Introducing CSS

Nic Aguirre   j363 Fall 2018

What is CSS?

CSS defined

Up until now, we’ve used **HTML**, which is for **content** - text, images, links

We will use **CSS** *(cascading style sheets)* to control **style** and **appearance** of websites

Example uses:

- Changing font size
- Changing color of elements
- Changing size and position of elements

Constructing a Style Rule

CSS is made up of **rules** that define how a webpage looks. Each rule has two main parts:

1) The **selector -** determines which part of your page is affected (e.g., paragraphs, headings, elements with a given class)

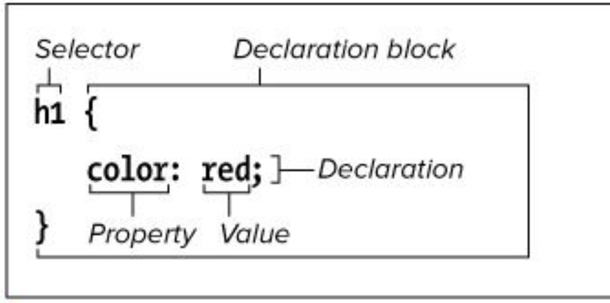
2) The **declaration block** - Made up of **property: value;** pairs that specify the design treatments to apply. (e.g., changing text color or position)

Constructing a Style Rule

CSS is made up of **rules** that define how a webpage looks. Each rule has two main parts:

1) The **selector -** determines which part of your page is affected (e.g., paragraphs, headings, elements with a given class)

2) The **declaration block** - Made up of **property: value;** pairs that specify the design treatments to apply. (e.g., changing text color or position)



```
h1 {
    color: red;
}
```

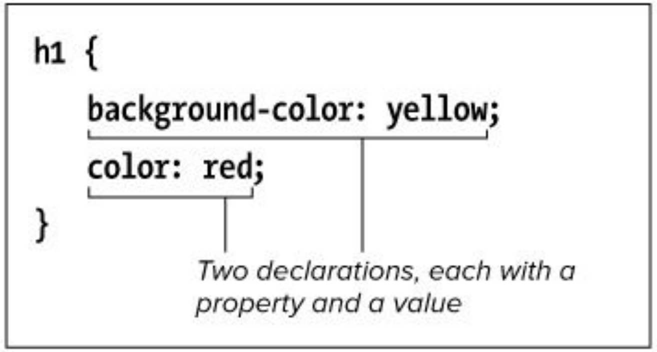
Constructing a Style Rule

Our declaration is made up of **property: value;** pairs.

The order of the declarations doesn't matter unless the same property is defined twice.

Beware of syntax:

- Declaration block begins and ends with curly braces { }
- Properties have a colon after them :
- Values have a semicolon after them ;

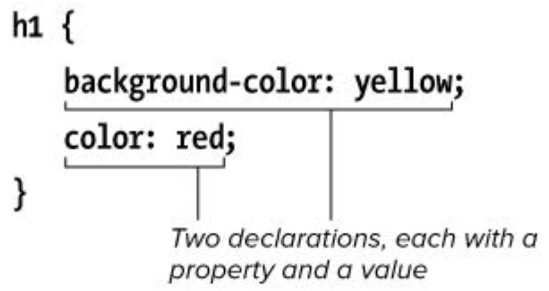


```
h1 {
   background-color: yellow;
   color: red;
}
```

HTML

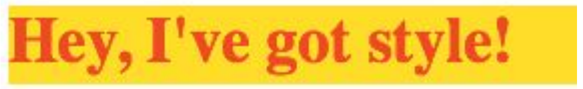
```
<body>
     <h1>Hey, I've got style!</h1>
</body>
```

