Cascading Style Sheets (CSS3)

PIC 40A, UCLA

CSS

CSS (Cascading Style Sheets) are a means for specifying format/display instructions to the web browser when it parses HTML. A browser can have its own defaults, but the CSS gives the control back to the web developer.

The current version of CSS is CSS3. Some of the changes this new standard brings about compared to CSS2 come in giving the web developer more control over selecting elements, being able to tailor display to a user’s device (tablet vs PC screen size). Plus more design options.

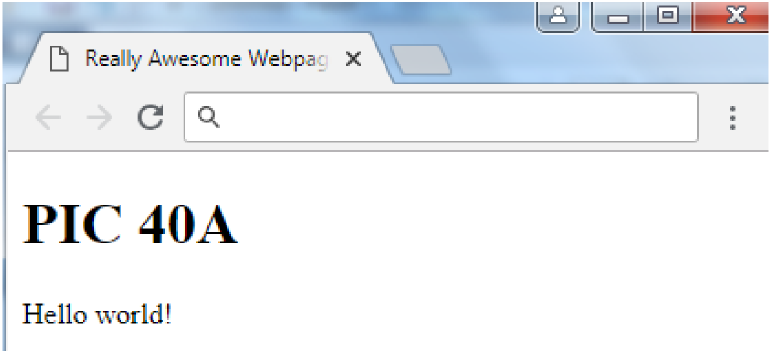
A Bit of Style

Let's add some style to that very simple HTML5 webpage:

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

<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" type="text/css" href="page1.css" />
  <title>Really Awesome Webpage</title>
</head>

<body>
  <h1>PIC 40A </h1>
  <p>
    Hello world!
  </p>
</body>

</html>
```

A Bit of Style

Here's how it used to display without any style.

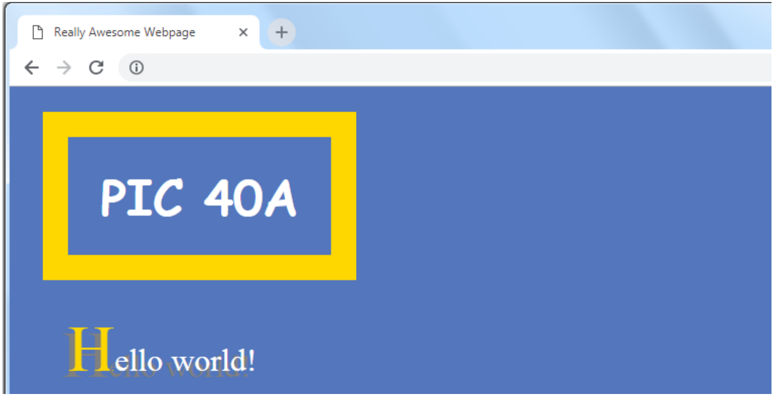


PIC 40A

Hello world!

A Bit of Style

Without modifying the HTML in any way, through the help of that **page1.css** file, here is how it looks:



A Bit of Style I

Here is **page1.css**:

```css
body{
  background-color: #5376BC;
}

h1{
  color: white;
  text-align: center;
  font-family: Comic Sans MS, cursive;
  font-size: 3em;
  width: 25%;
  border: 25px solid #FFD700;
  padding: 25px;
  margin: 25px;
}

p{
  text-indent: 50px;
  display: block;
  color: white;
  font-size: 2em;
  text-shadow: -6px 6px gray;
}

p::first-letter {
  font-size: 200%;
  color: #FFD700;
}
```

CSS Gist

CSS operates by selecting a set of one or more elements.

Once these elements have been selected, it changes their properties.

The general syntax it follows is:

```
selected elements {
   property: value;
   another_property: another_value;
}
```

The styles can all be done on one line, but that is harder to read so we will focus on the multiline format above.

CSS Properties

A CSS comment starts with /* and ends with */.

To get started with CSS, we comment what some of the preceding code does before going into more depth:

```
p{ /* select all p elements */
  text-indent: 50px; /* indent by 50 pixels */
  color: white; /* make the text white */
  font-size: 2em; /* make the font twice its default size */
  text-shadow: -6px 6px gray; /* add a gray shadow left&down of 6 pixels */
}
```

CSS Properties

There are many properties that can be changed with CSS. We will only study some of them in these notes including:

- Background
- Font
- Text
- Dimensions
- Padding
- Border
- Margin
- Display
- Classification and Positioning

CSS Properties

We will study the properties before studying the selection process. For the examples that follow, we need only concern ourselves with the following:

```
section { /* stuff that applys to all sections */ }

p { /* stuff that applies to all ps */ }

#foo { /* stuff that applies to all elements with id foo */ }

.bar {
   /*stuff that applies to all elements with attribute of class with value bar*/
}
```

CSS Background

The background of an element can be set. The properties we consider are:

- **background-color**: can specify the fill color of the element
- **background-image**: can specify a url to a desired image

For example,

```
background-color: red;
```

makes the background red. And

```
background-image: url("cat.png");
```

sets the background image to that of the **cat.png** image.

CSS Background

Colors can be specified in different ways in CSS:

- in words, certain colors are defined like "red", "blue", etc.
- in their red-green-blue color values from 0-255 as in **rgb(255,0,0)** being fully red with no blue or green
- in their red-green-blue color values written in hexademical juxtaposed next to each other as in **#cc0024** meaning
  a red component of **CC** $= 12 \times 16 + 12 = 204$,
  a green component of **00 =** $0 \times 16 + 0 = 0$, and
  a blue component of **24** $= 2 \times 16 + 4 = 36$.

Some other shorthands exist but we won't discuss them.

CSS Background

With HTML

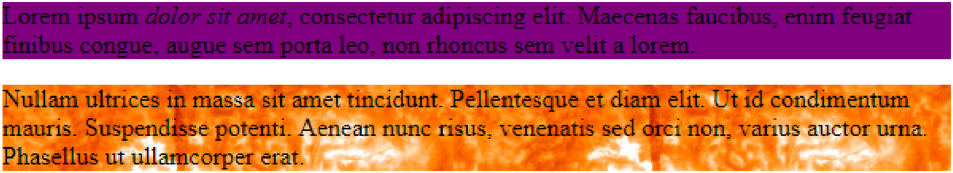
```html
<p id="first">
  Lorem ipsum <i>dolor sit amet</i>, consectetur adipiscing elit. Maecenas faucibus, enim
        feugiat finibus congue, augue sem porta leo, non rhoncus sem velit a lorem.
</p>
<p id="second">
  Nullam ultrices in massa sit amet tincidunt. Pellentesque et diam elit. Ut id condimentum
        mauris. Suspendisse potenti. Aenean nunc risus, venenatis sed orci non, varius
        auctor urna. Phasellus ut ullamcorper erat.
</p>
```

and CSS

