define the properties of the device. As an example, when a page is printed you may want to remove the background color.

```
@media print {
  body { background-color:white; }
}
```

Media queries can also be used in <link> tags to specify when to use the linked stylesheet.

```
<link rel="stylesheet" media="print" href="print.css">
```

Media Types

- print – Used for printers.
- screen – Used for computer and device screens.
- speech – Used for screen reader applications. (You may want to hide some content while showing alternate content, or maybe reorder the display of DIVs.)

Many of the media types have been deprecated with HTML 5. These include aural, braille, embossed, handheld and projection.

Media Features

Media features are used to create rules based on the number of colors supported and screen resolution. A CSS block or link is used when the media query is true.

Examples:

- Use when the display width does not exceed 400px.
  ```
  @media (max-width: 400px) { … }
  ```
- Use when the display width equals or exceeds 800px and device is in landscape mode.
  ```
  @media (min-width: 800px and (orientation: landscape) { … }
  ```

For a complete list of the media features see:

- https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries
- http://www.w3schools.com/cssref/css3_pr_mediaquery.asp

A Sample CSS File with Media Queries

```
/* CSS common the all devices and resolutions */
body { background-color:lightgray; }
h1 { color:red; font-size:24pt; }
h2 { color:blue; font-size:20pt; }
#top { color:blue; text-align:center; }
header { width:100%; font-size:36px; text-align:center;  }
#left { width:25%; float:left; }
article { width:75%; float:right; background-color:white; }
footer { border:1px solid; clear:both; text-align:center; }

/* hide the left nav for small screens */
@media (max-width: 400px)
{
  h1 { font-size:18pt; }
  h2 { font-size:16pt; }
  #left { display:none; }
  article { width:100%; float:none; }
}

@media print {
  #top { display:none; }
  #left { display:none; }
  article { width:100%; float:none; }
}
```

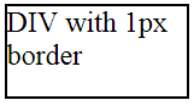
The CSS Box Model

CSS can be used to add borders, margins and padding around most displayed objects.

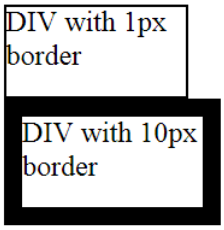
- Height and Width set the size of the content box.
- Border thickness and padding push text away from the edge of the box and are added to the outside of the box.
- Margins push the object away from its neighbors and is added outside of the box.
- Total width consumed on the page for the box is width + padding + border + margin.
- Margins, borders and padding can be individually set for the left, right, top and bottom of the object.
- Borders and (content + padding) can have background colors or images. Margins cannot.

🗁 **Sample file: BoxModel.html**

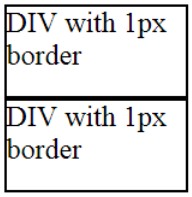
The following DIV has a width of 100px, a height of 50px and a border of 1px. Total width is 102px.



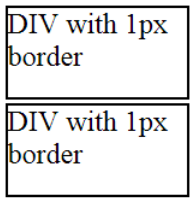
Total width of the second box is content width (100px) plus border (2*10px).



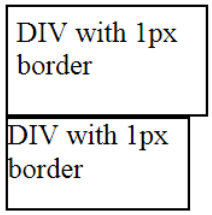
Two DIVs with no margin.



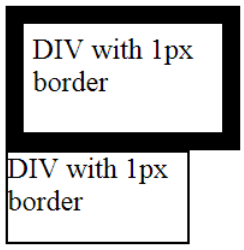
Top DIV has a 2px bottom margin. Total height is 50px plus border (2*1px) plus 2px bottom margin, or 54px.



Top DIV has a 5px padding. Total width is content width (100px) plus border (2*1px) plus padding (2*5px), or 112px.

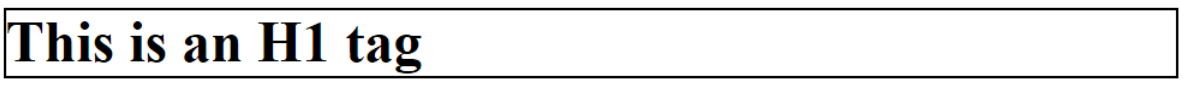


Top DIV has a 10px border and 5px padding. Total width is content width (100px) plus border (2*10px) plus padding (2*5px).



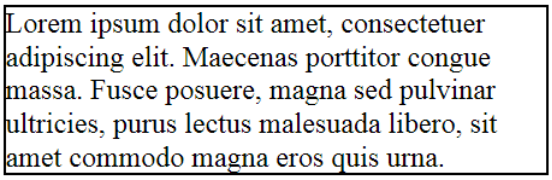
Paragraphs, spans and most other tags can be formatted using the CSS box model.

H1 with a 1px border. By default the width is 100%.

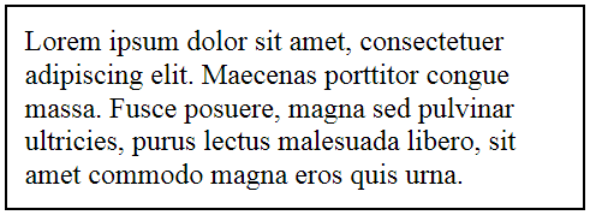


H1 with a 1px border and a width of 300px.

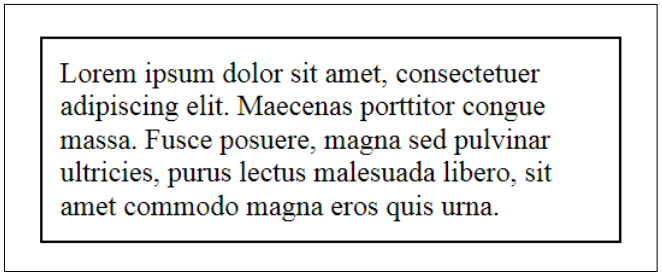
This is a P tag with a 1px border and a width of 300px.



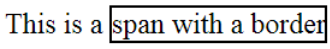
This is a P tag with a 1px border, a width of 300px and a padding of 10px.



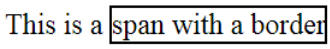
This is a P tag with a 1px border, a width of 300px, a padding of 10px and a margin of 20px



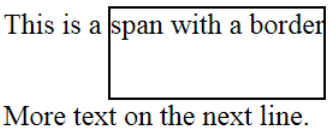
This has a span with no width or height set, but has a border of 1px.



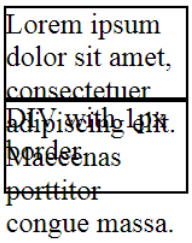
This has a span with no width set, but has a height set to 50px. Height and width are ignored on inline objects by default.



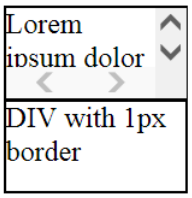
This has a span with no width set, but has a height set to 50px and "display:inline-block". Height and width are ignored on inline objects by default, but are honored on items displayed as a block.



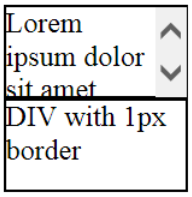
This DIV has a height of 50px and a width of 100px, and too much text, which flows out of the bottom overwriting anything below it.



This DIV has a height of 50px and a width of 100px, and too much text, but also has "overflow:scroll;" and scrollbars.



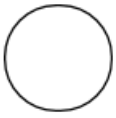
This DIV has a height of 50px and a width of 100px, and too much text, but also has "overflow-x: hidden; overflow-y: scroll;" and scrollbars.



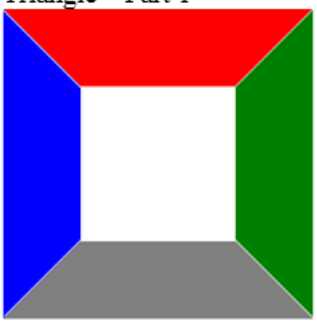
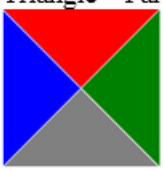
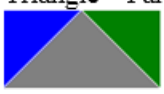
CSS Border Tricks!

CSS can be used to create several common shapes from a simple .

🗁 **Sample file: CssBorderTricks.html**

Square 	Just a simple DIV with a border: <div style="**width:50px;** **height:50px;** border:1px solid" >
Circle 	Just a DIV with large round corners: <div style="width:50px; height:50px; border:1px solid; **border-radius: 25px**;" >
Half Circle 	Just a DIV with only two round corners (top-left and bottom-left): <div style="width:25px; height:50px; border:1px solid; **border-radius:25px 0 0 25px**;" >

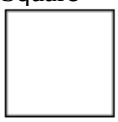
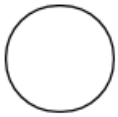
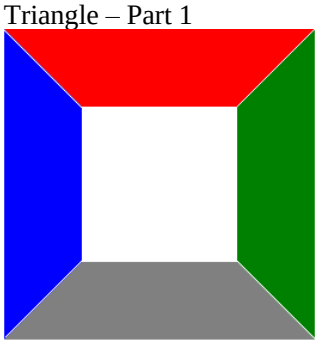
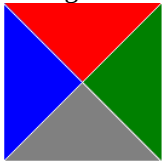
Half Circle – open 	Just a DIV with only two round corners (top-left and bottom-left) and no right border: <div style="width:25px; height:50px; border-width:1px 0 1px 1px; border-style:solid; border-radius:25px 0 0 25px;" >
Triangle 	Just a simple... well not really...: see the next seven examples… <div style="width: 0px; height: 0px; border-left: 50px solid transparent; border-right: 50px solid transparent; border-bottom: 50px solid gray;" >

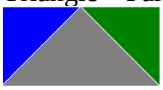
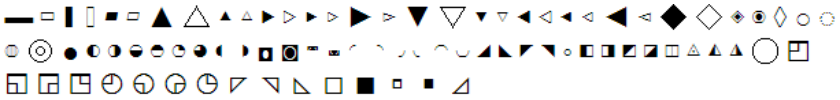
Triangle – Part 1 	Start with a box with thick borders (shown in different colors here): <div style="width: 100px; height: 100px; **border-left: 50px solid blue;** **border-right: 50px solid green;** **border-top: 50px solid red;** **border-bottom: 50px solid gray;**" >
Triangle – Part 2 	Set the height and width to zero and you now only have thick borders: <div style="**width: 0px**; **height: 0px**; border-left: 50px solid blue; border-right: 50px solid green; border-top: 50px solid red; border-bottom: 50px solid gray;" >
Triangle – Part 3 	Remove the top border: (depending on the desired direction of the triangle) <div style="width: 0px; height: 0px; border-left: 50px solid blue; border-right: 50px solid green; border-bottom: 50px solid gray;" >

Triangle – Part 4 	Make the left and right borders transparent (or the same color as the background): <div style="width: 0px; height: 0px; border-left: 50px solid **transparent**; border-right: 50px solid **transparent**; border-bottom: 50px solid gray;" >
Triangle – Part 5 	Tweak the width of each border as needed: <div style="width: 0px; height: 0px; border-left: **30px** solid transparent; border-right: **30px** solid transparent; border-bottom: 50px solid gray;" >

Triangle – Part 6 	And rotate it a little if you like: <div style="width: 0px; height: 0px; border-left: 30px solid transparent; border-right: 30px solid transparent; border-bottom: 50px solid gray; **transform: rotate(-15deg);**" >
Triangle – Part 7 	A variation with just the top left borders: <div style="width: 0px; height: 0px; border-left: 50px solid blue; border-bottom: 50px solid transparent;" >

Instead of CSS, you can also use HTML Entities for many common symbols: ▲ ▲ ► ► ▼ ▼ ◄ ◄	See here for more symbol entities: http://www.w3schools.com/charsets/ref_utf_geometric.asp 
Instead of CSS, you can also use the <canvas> or <svg> tags to draw shapes.	<canvas> uses JavaScript to draw shapes while <svg> uses XML to define shapes.

Square 	Just a simple DIV with a border: <div style="**width:50px;** **height:50px;** border:1px solid" >
Circle 	Just a DIV with large round corners: <div style="width:50px; height:50px; border:1px solid; **border-radius: 25px**;" >
Half Circle 	Just a DIV with only two round corners (top-left and bottom-left): <div style="width:25px; height:50px; border:1px solid; **border-radius:25px 0 0 25px**;" >
Half Circle – open 	Just a DIV with only two round corners (top-left and bottom-left) and no right border: <div style="width:25px; height:50px; border-width:1px 0 1px 1px; border-style:solid; border-radius:25px 0 0 25px;" >
Triangle 	Just a simple... well not really...: see the next seven examples… <div style="width: 0px; height: 0px; border-left: 50px solid transparent; border-right: 50px solid transparent; border-bottom: 50px solid gray;" >
Triangle – Part 1 	Start with a box with thick borders (shown in different colors here): <div style="width: 100px; height: 100px; **border-left: 50px solid blue;** **border-right: 50px solid green;** **border-top: 50px solid red;** **border-bottom: 50px solid gray;**" >
Triangle – Part 2 	Set the height and width to zero and you now only have thick borders: <div style="**width: 0px**; **height: 0px**; border-left: 50px solid blue; border-right: 50px solid green; border-top: 50px solid red; border-bottom: 50px solid gray;" >

Triangle – Part 3 	Remove the top border: (depending on the desired direction of the triangle) <div style="width: 0px; height: 0px; border-left: 50px solid blue; border-right: 50px solid green; border-bottom: 50px solid gray;" >
Triangle – Part 4 	Make the left and right borders transparent (or the same color as the background): <div style="width: 0px; height: 0px; border-left: 50px solid **transparent**; border-right: 50px solid **transparent**; border-bottom: 50px solid gray;" >
Triangle – Part 5 	Tweak the width of each border as needed: <div style="width: 0px; height: 0px; border-left: **30px** solid transparent; border-right: **30px** solid transparent; border-bottom: 50px solid gray;" >
Triangle – Part 6 	And rotate it a little if you like: <div style="width: 0px; height: 0px; border-left: 30px solid transparent; border-right: 30px solid transparent; border-bottom: 50px solid gray; **transform: rotate(-15deg);**" >
Triangle – Part 7 	A variation with just the top left borders: <div style="width: 0px; height: 0px; border-left: 50px solid blue; border-bottom: 50px solid transparent;" >
Instead of CSS, you can also use HTML Entities for many common symbols: ▲ ▲ ► ► ▼ ▼ ◄ ◄	See here for more symbol entities: http://www.w3schools.com/charsets/ref_utf_geometric.asp 
Instead of CSS, you can also use the <canvas> or <svg> tags to draw shapes.	<canvas> uses JavaScript to draw shapes while <svg> uses XML to define shapes.

Module 4: Fonts and Text

🗁 **Sample file: font.html**

Fonts

A font is a collection of characters sharing a common style, height, angle and other characteristics.