CSS



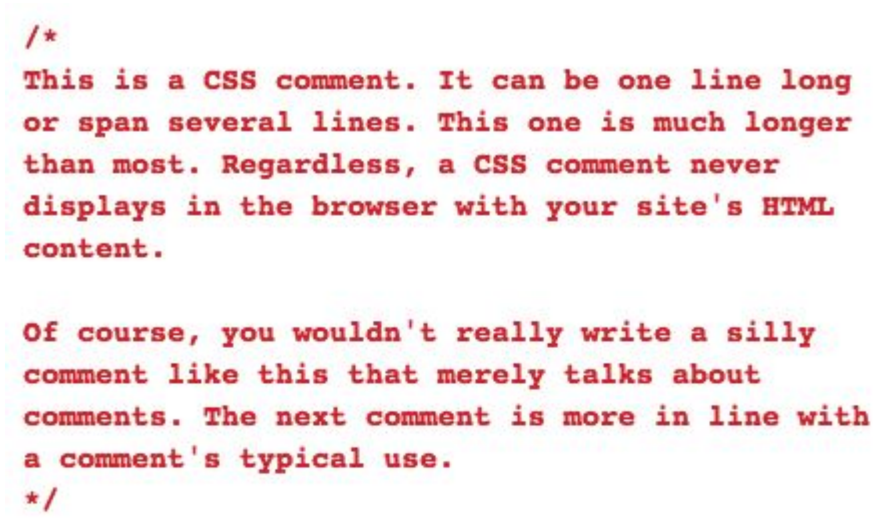
```
h1 {
   background-color: yellow;
   color: red;
}
```

RESULT



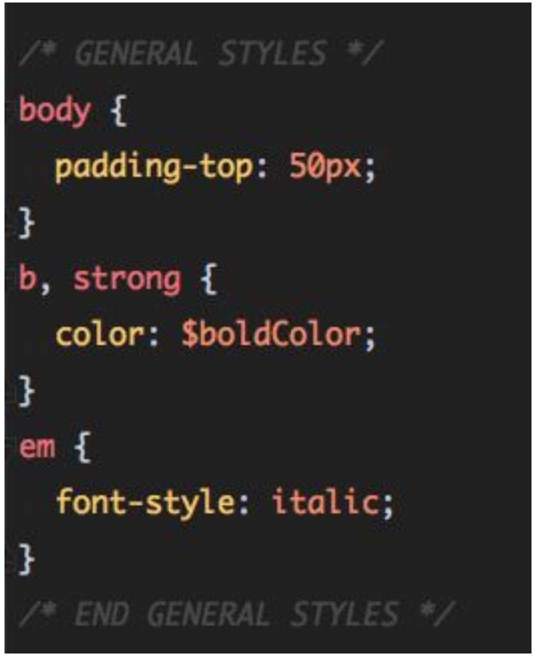
Hey, I've got style!

CSS comments



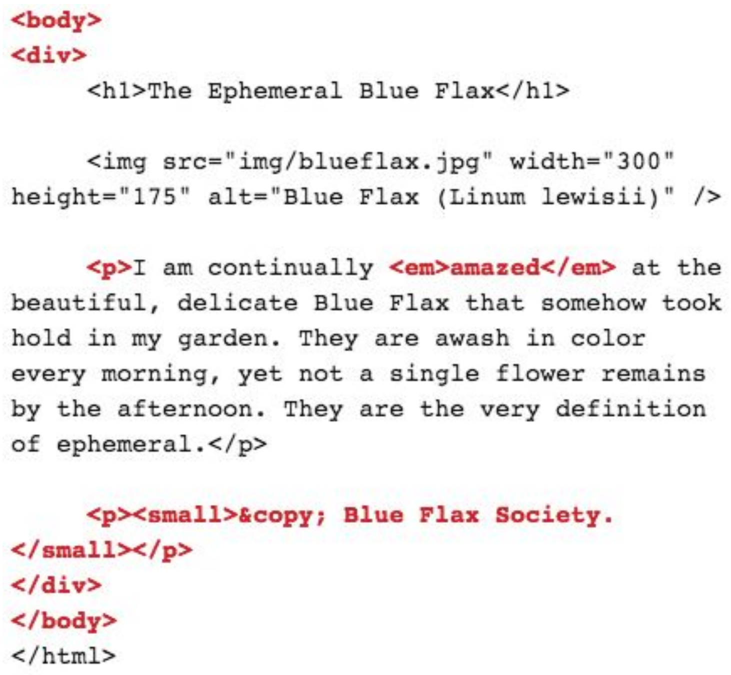
```
/*
This is a CSS comment. It can be one line long
or span several lines. This one is much longer
than most. Regardless, a CSS comment never
displays in the browser with your site's HTML
content.

Of course, you wouldn't really write a silly
comment like this that merely talks about
comments. The next comment is more in line with
a comment's typical use.
*/
```