```css
#first{
  background-color: purple;
}

#second{
  background-image: url("Sun.png")
}
```

CSS Background

we obtain the output



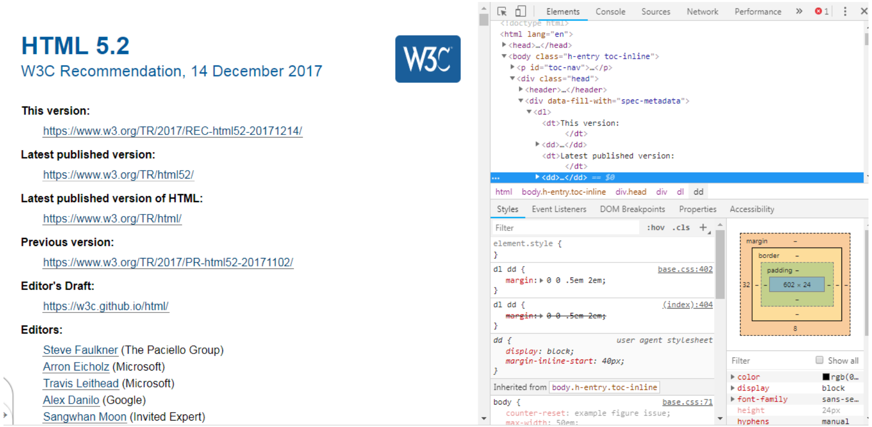
Lorem ipsum *dolor sit amet*, consectetur adipiscing elit. Maecenas faucibus, enim feugiat finibus congue, augue sem porta leo, non rhoncus sem velit a lorem.

Nullam ultrices in massa sit amet tincidunt. Pellentesque et diam elit. Ut id condimentum mauris. Suspendisse potenti. Aenean nunc risus, venenatis sed orci non, varius auctor urna. Phasellus ut ullamcorper erat.

(assuming we have the file **Sun.png**).

CSS and HTML Investigation

Virtually all web browsers allow their users to inspect a webpage to view the HTML, CSS, and JavaScript processes. In Chrome as one example, one can **right click**, **select Inspect**, and then access a lot of information.



HTML 5.2

W3C Recommendation, 14 December 2017

This version:

https://www.w3.org/TR/2017/REC-html52-20171214/

Latest published version:

https://www.w3.org/TR/html52/

Latest published version of HTML:

https://www.w3.org/TR/html/

Previous version:

https://www.w3.org/TR/2017/PR-html52-20171102/

Editor's Draft:

https://w3c.github.io/html/

Editors:

Steve Faulkner (The Paciello Group)
Arron Eicholz (Microsoft)
Travis Leithead (Microsoft)
Alex Danilo (Google)
Sangwhan Moon (Invited Expert)

```
<!doctype html>
<html lang="en">
 ▶<head>…</head>
 ▼<body class="h-entry toc-inline">
   ▶<p id="toc-nav">…</p>
   ▼<div class="head">
     ▶<header>…</header>
     ▼<div data-fill-with="spec-metadata">
       ▼<dl>
          <dt>This version:
               </dt>
         ▶<dd>…</dd>
          <dt>Latest published version:
               </dt>
...      ▶<dd>…</dd> == $0
```

```
element.style {
}
dl dd {                                   base.css:402
   margin: ▶ 0 0 .5em 2em;
}
dl dd {                                   (index):404
   ~~margin: ▶ 0 0 .5em 2em;~~
}
dd {                              user agent stylesheet
   display: block;
   margin-inline-start: 40px;
}
Inherited from body.h-entry.toc-inline
body {                                     base.css:71
   counter-reset: example figure issue;
   max-width: 50em;
```

```
▶ color          ■rgb(0…
▶ display        block
▶ font-family    sans-se…
  height         24px
  hyphens        manual
```

These tools are worth exploring!

CSS Font

We will consider the following subset of font properties:

- **font-family**: can specify a list of fonts to be used, in order of decreasing preference
- **font-size**: can specify how large the font is
- **font-weight:** can specify the weight of the font

CSS Font - font-family

There are five font families:

- **serif:** fonts that have decorative finishing strokes or slabbed endings called serifs, e.g., Times New Roman, New York - usually better for print media
- **sans-serif:** fonts that do not have these serifs, having little or no flare, e.g., Arial, Helvetica - usually better for the web
- **monospace:** fonts where every character has the same width, e.g., Courier, Prestige
- **cursive:** fonts that resemble handwriting, e.g., Caflisch Script, Comic Sans
- **fantasy:** fonts that are more for decoration, e.g., Cottonwood, Impact

CSS Font - font-family

To specify a font family, we give a comma separated list of families to try in order of preference, followed by the family name if the browser cannot support the previous items on the list.

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

CSS Font - font-family I

With HTML

```html
<p id="first">
  Times New Roman
</p>
<p id="second">
  Arial
</p>
<p id="third">
  Courier
</p>
<p id="fourth">
  Cedarville Cursive
</p>
<p id="fifth">
  Lobster
</p>
```

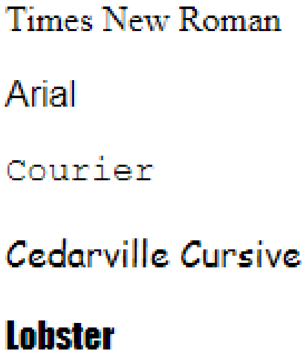
and CSS

CSS Font - font-family II

```css
#first{
  font-family: Times New Roman, serif;
}

#second{
  font-family: Arial, sans-serif;
}

#third{
  font-family: Courier, monospace;
}

#fourth{
  font-family: Cedarville Cursive, cursive;
}

#fifth{
  font-family: Lobster, fantasy;
}
```

CSS Font: font-family

we obtain the output



Times New Roman

Arial

Courier

Cedarville Cursive

Lobster

CSS Font - font-size

The **font-size** can be given by specifying the height in pixels **px**, or, more appropriately, its size in **em** or **%.**

1em equals the normal font size (based on how the browser wants to display the text) and the font-size can scale with the ems: 2em is twice as big, 0.8em is 80% as big, etc.

Remark: do not put a space between the number and the **em**!

The same idea holds with the percents except 100% equals the current font size.

Subtle remark: the **em**s can have a nested effect. Relative to its default size, if a span of text was given a size of 2em within a paragraph styled to 3em, the text would be 6 times as large as its default.

CSS Font - font-size

With HTML

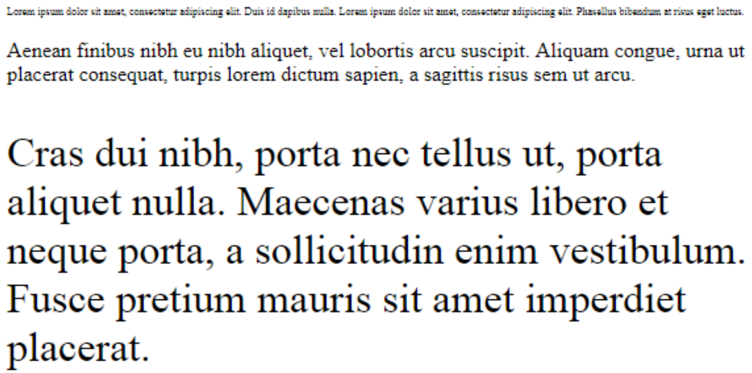
```html
<p id="small">
  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis id dapibus nulla. Lorem
        ipsum dolor sit amet, consectetur adipiscing elit. Phasellus bibendum at risus eget
        luctus.
</p>
<p id ="normal">
  Aenean finibus nibh eu nibh aliquet, vel lobortis arcu suscipit. Aliquam congue, urna ut
        placerat consequat, turpis lorem dictum sapien, a sagittis risus sem ut arcu.
</p>
<p id="big">
  Cras dui nibh, porta nec tellus ut, porta aliquet nulla. Maecenas varius libero et neque
        porta, a sollicitudin enim vestibulum. Fusce pretium mauris sit amet imperdiet
        placerat.
</p>
```

and CSS

