HTML5, CSS3 & Javascript

Chapter 1: Introduction to HTML & CSS

Chapter Objectives

In this chapter, we will discuss:

- Standardizing presentation with HTML and CSS
  - HTML essentials
  - Styling with CSS
  - Creating CSS in external files

Chapter Concepts

HTML 2 to HTML5

The beginning

- HTML was first published as an Internet draft in 1993.
- During the '90s HTML grew up faster and faster and finally, in 1999, version 4.01 was released.
- In the course of its development, the *World Wide Web Consortium* (W3C) assumed control of the specification.

HTML vs XML

- After the rapid delivery of these four versions though, HTML was widely considered a dead-end; the focus of web standards shifted to XML and XHTML, and HTML was put on the back burner.
- In the meantime, HTML refused to die, and the majority of content on the web continued to be served as HTML.
- To enable new web applications and address HTML's shortcomings, new features and specifications were needed for HTML.

HTML's Rebirth

- Wanting to take the web platform to a new level, a small group of people started the *Web Hypertext Application Working Group* (WHATWG) in **2004**. They created the **HTML5 specification**
- New features specifically geared to web applications saw the light.
- The **term Web 2.0 was coined**
- The first working draft for HTML5 in 2008
- The XHTML 2 working group stopped in 2009
- HTML5 officially released in 2015

A new vision

Compatibility

- If HTML5 features are not supported, the behavior must degrade gracefully
- Backward compatibility - The old Internet must survive

Utility

- Priority of Constituencies
- *User is King*
- This means, when in doubt, the specification values users over authors, over implementers (browsers), over specifiers (W3C), and over theoretical purity.
- HTML5 must be practical

A new vision

Interoperability (*Simple is better*)

- A new, simplified DOCTYPE
- Native browser ability instead of complex JavaScript code
- A new, simplified character set declaration

Universal access

- *Web Accessibility Initiative* (WAI)
- *Accessible Rich Internet Applications* (ARIA)

- New features should be based on HTML, CSS, DOM, and JavaScript
- Reduce the need for external plugins (like *Flash*)
- Better error handling
- More markup to replace scripting
- HTML5 should be **device independent**
- The development process should be visible to the public

The doctype

XHTML: the old one

```
<!DOCTYPE html PUBLIC
"-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1- transitional.dtd">
```

HTML5: simplicity

```
<!DOCTYPE html>
```

Charset

Short form - Supported by HTML5 and simplest

```
<meta charset=’utf-8’>
```

Long form - also supported

```
<meta http-equiv=’Content-Type’ content=’text/html; charset=utf-8’>
```

I never use funny characters. Do I still need to declare my character encoding?

- Yes! You should always specify a character encoding *a* on every HTML page you serve. Not specifying an encoding can lead to security vulnerabilities.

Namespace & Language

XHTML root element

- A valid XHTML has to specify the namespace with **xmlns** (that stands for "XML namespace")

```
<html xmlns="http://www.w3.org/1999/xhtml"
     lang="en" xml:lang="en">
```

Long form - also supported

- Now you can easily declare the HTML root element like below:

```
<html lang="en">
```

- The namespace is setted for you placing the HTML elements in the XHTML namespace.

HTML5 is not XML

- It is not case sensitive

```
<div>, <DIV>, <dIV> <!-- all valid -->
```

- You have not to close every tag

```
<BR>, <br>, <br /> <!-- all valid -->

<img>, <ImG>, <img /> <!-- all valid -->
```

- However it is better to use the XHTML code style to maintain a cleaner code
- Now it is possible to nest elements inside the **a** tag

A Basic HTML Document

- A basic HTML document has a document type declaration and an `html` element that contains one `head` and one `body` element