CSS comments



```
/* GENERAL STYLES */
body {
  padding-top: 50px;
}
b, strong {
  color: $boldColor;
}
em {
  font-style: italic;
}
/* END GENERAL STYLES */
```

Understanding Inheritance



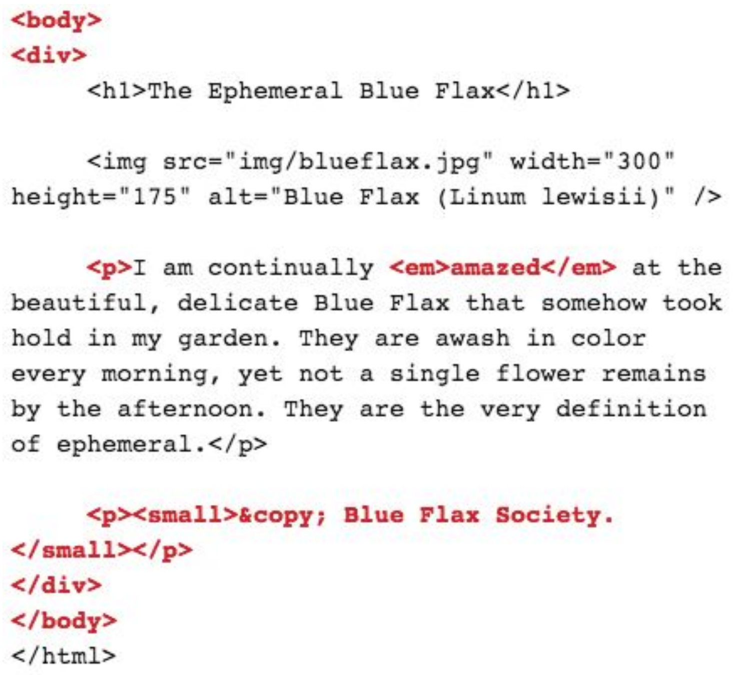
```
<body>
<div>
     <h1>The Ephemeral Blue Flax</h1>

     <img src="img/blueflax.jpg" width="300"
height="175" alt="Blue Flax (Linum lewisii)" />

     <p>I am continually <em>amazed</em> at the
beautiful, delicate Blue Flax that somehow took
hold in my garden. They are awash in color
every morning, yet not a single flower remains
by the afternoon. They are the very definition
of ephemeral.</p>

     <p><small>&copy; Blue Flax Society.
</small></p>
</div>
</body>
</html>
```

The browser sees HTML as a document tree.

When certain CSS properties are applied to an element, they affect not only that element but also flow down the branch or branches beneath them.

Lower elements are said to ***inherit*** from their ancestors.