```css
#small{
  font-size: 0.5em;
}

#normal{
  font-size: 1em;
}

#big{
  font-size: 2em;
}
```

CSS Font: font-size

we obtain the output



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis id dapibus nulla. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus bibendum at risus eget luctus.

Aenean finibus nibh eu nibh aliquet, vel lobortis arcu suscipit. Aliquam congue, urna ut placerat consequat, turpis lorem dictum sapien, a sagittis risus sem ut arcu.

Cras dui nibh, porta nec tellus ut, porta aliquet nulla. Maecenas varius libero et neque porta, a sollicitudin enim vestibulum. Fusce pretium mauris sit amet imperdiet placerat.

CSS Font - font-weight

The **font-weight** can have various values depending how how **bold** it should appear. It tends to run on a scale from 100-900 in increments of 100.

Standard text is 400. Normal bold text is 700.

CSS Font - font-weight I

With HTML

```html
<p id="w100">
  100
</p>
<p id="w300">
  300
</p>
<p id="w500">
  500
</p>
<p id="w700">
  700
</p>
<p id="w900">
  900
</p>
```

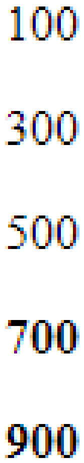
and CSS

CSS Font - font-weight II

```
#w100{
  font-weight: 100;
}

#w300{
  font-weight: 300;
}

#w500{
  font-weight: 500;
}

#w700{
  font-weight: 700;
}

#w900{
  font-weight: 900;
}
```

CSS Font: font-weight

we obtain the output



100

300

500

700

900

CSS Text

We will consider the following subset of text properties:

- **color**: can specify the font color, just like **background-color**.
- **text-align**: can specify the justification.
- **text-indent**: can specify the indent for the first line of an element.
- **text-decoration**: can specify the styling of the text.

CSS Text

Related to **color** is the **rgba** value: a 4-tuple of red/green/blue/alpha, where alpha is how opaque the text/element is.

There is also **opacity.** This can be used for any element to specify how opaque it is. It ranges from 0-1 with 1 being opaque and 0 being transparent. It applies to all child elements as well.

CSS Text - text-align

Some values of **text-align** include:

- **left** (aligns to the left)
- **right** (aligns to the right)
- **center** (centered)
- **justify** (all lines have same width)

CSS Text - text-indent

Some values of **text-indent** include:

- **%** (percent of element width to indent)
- **px** (number of pixels to indent)

CSS Text - text-decoration

Some values of **text-decoration** include:

- **underline** (underlines the text)
- **overline** (overlines the text)
- **line-through** (places a line through the text)

CSS Text I

With HTML

```html
<p id="first">
  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris in varius dolor, eu
        fermentum elit. In malesuada urna sit amet dui lacinia venenatis. Nunc gravida
        porttitor quam id blandit.
</p>
<p id="second">
  Fusce eget dictum nibh. Donec condimentum placerat tortor, in molestie tellus congue ac.
        Nullam consequat vitae quam eget sagittis. Quisque a imperdiet enim, vel faucibus
        velit.
</p>
<p id="third">
  Sed nec interdum tortor, ac egestas leo. In libero nisi, faucibus vel turpis sit amet,
        auctor fermentum tortor. Vestibulum convallis sagittis libero, at viverra enim
        gravida non.
</p>
<p id="fourth">
  Donec tincidunt tellus et aliquam placerat. Vivamus mattis molestie rutrum. Nullam
        tincidunt risus eu arcu consequat, vitae ullamcorper enim finibus. Pellentesque
        dapibus mi eu iaculis finibus.
</p>
```

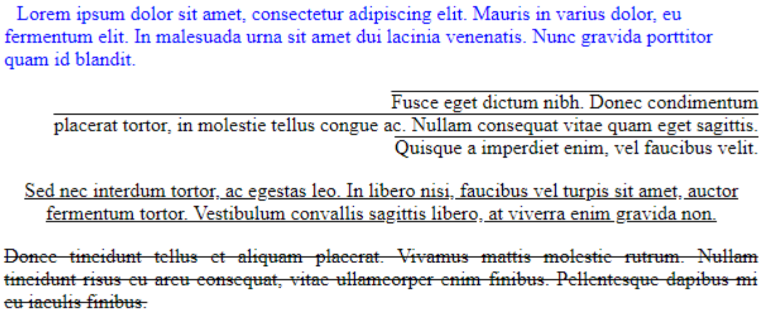
and CSS

CSS Text II

```css
#first{
  color: blue;
  text-align: left;
  text-indent: 10px;
}

#second{
  text-align: right;
  text-indent: 50%;
  text-decoration: overline;
}

#third{
  text-align: center;
  text-decoration: underline;
}

#fourth{
  text-align: justify;
  text-decoration: line-through;
}
```

CSS Text

we obtain the output



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris in varius dolor, eu fermentum elit. In malesuada urna sit amet dui lacinia venenatis. Nunc gravida porttitor quam id blandit.

Fusce eget dictum nibh. Donec condimentum placerat tortor, in molestie tellus congue ac. Nullam consequat vitae quam eget sagittis. Quisque a imperdiet enim, vel faucibus velit.

Sed nec interdum tortor, ac egestas leo. In libero nisi, faucibus vel turpis sit amet, auctor fermentum tortor. Vestibulum convallis sagittis libero, at viverra enim gravida non.

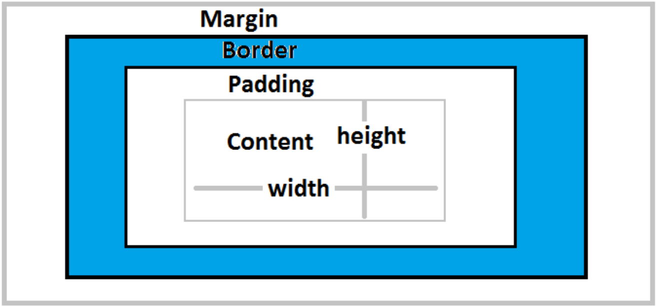
~~Donec tincidunt tellus et aliquam placerat. Vivamus mattis molestie rutrum. Nullam tincidunt risus eu arcu consequat, vitae ullamcorper enim finibus. Pellentesque dapibus mi eu iaculis finibus.~~

CSS Text

By using the **direction: rtl** instead of the default **ltr** (left-to-right), we could have the right-aligned paragraph indent from the right rather than the left.

CSS Box Model

The dimensions, padding, border, and margin properties relate to the **box model** of an element.



CSS Box Model

The content of an element has a height/width.

The content in turn is padded before reaching the element’s border.

The border can be specified to have a color, thickness, style, etc.

Then the outer border has a margin with respect to its containing element.

CSS Box Model

Some values for **height**, **width**, **padding**, **border-width**, **margin**:

- **px**: number of pixels
- **%**: percent of containing block's width

For example:

```
padding: 20px;
```

gives a uniform 20px of padding on the top, right, bottom, and left of the element.

It is also possible to specify the top/right/bottom/left individually with **padding-top**, **padding-right**, **padding-bottom**, and **padding-left** and similarly for **margin**. For **border** we use **border-top-width**, etc.

CSS Box Model

The **border**s can be given a **border-color** and **border-style**, too. The color values are the obvious ones.

Some values of **border-style** include:

- **dotted**
- **dashed**
- **solid**
- **double**

CSS Box Model

As one piece of shorthand we will consider (there are lots more), we can specify many border properties simultaneously with:

border: width style color;

where **width** is the width in % or pixels, **style** is the style value, and **color** is the color value.

CSS Box Model I

With HTML

```html
<section>
  <p id="first">
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris in varius dolor, eu
          fermentum elit. In malesuada urna sit amet dui lacinia venenatis. Nunc gravida
          porttitor quam id blandit.
  </p>
  <p id="second">
    Fusce eget dictum nibh. Donec condimentum placerat tortor, in molestie tellus congue ac.
           Nullam consequat vitae quam eget sagittis. Quisque a imperdiet enim, vel faucibus
           velit.
  </p>
  <p id="third">
    Sed nec interdum tortor, ac egestas leo. In libero nisi, faucibus vel turpis sit amet,
          auctor fermentum tortor. Vestibulum convallis sagittis libero, at viverra enim
          gravida non.
  </p>
  <p id="fourth">
    Donec tincidunt tellus et aliquam placerat. Vivamus mattis molestie rutrum. Nullam
          tincidunt risus eu arcu consequat, vitae ullamcorper enim finibus. Pellentesque
          dapibus mi eu iaculis finibus.
  </p>
</section>
```

and CSS

CSS Box Model II

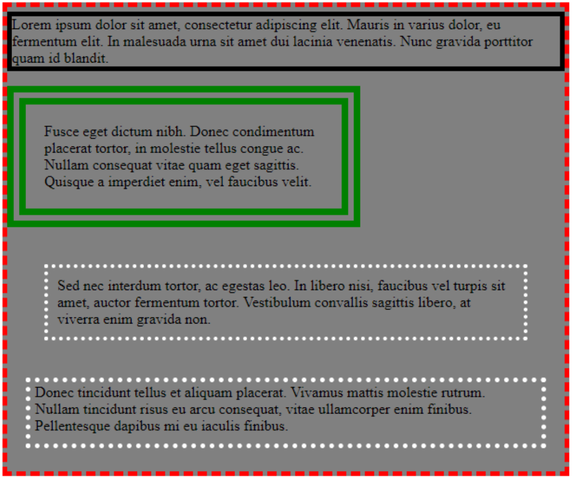
```css
section{
  border: 5px dashed red;
  background-color: #808080;
  height: 500px;
  width: 600px;
}

#first{
  padding: 0px;
  margin-top: 5px;
  border: 5px solid black;
}

#second{
  width: 50%;
  padding: 20px;
  border: 20px double green;
  margin: 0px;
}

#third{
  padding: 10px;
  border: 4px dotted white;
  margin: 40px;
}

#fourth{
  padding-top: 2px;
  padding-bottom: 10px;
  padding-left: 5px;
  padding-right: 40px;
  border-width: 5px;
```

CSS Box Model III

```
  border-style: dotted;
  border-color: white;
  margin: 20px;
}
```

CSS Box Model

we obtain the output



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris in varius dolor, eu fermentum elit. In malesuada urna sit amet dui lacinia venenatis. Nunc gravida porttitor quam id blandit.

Fusce eget dictum nibh. Donec condimentum placerat tortor, in molestie tellus congue ac. Nullam consequat vitae quam eget sagittis. Quisque a imperdiet enim, vel faucibus velit.

Sed nec interdum tortor, ac egestas leo. In libero nisi, faucibus vel turpis sit amet, auctor fermentum tortor. Vestibulum convallis sagittis libero, at viverra enim gravida non.

Donec tincidunt tellus et aliquam placerat. Vivamus mattis molestie rutrum. Nullam tincidunt risus eu arcu consequat, vitae ullamcorper enim finibus. Pellentesque dapibus mi eu iaculis finibus.

CSS Box Model

It is important not to overspecify details. For example, if a **p** element is nested inside a **section** element of a given size, we cannot tell the **p** element to take up a width of 100% but have a margin of 80 pixels...

There is also something known as **margin collapsing**: if there are two adjacent elements and one has a margin of **x**, say, while the other has a margin of **y**, then the margin between them will be their maximum, i.e., **max(x,y)**.

CSS Display

The **display** property can change how elements behave in the flow of a webpage. Its values include:

- **block** (the elements will "stack" like paragraphs do by default, not sharing a line with other elements, inserting line breaks before/after themselves, and taking up the maximum width possible)
- **inline** (the elements will flow inline and from line-to-line, staking up the least amount of space possible)
- **inline-block** (a hybrid: can share a line with other elements but cannot be split into multiple lines)
- **none** (the element will not be displayed)

CSS Display I

With HTML

```html
<p class="block">
  Lorem ipsum dolor sit amet, consectetur adipiscing elit.
</p>
<p class="block">
  Mauris in varius dolor, eu fermentum elit.
</p>
<p class="block">
  Donec condimentum placerat tortor, in molestie tellus congue ac. Nullam consequat vitae
        quam eget sagittis. Quisque a imperdiet enim, vel faucibus velit.
</p>
<p class="inline">
  Lorem ipsum dolor sit amet, consectetur adipiscing elit.
</p>
<p class="inline">
  Mauris in varius dolor, eu fermentum elit.
</p>
<p class="inline">
  Donec condimentum placerat tortor, in molestie tellus congue ac. Nullam consequat vitae
        quam eget sagittis. Quisque a imperdiet enim, vel faucibus velit.
</p>
<p class="inlineblock">
  Lorem ipsum dolor sit amet, consectetur adipiscing elit.
</p>
<p class="inlineblock">
  Mauris in varius dolor, eu fermentum elit.
</p>
<p class="inlineblock">
  Donec condimentum placerat tortor, in molestie tellus congue ac. Nullam consequat vitae
        quam eget sagittis. Quisque a imperdiet enim, vel faucibus velit.
</p>
```

CSS Display II

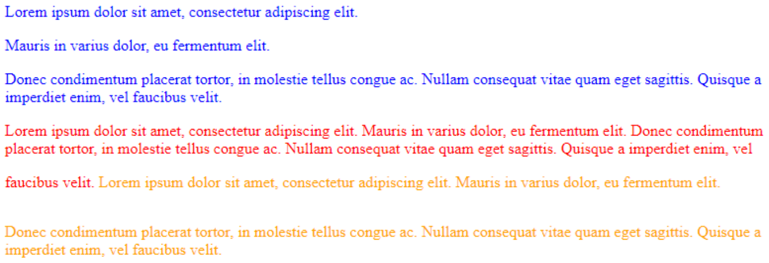
```html
<p class="none">
  I do not display
</p>
```

and CSS