This is Calibri
This is Arial
This is Times New Roman
This is Courier New
This is Comic Sans MS
That is Webdings →

You have several sources of fonts:

- Those built into your site visitor's browsers.
- Custom fonts that you have created and host on your own web servers.
- Custom fonts that you have licensed and host on your own web servers.
- Custom fonts that you have licensed and linked to.
  Examples:
  - Adobe's Typekit site https://typekit.com/
  - Google Fonts: https://fonts.google.com/

Prior to using any third party fonts, verify that the license requirements do not conflict with your organization's policies for licensed or open source content.

Fallback

Web browsers include several fallback fonts that be used when your selected fonts are not available. These are typically the last font listed in the CSS.

- Sans-serif – Fonts without serifs or decorations. Example: **Sans-serif**
- Serif – Fonts with decorations. Example: **Serif**
- Monospace – Fonts with all equal width characters. Example: **Monospace**

Images for Unusual Fonts

When you need to use an uncommon font for a logo or graphic design, and you don't want to license a font for web distribution, you can create an image of the design. Remember to add an "alt" attribute with the text that's displayed in the image for the benefit of screen readers and search engines!


```
<img src="specialtext.png" alt="This is an Uncommon Font!">
```

CSS for Fonts

Font characteristics can be controlled using the "font" CSS styles. These include:

- font-family - The font face. Examples: Time Roman, Arial, Courier
- font-style - Normal, Italics or Oblique. Example: Normal, *Italic*, *Oblique*
- font-size - Size of font in any of the supported absolute or relative units.
- font-weight - Boldness of the font.

🗁 **Sample file: fonts.html**

font-family

While we refer to a "font", a font is typically a family of fonts that includes a default design, and optionally, bold, italic and other versions. When a font does not include one of the styles such as italic, most browsers will create one using an algorithm. Font families are usually unique to an operating system and may not be available on all devices or supported by all browsers. In order to have as much control as possible of the appearance of your page, you should specify your preferred list of fonts to use if your first choice is not available. In the example below we state that we want to use "Arial", but if it is not available to use Helvetica, and if that is not available, use the browser's default sans-serif font.

```
font-family: Arial, Helvetica, sans-serif
```

Default fonts

Browsers support several generic fonts. These will typically be mapped to the equivalent fonts installed on the user's device.

- Sans-serif
- Serif

- Monospace
- Cursive

Safe Fonts and Fallbacks

A few fonts are available on just about every device and browser and are referred to as “safe fonts”.

```
font-family: Arial, Helvetica, sans-serif
font-family: 'Times New Roman', Times, serif
font-family: 'Courier New', Courier, monospace
```

If you have a specific font that you want to use, and most of your users will have that font available, list it first, but still provide a list of fallback fonts.

```
font-family: Tahoma, Arial, Helvetica, sans-serif
```

Test, test, test!

To see how some of the common fonts display in each of your browsers visit this W3 page:
http://www.w3.org/Style/Examples/007/fonts.en.html

font-style (italics)

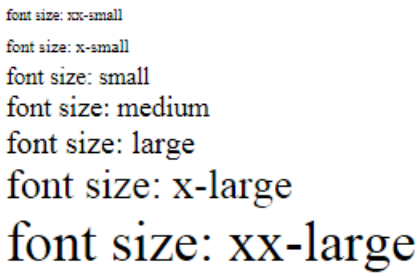
Most fonts will either have an italics variation, or the computer will create one.

Note: Italic vs. Oblique. In an ideal situation, an Italic variation of a font is a custom designed angled version of the primary font while Oblique is simply an angled, computer generated, version of the primary font. If a true Italic font is not available, most computers will create an Oblique from the primary font. In the web world you will almost never see a difference.

font-size

Font size can be set using any of the CSS size units or a list of predefined sizes:

- Numeric value in any of the CSS units. Example:
  `font-size:16pt`
- Absolute sizes:
  `xx-small, x-small, small, medium, large, x-large, xx-large`
- Relative sizes:
  `larger, smaller` (These roughly switch between the absolute sizes above.)



Notes:

- It is often stated that the absolute sizes roughly correspond to the numbered sizes, 1..7, used by the obsolete <font> tag.
    - In testing, sizes 1..7 correspond to 10px, 13px, 16px, 18px, 24px, 32px and 48px.
    - In testing, the CSS absolute sizes correspond to 9px, 10px, 13px, 16px, 18px, 24px and 32px.
    - See the font.html sample file for examples.
- The W3.org specification does not define the sizes of the absolute size names. This is left to the browser vendor to define. You generally should define your font sizes in one of the CSS units.
- The display size of a font depends on the browser. I.e. 48px is not the same in every browser. As an example: on a Windows 7 computer, IE 11's Zoom set to 125% is about the same display size as Firefox version 47 at 100%.

font-weight (bold)

Possible values: normal, bold, bolder, lighter, or a numeric value from 100-900.

Most browsers will only render "bold" and "normal" and will render 100-400 as "normal", 500-900 as "bold", "lighter" as "normal" and "bolder" as "bold"

<b> vs. and <i> vs.

The <b> and <i> tags are exceptions to the "tags are markup, not presentation" rule. These simply defined the use of **bold** and *italics* styles to display text and add no other importance. and are semantic and imply the context or purpose of the enclosed text. The distinction becomes important when you start thinking about text-to-speech and other alternate delivery of web content where <b> and <i> might be ignored while might be spoken louder and might be in another voice. Also, search engines may choose to ignore <b> and <i> while considering text inside of and to be more important. Browsers render the same as <b> and the same as <i> for text displays.

Note that both <b> and <i> can be styled using CSS and could be displayed as anything!

For a long article on the differences and proposed best practices see:
http://html5doctor.com/i-b-em-strong-element/

<b> vs. font-weight and <i> vs. font-style

The <b> and <i> tags are inline, manually added, tags that change the appearance of the enclosed text. The CSS font-weight and font-style are presentation features applied to any text container, or globally using a style sheet.

When is Bold not Bold?

When you style it that way! <b> is a tag, and tags can be styled. If you wanted to make all of the text someone else enclosed in <b> tags to be displayed italic and not bold then just add some CSS! Remember, CSS is about the presentation of content.

```
b { font-weight:normal; font-style:italic; }
➔   This is <b>bold</b> text!
```

This is *bold* text!

Note that the same also applies to <u> vs. text-decoration:underline.

CSS for Text

Text Alignment

```
<p style="text-align:left">
  Lorem ipsum dolor sit amet, …
</p>

<p style="text-align:right">
  Lorem ipsum dolor sit amet, …
</p>

<p style="text-align:center">
  Lorem ipsum dolor sit amet, …
</p>

<p style="text-align:justify">
  Lorem ipsum dolor sit amet, …
</p>
```

Lorem ipsum dolor sit amet,
consectetuer adipiscing elit.
Maecenas porttitor congue massa.

Lorem ipsum dolor sit amet,
consectetuer adipiscing elit.
Maecenas porttitor congue massa.

Lorem ipsum dolor sit amet,
consectetuer adipiscing elit.
Maecenas porttitor congue massa.

Lorem ipsum dolor sit amet,
consectetuer adipiscing elit.
Maecenas porttitor congue massa.

Overline, Line-through, Underline

```
<p style="text-decoration:overline">
  Lorem ipsum dolor sit amet
</p>
<p style="text-decoration:line-
through">
  Lorem ipsum dolor sit amet
</p>
<p style="text-decoration:underline">
Lorem ipsum dolor sit amet
</p>
```

Lorem ipsum dolor sit amet

~~Lorem ipsum dolor sit amet~~

Lorem ipsum dolor sit amet

All three combined:

```
<p style="text-decoration:overline
line-through underline;">
Lorem ipsum dolor sit amet
</p>
```

~~Lorem ipsum dolor sit amet~~

Note: You can change the colors of the lines using "text-decoration-color: red;". But at the time this was written this is only supported in Firefox.

Note: You can use border-top and border-bottom to emulate overline and underline. These will also give you more options to control the location and color of the lines.

Changing capitalization

```
<span style="text-
transform:uppercase">Hello
World!</span>
<span style="text-
transform:lowercase">Hello
World!</span>
<span style="text-
transform:capitalize">hello
world!</span>
```

HELLO WORLD!

hello world!

Hello World!

Letter spacing

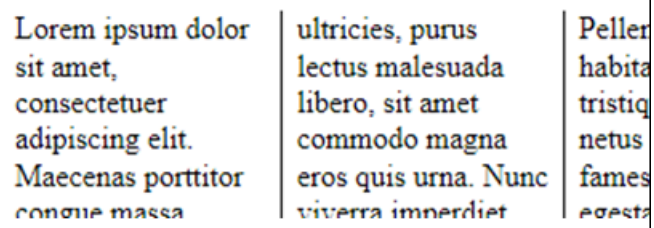
```
<span style="letter-
spacing:3px">Letter spacing</span>
<span style="letter-
spacing:6px">Letter spacing</span>
```

L e t t e r s p a c i n g

L e t t e r s p a c i n g

Multiple Columns

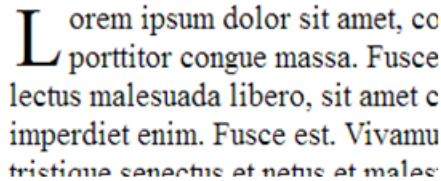
```
<style>
p {
  -moz-column-count: 3;
  -moz-column-rule-width: 1px;
  -moz-column-rule-style: solid;
  column-count: 3;
  column-rule-width: 1px;
  column-rule-style: solid;
}
</style>

<p>
  Lorem ipsum dolor sit amet, …
</p>
```



To support older browsers you may need to add the experimental prefixes ("-moz-", "-ms-", "-webkit-", "-o-").

Drop Cap

```
p:first-letter {
        font-size: 250%;
        margin:5px 3px 0;
        float: left;
```



CSS Text Tricks!

CSS can be used to transform text in many ways. You can add shadows, rotate and distort to create interesting effects. Below are a few examples. Search the web for more!

Shadows

Shadows can be created by offsetting the text while using a lighter color. In the examples below, the first two numbers are the offset and can be positive or negative. The third value is the color and can be specified using any of the CSS color values. You can also specify a third number to add a "blur-radius" to soften the shadow.

Examples:

```
h1 { text-shadow: 4px 4px lightgray; }

h1 { text-shadow: -4px 4px lightgray; }

h1 { text-shadow: 4px 4px 4px gray; }

h1 { text-shadow: 4px 4px 8px gray; }
```

Sales Meeting
Sales Meeting
Sales Meeting
Sales Meeting

For an embossed look, try two opposite shadows and a light color:

```
<h2 style="color: #EEEEEE; text-shadow: -1px -1px 1px #FFFFFF, 1px 1px 1px #888888;"> Embossed </h2>
```

Embossed

"text-shadow" has four parameters: horizontalOffset verticalOffset blurRadius color

Rotated Text

One of the features of a CSS "transform" is the ability to rotate an image, text or other HTML object. The unit of rotation is degrees with positive numbers for clockwise rotation.

Examples:

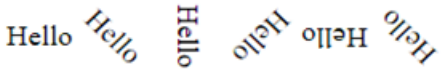
```
.rotate90
{
  display:inline-block; /* needed for Firefox */
  transform: rotate(-90deg);
}

Hello<span class="rotate90">World</span>!
```

Hello World !

```
.rotate { transform: rotate(90deg); }

<td rowspan=5 class="rotate">Closed</td>
```

January, 2017						
Su	Mo	Tu	We	Th	Fr	Sa
Closed	1	2	3	4	5	Closed
	9	10	11	12	13	
	16	17	18	19	20	
	23	24	25	26	27	
	30	31				

```
<span style="transform:rotate(0deg);
  display:inline-block;">Hello</span>
<span style="transform:rotate(45deg);
  display:inline-block;">Hello</span>
<span style="transform:rotate(90deg);
  display:inline-block;">Hello</span>
<span style="transform:rotate(135deg);
  display:inline-block;">Hello</span>
<span style="transform:rotate(180deg);
  display:inline-block;">Hello</span>
<span style="transform:rotate(225deg);
  display:inline-block;">Hello</span>
```



Notes:

Firefox and other browsers may need the addition of “display:inline-block” when rotating a .

To support older browsers, you may need to add the experimental prefixes (“-moz-”) and/or a shim. Visit http://caniuse.com, search for “transform” and then click “Show all” for details. Also see “Experimental Vendor Prefixes” in the CSS module of this course.

```
-moz-transform: rotate(90deg);
-ms-transform: rotate(90deg);
-webkit-transform: rotate(90deg);
-o-transform: rotate(90deg);
transform: rotate(90deg);
```

Working with Lists

Bulleted and numbered lists are common structures in documents and web pages. By using CSS these simple lists can be converted into menus and ribbons. While these elements have default display properties, you will probably want to restyle them in to something a bit nicer.

Ordered and Unordered Lists

An ordered list uses numbers, letters or Roman numerals to build ordered lists of content. An unordered list uses bullets (or squares or stars or …) to display the list of items.

1. First item
2. Second item
3. Third item
4. Forth item

- First item
- Second item
- Third item
- Forth item

Both kinds of lists follow the same pattern: <ol> or <ul> tags that contain one or more list item (<li>) tags.

```
<ol>
  <li>First item</li>
  <li>Second item</li>
  <li>Third item</li>
  <li>Fourth item</li>
</ol>
```

```
<ul>
  <li>First item</li>
  <li>Second item</li>
  <li>Third item</li>
  <li>Fourth item</li>
</ul>
```

The content of the <li> tags can be text or other HTML including child lists.

- **First** item
- 
- Level 2:
    1. First item
    2. Second item
    3. Third item
    4. Fourth item
- Fourth item

```
<ul>
  <li><b>First</b> item</li>
  <li><img src="../images/flower030.png"></li>
  <li>
    Level 2:
    <ol>
      <li>First item</li>
      <li>Second item</li>
      <li>Third item</li>
     <li>Fourth item</li>
    </ol>
  </li>
  <li>Fourth item</li>
</ul>
```

List Styling

The first change in most lists is to customize the numbers or bullets being used. Note that the bullet and numbering option support varies by browser. See here for a list of supported options: http://www.quirksmode.org/css/lists.html

To select from one of the built-in bullets use list-style-type and one of these values:

- disc (default)
- circle
- square
- Documented, but not supported yet:
    - check
    - diamond
    - hypen

```
<ul>
  <li>First item</li>
  <li>Second item</li>
  <li>Third item</li>
  <li>Fourth item</li>
</ul>
```

- First item
- Second item
- Third item
- Fourth item

For ordered lists you can select from these built-in types:

- decimal (default), decimal-leading-zero
- lower-roman, upper-roman
- lower-greek
- lower-latin, upper-latin (a,b,c…)
- armenian

- georgian
- lower-alpha, upper-alpha (a,b,c…)

Examples of decimal-leading-zero, upper-roman and lower-greek:

01. First item	I. First item	α. First item
02. Second item	II. Second item	β. Second item
03. Third item	III. Third item	γ. Third item
04. Fourth item	IV. Fourth item	δ. Fourth item

You can even create custom images for unordered lists.

```
list-style-image: url(MyCustomBullet.gif)
```

- First item
- Second item
- Third item
- Fourth item

For more complex lists you can use <table> and tags to build very complex structures.