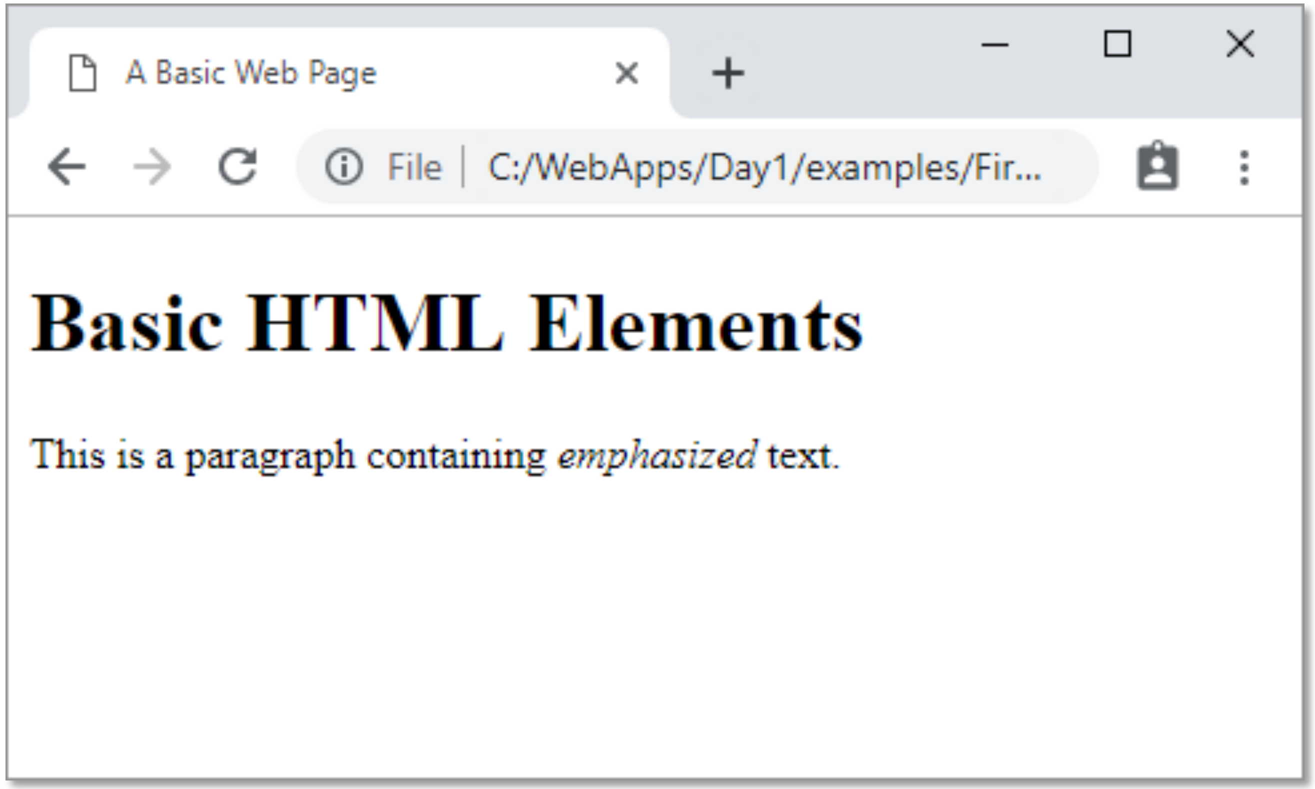
```html
<!doctype html>
<html>

<head>
    <title>A Basic Web Page</title>
</head>

<body>
    <h1>Basic HTML Elements</h1> <!-- comment won’t show up in browser view -->
    <p>This is a paragraph containing <em>emphasized</em> text.</p>
</body>

</html>
```

Basic Syntax

- These are the basic elements seen on the previous slide:
  - `<!doctype html>`: specifies HTML5 and higher (more on this later)
  - `<html>`...`</html>`: the root element of the document
  - `<head>`...`</head>`: contains information about the document, such as title
  - `<title>`...`</title>`: title of document, shown as window title in web browser
  - `<body>`...`</body>`: contains all the (visible) content of the web page
  - `<h1>`...`</h1>`: is a level 1 heading (important)
    - Also `<h2>` … `<h6>`
  - `<p>`...`</p>`: is a paragraph
  - `<em>`...`</em>`: emphasis
- Most elements are indicated by an opening and closing tag
  - E.g., opening tag `<head>`, closing tag `</head>`
- HTML comments are enclosed in `<!--` … `-->`
  - May not be nested

Appearance

- The browser (technically known as the User Agent) gives each element a default appearance
  - Later we will see how to customize this