```css
.inline{
  color: red;
  display: inline;
}

.block{
  color: blue;
  display: block;
}

.inlineblock{
  color: orange;
  display: inline-block;
}

.none{
  color: purple;
  display: none;
}
```

CSS Text

we obtain the output



Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Mauris in varius dolor, eu fermentum elit.

Donec condimentum placerat tortor, in molestie tellus congue ac. Nullam consequat vitae quam eget sagittis. Quisque a imperdiet enim, vel faucibus velit.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris in varius dolor, eu fermentum elit. Donec condimentum placerat tortor, in molestie tellus congue ac. Nullam consequat vitae quam eget sagittis. Quisque a imperdiet enim, vel

faucibus velit. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris in varius dolor, eu fermentum elit.

Donec condimentum placerat tortor, in molestie tellus congue ac. Nullam consequat vitae quam eget sagittis. Quisque a imperdiet enim, vel faucibus velit.

CSS Classification

For properties that fit within the Classification category, we consider:

- **position:** how the element should be placed on the page, relative to what other positions, etc.
- **float:** whether text can float around it (for images)
- **clear:** specifies which sides of an element where floats are not allowed
- **z-index**: the larger an element's z-index, the more it will cover other elements occupying the same space

CSS Classification - position

Each element can be given the following values of **position:**

- **static** (its default placement)
- **fixed** (maintaining a fixed placement with respect to the browser window, useful to keep a menubar always in view, say)
- **absolute** (being placed with respect to its closest non-static ancestor element - or the window itself if there are none)
- **relative** (being placed relative to where it would normally be in the text)
- **sticky** (a hybrid of relative and fixed: element remains in its relative position until the user has scrolled sufficiently, then it moves to keep a fixed position)

When giving non-static values, we can specify the offsets of **top**, **right**, **bottom**, or **left**.

CSS Classification - position I

Consider HTML

```
<section>
  <p id="static">
    STATIC
  </p>
  <p id="fixed">
    FIXED
  </p>
  <p id="absolute">
    ABSOLUTE
  </p>
  <p id="relative">
    RELATIVE
  </p>
  <p id="sticky">
    STICKY
  </p>
</section>
```

and CSS

CSS Classification - position II

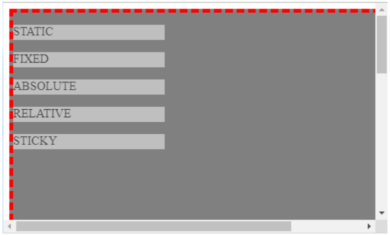
```css
section{
  position: relative;
  border: 5px dashed red;
  background-color: #808080;
  height: 900px;
  width: 600px;
}

p{
  opacity: 0.5;
  background-color: white;
  height: 40px;
}

#static{
  position: static;
  top: 30px;
  right: 30px;
}

#fixed{
  position: fixed;
  top: 30px;
  right: 30px;
}

#absolute{
  position: absolute;
  top: 30px;
  right: 30px;
}

```

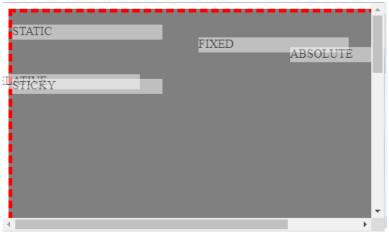
CSS Classification - position III

```
#relative{
  position: relative;
  top: 30px;
  right: 30px;
}

#sticky{
  position: sticky;
  top: 30px;
  right: 30px;
}
```

CSS Classification - position

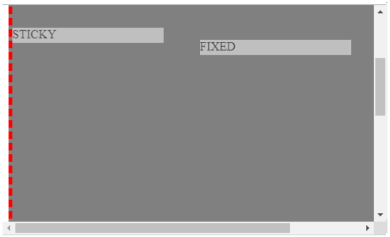
Left: if the CSS were **not applied**. **Right:** with the CSS.

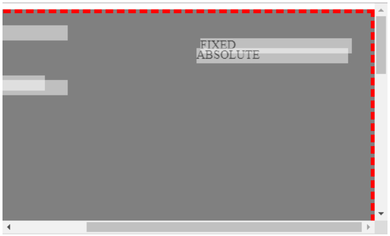




CSS Classification - position

Left: scrolling down. **Right:** scrolling right.





CSS Classification - position

The **position** values of **absolute** and **fixed** and the **float** values (to come) of **left** and **right** take an element out of its normal flow. Other elements ignore the presence of the elements that are "out of flow". In the example, the **relative** paragraph is just slightly down- and left-shifted from where it would be if only the **static** paragraph were present.

CSS Classification - float and clear

Values of the **float** property include:

- **left** (so an element floats to the left in its container)
- **right** (so an element floats to the right in its container)
- **none** (so the element does not float)

Values of the **clear** property include:

- **left** (so elements cannot float to the left)
- **right** (so elements cannot float to the right)
- **both** (so elements cannot float either left or right)
- **none** (default, allows floating)

CSS Classification - float and clear I

Consider HTML

```
<p>
  <img src="Galaxy.png" alt="picture of spiral galaxy" class = "float_left" />
  <b>ALLOWS FLOATS</b>. Nullam ultrices in massa sit amet tincidunt. Pellentesque et diam
       elit. Ut id condimentum mauris. Suspendisse potenti. Aenean nunc risus, venenatis
       sed orci non, varius auctor urna. Phasellus ut ullamcorper erat. Donec condimentum
       placerat tortor, in molestie tellus congue ac. Nullam consequat vitae quam eget
       sagittis.
</p>
<p>
  <img src="Galaxy.png" alt="picture of spiral galaxy" class = "float_left" />
  <img src="Galaxy.png" alt="picture of spiral galaxy" class = "float_right" />
  <b>ALLOWS FLOATS</b>. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
  </p>
<p>
  <b>ALLOWS FLOATS</b>. Donec condimentum placerat tortor, in molestie tellus congue ac.
     Nullam consequat vitae quam eget sagittis. Quisque a imperdiet enim, vel faucibus
     velit.
  <img src="Galaxy.png" alt="picture of spiral galaxy" class = "float_right" />
</p>
<p class="clear_right">
  <b>CLEAR RIGHT</b>. Donec condimentum placerat tortor, in molestie tellus congue ac.
     Nullam consequat vitae quam eget sagittis. Quisque a imperdiet enim, vel faucibus
     velit. Mauris in varius dolor, eu fermentum elit. In malesuada urna sit amet dui
     lacinia venenatis. Nunc gravida porttitor quam id blandit.
</p>
```

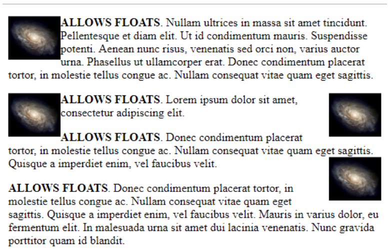
and CSS

CSS Classification - float and clear II