Module 5: Colors and Backgrounds

In this course we are taking a developer's view of design, the tags, styles, and tools for building web pages. You will add the graphic designer's skills to create great looking pages.

🗁 **Sample file: Colors.html**

Specifying Colors

Colors can be selected using color names or numerically by specifying the Red, Green and Blue values.

Color Names

HTML and CSS define 140 color names, some obvious, and some not. Just what color is "peru" or "rebeccapurple"? And what's the difference between "yellowgreen" and "greenyellow"? ☺ Older browsers even mixed spellings of "gray" so that lightgrey, grey, darkgrey, lightgray, gray, and darkgray were not all shades of gray. Modern browsers support a few "alternate" color spellings so that both "grey" and "gray" are now the same color.

RGB Color Numbers

Even with the spelling issue taken care of, not everyone agrees on what color "gray" really should be. Instead of using the color names, you can mix your own colors by using 16,777,216 combinations of Red, Green and Blue.

256 Colors

You can create anyone of 256 colors by using three hexadecimal values with each value assigning an amount of Red, Green or Blue, each within a range of 0 to 15. In hexadecimal, the values would be 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. The letters can be typed in upper or lower case.

```
#f00 = red, #0f0 = blue, #00f = green, #000 = black, #fff = white
```

16,777,216 Colors

Instead of using 16 increments each of Red, Green and Blue to get 256 different colors, you can use 256 increments of Red, Green and Blue to get 16,777,216 different colors.

```
#ff0000 = red, #00ff00 = blue, #0000ff = green, #000 = black, #ffffff = white
```

When working with CSS you can also use the RGB function to define colors using decimal instead of hexadecimal.

```
rgb(0,0,255) = blue
```

HSL Colors

HSL colors (hue, saturation, and lightness) can be defined in CSS using the hsl function. Hue is measured in degrees on a color wheel and ranges from 0 to 360 where 0 is red, 120 is green and 240 is blue. Saturation is a percentage from 0 to 100. Lightness is also a percentage with 0% as black and 100% as white.

```
hsl(0, 100%, 50%) = red,    hsl(240, 100%, 50%) = blue,    hsl(120, 100%, 50%) = green
```

HSL is supported in most modern browsers. For an HSL color picker see: http://www.w3schools.com/colors/colors_hsl.asp. For a list of supported browsers see: http://caniuse.com/#search=hsl

The Future

CSS4 suggests the addition of several more color selection standards. None of these are available in current browsers.

- HWB (Hue, Whiteness, Blackness)
- CMYK (CYAN, MAGENTA, YELLOW , and BLACK)

Color Tools

- **Color Converter**: The w3schools.com site has a great tool for experimenting with each of the color selection options. Visit http://www.w3schools.com/colors/colors_converter.asp and experiment with each of the options.
- **HTML Color Picker**: http://www.w3schools.com/colors/colors_picker.asp
- **Crayola crayon colors**: http://www.w3schools.com/colors/colors_crayola.asp
- **Color wheel, palette selector**: http://paletton.com (Includes tools to show how the colors would display for color blindness and “bad monitors”.)
- There are many tools to help you pick color palettes for your web project. Here’s one site that lists 28 tools: http://www.creativebloq.com/colour/tools-colour-schemes-12121430

Applying Colors

Prior to CSS, colors were applied using tag attributes. Here’s the obsolete <font> tag:

```
<font face="arial" color="red">this is red text!</font>
```

CSS lets you apply color to just about every aspect of an HTML object.

```
div { color:red; background-color:yellow; border-color:green; scrollbar-face-color:blue }
```

CSS also supports all color selection options listed earlier, color names, #fff, #ffffff, rgb(0,0,0) and hsl(0, 100%, 50%).

Gradients

CSS supports adding smooth transitions from one color to another color using linear and radial gradients

In CSS 3, gradients come in two flavors: linear and radial. Gradients are defined using these patterns:

- **Default (top to bottom)**
  property: linear-gradient(*startColor, color, color, endColor*)
  background: linear-gradient(red, green)

- **With a direction:**
  property: linear-gradient(*directionToStartFrom, startColor, endColor*)
  direction= to left, to right, to top, to bottom
  background: linear-gradient(to top, red, green)

- **With a direction (diagonal):**
  property: linear-gradient(*directionToStartFrom, startColor, endColor*)
  direction= to bottom left, to bottom right, to top right, etc.
  background: linear-gradient(to top right, red, green)

- **With any angle:**
  property: linear-gradient(*angle, startColor, endColor*)
  background: linear-gradient(30deg, red, green)

- **Radial gradient:**
  property: radial-gradient(*startColor, color, color, endColor*)
  background: radial-gradient(red, green)

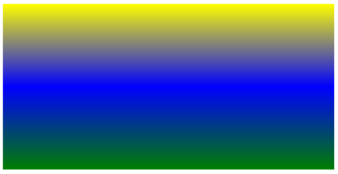
- **Radial gradient:**
  property: radial-gradient(*startColor, color, color, endColor*)
  shape= ellipse (default), circle
  background: radial-gradient(circle, red, green)

🗁 **Sample file: gradients.html**

background:linear-gradient(yellow, green)

background:linear-gradient(15deg, yellow, green)

background:linear-gradient(yellow, blue, green)



background: radial-gradient(yellow, green)

background:linear-gradient(to top, yellow, green)

background: radial-gradient(circle, yellow, green)

background:linear-gradient(to top right, yellow, green)

Module 6: Anchors and Hyperlinks

The “H” of HTML refers the ability to link from one document to another. While we refer to links or hyperlinks, the tag used for links is the “anchor” tag.

The user’s mouse will change when they mouse over an anchor tag. (You can change the mouse pointer using CSS.)

The anchor tag has several uses:

- Link to other pages or sites.

```
<a href="http://www.w3.org">World Wide Web Consortium</a>
```

- Link to locations in the same page. If the link destination is on the same page, then the href is just a pound sign (“#”) and the ID of any tag on the page.

```
<a href="#details">Click here for details</a>      ← The link
…
<h1 id="details">Contest Details</h1>              ← The destination
```

  Note: Prior to HTML 4 an <a> tag with a name attribute was used as the destination. For compatibility, modern browsers will treat the “name” attribute of an <a> tag as if it is the ID for the tag. While obsolete, anchor tags with the “name” attribute are still found in many web pages.

```
<a href="#details">Click here for details</a>          ← The link
…
<a name="details"></a><h1>Contest Details</h1>  ← The destination
```

- Link to another page and scroll down to content:

```
<a href="http://www.example.com/contest.html#details">Click here for details</a>
```

- Act as button to run JavaScript.

```
<a href="javascript:alert('you clicked me!'">Click me!</a>
```

  or

```
<a href="" onclick="javascript:alert('you clicked me!'); return false;">Click me!</a>
```

Attributes:

- href – Destination URL, target or script.

```
href="http://www.w3.org"              (absolute URL)
href="home.html"                      (relative URL)
href="../accounts/youraccount.html"   (relative URL)
href="#privacyinfo"                   (target in the same page)
```

  name – The name for used when <a> is a target in the same page. This is obsolete starting with HTML 4.0, but is still

commonly used. The target can now be any tag with an ID attribute.

```
<a name="details"></a>
```

- download – (HTML 5 only) The linked target will be downloaded when clicked. *Not currently supported in Internet Explorer or Safari.* Supported in Chrome 14+, Edge, Firefox 20+ Opera 15+

```
<a href="Flyer.pdf" download>Download a flyer</a>
```

- target – Where to open the linked page.
  - _blank – open in new window or tab, depending on browser settings.
  - *aName* – Any other text can be used for the target. This will open a new window or tab. Additional links using this name will open in the same window.
  - _self – Opens in the same window. (This is the default for target.)
  - _parent, _top – Obsolete. These were with framesets.
- rel – Describes the relationship to the current page. While there are many options, the most commonly used in "nofollow".
  - rel="nofollow" tells search engines not to follow the link, or relate it to your page.

<a> and CSS

The anchor tag has four states, unvisited, visited, hover and active, and each can be styled using CSS "Pseudo-classes". While these Pseudo-classes will style the contents of the <a> tag, styles defined between the <a> tags will cascade and "win".

Changing the link style

The change the appearance of these three states use these Pseudo-classes:

- Unvisited link:
  ```
  a:link { color:blue; }
  ```
- Visited link:
  ```
  a:visited {color:red; }
  ```
- Mouse is over the link: (The a:hover definition must follow a:link and a:visited.)
  ```
  a:hover { color:black; }
  ```
- Active link:
  ```
  a:active { color:green; }
  ```

The above can be combined with other CSS selectors as needed:

- To style the following link:
  Product 123
  use:
  a.productlink:link { color:blue; }

Changing the Mouse Pointer

The CSS cursor feature can be used to change the mouse pointer.

To change the pointer displayed over this link:

```
<a href="help.html">Help!</a>
```

use:

```
cursor:help;
```

to get the help (?) pointer:

There are forty predefined mouse pointers available: alias, all-scroll, auto, cell, context-menu, col-resize, copy, crosshair, default, e-resize, ew-resize, grab, grabbing, help, move, n-resize, ne-resize, nesw-resize, ns-resize, nw-resize, nwse-resize, no-drop, none, not-allowed, pointer, progress, row-resize, s-resize, se-resize, sw-resize, text, URL, vertical-text, w-resize, wait, zoom-in, zoom-out, initial, inherit

You can create custom pointers using your image. Internet Explorer only supports the .cur and .ani file types. Firefox and other browsers support most of the common image formats. There are a number of tools to create .cur and .ani files. One freeware tool: JustCursors (available from download.cnet.com).


```
<a style="cursor:url(face.cur), auto" href="help.html">Help!</a>
```

You can supply “fallbacks” to deal with the various browsers, including the built-in cursor types.

```
a.help {
  cursor: help;
  cursor: url(face.cur), help;  /* try this 2nd (for IE) */
  cursor: url(face.jpeg) 4 12, help; /* try this 1st */
}
```

🗁 **Sample file: customcursors.html (and face.cur)**

Hyperlinks with Images and Other Objects

The <a> tag can wrap just about any content. When working images, and other less than obvious click targets, you give your users hints and alternate text.

```
<a href="help.html"><img href="help.jpeg" alt="click here for help"></a>
<a href="help.html" title="click here for help"><p>lots of text</p></a>
```

Buttons

HTML buttons are designed to be used inside of forms and are not a direct replacement for <a> tags as they are missing the "href" attribute. You can add JavaScript to a <button> tag to make it redirect to a URL.

```
<button onclick="document.location='home.html';">Click Me</button>
```

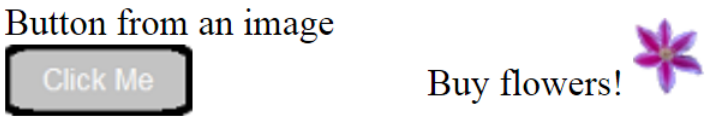
If you need buttons that use <a> tags then just create them from images, CSS or JavaScript libraries.

🗁 **Sample file: LinksAndButtons.html**

Buttons from Images

Images can be wrapped inside of anchor tags to create just about any shape or size buttons.

```
<a href="saledetails.html"><img src="..\images\clickmeButton.png"></a>
Buy flowers! <a href="saledetails.html"><img src="..\images\flower030.png"></a>
```



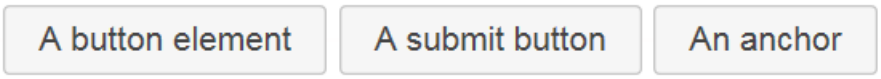
Buttons from CSS

CSS will let you put borders and backgrounds on just about everything. Wrap a in an <a> and you can create your own buttons and controls.

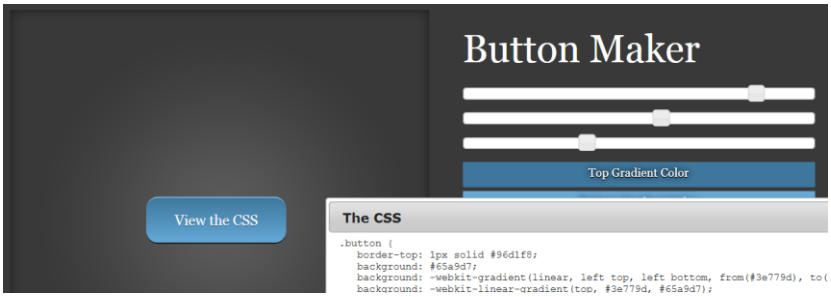
```
<style>
 a.SpanButton {
  color:red;
  padding:0px 5px;
  border-width:3px; border-style:solid;
  border-radius: 5px;
  border-color:gray;
  text-decoration:none;
 }
 a.SpanButton:hover
 {
  color:green;
 }
</style>

<a class="SpanButton" href="saledetails.html"><span>Click Me</span></a>
```

There more examples at w3schools.com: http://www.w3schools.com/Css/css3_buttons.asp

JavaScript developers can use tools like jQueryUI to create buttons from <button>, <input> and <a> tags.



There are even “button maker” tools on the web: https://css-tricks.com/examples/ButtonMaker/



Module 7: Page Layout

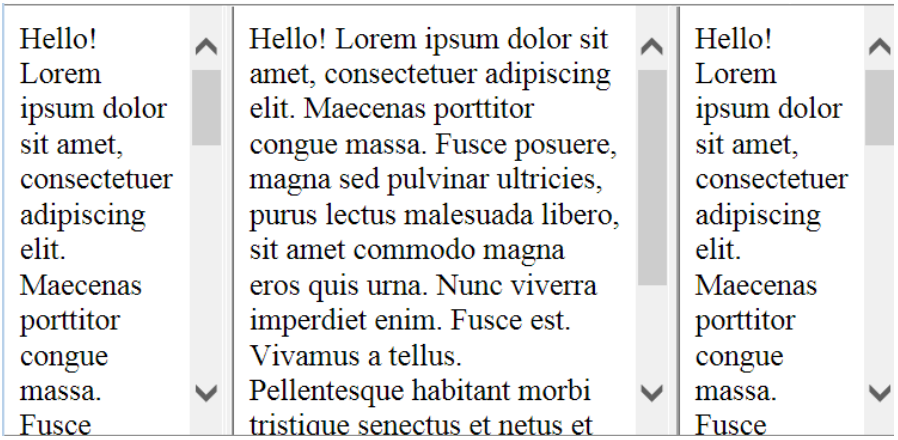
Page Layout Options

In this module we will look at HTML's options to control where and how content is arranged on a page. We will look at both the basic tags and what can be done using CSS.