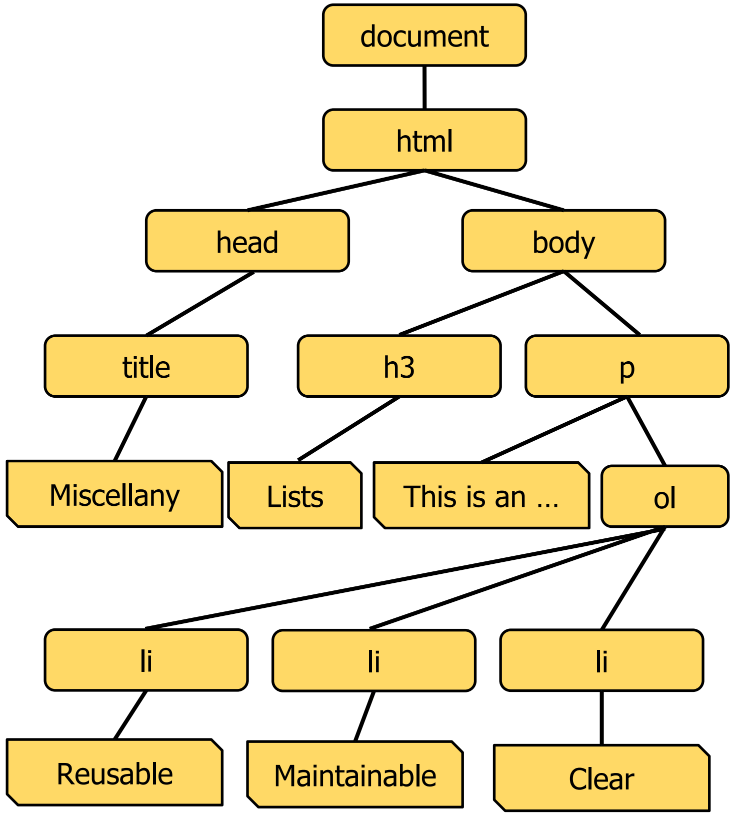
Basic HTML Elements

This is a paragraph containing *emphasized* text.

The DOM

- The browser parses a HTML document and creates a DOM

```
<html>
<head>
   <title>Miscellany</title>
</head>
<body>
  <h3>Lists</h3>
  <p>
     This is an ordered list:
     <ol>
       <li>Reusable</li>
       <li>Maintainable</li>
       <li>Clear</li>
     </ol>
  </p>
</body>
</html>
```



- Can use the DOM API to directly affect the DOM
  - Normally done from an event handler method

DOs and DON'Ts

<html>

<head>

Use only for styles, loading JS code, title , meta elements

Never put page related HTML elements in here!

</head>

<body>

ALL HTML related tags belong in here including semantic tags!

Some JS code is OK, some JS code can be loaded here as well

Don't put style tags in body

</body>

</html>

There is no neck!

No code or tags EVER go between </head> and <body>!

- Browsers forgive a lot but bad or faulty coding styles can lead to unpredictable behavior!

When Is a Browser Not a Browser?

- Most web content is consumed by browsers where a human reads the content
- Not all browsers are created equal, especially mobile browsers
  - Some provide an “accelerated” mode where pages are pre-processed on a server
    - Reduces bandwidth usage and optimizes pages for smaller screen size
- Search companies scan web pages using programs called web crawlers
- Many people, especially those with sight impairment, use screen readers
  - A program that interprets the page and reads it out
- Personal assistants (e.g., Siri, Alexa) are becoming more popular
  - When you ask a question, the assistant often scans web pages for the answer
- All these features rely on being able to understand the structure of the web page
  - This course uses the term “screen reader” to indicate relevant features

Images

- Images cannot be embedded in or attached to the HTML
  - HTML is a purely text format
  - Unicode text, to account for international languages, but just text
- Images are "sourced" from another document using a URL
  - Either a path relative to the current page or an absolute path
  - Browser downloads and places images in the document

Attribute of the `img` tag

Tag does not require closing, or may be marked as self-closing (/>)

```
<img src="ftse.svg" alt="Chart of FTSE All Share Index, 1 year, 16 May 2019">
<img src="photo.jpg" alt="A photograph from Copenhagen, Denmark, showing air
conditioning units. The photo has a highly graphical, modernist quality.">
```

- The `src` attribute is mandatory, all other attributes are optional
  - The `alt` information is displayed if the end-user has disabled images
    - And by screen readers

Image Formats

- It is recommended that you use one of four image formats:
  - JPEG (jpg) for photographs and smoothly varying images
    - Lossy compression based on how the eye sees the world
    - Good for smooth variations and complex scenes where edge detail is not important
  - PNG or SVG for images that contain mostly lines or text (e.g., diagrams, charts)
    - PNG is a “bitmap” format that supports lossless compression and alpha transparency
    - SVG is a “vector” format that can be scaled perfectly to any size
  - GIF when simple animation is required
    - Supports a limited set of colors and limited transparency
    - Has support for animation
- If an image format supports interlacing, use that
  - Users initially see a low-resolution image, while the rest of the image loads
- Reduce the image size to the desired width and height
  - Downsampling an image to a smaller size will help pages load faster