```
img{
  height: 60px;
}

div{
  clear: both;
}

.clear_left{
  clear: left;
}

.clear_right{
  clear: right;
}

.clear_both{
  clear: both;
}

.float_left{
  float: left;
}

.float_right{
  float: right;
}
```

CSS Classification - float and clear

The display with this code is on the **left.**
Without the *clear: right* setting, the display is on the **right.**

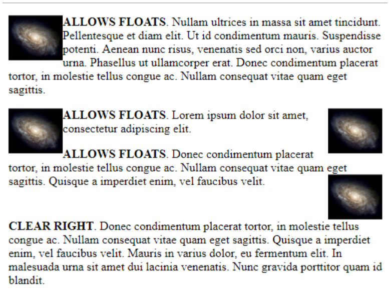


ALLOWS FLOATS. Nullam ultrices in massa sit amet tincidunt. Pellentesque et diam elit. Ut id condimentum mauris. Suspendisse potenti. Aenean nunc risus, venenatis sed orci non, varius auctor urna. Phasellus ut ullamcorper erat. Donec condimentum placerat tortor, in molestie tellus congue ac. Nullam consequat vitae quam eget sagittis.

ALLOWS FLOATS. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

ALLOWS FLOATS. Donec condimentum placerat tortor, in molestie tellus congue ac. Nullam consequat vitae quam eget sagittis. Quisque a imperdiet enim, vel faucibus velit.

ALLOWS FLOATS. Donec condimentum placerat tortor, in molestie tellus congue ac. Nullam consequat vitae quam eget sagittis. Quisque a imperdiet enim, vel faucibus velit. Mauris in varius dolor, eu fermentum elit. In malesuada urna sit amet dui lacinia venenatis. Nunc gravida porttitor quam id blandit.



ALLOWS FLOATS. Nullam ultrices in massa sit amet tincidunt. Pellentesque et diam elit. Ut id condimentum mauris. Suspendisse potenti. Aenean nunc risus, venenatis sed orci non, varius auctor urna. Phasellus ut ullamcorper erat. Donec condimentum placerat tortor, in molestie tellus congue ac. Nullam consequat vitae quam eget sagittis.

ALLOWS FLOATS. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

ALLOWS FLOATS. Donec condimentum placerat tortor, in molestie tellus congue ac. Nullam consequat vitae quam eget sagittis. Quisque a imperdiet enim, vel faucibus velit.

CLEAR RIGHT. Donec condimentum placerat tortor, in molestie tellus congue ac. Nullam consequat vitae quam eget sagittis. Quisque a imperdiet enim, vel faucibus velit. Mauris in varius dolor, eu fermentum elit. In malesuada urna sit amet dui lacinia venenatis. Nunc gravida porttitor quam id blandit.

CSS Classification - float and clear I

When content is floated within a container, its size has no effect upon the border area of the enclosing container so sometimes this can lead to floated content extending beyond the boundary of its enclosing container.

To force the enclosing container to include the floated element, designers often insert a

```
<div class="clear"></div>
```

with styling

```
.clear {
   clear: both;
}
```

CSS Classification - float and clear

Remark: there is no such thing as **float: center**!

To center an element in a container, we can use:

```
margin: 0 auto;
display: block;
```

to give 0 margins at the top/bottom and center with respect to left/right.

CSS Classification - z-index

The **z-index** can take integer values (negative allowed). If two elements take up the same area, the one with the higher z-index goes on top.

Power of CSS

CSS can be very technical, but it is immensely powerful in display.

For some inspration, check out the **CSS Zen Garden**! By toggling the style sheets, the same HTML can appear drastically different.

CSS Selectors

So now we consider how it selects HTML elements. CSS works with what are called **selectors.**

For a first pass, we can say there are three ways to select items:

- **element type**: by explicitly listing the elements like **body** or **h1**;
- **id**: by listing the id of an element like **#about**; and
- **class**: if an element is given a **class="blargh"** attribute-value pair, all elements of that class can be selected with **.blargh.**

All elements can be given a **class** attribute with value that we decide, which can later be used for styling.

CSS Selectors

Consider:

```html
<article id="intro">
  <h1>Personal Introduction</h1>
  <p>
    Lorem ipsum dolor sit amet, <i>consectetur adipiscing elit</i>, sed do eiusmod tempor
          incididunt ut labore et dolore magna aliqua.
  </p>
  <p class="very_important">
    Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat
          nulla pariatur.
  </p>
  <p class="very_important">
    Leo vel fringilla est ullamcorper eget nulla facilisi etiam.
  </p>
</article>
<article>
  <h1>Trip to Hawaii</h1>
  <p>
    Leo vel fringilla est ullamcorper eget nulla facilisi etiam.
  </p>
</article>
```

Then:

- **p** selects all paragraphs;
- **#intro** selects the article element;
- **.very_important** selects the second and third p-elements.

CSS Combinators

A CSS **combinator** allows for multiple elements to be selected and for more specific selection based on an element's family tree.

We can select multiple sets of elements using a comma. We can also select elements based on their ancestry/lineage by concatenating selectors with a space.

- **article, h1**: selects all articles and all h1 elements;
- **#intro, h1**: selects the first article and all h1 elements;
- **#intro p**: selects only p elements within the first article;
- **#intro i**: selects the i element that has #intro as an ancestor;
- **#intro .very_important**: selects only elements with class "very important" that are found in the first article;
- **article h1**: both h1 elements since they are both nested within an article.

There is also the *, the universal selector that selects everything.

CSS Combinators

There is also the **~**, **>**, and **+** syntax to represent siblings, a parent-child relation, and an immediately-follows relationship.

- **h1 ~ p**: selects all of the **p** elements because they are all siblings that come after an **h1** element
- **p > i**: selects the **i** that is a child of **p**
- **article > i**: selects nothing since there is no **i** being a direct child of **article**
- **h1 + p**: selects only the first paragraphs of both articles because they both immediately follow an **h1**

The ~ is new to CSS3 along with several others.

Pseudo-Classes

Through CSS we can also select what are called **pseudo-classes.** These are used to define special states of an element, such as when a link is moused over, when a link has been clicked, if an element is the first of its parent, if an input element is in focus, etc.

The syntax for a pseudo-class is **element:special_state**

We will consider a few in context:

```
p:hover{
   color: green;
}
```

When a paragraph is moused over, its text turns green.

Pseudo-Classes

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

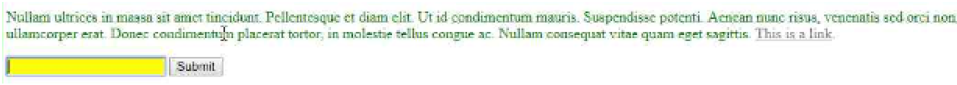
When an anchor has been clicked on, its text becomes gray. Likewise we can select all unvisited links with **a:link** and links being clicked on with **a:active**.