As HTML has evolved so have the tools and best practices for page layout.

Frames

In the early days of HTML pages were often broken up into areas called frames. The root page defined the "frameset" and each frame contained its own HTML page. The result was "scrollbars everywhere" and an overall inflexible and ugly design.

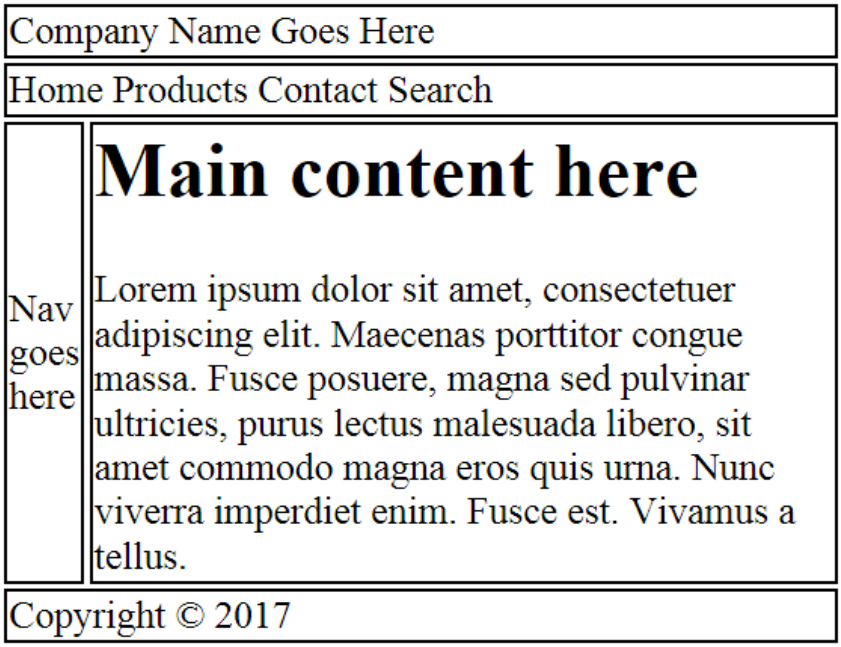


Tables

HTML tables are used to layout data in rows and columns, similar to a spreadsheet.

Fish	price
Catfish	19.95
Cod	17.95
Trout	24.95

Many web developers started using tables as a replacement for frames to layout entire pages. This has several problems, including the complexity of the HTML, the manual labor required to change the design, and the biggest issue of not being dynamic when used with phones and other devices.



🗁 **Sample file: Tables.html**

DIVs

DIVs, while very simplistic and plain by default, can be quickly restyled by simply changing the CSS, either by linking to another file, or by designing the CSS to adapt to the user's device.

🗁 **Sample file: html5contextual.html**

For an example of DIVs and CSS in action, visit the CSS Zen Garden (http://www.csszengarden.com/).

1. View the source of the page and note that all content is either in DIVs or the new HTML 5 DIV-like tags (header, footer, section, etc.).
2. Now find and click the links for the other designs. Again, view the source of the page and note that the only thing that has changed is the linked CSS file. Major look and feel design changes with just a change of CSS! (The JavaScript in the page is only being used to manage the loading of fonts. All of the cool stuff is from pure CSS!)
3. Resize your browser and note if the design adapts to the display size.
4. Access the CSS Zen Garden from your cell phone or tablet and see if the design still works!

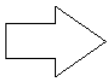
Tables

In modern web design, tables are used for the display of data that is best arranged in formal rows and columns. The contents of a table is often dynamically created by client side or server side code. In the past, tables have been used to design page layouts. This is now best done using DIVs and CSS.

Tables use four tags: <table>, <tr>, <th> and <td> where <tr> represents a row, <th> represents a header cell and <td> represents the data in a cell.

🗁 **Sample file: Tables2.html** and **Tables3.html**

```
<table>
  <tr>
    <th>Fish</th><th>price</th>
  </tr>
  <tr>
    <td>Catfish</td><td>19.95</td>
  </tr>
  <tr>
    <td>Cod</td><td>17.95</td>
  </tr>
  <tr>
    <td>Trout</td><td>24.95</td>
  </tr>
</table>
```



Fish	price
Catfish	19.95
Cod	17.95
Trout	24.95

Attributes: In general the table elements use of attributes to control the size of the table, borders and margins are deprecated in favor of CSS styling. Two exceptions are COLSPAN and ROWSPAN. (see Merging Cells below)

<table>

A table has one <table> tag pair and at least one <tr> (row) that has at least one <td> (cell or data). This tag can contain: <caption>, <colgroup>, <thead>, <tbody>, <tfoot>

<caption>

A caption or title for the table. This will span the total width of the table and by default will have its contents centered. The contents can include text and inline elements. This tag must immediately follow the <table> tag.

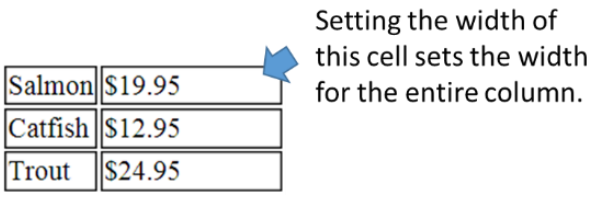
```
<caption>Today's fish</caption>
```

Today's fish	
Salmon	$19.95
Catfish	$12.95
Trout	$24.95

Widths

By default, the content of the cells will determine the widths of the columns. All cells in a single column will be the same width. Column widths only need to be set on one cell of a column. If more than one width has been specified, the largest will be used. Column width is set by applying the CSS “width” to a <td>.

```
<table>
  <tr><td>Salmon</td><td style="width:100px;">$19.95</td></tr>
  <tr><td>Catfish</td><td>$12.95</td></tr>
  <tr><td>Trout</td><td>$24.95</td></tr>
</table>
```



CSS Frequently used for Tables

All of the CSS used for the CSS Box Model can be used for tables.

Borders

Borders are often used with tables, possibly with a side effect that's not needed. In the example below note the white space between the cells.

Fish		
Salmon	$19.95	fresh!
Catfish	$12.95	fresh!
Trout	$24.95	frozen
Burgers		
Classic burger	$6.95	Comes
Cheese burger	$7.95	with

By applying border-collapse:collapse to the <table> tag you can close those gaps. (border-collapse:separate is the default.)

```
table { border-collapse:collapse; }
```

Fish		
Salmon	$19.95	fresh!
Catfish	$12.95	fresh!
Trout	$24.95	frozen
Burgers		
Classic burger	$6.95	Comes
Cheese burger	$7.95	with

Merging Cells

HTML tables can be used to create very complex layouts of data where some cells (td's) span multiple rows or columns.

colspan (column span)

In the example below, the title "Fish" is in a spanned <td> that crosses all of the columns of the table. To create a span, add a colspan attribute to the first <td> in the span area. Note that in this example, the <td> with the "Fish" text is the only <td> in the row. It spans or fills the space of three <td> cells.

```
<table>
 <tr><td colspan=3>Fish</td></tr>
 <tr>
  <td>Salmon</td>
  <td>$19.95</td>
  <td>fresh!</td>
 </tr>
 …
</table>
```

Fish		
Salmon	$19.95	fresh!
Catfish	$12.95	fresh!
Trout	$24.95	frozen

rowspan

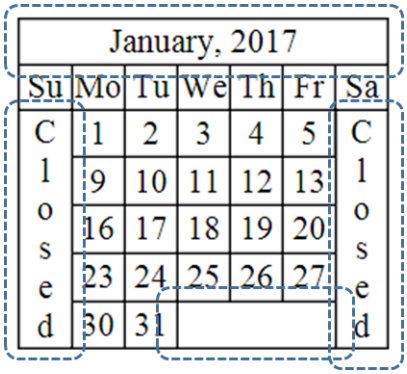
A rowspan is a single <td> that spans vertically. In the third and fourth rows of this table there are no 3rd column <td> cells. The <td> from row 2 spans or drops down to where those cells would have been.

```
<table>
 <tr><td colspan=3>Burgers</td></tr>
 <tr>
  <td>Classic burger</td>
  <td>$6.95</td>
  <td rowspan=3>Comes<br>with<br>fries!</td>
 </tr>
 <tr>
  <td>Cheese burger</td>
  <td>$7.95</td>
  <!-- no cell here! Spanned from row 2 -->
 </tr>
 <tr>
  <td>The Everything!</td>
  <td>$8.95</td>
  <!-- no cell here! Spanned from row 2 -->
 </tr>
</table>
```

Burgers		
Classic burger	$6.95	Comes with fries!
Cheese burger	$7.95	
The Everything!	$8.95	

Tables can get complex!

The use of column and row spans can get complex and prone to error. The results can be quite useful. Here's a calendar built from a table with four spans. One for the Month across the top, two for the weekend columns and one for the blank part of the calendar:



January, 2017						
Su	Mo	Tu	We	Th	Fr	Sa
C l o s e d	1	2	3	4	5	C l o s e d
	9	10	11	12	13	
	16	17	18	19	20	
	23	24	25	26	27	
	30	31				

🗁 **Sample file: calendartable.html**

Here's the HTML for the calendar's table. Note the rowspan and the colspan attributes.

```
<style>
  table { border-collapse:collapse; }
  table, td { border-width:1px; border-style:solid }
  td { text-align:center; width:22px; }
</style>

<table>

  <tr><td colspan=7>January, 2017</td></tr>

  <tr>
    <td>Su</td><td>Mo</td><td>Tu</td><td>We</td><td>Th</td><td>Fr</td><td>Sa</td>
  </tr>

  <tr>
    <td rowspan=5>C<br>l<br>o<br>s<br>e<br>d</td>
    <td>1</td><td>2</td><td>3</td><td>4</td><td>5</td>
    <td rowspan=5>C<br>l<br>o<br>s<br>e<br>d</td>
  </tr>
  <tr>
    <td>9</td><td>10</td><td>11</td><td>12</td><td>13</td>
  </tr>
  <tr>
    <td>16</td><td>17</td><td>18</td><td>19</td><td>20</td>
  </tr>
  <tr>
    <td>23</td><td>24</td><td>25</td><td>26</td><td>27</td>
  </tr>
  <tr>
    <td>30</td><td>31</td><td colspan=3> </td>
  </tr>
</table>
```

CSS to Control Wrapping

Tables will wrap text when there is not enough room for the text to be displayed in a single cell.

Burgers		
Classic burger with lots of mustard	$6.95	Comes with
Cheese burger	$7.95	

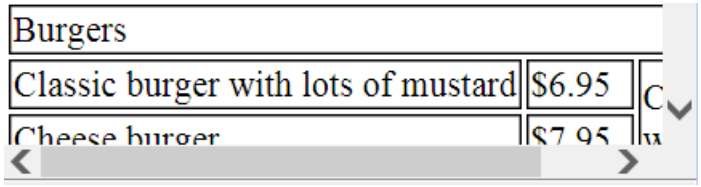
You can prevent unwanted text wrapping two ways:

- Style the text using: white-space: nowrap. You can apply this to a single class, a cell or all <td> tags.

```
td { white-space: nowrap; }
```

- Replace spaces with non-breaking spaces using the HTML entity .

```
Classic burger with lots of mustard
```



When the table is wider than the space available, your users may need to scroll back and forth across the data.

CSS Pseudo-classes for Tables

Tables are an ideal place to use the CSS pseudo-classes to define formatting. As an example, let's say we want the first column (the first <td>) to be bold, the next column right aligned and the third italic. All we need is the "td" selector and the "nth-child(*x*)" pseudo-class:

```
<style>
  table tr td { ; border-width:1px; border-style:solid;  }

  td:nth-child(1) { font-weight:bold; }  /* menu item */
  td:nth-child(2) { text-align:right; }  /* price */
  td:nth-child(3) { font-style:italic; } /* notes */
</style>
```

Classic burger	$6.95	*Comes*
Cheese burger	$7.95	*with*
The Everything!	$8.95	*fries!*

Note that the word "child" is not referring to the nth child of a <td>. It's the first child of the parent (<tr>) that is a <td>.

If you wanted each row to have a different background color then change the color for just the even rows. Use CSS to set the background color for even rows:

```
tr:nth-child(even) { background-color:lightgray }
```

Fish		
Salmon	$19.95	
Catfish	$12.95	
Trout	$24.95	

Some of the CSS Pseudo-classes useful for tables are listed below. Note that these are not limited to only tables and can be used with lists (<ol>, <ul>) and other tags.

:first-child :last-child	The first or last of the element in the same parent element. For example: td:last-child will select the last td in a row (tr). :first-child is the same as :nth-child(1)
:first-of-type :last-of-type	Select the first element of the selected type. The difference between :first-child and :first-of-type is when a parent tag has mixed child types. As an example, a row could have a <th> as the first cell and <td>'s for the remaining cells. td:first-child will not select any cell. (There is no <td> that is the first child!) td:first-of-type would select the first <td> (which happens to be the second cell in the row).
:nth-child(*n*) :nth-last-child(n)	Values for "n": a number (if n is "5" then this would only select the 5th element) odd, even 2n Same as even 3n+2 Every 3rd element starting with 2 (2, 5, 7…) 2n+1 Same as odd (1, 3, 5…)

Table Sections

A table can be broken into three sections: thead, tbody and tfoot. These can be used for styling or to add contextual value. By default, these three tags have no styling and will not impact the layout of a table.

<thead>

There can be at most only one <thead>. This section typically contains column headings.

<tfoot>

There can be at most only one <tfoot>. This section typically contains totals or summary info.

<tbody>

A <tbody> contains the table's data. A <tbody> exists for every table even if not explicitly defined. Multiple <tbody> tags can be added to define groups of rows.