Scalable Vector Graphics

- XML-based vector format for 2D images
- Defines the image as a series of vector primitives:
  - E.g., circle, line, text
  - Since the primitives are vectors, they can be scaled to any size
  - Usually much smaller than a “bitmap” format for appropriate content
- Requires special software to create manually
  - Programmatically simple to create, good for graphs
- Contents of an SVG image become part of the page that displays them
  - Can be styled (e.g., colored) in the same way as any other HTML element
  - Can interact programmatically, e.g., react as the mouse moves

Try It Now: Review Image Formats

Do Now

10 **min**

- In the folder `Ch01\ImageDemo`, open `image-demo.html`
  - Launch it in the default browser
  - When you move the mouse pointer over any of the images, they will all zoom 8x
- What can you see? Perhaps something like this:

	Graph	Photo
JPEG	Some blurring around text is clear when magnified, leads to softness at normal size.	Photo has some edge artefacts, not noticeable at normal size. Blocks of color are smooth.
GIF	Some blurring of text, but better than JPEG.	Worse edge artefacts. Speckling in blocks of color that is visible even at normal size.
PNG	Similar textual quality to GIF. Similar file size.	Almost as good as JPEG, but much large file size.
SVG	Pin sharp at all magnifications. Much smaller file size.	Cannot handle photos at all.

Links

- The “hyper” in HTML refers to the capability to click links
  - A link takes the user to a different web resource
  - A link is “anchored” (`<a>`) by some part of an HTML page, often text or an image
  - Specify the URL of the destination in the href attribute of the `<a>` tag
    - Could be a relative path, as here, or an absolute path

```html
<p>
    Here is an example of a JPG image, which is a particularly good format for
photographs.
    <a href="links.html">
        <img src="images/photo.jpg" alt="A photograph from Copenhagen, Denmark"></a>
    This photo is also a link, if you click on it, you will be taken to the page
about links.
</p>
```

Link to Sections of a Document

- Can link to a particular section of a document
  - Give the section to be linked an id
    - Any element can have an id and it must be unique within that document

```html
<h2 id="pre">Pre-formatted Text</h2>
```

- Link to this from the same document as:

```html
<p>
    You can jump ahead to <a href="#br">Line Breaks</a>, <a href="#pre">Pre-formatted Text</a>,
    <a href="#hr">Horizontal Rule</a> or <a href="#nbsp">Non-breaking space</a>
</p>
```

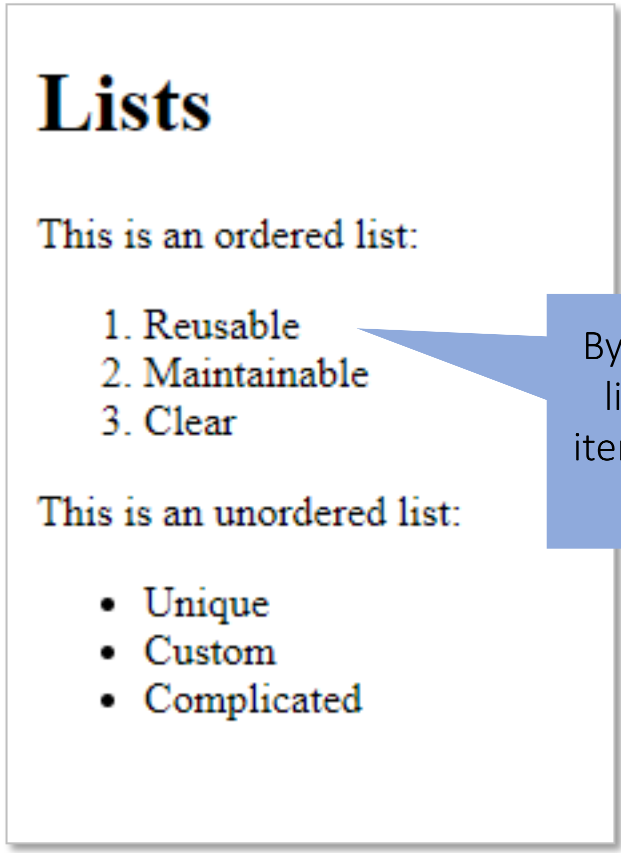
- Link to a section from a different document by adding section ID to path
  - Can add section ID to both absolute and relative paths

```html
<a href="https://en.wikipedia.org/wiki/Chichen_Itza#Location">Where is it?</a>
```

Lists

- Can create ordered and unordered lists of items
  - Use `<li>` for items and `<ol>` for ordered lists; `<ul>` is unordered

```
<h1>Lists</h1>
<p>
    This is an ordered list:
    <ol>
        <li>Reusable</li>
        <li>Maintainable</li>
        <li>Clear</li>
    </ol>
</p>
<p>
    This is an unordered list:
    <ul>
        <li>Unique</li>
        <li>Custom</li>
        <li>Complicated</li>
    </ul>
</p>
```