Understanding Inheritance



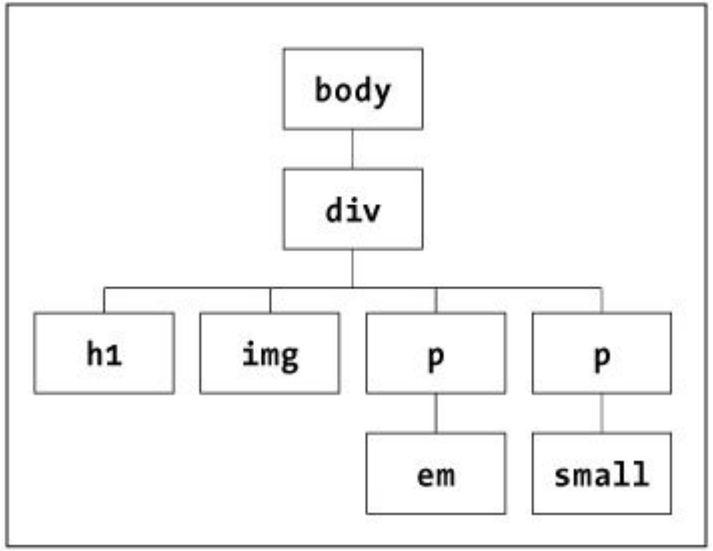
```
<body>
<div>
     <h1>The Ephemeral Blue Flax</h1>

     <img src="img/blueflax.jpg" width="300"
height="175" alt="Blue Flax (Linum lewisii)" />

     <p>I am continually <em>amazed</em> at the
beautiful, delicate Blue Flax that somehow took
hold in my garden. They are awash in color
every morning, yet not a single flower remains
by the afternoon. They are the very definition
of ephemeral.</p>

     <p><small>&copy; Blue Flax Society.
</small></p>
</div>
</body>
</html>
```

Lower elements are said to ***inherit*** from their ancestors.

Example - If you change font color for the **div** element, it will affect elements inside of it, such as <p> and <h1>.

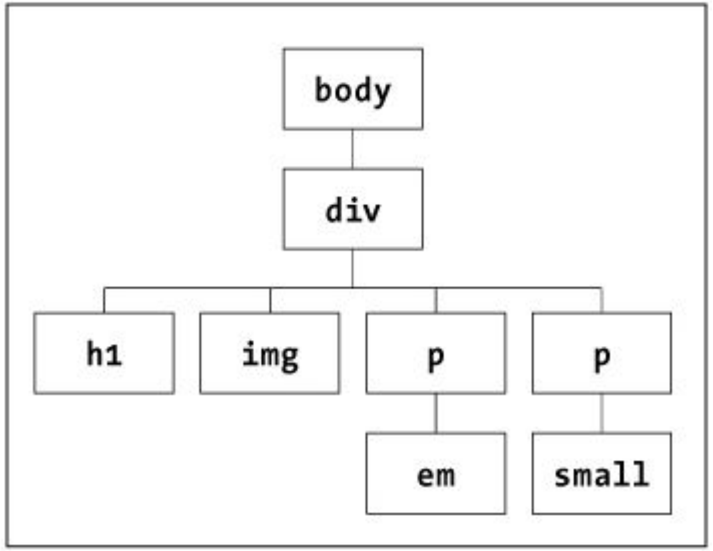
Understanding Inheritance

```
<body>
<div>
     <h1>The Ephemeral Blue Flax</h1>

     <img src="img/blueflax.jpg" width="300"
height="175" alt="Blue Flax (Linum lewisii)" />

     <p>I am continually <em>amazed</em> at the
beautiful, delicate Blue Flax that somehow took
hold in my garden. They are awash in color
every morning, yet not a single flower remains
by the afternoon. They are the very definition
of ephemeral.</p>

     <p><small>&copy; Blue Flax Society.
</small></p>
</div>
</body>
</html>
```