An example with styling:

```
<style>
  /* required for borders on tbody */
  table { border-collapse: collapse; }

  /* add a border on the tbody */
  tbody { border-width:1px; border-style:solid; }

  /* additional styling */
  thead { font-size:16pt; font-weight:bold; background-color:lightgray; }
  tbody { font-size:12pt; }
  tfoot { font-size:12pt; font-style:italic; background-color:lightgray; }

  td:nth-child(2) { text-align:right; }
</style>
```

```
<table>
  <thead>
    <tr><td>Item</td><td>Price</td></tr>
  </thead>

  <tfoot>
    <tr><td>Total</td><td>$105.55</td></tr>
  </tfoot>

  <tbody>
    <tr><td>Salmon</td><td>$19.95</td></tr>
    <tr><td>Catfish</td><td>$12.95</td></tr>
    <tr><td>Trout</td><td>$24.95</td></tr>
  </tbody>

  <tbody>
    <tr><td>Classic burger</td><td>$6.95</td></tr>
    <tr><td>Cheese burger</td><td>$7.95</td></tr>
    <tr><td>The Everything!</td><td>$8.95</td></tr>
  </tbody>

  <tbody>
    <tr><td>Classic hot dog</td><td>$6.95</td></tr>
    <tr><td>Super HOT dog</td><td>$7.95</td></tr>
    <tr><td>Cheese dog</td><td>$8.95</td></tr>
  </tbody>
</table>
```

Item	Price
Salmon	$19.95
Catfish	$12.95
Trout	$24.95
Classic burger	$6.95
Cheese burger	$7.95
The Everything!	$8.95
Classic hot dog	$6.95
Super HOT dog	$7.95
Cheese dog	$8.95
Total	*$105.55*

DIVs

A is simply a container and can be thought of as a division of a document. It is often used to group blocks of content.

- By default:
    - Has a break between it and the previous and next object.

  - Has no visual features.
- Can contain most other HTML tags.
- Can be styled to be displayed in many ways.

Try it!

1. Open your HTML editor. (Notepad will do.)
2. Add two tags as follows:
```
<div>This is DIV 1</div><div>This is DIV 2</div>
```
3. Save the file and open in a browser.

Question:

- Is the text on one or two lines? _____

4. Add the following CSS to add a border around the DIVs and refresh the page in the browser.
```
<style>
  div { border-width: 1 px; border-style: solid; }
</style>
```

Question:

- How wide is a DIV by default? 1. Width of the text. 2. Width of the page.

5. Add "display: inline" to the DIV's style and refresh the page in the browser.
```
<style>
  div { border-width: 1 px; border-style: solid; display: inline; }
</style>
```

Questions:

- Is the text on one or two lines? _____
- How wide is a DIV without the default line break? 1. Width of the text. 2. Width of the page.

Float

CSS can be used to "float" content so that text and other content can flow around the floated content. Float can be used to:

- Wrap text around images.
- Wrap text around sidebars.
- Create drop caps. (The first letter of a paragraph is enlarged and dropped down two or more lines.)
- Create adaptive designs where several DIVs will be displayed across a page on a wide monitor, but be moved into multiple lines of DIVs on smaller displays,

🗁 **Sample file: flowers.html**

Float options

- float:left – object stays to the left and other content flows around the right side.
- float:right – object stays to the right and other content flows around the left side.
- float:none

Clear

When using float you often need to make sure that the next element starts below the previous floated item. You could also describe this a making sure that nothing is floating next to the styled item.

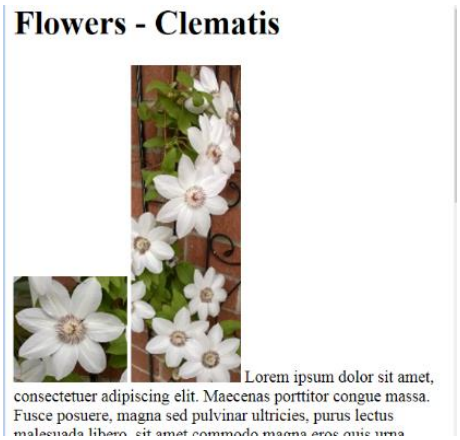
- clear:none
- clear:left – clear items floated ot the left.
- clear:right – clear items floated ot the right.
- clear:both

Float with Images

The following HTML displays two pictures and some text without using float. Notice that the images behave as text as if they were characters in the line.

```
<h1>Flowers - Clematis</h1>

<img src="..\images\flowers1-10.jpg">
<img src="..\images\flowers3-10.jpg">

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas
Vivamus a tellus. Pellentesque habitant morbi tristique senectus et
In porttitor. Donec laoreet nonummy augue. Suspendisse dui purus, s
```

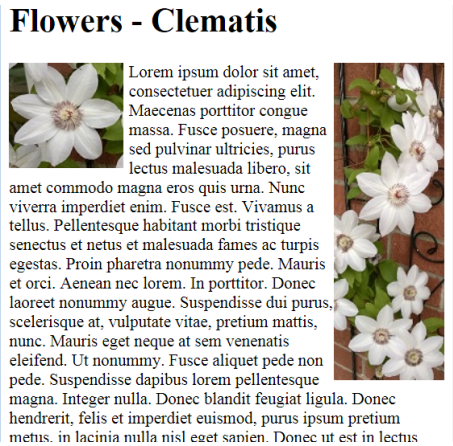
The result:



If we float the first picture the left and the second to the right, we get a layout where the text wraps around the images. You can adjust the image margins as needed to keep the text from touching the pictures.

```
<h1>Flowers - Clematis</h1>

<img src="..\images\flowers1-10.jpg" style="float:left;margin-left:5px">
<img src="..\images\flowers3-10.jpg" style="float:right">

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas portti
Vivamus a tellus. Pellentesque habitant morbi tristique senectus et netus
In porttitor. Donec laoreet nonummy augue. Suspendisse dui purus, scelori
```

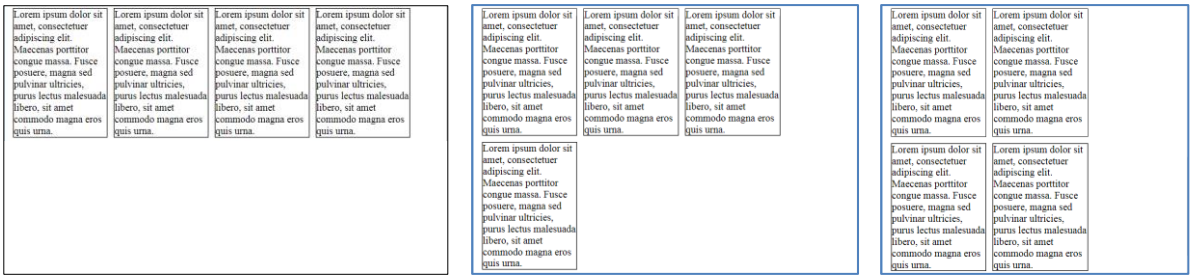
Float adapts to the page size! You can also design your CSS code so the size of the images also changes with the page size.



Float for Wrapping

DIVS, images and other block-level objects can float and adapt to the space available in a page. In the example below, the four DIVs have been styled to “float”. As the page is resized they move from four across to whatever will fit for the page width.

```
<div style="width:150px; float:left;margin:5px; border:1px solid">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas por
</div>
<div style="width:150px; float:left;margin:5px; border:1px solid">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas por
</div>
<div style="width:150px; float:left;margin:5px; border:1px solid">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas por
</div>
<div style="width:150px; float:left;margin:5px; border:1px solid">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas por
</div>
```



SPAN

A has no default display characteristics and is most often used to style a block of text. It is also used with an ID to allow JavaScript to change text within the .

- A can contain text and other inline elements, including other tags.
- A cannot contain block-level elements. (such as DIVs)
- A by default does not add before or after line breaks.
- A without CSS will not change the display of its contents.

Example:

```
The quick <span style="color:brown">brown</span> fox …
```

DIV vs. SPAN

While a is a container of text and/or other HTML tags, a is only a container of "inline elements". A cannot contain block elements.

Examples of inline elements: <b> <i> <emp> <img>
 <a>

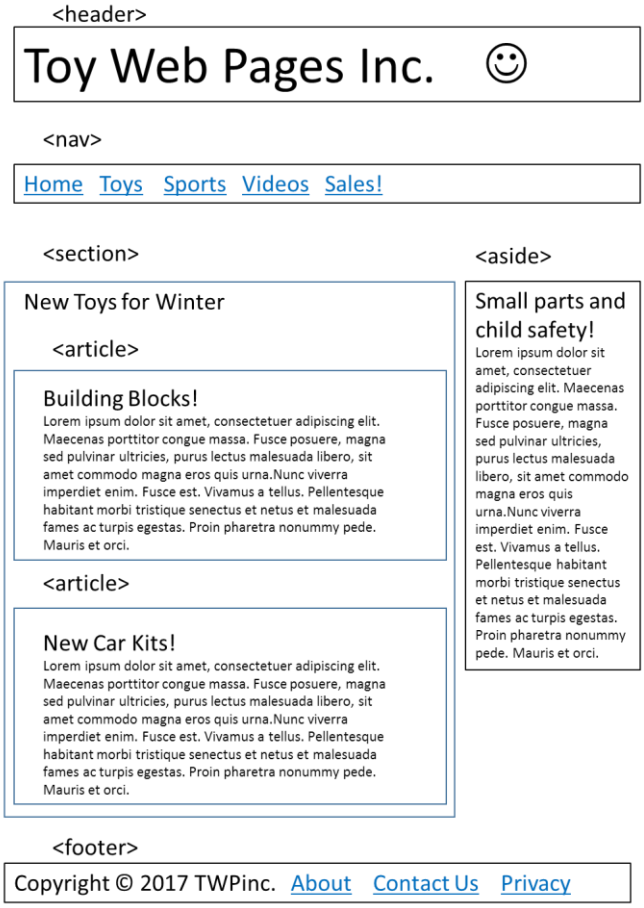
Examples of block elements: < h1> <header> <section> <form> <hr>   <video>

HTML 5 DIV-like Tags

HTML 5 introduced a number of DIV-like tags to better define the purpose of content. These contextual tags provide valuable information to search engines. Before HTML 5, search engines and developers have guessed the purpose of a by looking at its ID attribute. For example, " " would imply that the DIV contains company, copyright and privacy links. But, this only works if the web designer used common words like "footer", "foot" and "copyright" for the ID.

HTML 5 now provides a collection of contextual tags, such as <footer>, to make determining the purpose of the contents of tags a lot more consistent. These include: <article>, <aside>, <details>, <figcaption>, <figure>, <footer>, <header>, <main>, <mark>, <nav>, <section>, <summary> and <time>.

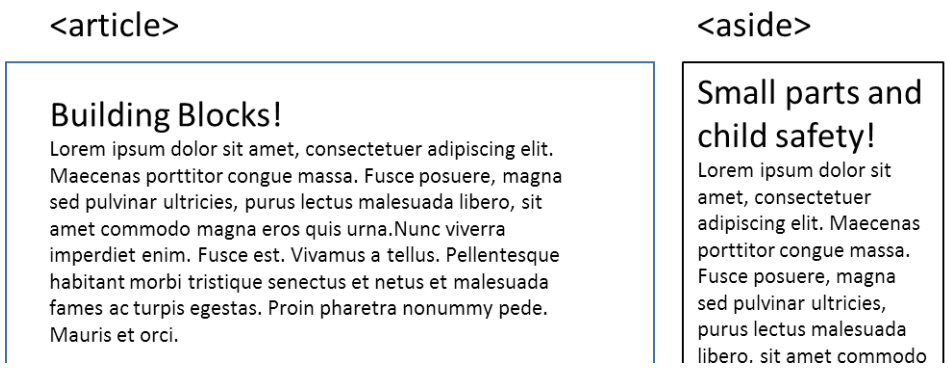
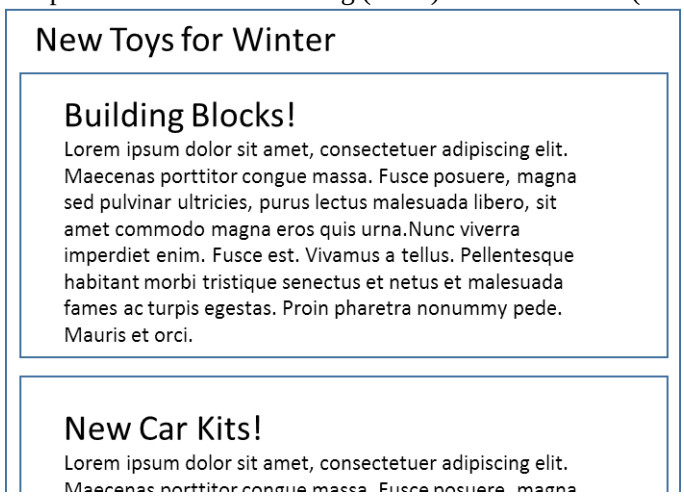
🗁 **Sample file: html5contextual.html**



- The primary use of these tags is to assist search engines to understand the purpose of each part of the page.
- By default, these tags are treated as DIVs and do not have any built-in formatting. Unless custom styled using CSS, these tags behave the same as a DIV.
- These tags are not recognized by many older, pre-HTML 5, browsers. For example, Internet Explorer 8 and earlier ignore these tags.
    - CSS support for these tags can be added by adding a JavaScript library to force the older browsers to recognize these new tags. This kind of library is often referred to as a “shiv” or a “shim”. The most common shiv for these new tags is probably “HTML 5shiv” and is available from github (https://github.com/aFarkas/HTML 5shiv/) and many Content Delivery Networks (CDNs).

- As this is often an Internet Explorer related issue, Internet Explorer comments are often used to insure that this library is only loaded for these older browsers.

```
<!--[if lt IE 9]>
  <script src="HTML 5shiv.js"></script>
<![endif]-->
```

<header>	Typically includes introductory or navigational aids and often includes search boxes and logos. It does not contain the primary page content.
<footer>	Typically includes copyright messages and links to "about us", "privacy statement", etc. Copyright © 2017 TWPinc. About Contact Us Privacy
<nav>	Typically includes site navigation. Home Toys Sports Videos Sales!
<aside>	A common use is as a side bar. While an <aside> should in some way be related to the nearby content, it is just a container. 
<section>	A block of related content. Typically includes a heading. Example: Section with a heading (<h1>) and two articles (<article>): 
<figure> <figcaption>	A block of independent content, typically with a caption.

```
<figure>
  <img
  src="..\images\flowers1-10.jpg"
  alt="A flower!">
  <figcaption>Fig 1. Clematis</figcaption>
</figure>
```

Fig 1. Clematis

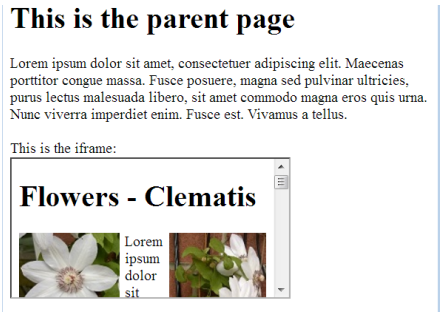
🗁 **Sample file: flowers.html**

IFRAMEs

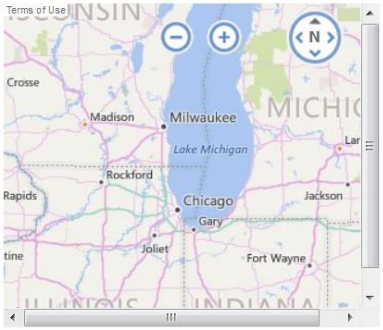
IFRAMEs are used to embed an HTML document in an HTML document. IFRAMEs are often used to display content from another web site such as a map service.