Lists

This is an ordered list:

1. Reusable
2. Maintainable
3. Clear

This is an unordered list:

- Unique
- Custom
- Complicated

By default, list items in an ordered list are given numbers, whereas items in an unordered list are given bullets

Line Breaks and Horizontal Lines

- Can force a line break by using `<br>`
  - Use this when the line break is part of the content, as in poetry
  - Use `<p>` or `<hr>` to indicate a thematic break
- Can have the browser draw a horizontal line by using `<hr>`
  - This should not be used just for appearance, it should be thematic

```html
<hr>
<h2>Line Breaks</h2>
<p>They could be used to lay out poetry:</p>
<h3>A Poem Never Says Anything</h3>
<p>A poem never says anything.<br>
It just opens a door, quietly.<br>
<br>
Sleepless and bent<br>
just like my aged father<br>
waiting for me in a lonely winter night.<br>
<br></p>
<p><em>Uttaran Chaudhuri</em></p>
<hr>
```

Preformatted Input

- If you have plain text that should not be wrapped, can use `<pre>`
    - Preserves line breaks and whitespace
    - Often used for code listings, but can also be used for poetry

```
<pre>
I will arise and go now, and go to Innisfree,
And a small cabin build there, of clay and wattles made;
Nine bean-rows will I have there, a hive for the honey-bee,
And live alone in the bee-loud glade.

And I shall have some peace there, for peace comes dropping slow,
Dropping from the veils of the morning to where the cricket sings;
There midnight’s all a glimmer, and noon a purple glow,
And evening full of the linnet’s wings.

I will arise and go now, for always night and day
I hear lake water lapping with low sounds by the shore;
While I stand on the roadway, or on the pavements grey,
I hear it in the deep heart’s core.
</pre>
```

Non-Breaking Space

- Normally, the web browser wraps text by looking for spaces
  - If you have a space separating words that should not be broken, use `&nbsp;`

```
<p>Try re-sizing the browser and see whether the company name is ever
split over two lines.</p>
<p>
    Fidelity&nbsp;Investments&nbsp;Inc Fidelity&nbsp;Investments&nbsp;Inc
    Fidelity&nbsp;Investments&nbsp;Inc Fidelity&nbsp;Investments&nbsp;Inc
    Fidelity&nbsp;Investments&nbsp;Inc Fidelity&nbsp;Investments&nbsp;Inc
    Fidelity&nbsp;Investments&nbsp;Inc Fidelity&nbsp;Investments&nbsp;Inc
    Fidelity&nbsp;Investments&nbsp;Inc Fidelity&nbsp;Investments&nbsp;Inc
    Fidelity&nbsp;Investments&nbsp;Inc Fidelity&nbsp;Investments&nbsp;Inc
</p>
```

- This is an example of an "entity"
  - Entities start with the ampersand and end with the semicolon
  - Entities are used to display special characters

Some Entities in HTML

- Some entities are used to escape special characters in HTML
  - E.g., less than (<) and greater than (>) that otherwise are interpreted as tags
- Entities can also be used to represent Unicode characters
  - Useful for characters that aren’t on your keyboard
  - See http://www.w3.org/TR/REC-html40/sgml/entities.html for full list

Double Quote	"	&quot;
Apostrophe	'	&apos;
Less Than	<	&lt;
Greater Than	>	&gt;
Ampersand	&	&amp;

Copyright	©	&copy;
Registered	®	&reg;
Trademark	™	&#8482;
Indian Rupees	₹	&#x20b9;
U umlaut	ü	&uuml;

decimal or hex

Try It Now: Basic HTML

Do Now

20 **min**

- In Visual Studio Code, open the folder `Ch01\SimpleExamples`
  - Open the file `index.html`
- Also open the file in a web browser
  - Click the links in the table of contents
  - Which files are being displayed?
  - View the source of those files
  - Make changes and verify that you see what you expect in the browser
    - Remember to save the files with `CTRL-S`
  - Optimize formatting by pressing `Shift-Alt-F`
    - Let Visual Studio Code improve your code format for you!