```
input:focus {
   background-color: yellow;
}
```

When an input field is receiving input from the user, the field is yellow. Likewise we can select all inputs that are checked with **input:check**, all inputs that are invalid with **input:invalid**, and all valid inputs with **input:valid**.

Pseudo-Classes

Using the preceding CSS pseudo-classes we have something such as:



Nullam ultrices in massa sit amet tincidunt. Pellentesque et diam elit. Ut id condimentum mauris. Suspendisse potenti. Aenean nunc risus, venenatis sed orci non, ullamcorper erat. Donec condimentum placerat tortor, in molestie tellus congue ac. Nullam consequat vitae quam eget sagittis. This is a link.

Pseudo-Classes

With HTML excerpt:

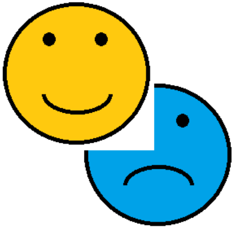
```html
<main>
  <section>
    <img src="happy.png" alt="smile" id="smiley" />
    <img src="sad.png" alt="frown" id="frowny" />
  </section>
</main>
```

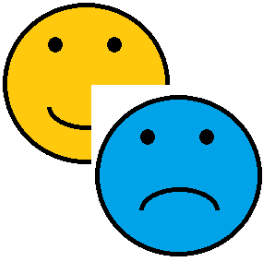
and CSS

```css
section{
  position: relative;
}

#smiley{
  position: absolute;
  top: 50px;
  left: 50px;
  z-index: 2;
}

#frowny{
  position: absolute;
  top: 150px;
  left: 150px;
  z-index: 1;
}

#frowny:hover{
  z-index: 3;
}
```

Pseudo-Classes

Depending on whether the user’s mouse is hovering over the frowney face, we can have one of two displays:





Pseudo-Elements

A CSS **pseudo-element** allows for a specific part of an element to be selected. The syntax for a pseudo-element is: **element::specific_part**. Here are some examples:

```
p::before {
   content: "~";
}
```

Before every **p** element, a tilde ~ will be inserted. Likewise we can insert content after each paragraph element with **p::after**.

The **content** property simply dictates what should be placed in the space.

Pseudo-Elements

```
p::first-line {
   font-weight: 900;
}
```

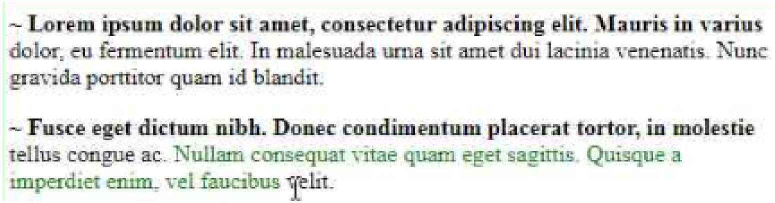
The first letter of every paragraph is given a heavy font weight. We can also select the first line of every paragraph with **p::first-letter**. Both **first-letter** and **first-line** can only be applied to block-level elements.

```
p::selection {
   font-color: green;
}
```

Above, whatever the user has selected within a paragraph, the font size increases and its color goes blue.

Pseudo-Elements

Using the preceding CSS pseudo-elements we have something such as:



~ **Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris in varius** dolor, eu fermentum elit. In malesuada urna sit amet dui lacinia venenatis. Nunc gravida porttitor quam id blandit.

~ **Fusce eget dictum nibh. Donec condimentum placerat tortor, in molestie** tellus congue ac. Nullam consequat vitae quam eget sagittis. Quisque a imperdiet enim, vel faucibus velit.

Remark: ::selection is new to CSS3 and may soon be obsolete. As of September 11, 2018, the W3C Recommendations have removed their documentation of this pseudo element. Reading from their current documentation on **::selection**:

This section intentionally left blank. (This section previously defined a ::selection pseudo-element.) ~W3C

It is unclear what will happen to **::selection** in the future.

Inserting CSS

CSS code can be inserted in one of three ways:

- **inline** (should be avoided at all costs because it is hard to maintain) by adding a **style** attribute to an element:

  ```
  <p style="font-size: 2em;">A paragraph</p>
  ```

- **style element** in the **head** (should also be avoided as it can also be difficult to maintain and makes code look sloppy):

  ```
  <head>
    ...
    <style>
      p {
        font-size: 2em;
      }
    </style>
  ...
  </head>
  ```

Inserting CSS

- **external css** by **link**ing another style sheet in the **head**

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

This is the way most web developers prefer.

CSS Inheritance and Specificity

The "cascading" part of CSS comes from the cascade of rules that get applied to determine the styling of an element.

Firstly, many properties are **inherited**. For example, if the **color** of an **article** was set to red, by default the **p** elements would also have their text red, as would any **b** elements, say, nested within them.

But not all properties are inherited. If an **article** was given a dashed blue border, the **p**aragraphs within it would not themselves be given such a border by default.

CSS Inheritance and Specificity

Sometimes CSS stylings could override inheritance and if we don’t want that, we can specify a value of **inherit** for most properties.

Another value many properties can be assigned is **initial**, representing their unmodified state. For example,

```
p {
   color: red;
}
b {
   color: initial;
}
```

Given code above, a **b** within **p** would appear in its default color, not red.

CSS Inheritance and Specificity

When two different stylings could apply to a single element, the styling that is applied is set by the **specificity.** Specificity requires calculating a 4-tuple, (d_1, d_2, d_3, d_4), for each element.

- $d_1 = 1$ if there is an inline style applied (shouldn't be done!) and $d_1 = 0$ otherwise.
- $d_2 =$ the number of id selectors, the #'s used.
- $d_3 =$ the number of classes and pseudo-classes used.
- d_4 = the number of elements and pseudo-elements used.

Specificity is ordered **lexicographically** based on the 4-tuple.

CSS Inheritance and Specificity

For some examples:

```
* { /* (0,0,0,0): select all does not list any elements, classes, ids, etc. */
   color: red;
}

p::selection { /* (0,0,0,2): element and pseudo-element */
   color: red;
}

#setup .important { /* (0,1,1,0): one id, one class */
   color: red;
}

.first_part .important { /* (0,0,2,0): two classes */
   color: blue;
}
```

CSS Inheritance and Specificity

When two items tie for specificity, the most recent (farthest down on the page) wins out.

The most specific style possible will apply so that stylings for **p** will override styles for **main**, for example.

CSS Inheritance and Specificity I