- The content of the IFRAME is only from the linked document.
  `<iframe src="OnSalePage.html"></iframe>`
- A message can be added between the tags for browsers that do not support IFRAMEs.
  `<iframe src="OnSalePage.html">iframes are not supported by this browser</iframe>`
- An IFRAME will often display with scrollbars.
- By default an IFRAME is an inline element.
- Attributes:
  - height - pixels
  - width - pixels
  - name
  - id
  - src

```
<h1>This is the parent page</h1>

Lorem ipsum dolor sit amet, consectetu
<br><br>

This is the iframe:<br>

<iframe src="flowers.html"></iframe>
```



The mapping sites usually supply linkable maps that you can display in an IFRAME. As an example, Bing Maps has a page that will write the code for you: http://www.bing.com/maps/embed/Customize.aspx

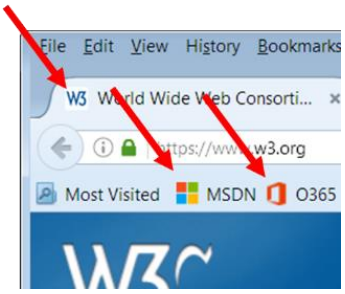



```
<div>
 <iframe
   width="325"
   height="280"
   frameborder="0"
   src="http://www.bing.com/maps/embed/viewer.aspx?v=
 </iframe>
</div>
```

Module 8: Images

Favicon

One very special image you can add to a web site is the icon to be displayed in the browser's tab or address bar. Each brand of browser will display these icons differently. Most will display them in the tabs and bookmarks, and some will display them in the address bar or links bar.



The image must be 16x16 pixels or 32x32 pixels, using either 8-bit or 24-bit colors and in one of the following formats: PNG, GIF, or ICO. (IE 10 and earlier only support ICO.)

The original way browsers found this icon was to simply look in the root of your site for a file named favicon.ico. (I.e. http://www.example.com/favicon.ico) The preferred approach is to add a <link> to the <head> section of the page. The "rel" attribute can be either "shortcut icon" or "icon". The "type" attribute can generally be omitted. Here are three examples:

```
<link rel="shortcut icon" href=" /images/favicon.ico" type="image/x-icon" />

<link rel="icon" type="image/png" href="http://example.com/myicon.png">

<link rel="icon" href="myicon.gif" />
```

Favicon has had an unusual browser support history. For more details see:
https://en.wikipedia.org/wiki/Favicon

Preparing Images

Whenever possible, pre-size and process images before using in a web page. While HTML and CSS include many options to resize, grayscale and recolor images, you will get best results by first adjusting the images in a paint or photo editing program.

Some examples of photo editors include:

- Windows Paint (free!)
- Photoshop Online Tools / Photoshop Express Editor http://www.photoshop.com/tools (free and online)

- GIMP (GNU Image Manipulation Program) (free)

Image Files

File Size

Each pixel requires 1 to many bits to describe the color. Black and white images may only need one bit while gray scaled or color images may require 24 bits to represent 2^{24} colors (16,777,216).

Compression

Some file types support compression to reduce the size of the files.

- Lossless compression will result in a smaller file size without any data loss. A pixel by pixel comparison will show no differences.
- Lossy compression will result in a smaller file, but will result in an image that is not an exact copy of the original.

Image Files Types and Features

When selecting an image format consider:

- Number of colors supported
- File size / compressibility
- Lossless or lossy compression
- Support for animation
- Support by all browsers

	BMP Winows **Bit**map	**JPG / JPEG** **J**oint **P**hotographic **E**xperts **G**roup	**GIF** **G**raphics **I**nterchange **F**ormat	**PNG** **P**ortable **N**etwork **G**raphics
Colors	2^4 (16), 2^{16} (65,536) or 2^{24} (16,777,216)	2^{24} (16,777,216)	2^8 (256)	2^{24} (16,777,216) or 2^{48}
Compression	None	Lossy	Lossless	Lossless
Compatibility	Typically limited to Windows computers			
Animation			Yes	APNG files support animation but have limited browser support.
Supports transparent backgrounds			Yes	Yes
Notes	Default format used by Windows Paint.		Conversions from other formats often causes a loss of color.	

Browser Support

JPG, GIF and PNG are supported by all of the modern browsers. For a more complete list see:
https://en.wikipedia.org/wiki/Comparison_of_web_browsers#Image_format_support

The IMG Tag

Images can be displayed in a page using the <img> tag or by loading the image as a background to other elements.

The simplest <img> tag has only a “src” attribute to specify the image to be loaded. The image tag has no end tag.

```
<img src="logo.gif">
<img src="logo.gif" alt="Team logo" height="100" width="200">
```

Defaults:

- Height and width are set to the size of the image.
- <img> is an inline tag and the bottom of the image is aligned with the baseline of the text.

Best Practices:

- Images should be pre-sized and not rescaled by the height and width attributes.

- The height and width attributes should be provided to reserve space for the image.
- The “alt” attribute should contain text to describe the image. This used for non-graphical interfaces, screen reader applications and images with broken links.

Common Attributes

src	Absolute or relative URL to an image file. src="logo.gif" src="http://www.example.com/logo.gif" src="..\images\logo.gif" Notes: “src” can also point to a server side page that dynamically creates an image and returns it as a stream of bytes. src="createchart.aspx?customer=123" “src” can include the image as Base64 encoded data. The following represents a single black pixel. <img src=" AAAABAAEAAAAAAAQAAAB0EgAAdBIAAAAAAAAAAAAAAAAAAAAP///wAAAAAA" >
height, width	Height and width in pixels. Use CSS to use other units. Typically should match the actual image.
longdesc	A URL to a page with a detailed description of the image.
usemap	Partial URL to an image map (<map>…</map>) that defines clickable regions of the image. See Image Maps below.
ismap	When the <img> tag is included inside of an anchor tag (<a>) then the coordinates of the user’s click are appended to the anchor’s href URL as a query string. This is a Boolean value. Clicking <img src="products.png" ismap> will redirect to productdetails.aspx**?x,y** where x and y are the x and y coordinates of the user’s click.

File Paths

When working with images from the same server as the containing page, you will usually work with a relative path to the file. Absolute paths should only be used when linking to resources on another web server.

A path that does not include a folder name will refer to a file in the same folder as the page while a path with a folder will refer to a file in a folder below the page’s location.

Absolute paths to external files:

```
http://www.example.com/images/puppy.jpeg
http://www.example.com/logo.gif
```

Relative paths (relative to the page's location):

```
images/puppy.jpeg
logo.gif
```

Relative paths (relative to the root of the site):

```
/images/puppy.jpeg
/logo.gif
```

Relative paths (up one folder from the page's folder):

```
../images/puppy.jpeg
```

The ".." in the last sample path means "up one directory from here". I.e. go up one directory, go into the images directory and get the puppy.jpeg file. If the page "report1.html" in this example was in the "Reports" folder then "../images/puppy.jpeg" would be found this way:

```
C:\website\home.html
C:\website\images\puppy.jpeg
C:\website\images\logo.gif
C:\website\reports\reports.html
C:\website\reports\report1.html
```

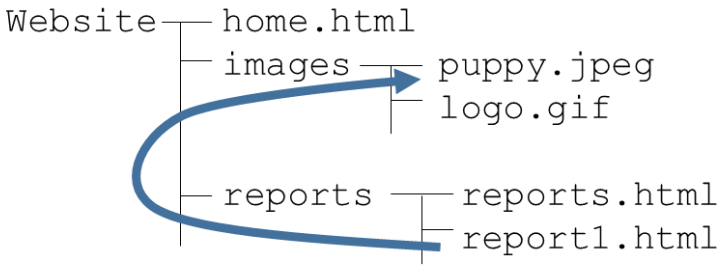
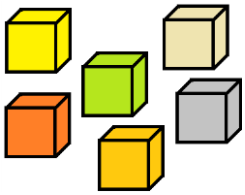


Image Maps

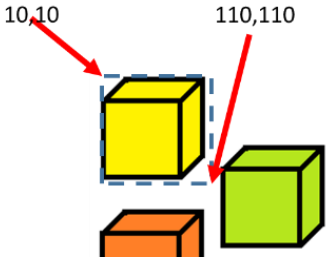
Image maps are used to add clickable areas to an image. You will need to identify the locations of each clickable spot and add the x and y coordinates to a <map> tag. Many web editors include an image map tool that creates the <map> HTML for you. Several online image map tools are also available.

🗁 **Sample file: ImageMap.html**

For an example, each of the boxes in the image below should link to a different URL.



For the first box, the top left corner is located at x=10 and y=10 and the bottom right corner is at x=110 and y=110.



To make the box clickable we need an <img> tag, a <map> tag and an <area> tag.

- The image tag needs a “usemap” attribute with the ID of the <map> tag.
- The <map> tag needs an ID attribute.
- The <area> tags define:
    - The ID for the area. (for JavaScript use)
    - An optional title to use as a mouse over tip.
    - An HREF for the click destination.
    - A shape type: rect, circle or poly.
    - A set of coordinates that outline the clickable area.
    - Optional target to specify where the link opens.

```
<img src="../images/ClickABox.png" usemap="#BoxMap">

<map id="BoxMap">
 <area id="Box1" alt="" title="Box 1" href="../images/flowers1.jpg"
  shape="rect" coords="10,10,110,110" style="outline:none;" target="_self"/>
 <area id="Box2" alt="" title="Box 2" href="../images/flowers2.jpg"
  shape="rect" coords="7,142,114,246" style="outline:none;" target="_self"/>
</map>
```

The “coords” attribute:

- x1,y1,x2,y2 - is used to define a “shape=rect” area.
- x,y,radius - is used to define a “shape=circle” area. X and y are the center of the circle.
- x1,y1,x2,y2,..,xn,yn - is used to define a polygon, “shape=poly” area. Each x,y is one point of the polygon.

Background Images

You can supply background images to the <body> tag to include an image for the entire page. You can also supply background images for almost every tag and for borders.

Make sure that your background images do not make text unreadable. Either lighten, fade or grayscale the image in a photo editor, or apply CSS to lighten the image. (Example: opacity: 0.3;)

For a Page

The background image for a page can be specified in the <body> tag. The image will be repeated to fill the available space. While you can use the "background" attribute, this has been deprecated in HTML 5 in favor of using CSS.

```
<body background="greylogo.jpg">
<body style="background-image:url(greylogo.jpg)">
```

Or better:

```
<style>
  body { backgroud-image:url(greylogo.jpg); }
</style>
…
<body>
```

To keep background images from scrolling add:

```
background-attachment: fixed;
```

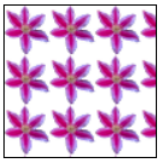
For a DIV or other Tag

Images are often added as backgrounds to or other tags so that the image can be selected from a style sheet.

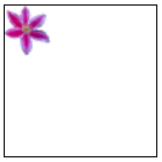
```
div.bg1 { backgroud-image:url(background1.jpg); }
<div class="bg1"> some content </div>
```

By default the image will be repeated to fill the available space. You can control this with "background-repeat".

No repeat specified.

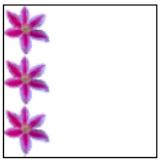


background-repeat:no-repeat



background-repeat:repeat-x

background-repeat:repeat-y

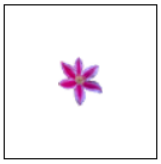


🗁 **Sample file: backgrounds.html**

If the background image is not repeated, you may want to set the location of the image:

- background-position: *value*
  where value is one of the following:
  - center top
  - center center
  - center bottom
  - left top
  - left center
  - left bottom
  - right top
  - right center
  - right bottom
  - x% y% where the top left is 0% 0%
  - x y where the top left is 0px 0px

background-repeat:repeat-y;
background-position:center center;



background-repeat:no-repeat;
background-position:botttom right;

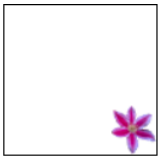


Image Best Practices

- If the image is going to be changed using CSS based on the device or screen size then use CSS background-image and not the <img> tag.
- Pre-size the image to fit the planned use. Do not download big images and then shrink using Height and Width.
- Test images for all uses: printing, high contrast, mobile devices, etc.

CSS Sprites

Consider a page with 21 icons. Each visitor has to download 21 separate files using 21 requests. If we combine all of those images into a single file we can then download them in one request. To use any one of the images we will need three things:

- The sprites file.
- Some tag were we can set the background image. A will do, but any element that supports a background image will work.
- Some CSS to select the part of the sprite to display.
    - Set the height and width of the span.
    - Set the span to display as a block.
    - Shift the image inside of the span to put our icon within the span.

As an example, let's take this collection of icons. Each is 24px by 24px. (Five have boxes drawn around them for clarity.)

Now let's create a span to hold the icon. It's just a simple span with a class name.

```
The <span class="iconRightArrow"></span> icon starts the video!
```

Add a little CSS and will get just the one icon we need. Note that to get the right arrow icon we will need to shift the image 4 times 24 pixels to the left. I.e. -96px and shift the image up 24 pixels, or -24px.

```
<style>
 .iconRightArrow {
  display:inline-block;  /* expand the span to it's height and width */
  height:24px;  /* height and width of a single icon */
  width:24px;
  background:url(samplesprites.png) -96px -24px;  /* shift and display the image */
 }
</style>
```

The result:

The  icon starts the video!

If we wanted the down arrow we would change the "shift" to 0px -48px.

```
<style>
 .iconDownArrow {
  display:inline-block;  /* expand the span to it's height and width */
  height:24px;  /* height and width of a single icon */
  width:24px;
  background:url(samplesprites.png) 0px -48px;  /* shift and display the image */
 }
</style>

Press <span class="iconDownArrow"></span> for the next video.
```

To use the sprite icon as a button just wrap the span in an <a> tag.

```
<a href="helloworld.html"><span class="iconRightArrow"></span></a>
```

To make the image change during a mouse over just use the "hover" Pseudo-class and select a different image.

```
.iconRightArrow:hover {
  display:inline-block;
  height:24px;
  width:24px;
  background:url(samplesprites.png) -48px -24px;   /* location of some other icon */
 }
```

Sprites can be used to create all kinds of interesting effects. Do a web search for "CSS Sprites" for examples. If you don't want to create your own sprite files then search the web for "CSS Sprite creator" or "CSS Sprite editor". Some of these can even scan your page and give you both the sprite file and the sample CSS to use it.

Module 9: HTML Forms

Early HTML was designed to create electronic documents using markup tags. HTML 2.0 added forms so we could collect data from site visitors. Form controls include text boxes, select lists and select areas along with radio buttons, check boxes and submit buttons. Forms can also include most of the HTML and CSS you have already seen to layout and style the form.