Chapter Concepts

From Attributes to Style

- In older versions of HTML, appearance was specified through attributes

```
<hr align="center" width="50%">
```

- These are deprecated in modern HTML, in favor of styles:

```
<hr style="width: 50%; margin-inline-start: auto;">
```

  - This type of style is known as an in-line style

There is no direct equivalent of align, and setting the margin to auto (in other words, centering) is the default, but shown here for comparison

- In-line styles need to be set on every element indi
  - Hard to maintain consistency
  - When creating a website, we usually have a “house style”

Centralizing Presentation

- We can use style rules to apply a consistent style to a range of elements
  - We will explore style rules in more detail shortly
  - This rule applies the same style declarations to all `hr` elements

```
hr {
    width: 50%;
    margin-inline-start: auto;
}
```

- One way to use these is to put a `style` element in the `head`

```
<head>
…
    <title>Breaks</title>
    <style>
        hr {
            width: 50%;
            margin-inline-start: auto;
        }
    </style>
</head>
```

Try It Now: Style

Do Now

15 **min**

- View the file `Ch01\StyleExamples\index.html`
  - Load it into the browser
  - Click the “Basic” link
- Also open the file `basic.html`
  - Delete everything to the right of `font-family`
  - Hit `CTRL-SPACE` and look at the options
    - Choose something from the list, save the file, and then click the link again
    - Try a few options—don’t neglect those further down the list (such as `cursive`)
  - Do the same with `font-weight`
    - Make sure to try both numeric and text values

Setting Style on Multiple Pages

- A common requirement is to have a consistent presentation style across many pages
  - Ideally, we would apply exactly the same style definitions
- We can do this by writing a separate stylesheet

style.css

```
hr {
    width: 50%;
    margin-inline-start: auto;
}
```

- And linking to it from each HTML page's `head`

```
<head>
…
    <title>Breaks</title>
    <link rel="stylesheet" href="style.css">
</head>
```

Simple CSS Selectors

- The part of the CSS rule that indicates which elements it should apply to:
  - `E`: match all elements of type `E`
  - `*`: match all elements
    - Avoid this, see later
  - `#id`: match element with that id
  - `selector, selector`: match either
    - Can be multiple
  - `.class`: match all elements of class
    - See next slide

```css
hr {
    width: 50%;
    margin-inline-start: auto;
}

* {
    font-family: Verdana, Arial, Helvetica, sans-serif
}

#bl {
    background-color: blue;
}

#rp {
    background-color: rebeccapurple;
}

#bl, #rp {
    color: white;
}

.intro {
    color: #333;
}
```

HTML Class

- Useful to be able to apply styles to all elements of a type and to a single element
  - But often want to apply to a collection of elements of the same or different types
  - Apply a class to the HTML elements

```html
<h1 class="intro">Breaks</h1>
<p class="intro">We have already seen the p element, used to create a paragraph, but
there are other types of breaks available in HTML.</p>
```

    - (Note that elements of different types can share the same class)
  - Then apply style rules to that class

```css
.intro {
    color: #333;
}
```

    - Note that the selector starts with a dot (period or full stop)
  - To select just the `h1`, could use `h1.intro`
- Classes are also useful for programmatic interaction with the DOM

Common Declarations

- Each name-value pair is called a declaration
  - Declarations end with a semicolon

Property	Description
`height`	Sets the height of an element. Not every element can have a defined height, see discussion of layout later. `height: 100px;`
`width`	Sets the width of an element, as with height. `width: 50%;`
`max-height`, `min-height`, `max-width`, `min-width`	Prevent size of element from varying outside certain limits, but otherwise let it vary with the display size.
`background-color`	Set the fill color of an element. `background-color: blue;`

Units for Dimensions

- There are many ways to set dimensions, these are some of the most used
  - Generally, favor relative dimensions as these will scale better

Unit	Description
em	Font size (historically the size of capital M) verttical
rem	Font size of root element
ex	Size of lower case x, 1ex ≈ 0.5em in many fonts horizontal
%	Not technically a unit, usually sets relative to parent

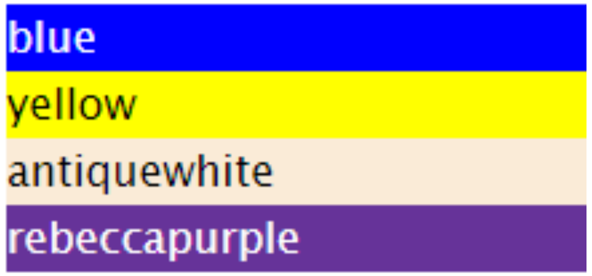
  - There are also absolute dimensions

Unit	Description
px	pixel (screen dot[1])
in	inch, defined as 96px on screens
cm	centimetre, defined as 96px / 2.54
pt	point = 1/72nd inch (96px / 72)