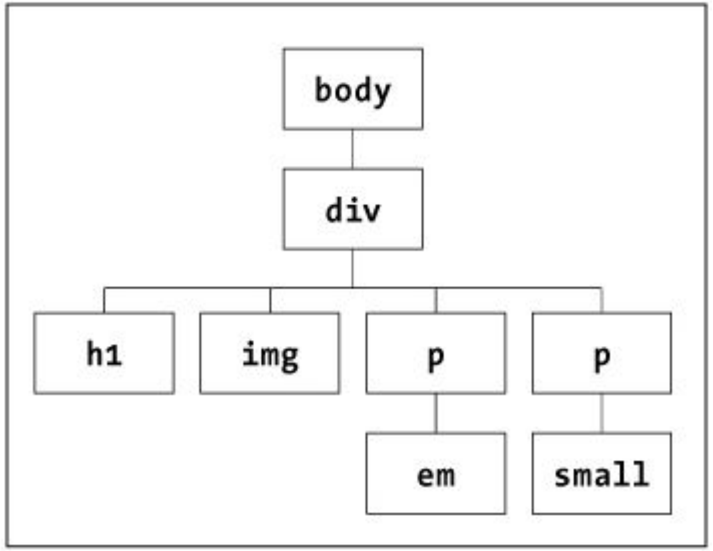
Understanding Inheritance

All elements are descendants of the **body** element. In this case, **div** wraps around everything inside that. Digging deeper, the **em** and **small** elements are contained within a **p** element and thus are descendants of **p**



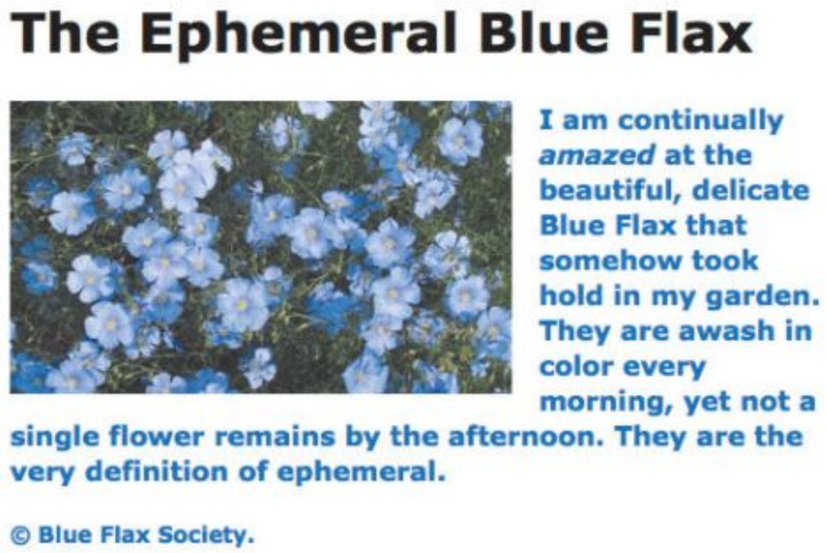
Understanding Inheritance

```
body {
      font-family: Verdana, Geneva, sans-serif;
}

div {
      border: 1px solid #000;
      overflow: hidden;
      padding: 0 1em .25em;
}

p {
      color: #36c; /* a blue color */
      font-weight: bold;
}

img {
      float: left; /* makes text wrap it */
      margin-right: 1em;
}
```



Understanding Inheritance

```
body {
      font-family: Verdana, Geneva, sans-serif;
}

div {
      border: 1px solid #000;
      overflow: hidden;
      padding: 0 1em .25em;
}

p {
      color: #36c; /* a blue color */
      font-weight: bold;
}

img {
      float: left; /* makes text wrap it */
      margin-right: 1em;
}
```



The Ephemeral Blue Flax

I am continually *amazed* at the beautiful, delicate Blue Flax that somehow took hold in my garden. They are awash in color every morning, yet not a single flower remains by the afternoon. They are the very definition of ephemeral.

© Blue Flax Society.