Forms have two major components, HTML in a web page, and server-side processing of the data. While there are a few browser based JavaScript processed form applications, most data collected by forms will be sent to a server to request data or to update SQL databases.

HTML 5 introduced a large number of form tags to make it easier to build self-validating forms. As with many newer features, there has been inconsistent implementations of these features. Always consult sites like http://caniuse.com to check the current support status of these features.

In this course, we will be manually building forms. Many of the HTML editors include drag and drop forms designers, as do many online tools, to make form creation easier.

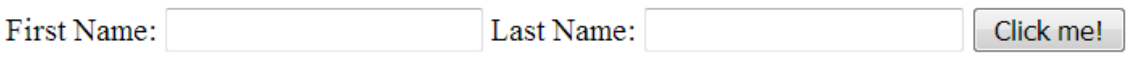
Form resources:

- W3.org - http://www.w3.org/TR/html51/sec-forms.html#sec-forms
- Mozilla/Firefox - https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms
- caniuse.com - http://caniuse.com/#search=HTML5%20form%20features
- http://www.w3schools.com/tags/tag_form.asp
- http://www.w3schools.com/html/html_forms.asp (tutorial)
- http://www.w3schools.com/html/html_form_elements.asp
- http://www.w3schools.com/html/html_form_input_types.asp
- http://www.w3schools.com/html/html_form_attributes.asp

A Basic Form

The most basic form has <form> tags, at least one <input>, <select> or <textarea> element, and a button to perform some action. The sample form below does not send any data to a server, instead it uses JavaScript to display a message.

🗁 **Sample file: forms1.html**



```
<form name="form1">

  First Name:  <input type="text" name="firstName">
  Last Name:  <input type="text" name="lastName">

  <button onclick="alert('Welcome ' +
                  form1.firstName.value + ' ' +
                  form1.lastName.value)"
  >Click me!</button>

</form>
```

Most forms send data to a web server. In the next example the form submits to another page on the server. (This page is just a dummy for testing.) As the default action of a button in a form is to submit the form, all it needs is some text to display.

Submitted forms need at least two attributes:

- method – How to send the data. Either GET or POST. (See POST vs. GET below.)
- action – The URL of the page to receive the data.

The form also needs either a <button> or <input type=submit> to submit the form.

A basic form that submits to another page using GET

First Name: Last Name: Click me!

```
<form name="form1" method="GET" action="formresultsdummy.html">

  First Name: <input type="text" name="firstName">
  Last Name: <input type="text" name="lastName">

  <button>Click me!</button>

</form>
```

After clicking the button, the browser will redirect to the page listed in the action. As this form is using method=GET, the data will be passed as a visible query string appended to the URL.

/formresultsdummy.html?firstName=Sam&lastName=Jones

POST vs. GET

The HTTP protocol has defined a number of "verbs" to describe the purpose of a web server request. While this list is a bit long; post, get, put, merge, delete, and a few others; only two are typically used with HTML forms: PUT and GET.

To understand the difference between PUT and GET:

- A GET in its simplest form is just a URL with data attached.
- A POST is a URL plus additional data sent with the request. This additional data is the HTTP Header. This data is in the form of name/value pairs for each item in the form.

- A GET can also have an HTTP Header sent with it.

Benefits of a GET

- Easy to use.
- Sent as part of a URL, so bookmarkable and can be typed into an email.

Benefits of a POST

- When using HTTPS the data in the POST header is encrypted. Even when using HTTPS, the data in a query string is sent in clear text, visible to network sniffers and often visible in your browser's history.
- No limit to the about of data sent. A GET is typically limited to around 2000 characters.
- POSTs cannot be bookmarked and are not saved in the browser cache or history.

In general:

- GETs should only be used to request data from a server, not save data to a server. I.e. GETs should not have "side effects", such as changing something.
- GETs should not be used with confidential data.
- Using the BACK button will only re-request data with a proper GET, but potentially could repeat the save or update operation of a POST. (Think about a "add to cart" or "post payment" click being repeated.)
- GET URLs are stored in the browser history.
- Use POSTs when submitting a large amount of data.
- A POST is more secure than a GET. A POST with HTTPS is fully encrypted.

name vs. id

Each form control with data to be submitted must have a "name" attribute. This attribute represents the name of the property. Controls without names will not have their data submitted. The name used in your control will be sent to the server where it will be used for updates. Your selected name must match what the server is expecting or it will be ignored.

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

When a form is submitted the name and its value are sent as a pair: firstName=Sam

IDs are used for JavaScript and CSS to select a control. IDs are not sent to the server and are not needed to insure that data will be sent when the form is submitted.

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

Form controls will often have the same value for both the name and ID attributes.

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

Basic Form Elements

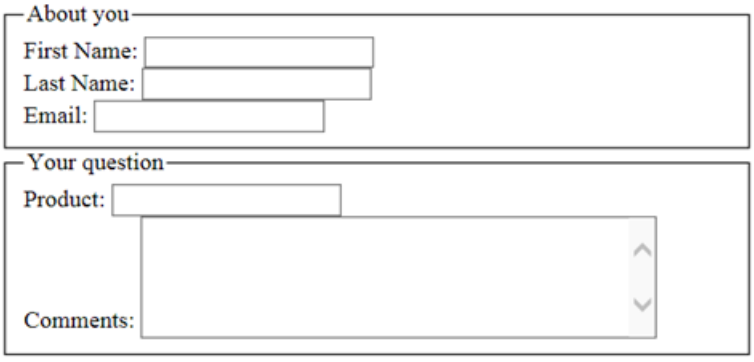
Also see “HTML 5 INPUT Types” later in this module.

🗁 **Sample file: forms2.html**

Type	Behavior when implemented:
input type="text"	General purpose text box. “text” is also the default type. <input type="text" name="firstName">
input type="checkbox"	Displays a checkbox. Use “checked” instead of “value” to set the default. Multiple checkboxes can be selected at a time from a group. All buttons in a group have the same name. The descriptive text will usually be displayed to the right of the checkbox. <input type="checkbox" name="options" value="rush"> Rush shipping
input type="file"	Used to upload files. See “Uploading Files” later in this module.
input type="hidden"	Used to submit data to a server, but not display it in the form. <input type="hidden" name="status" value="active">
input type="image"	Used to make an image into a button. <input type="image" src="../images/clickmeButton.png" alt="Submit">
input type="password"	Similar to type="text", but the characters are masked with a symbol to hide the password. This is not limited to passwords and can be used where data entry needs to hidden as it is typed. <input type="password" name="password">
input type="radio"	Displays a radio button. Use “checked” instead of “value” to set the default. Only one radio button can be selected at a time from a group. All buttons in a group have the same name. The descriptive text will usually be displayed to the right of the button. <input type="radio" name="payment" value="cash"> Cash
input type="reset"	Displays a button that resets all form values to their defaults (empty or the value of the “value” attribute). <input type="reset" value="Reset the form">

input type="submit"	Displays a submit button. When clicked will submit the form to the destination in the "action" attribute, or fire the onsubmit JavaScript event. The button text is set with the "value" attribute. <input type="submit" value="Submit">
select	Creates a dropdown list. <select> has both start and end tags and must contain one or more <option> tags. See "Select" later in this module. <select name="States"> <option value="CA">California</option> <option value="NY">New York</option> <option value="OH">Ohio</option> <option value="TX">Texas</option> </select>
textarea	Defines a multi-line input control that can hold an unlimited amount of text. Unlike the <input>tags, <textarea> has start and end tags. Default text is placed between the tags, not in a "value" attribute. <textarea> has two additional attributes not found in <input> tags. "rows" and "cols" define the displayed size of the <textarea>. "wrap" is either "soft" or "hard". "soft" sends the text as one line while hard preserves line breaks. <textarea name="comments" rows=5 cols=50 wrap="soft"></textarea>

label	A <label> is used to display text associated with a control. When the <label> "for" attribute is set to the ID of a control then a click on the <label> is treated as a click on the control. <label> tags make it easier for users to click radio buttons and checkboxes. <label> also offers sematic value to screen readers and other tools. <label **for="firstName"**>First Name</label> <input **id="firstName"** type="text" name="firstName"> Alternate use: **<label>**First Name<input type="text" name="firstName">**</label>**

fieldset	<fieldset> is used to group sections of a form together. The default formatting is to add a box around the fieldset and to add the caption into the top border line. `<form method="GET" action="formresultsdummy.html">` `  <fieldset>` `  <legend>About you</legend>` `  First Name: <input type="text" name="firstName">  ` `  Last Name: <input type="text" name="lastName">  ` `  Email: <input type="text" name="email">  ` `  </fieldset>` `  <fieldset>` `  <legend>Your question</legend>` `  Product: <input type="text" name="product">  ` `  Comments: <textarea name="comments" rows=5 cols=40></textarea>` `  </fieldset>` `  <input type="submit" value="Submit">` `</form>` 

Basic Form Attributes

These are the pre-HTML 5 attributes that still apply to HTML 5 pages. Older, pre-HTML 5, attributes are not listed here and generally should be avoided.

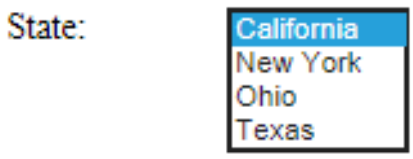
Also see “HTML 5 INPUT Attributes” later in this module.

Attribute	Behavior when implemented:
Alt	Alternate text, only for "type="image"" <input type="image" name="picture" **alt="Product picture"** >
checked	Pre-selects a checkbox or a radio button. This is a Boolean and is enabled by being present. Has no values. (But often seen typed as checked=checked) <input type="checkbox" name="ismember" **checked** >
disabled	The form element is disabled. Value is displayed if supplied, but is grayed and cannot be edited. Disabled fields are NOT sent to the server. This is a Boolean and is enabled by being present. Has no values. (Also see "readonly".) <input type="text" name="firstName" value="Susan" **disabled** >
maxlength	The maximum number of characters allowed. <input type="text" name="firstName" value="Susan" **maxlength="24"** >
name	Element name. This is sent to the server along with the data. firstName=Susan <input type="text" **name="firstName"** value="Susan" >
readonly	The form element is read-only and cannot be edited. It WILL be sent to the server using the value property. This is a Boolean and is enabled by being present. Has no values. (Also see "disabled".) <input type="text" name="firstName" value="Susan" **readonly** >
value	The default value of the field. This is often prepopulated from the server in pages used to edit server data. <input type="text" name="firstName" **value="Susan"** >

Select

The <select> tag is used to create dropdown lists. The content of a <select> is a series of <option> tags. The <option> tags contain the text to display in the dropdown, and optionally contain a "value" attribute. The "value" is sent when the form is submitted. If "value" is missing, then the displayed text is sent.

```
<select name="States">
  <option value="CA">California</option>
  <option value="NY">New York</option>
  <option value="OH">Ohio</option>
  <option value="TX">Texas</option>
</select>
```



Default

One of the options can be flagged as the default by adding the "selected" attribute.

```
<option value="OH" selected>Ohio</option>
```

optgroup

The <option> tags can be grouped into sets using <optgroup>. If <optgroup> is not supported by the user's browser, then the select list displays all of the <option> content as a single list. As each browser styles the optgroup text differently, you will want to add CSS to style these.

```
<select name="States">
  <optgroup label="Regional Offices">
    <option value="CA">California</option>
    <option value="NY">New York</option>
  </optgroup>
  <optgroup label="Sales Offices">
    <option value="OH">Ohio</option>
    <option value="TX">Texas</option>
  </optgroup>
</select>
```

Uploading Files

Uploading a file as part of a form submission requires an <input type=file> and two changes to the <form> tag. The HTML form only gets things ready for an upload. *You will have to create custom code on the server to receive and save the files.*

The <form> tag must have these two attributes:

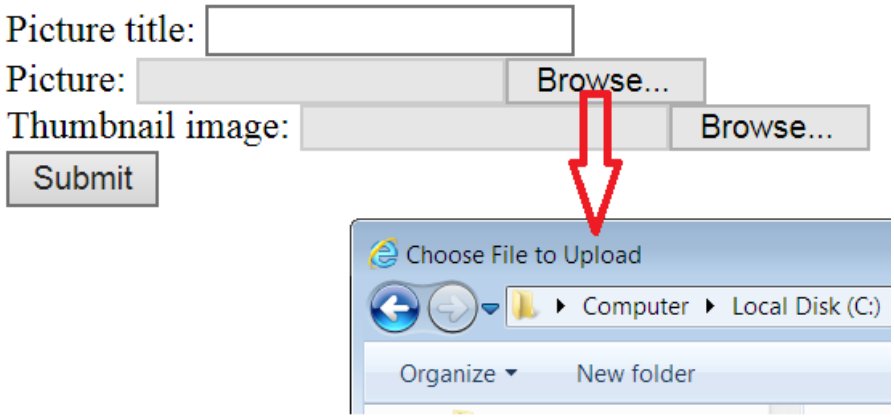
- method=POST
- enctype="multipart/form-data"

The form must include at least one <input type=file> element.

```
<form action="myuploadpage.aspx" method="post" enctype="multipart/form-data">
  Picture title: <input type="text" name="title" value=""> <br>
  Picture:  <input type="file" name="file1"> <br>
  Thumbnail image:  <input type="file" name="file2"><br>
  <button type="submit">Submit</button>
</form>
```

🗁 **Sample file: forms2.html**

The exact display of <input type=file> will vary by browser. Clicking the control's button will display a dialog box so you can select a file.

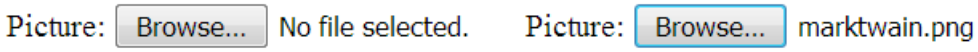


Internet Explorer displays a read-only text box and a "Browse" button.

Before and after selection:

Firefox displays a button followed by a message:

Before and after selection:



Input Type=file Attributes

- accept – Specifies the file type or mime type to accept. This causes the file selection dialog box to prefilter the files. The user can still override this and upload any file type.
    - *file_extension* Example: accept=".docx"
    - audio/* Common audio file types.
    - video/* Common video file types.
    - image/* Common image file types.
    - *media_type* For a list see: http://www.iana.org/assignments/media-types/media-types.xhtml

Better File Upload Options

As the file upload controls are not consistently implemented across all of the browsers, and because you might want to offer a nicer file upload experience, you may want to do a little research on the web on HTML file upload libraries and add-ins.

HTML 5 Form Enhancements

HTML 5 added many enhancements for forms. The only problem is that browsers and devices have very inconsistent support for them. Make sure you extensively test these in the browsers and devices you need to

support, and that you still validate the user's input by using both client side JavaScript and server side code. When testing with phones and tablets watch for changes to the touch keyboards. As an example, the <input type=email> may display a keyboard with ".com" and "@" keys.