[1] In fact, the standard defines a *recommended* "reference pixel", being the whole number of actual pixels that best approximate the visual angle of a device with pixel density 96ppi at arm's length (say, 28 inches) – approx. 0.0213 degrees

Colors

- By (case-insensitive) name, e.g.:
  - `blue`
  - `yellow`
  - `antiquewhite`
  - `rebeccapurple`[1]



- By 3-, 4-, 6-, or 8-digit hex code
  - Represent RGB (00 – ff, 0 – 255) and transparency (ff = 100% opaque)
  - All these are `rebeccapurple`
  - `#639`
  - `#663399`
  - `#639f` or `#639F`
  - `#663399ff` or `#663399FF`
- By `rgb` or `rgba` function (equivalent since CSS4), all these are `rebeccapurple`:
  - `rgb(102, 51, 153)`
  - `rgb(40%, 20%, 60%)`
  - `rgb(102, 51, 153, 100%)`
  - `rgb(102, 51, 153, 1.0)`
  - `rgb(102 51 153 / 1)`
- Using the `transparent` keyword
- By `hsl` or `hsla` function
  - Hue, saturation, and lightness model
  - Easy to create family of colors
  - Beyond the scope of this course, but favored by web designers

[1] https://en.wikipedia.org/wiki/Eric_A._Meyer#Personal_life

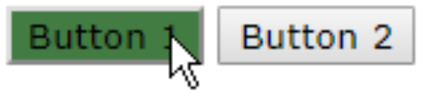
Text Styles

Property	Description
`color`	Sets the font color.
`font-family`	Sets the font of an element. If a particular font is not available, can specify fall-backs. `font-family: Arial, Helvetica, sans-serif;`
`font-size`	Sets the size of a font to a pre-defined keyword (such as `medium`), a percentage of the parent, or a fixed size. `font-size: 14px;`
`font-style`	`normal`, `italic`, or `oblique` (in most cases there is no difference between `italic` and `oblique`).
`font-weight`	E.g., `normal`, `bold`, or a numeric value (`100`, `200`, `300`, `400`, `500`, `600`, `700=bold`, `800`, `900`).
`line-height`	Vertical spacing of text, distinct from `font-size`. `line-height: 1.5;`
`text-align`	`left`, `right`, `center`, `justify`
`text-decoration`	`underline`, `overline`, `line-through`

Pseudo-Class Selectors

- Pseudo-classes may be used to match an element based on state or relationship:
  - E.g., `:hover` matches an element when the mouse is over it

```
button:hover {
    background: #437c43;
}
```

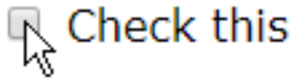


for links
-active
-visited
-hover
-link (unvisited)

- Examples for pseudo-classes include:
  - `:checked`

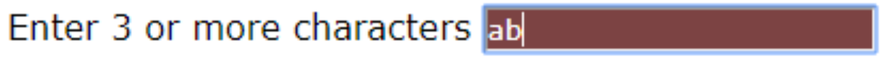
```
input:checked + label {
    color: #437c43;
    text-decoration: underline;
}
```

Label adjacent to input (we will cover this later)



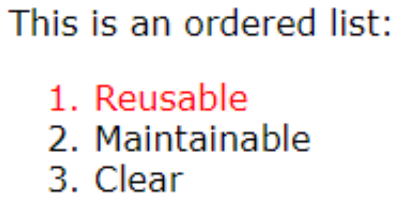
  - `:valid`, `:invalid`

```
input:valid { background-color: #437c43; }
input:invalid { background-color: #7c4343; }
```



  - `:first-child`

```
ol li:first-child {
    color: red;
}
```



This is an ordered list:

1. Reusable
2. Maintainable
3. Clear

Styling Links

- It is better to leave link colors and cues (underlining, etc.) alone
  - But if the default blue, underlined links will be hard to read on your background, you can change the color scheme using pseudo-classes:
    - `:link` (all links, used for unvisited)
    - `:visited`
    - `:hover`
    - `:active` (being selected)
  - Always specify in order LVHA
  - Can apply limited styling to visited links to avoid information leak

```
a:link {
font-family: cursive;
}

a:visited {
color: #437c43;
}

a:hover {
color: #7c4343;
}

a:active {
text-decoration: underline overline;
}
```

There is no programmatic way to detect the difference between a visited and unvisited link