```html
<p id="setup">
  <span class="first_part"><i>O</i>nce upon a <span class="important">time </span>, there
        were students studying PIC 40A</span>...
</p>
<p>
  <b>A</b>nd they all lived happily ever after.
</p>
```

with CSS:

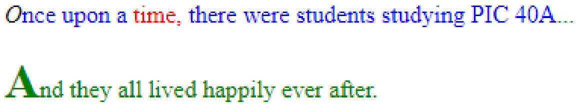
```css
i{ /* describes the O */
  color: black;
}

b{ /* describes b's */
  font-size: 20em;
}

b{ /* overrides above b font-size */
  font-size: 2em;
}

p {
  color: green;
  font-size: 1em;
}

#setup .important{
  color: red;
}

```

CSS Inheritance and Specificity II

```
.first_part .important{
  color: orange;
}

.first_part {
  color: blue;
}
```

CSS Inheritance and Specificity I

The following output would be generated:


Once upon a time, there were students studying PIC 40A...

And they all lived happily ever after.

By default all **p** elements are given green color and standard font-size.

The class **first_part** comes within the **p** element so we start to see the blue font specified by the **.first_part.**

The **i** element comes within the **important** class so we see the style applied to **i** of black is applied instead of the style specified by **.first_part.**

CSS Inheritance and Specificity II

The word "time" can be described as *either* descended from **#setup .important** or **.first_part .important**. The former has specificity (0,1,1,0) > (0,0,2,0) so the red of **#setup .important** wins *because it has a higher specificity*.

b is specified twice but only the style furthest down is considered, hence font-size of double vs twenty times. For the "A", **b** is more descriptive than **p**, so the **b** font size (2 em) wins out.

CSS Validation

Unlike prior CSS versions, CSS3 is released in modules. One can read the W3C Recommendations **here.**

CSS can be validated to meet CSS3 requirements**. Here is one** option.

CSS Style

Here are some important items when it comes to writing good, clean CSS. We will adopt these standards for the course:

- Never use inline styling and always link to external style sheets in the head.
- Never specify a rule as **!important**.
- Separate multiple *property: value* pairs on separate lines.
- Organize the style sheet in a roughly sequential order (stuff for header, stuff for the body, stuff for footer) and comment the part of the page each section is targeting.
- Avoid code-bloating and combine elements when it will not be confusing with the comma syntax: **h1, h2, h3** for example.
- Use descriptive class names, but don’t be too specific: prefer a class of **important** over **stuff_that_appears_red_and_bolded**, for example.
- In using font-families, always include a generic name as backup.
- Ensure the code validates.

CSS Style

Remark on **!important.** We actually haven't discussed it yet. We will look at it simply for completeness but it is considered very bad style.

Consider:

```
p {
   font-size: 2em !important;
}
```

Basically no matter what styles could apply to a **p**, even if other stylings have higher specificity, the font-size will be 2em. This sort of approach is hard to maintain and can be frustrating when the normal specifity rules appear to be broken.

Website Usability

Usability is the notion of how easy it is to use a product. You've probably been to websites that are sorely lacking in usability: you can't find your way around, you don't know the gist of the articles you are reading, the navigation is broken, it's hard to find contact information, you don't know how to log in, etc.

Designing a website that is usable is an art and a lot of research has gone into what makes a great website. We'll consider a bit of the research here.

Website Usability

Eye-tracking studies have shown that many users of a webpage scan the information in an **F-shape**.

This pattern suggests that users: first scan across the top of the page, presumably looking at some heading. Then they scan a ways down along the left and read another heading or the start of a paragraph. Finally they scan down along the side of the page. Other patterns exist but this is often the general idea.

This takes place very quickly and if they don’t find the information they are looking for, they may leave your page within a few seconds and a potential **conversion** (taking someone from a visitor to a client/user) has been lost.

Remark: it has been suggested this phenomena is most present in poorly formatted webpages with "walls of text" without formatting applied (sectioning, bold facing, bulleted lists, ec.).

Website Usability

Keeping usability in mind and how users will (quickly) interact with your webpage, the following are some recommended usability tips:

- Use headings and subheadings to clearly convey information.
- Use bolding and other text formats to draw attention to keywords.
- Use lists to organize information.
- Keep things as simple as possible: avoid unnecessary information, minimize the work a user has to do to log in or use your page.
- Optimize your webpage for different platforms (desktop and mobile are different).
- As your webpage changes, track how the conversion rate changes and fix things if necessary.
- Do A/B testing by varying the design and studying the impact that has on users.
- Include the most crucial information in the first paragraph or two of each section - even try to start paragraphs off with the most important words and information.

CSS3 Features

In this final section, we'll consider some more features of CSS that are unique to CSS3. The features we consider are:

- **opacity:** how opaque an element is. The range is from 0-1. By default elements are set to 1.
- **text-shadow:** gives a shadow taking properties: *horizontal_shadow vertical_shadow blur_radius color*. The **blur_radius** and **color** are optional.
- **media queries:** can change CSS styles based on the screen size of the viewer (think of optimizing this for smart phones vs tablets vs desktops)

CSS3 Features

To select based on screen size the syntax is:

```
@media screen and (max-width: nnn px) {
   /* regular CSS */
}
```

where **nnn** is the number of pixels to use as a threshold, **max-width** represents a screen of width less than or equal to **nnn**, and **min-width** can also be used representing a screen of width greater than or equal to **nnn.**

Other selection criteria can be used, including the number of bits used per colored pixel, screen resolution, screen orientation...

CSS3 Features

With the HTML

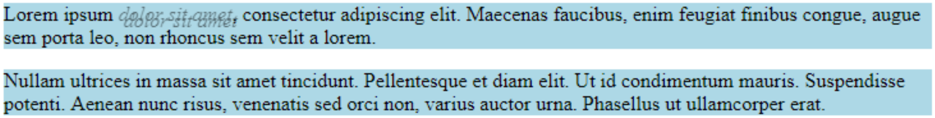
```html
<p>
  Lorem ipsum <i>dolor sit amet</i>, consectetur adipiscing elit. Maecenas faucibus, enim
        feugiat finibus congue, augue sem porta leo, non rhoncus sem velit a lorem.
</p>
<p>
  Nullam ultrices in massa sit amet tincidunt. Pellentesque et diam elit. Ut id condimentum
        mauris. Suspendisse potenti. Aenean nunc risus, venenatis sed orci non, varius
        auctor urna. Phasellus ut ullamcorper erat.
</p>
```

and CSS

```css
p {
  background-color: lightblue;
}

i {
  opacity: 0.5;
  text-shadow: 4px 5px black;
}

@media screen and (max-width: 600px) { /* on small screens */
  p { /* make paragraph font heavier */
    font-weight: 700;
  }
}
```

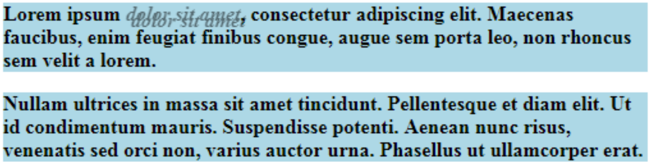
CSS3 Features

We get a display that illustrates the opacity, text-shadow, and a display that differs depending on the screen height/width.



Lorem ipsum *dolor sit amet*, consectetur adipiscing elit. Maecenas faucibus, enim feugiat finibus congue, augue sem porta leo, non rhoncus sem velit a lorem.

Nullam ultrices in massa sit amet tincidunt. Pellentesque et diam elit. Ut id condimentum mauris. Suspendisse potenti. Aenean nunc risus, venenatis sed orci non, varius auctor urna. Phasellus ut ullamcorper erat.



Lorem ipsum *dolor sit amet*, consectetur adipiscing elit. Maecenas faucibus, enim feugiat finibus congue, augue sem porta leo, non rhoncus sem velit a lorem.

Nullam ultrices in massa sit amet tincidunt. Pellentesque et diam elit. Ut id condimentum mauris. Suspendisse potenti. Aenean nunc risus, venenatis sed orci non, varius auctor urna. Phasellus ut ullamcorper erat.