New HTML 5 INPUT Types

When an HTML 5 type is not recognized by a browser, the input type will be treated as "text". Until these are more universally supported, you will want to use tools like jQueryUI to build the equivalent of these.

🗁 **Sample file: forms3.html**

Type	Behavior when implemented:
color	Displays a color picker, or access a hexadecimal color number. (#000000) (not supported by all browsers) <input type="color" name="colorchoice">
date	Displays a date picker and/or validates the date. (not supported by all browsers) Format: YYYY-MM-DD <input type="date" name="orderdate"> <input type="date" name="orderdate" max="2017-12-31" min="2017-01-01">
datetime	Currently displays the same as "text" in most browsers.
datetime-local	Currently displays the same as "text" in most browsers.
email	Mobile devices may display an email oriented keyboard that includes "@" and ".com" buttons.
month	Currently displays the same as "text" in most browsers. <input type="month" name="ordermonth"> Format: YYYY-MM
number	Limits data entry to just numeric values.
range	Limits input to a range using the "min" and "max" attributes.
search	Currently displays the same as "text" in most browsers. Could be styled to include a search icon.
tel	Currently displays the same as "text" in most browsers. You could add a regular expression to match phone numbers.
time	Used to enter a time. Format: HH:MM Currently displays the same as "text" in most browsers, while some may display a time picker. <input type="time name="ordertime">
url	Validates the entry as a URL. "http://www.example.com"
week	Used to enter a week and a year. Currently displays the same as "text" in most browsers, while some may display a date picker.

New HTML 5 INPUT Attributes

These attributes are new to HTML 5 and are not consistently implemented. There are JavaScript shims available to make these work in most new browsers. *Test, test, test!*

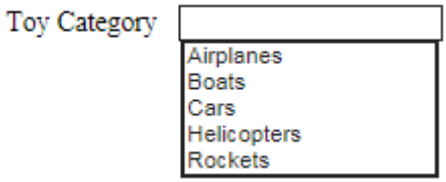
Attribute	Behavior when implemented:
autocomplete	"on" or "off". The browser will offer values previously entered in other fields with the same name. Works with text, search, url, tel, email, password, date pickers, range, and color. <input name="qty" type="number" autocomplete="on" >
autofocus	Cursor will jump to this field when the form is loaded. <input name="qty" type="number" autofocus >
list	Specifies the <datalist> element that contains a list of options for an <input> element.
min, max	Validation. Minimum or maximum value for a number or date. <input name="qty" type="number" min="5" max="50">
pattern	Validation. Specifies the regular expression to be used for validation. ZIP Code: <input name="zip" type="test" pattern="^\d{5}(-\d{4})?$" >
placeholder	Specifies a short tip. Usually displayed as the background for a textbox. <input name="desc" type="text" placeholder="enter a short description" >
required	Validation. Field must be supplied. <input name="qty" type="number" required >
step	Validation. Interval for entered values. Typically used with min and max. To only allow 5,10,15,20,… <input name="qty" type="number" min="5" max="50" step="5">

DataList

DataList is an HTML 5 update to the <select> tag that offers "select as you type" where the dropdown is filtered as the user types the first few characters.

If the browser does not support <datalist> then the <input> is displayed as a default <input> tag and the user simply types their text. At this time DataList is not well supported. See http://caniuse.com/#search=datalist

```
<input list="toys">
<datalist id="toys">
  <option value="Airplanes">
  <option value="Boats">
  <option value="Cars">
  <option value="Helicopters">
  <option value="Rockets">
</datalist>
```



Form Tools

There are form editors built into some of the HTML editors. There are also web sites where you can build, and then copy, forms. (Example: https://www.wufoo.com/)

Module 10: Multimedia

Video and Audio

You have two options for supporting video in a web page: using a plugin or using the HTML 5 <video> element. The use of plugins is disappearing as too many devices no longer support them.

Plug-ins:

- Adobe's Flash
- Apple's Quick Time
- Microsoft's Media Player plugin

HTML 5 Video

The HTML 5 standard added a <video> tag to play videos without the need for plugins. Most browsers supported at least some level of video support starting in 2009-2011. While the <video> tag is widely supported, not all browsers support the same list of video file formats.

Video Formats and Browser Support

- WebM – Chrome, Firefox and Opera
- Ogg/Theora – Chrome, Firefox and Opera
- MPEG-4/H.264 – Most modern browsers

For more details, issues and browser support see:

- caniuse.com - http://caniuse.com/#search=video
- W3Schools - https://www.w3schools.com/html/html5_video.asp
- Wikipedia - https://en.wikipedia.org/wiki/HTML5_video
- Mozilla.org - https://developer.mozilla.org/en-US/docs/Web/HTML/Supported_media_formats
- http://html5test.com

HTML 5 Video

While the HTML 5 <video> tag supports a lot of options, the most basic format only needs the height, width and src attributes.

```
<video height="178" width="320" src="snow320x178low.mp4"></video>
```

🗁 **Sample files: Videos/Video.html, Videos/VideoSizes.html, Videos/VideoOptions.html** (The individual video files are in the same folder.)

Controls

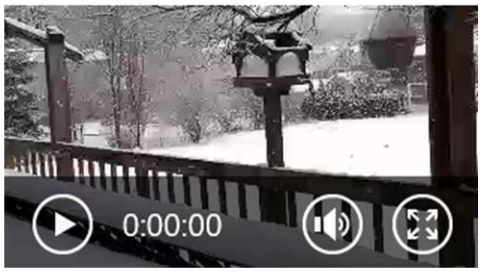
You can either use the browser's built-in video player controls or create your own using JavaScript and the HTML 5 video API. "controls" is an attribute that is "true" by being present. Without the controls, loop or autoplay attributes the user will need to know to right-click the video and click **Play**.

```
<video height="178" width="320" src="snow320x178low.mp4" controls></video>
```

Each browser's controls are different.

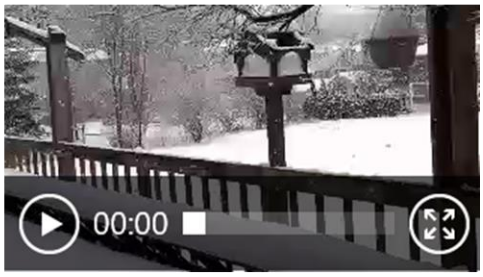
Firefox

Internet Explorer 11

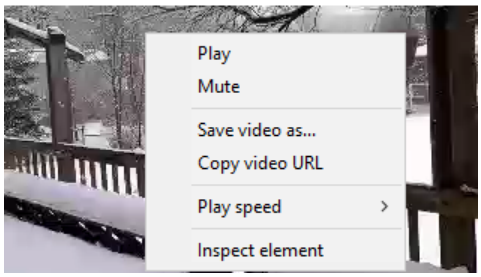


Chrome

Edge



If the controls are not displayed, the user can still right-click the video and start the video from a popup.



If you need a consistent display of controls then create your own HTML/CSS controls and write a little JavaScript to use the HTML 5 Video API.

Attributes

height, width	The browser will resize the video to fit the specified height and width. Best practice is to presize the video and only deliver minimum size needed.
src	Specifies the URL to the video file. Use the <source> child tag if multiple formats are available.
autoplay	Automatically starts the video.
controls	Displays the browser's default video control panel.
loop	Video will continuously play.
muted	The audio will be muted. The user can click the mute button to hear the audio if the controls are displayed.
poster	Sets the URL to an image to be displayed while the video is being downloaded or before the user clicks play.

Video Fallback

The HTML 5 <video> tag is not supported by all browsers, and not all common video formats are supported by all HTML 5 browsers. All of the data needed for the <video> tag is stored in the tag's attributes, or the tag's child element's attributes. Any content inside of the <video> tag is ignored by HTML 5 browsers.

```
<video controls width="300" height="200" >
  <source src="cutekittens.ogg" type="video/ogg">
  <source src="cutekittens.mp4" type="video/mp4">
    Your browser does not support HTML 5 video.
</video>
```

Browsers ignore unrecognized tags. Pre-HTML5 browsers would therefore ignore the <video> and <source> tags and display the "Your browser does not support HTML 5 video" text. Instead of embedding text, you can embed an alternate video technology.

```
<!-- First choice is HTML 5 -->
<video controls width="300" height="200" >
  <source src="cutekitten.mp4" type="video/mp4">
  <source src="cutekitten.m4v" type="video/x-m4v">

  <!-- Flash fallback! -->
  <object type="application/x-shockwave-flash" data="cutekitten.swf" width="300" height="200" >
    <param name="movie" value=" cutekitten.swf" />
    <param name="quality" value="high" />

    <!-- "none of the above" fallback! -->
    Your browser does not support HTML 5 video.

  </object>

</video>
```

CSS

While you cannot style the video itself, you can style the container, typically a , using standard CSS.

Example:

```
#video {  border:1px red solid; }
```

Example:

```
.videocssdemo {
    border:5px red solid; opacity: 0.4; position:absolute; top:20px; left:30px; transition: 2s
}
.videocssdemo:hover { transform:rotate(45deg); transition: 2s }
```

JavaScript

The <video> element has an API to control videos from JavaScript.

Examples:

```
var thevideo = document.getElementById('video')
thevideo.play();
thevideo.pause();
thevideo.volume = .5;   (0 – 1)
thevideo.muted = true;
```

Audio

Working with audio is very similar to working with video. Although you can even use the <video> tag with audio files, HTML 5 also includes an <audio> tag.

Audio Formats

- MP3 – All modern browsers
- AAC – a proposed successor to MP3 – All modern browsers, but with limited support in Firefox.
- WAV – Most browsers other than Internet Explorer

For more details, issues and browser support see:

- caniuse.com - http://caniuse.com/#search=audio
- W3Schools - https://www.w3schools.com/html/html5_audio.asp
- Wikipedia - https://en.wikipedia.org/wiki/HTML5_video
- Mozilla.org - https://developer.mozilla.org/en-US/docs/Web/HTML/Supported_media_formats
- http://html5test.com

HTML 5 Audio

```
<audio src="KittenPurr.mp3" controls></audio>
```

Notes:

- Unless the “controls” attribute is supplied, the <audio> tag is invisible.
- Most browsers will not auto start audio.
- Most of the <video> tags are supported except for height and width.
  autoplay, controls, loop, muted and src are supported.
- The <audio> tag also supports the <source> child tag to support multiple audio formats.

```
<audio controls>
    <source src="KittenPurr.wav" />
    <source src="KittenPurr.mp3" />
    Your browser does not support HTML5 audio.
</audio>
```

Hosting Videos in the Cloud

To reduce storage space and bandwidth needs for your local servers you can host your videos in cloud services such as Amazon Web Services, Microsoft Azure and YouTube.

Depending on where you host your content, and how you would like to deliver the content you might use one of these options:

- Using HTML 5 video tags and stream the video from:
  - Your own servers
  - Amazon Web Services
  - Microsoft Azure
- YouTube and other sources by using their <IFRAME> embedded player.

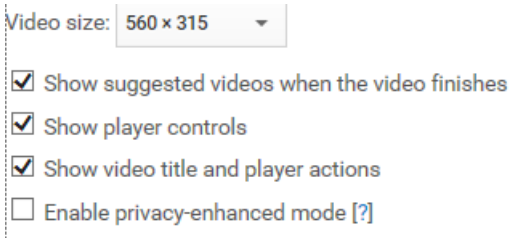
Embedding YouTube Videos

As an easy example of a hosted video we will take a look at embedding a YouTube video.

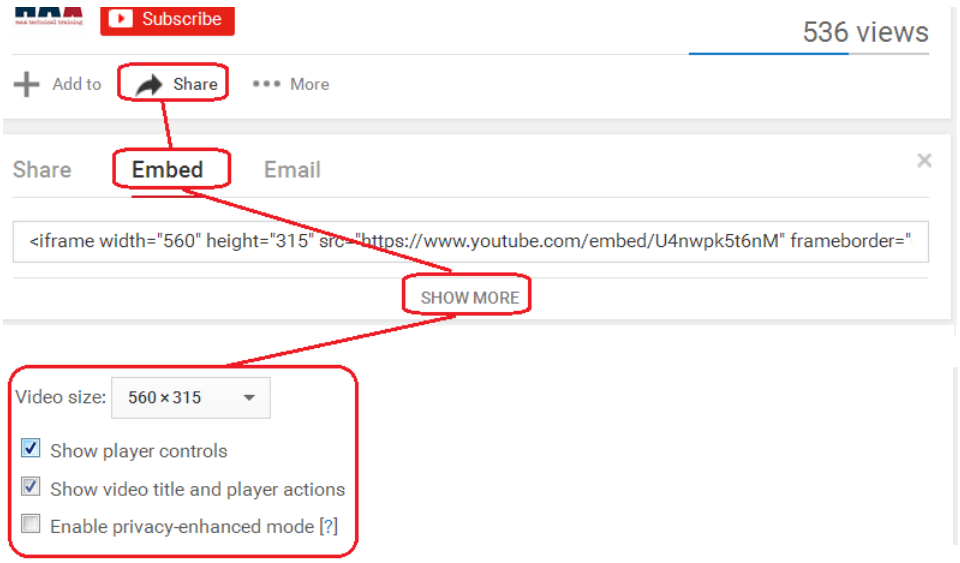
Steps:

1. Visit the video at the YouTube site.
2. Click the Share button just below the video.
3. Click the Embed link.

4. Immediately below the sample HTML click SHOW MORE and choose the video size and other options.



5. And of course read the “YouTube API terms of service”.
6. Copy the HTML for the <IFRAME> and add it to your web page.

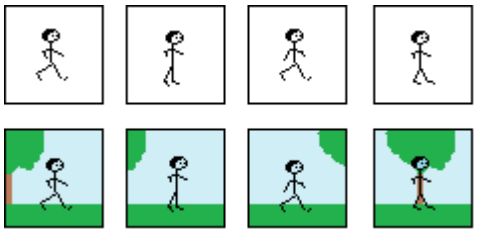


To see how to use additional features such as auto-play or to start the video at a certain point, see this article: https://support.google.com/youtube/answer/171780

Working with Animated GIFs

One of the earliest forms of animation on the web was the Animated GIF file. These files are basically a collection of GIF images that are displayed in sequence. All you need is GIF editor that supports animations and some artistic skills. (The following example shows no artistic skills!)

Start with some great art… a series of images. Then load these into a GIF animation editor.



🗁 **Sample file2: AnimatedGIF/StickManAndTrees.gif** and **AnimatedGIF/StickMan.gif** (The individual frames are in the same folder.)

While there are number of GIF editors around, you can use just about any paint tool to create the images and then upload them to one of the online GIF tools. (Do a search for “online animated gif editor”. The samples supplied with this class were created using http://gifcreator.me/.)