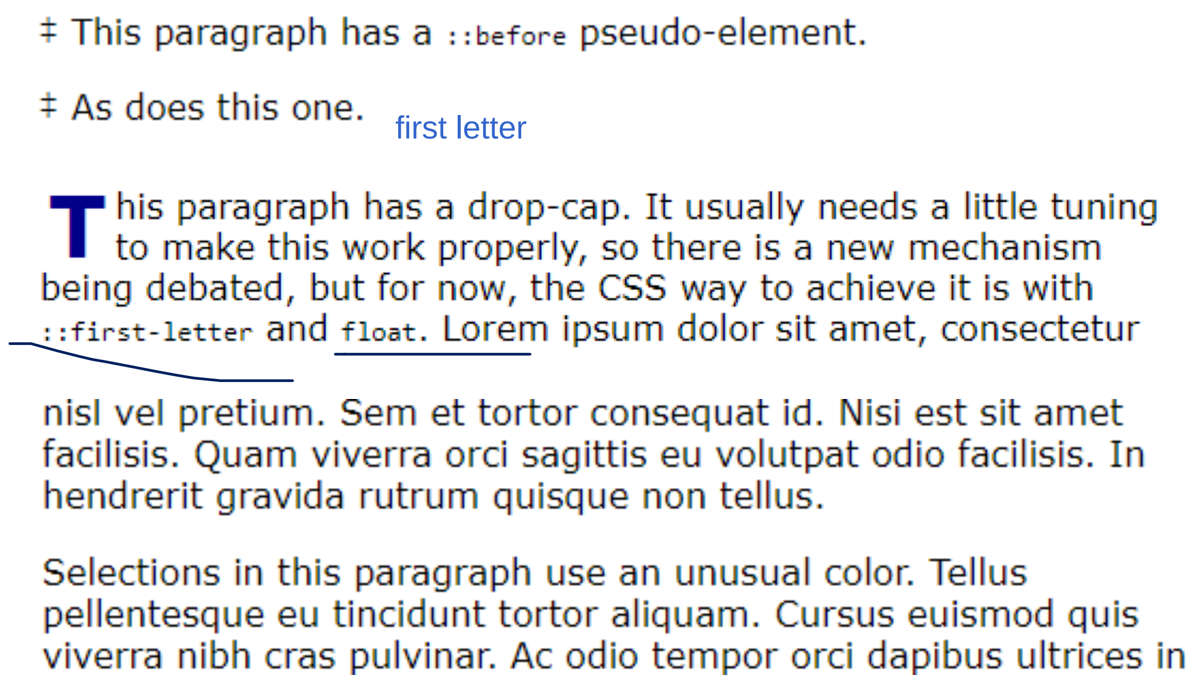
Pseudo-Element Selectors

- Select a part of an element
  - In contrast to pseudo-classes, which select based on state or relationship
  - Historically, both were indicated with a single colon, but they are now distinguished
- Examples of pseudo-elements:
  - `::after`, `::before`

```
.symbol::before {
    content: '\2021';
}
```

  - `::first-letter`, `::first-line`
  - `::selection`

```
.selection::selection {
    color: yellow;
    background-color: rebeccapurple;
}
```


‡ This paragraph has a `::before` pseudo-element.

‡ As does this one.

This paragraph has a drop-cap. It usually needs a little tuning to make this work properly, so there is a new mechanism being debated, but for now, the CSS way to achieve it is with `::first-letter` and `float`. Lorem ipsum dolor sit amet, consectetur

nisl vel pretium. Sem et tortor consequat id. Nisi est sit amet facilisis. Quam viverra orci sagittis eu volutpat odio facilisis. In hendrerit gravida rutrum quisque non tellus.

Selections in this paragraph use an unusual color. Tellus pellentesque eu tincidunt tortor aliquam. Cursus euismod quis viverra nibh cras pulvinar. Ac odio tempor orci dapibus ultrices in

Instructor Demo: Browser Developer Tools

DEMO

15 min

- Your instructor will show how to debug CSS problems including:
  1. Verify that the CSS has loaded for a page with Chrome developer tools
  2. Determine what CSS rules are being applied to elements
  3. Change CSS properties on the fly

Exercise 1.1: Applying CSS Styling

HANDS-ON EXERCISE

20 **min**

- Open the files Ch01\ApplyCssStyling\index.html and Ch01\ApplyCssStyling\style.css.
- Complete the TODO steps in both files.
- To see your changes, you will need to reload the web page.

Chapter Concepts

Introduction to HTML

Styling with CSS

➤ **Chapter Summary**

Chapter Summary

In this chapter, we have discussed:

- Standardizing presentation with HTML and CSS
  - HTML essentials
  - Styling with CSS
  - Creating CSS in external files