Defining style rules

We apply styles in one of three ways:

1) Inline styles (in HTML document)
2) Style blocks (in HTML document)
3) External stylesheet (a linked CSS file)

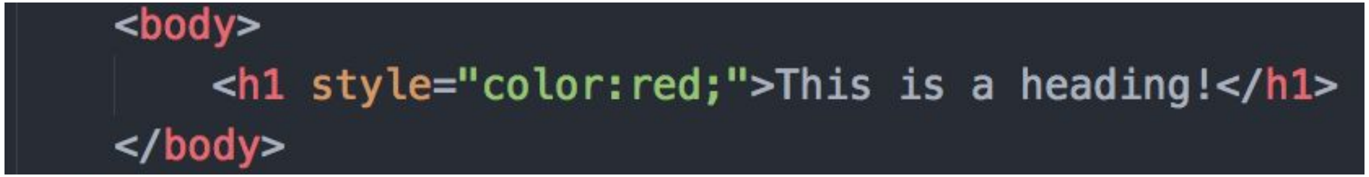
Suppose we want to make all **h1** elements red...

Defining style rules

We apply styles in one of three ways:

1) **Inline styles (in HTML document)**
2) Style blocks (in HTML document)
3) External stylesheet (a linked CSS file)

index.html



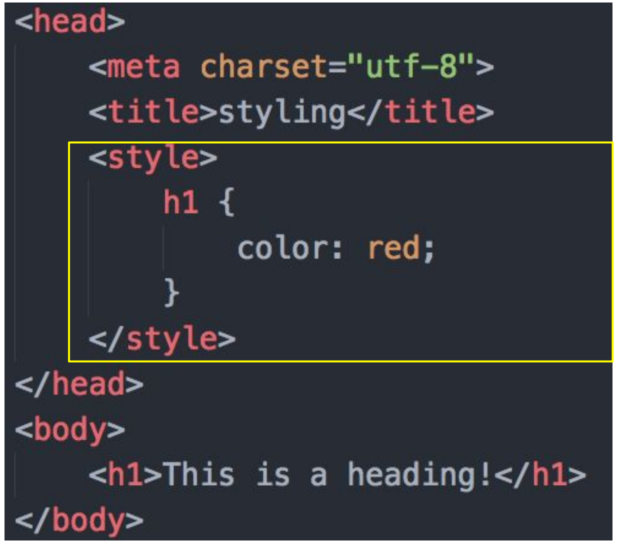
```
<body>
    <h1 style="color:red;">This is a heading!</h1>
</body>
```

Defining style rules

We apply styles in one of three ways:

1) Inline styles (in HTML document)
2) **Style blocks (in HTML document)**
3) External stylesheet (a linked CSS file)

index.html



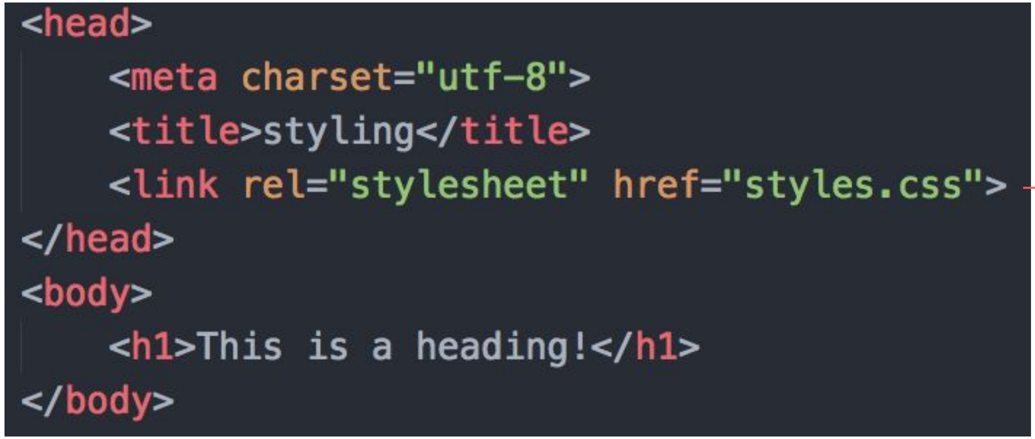
```
<head>
    <meta charset="utf-8">
    <title>styling</title>
    <style>
        h1 {
            color: red;
        }
    </style>
</head>
<body>
    <h1>This is a heading!</h1>
</body>
```

Defining style rules

We apply styles in one of three ways:

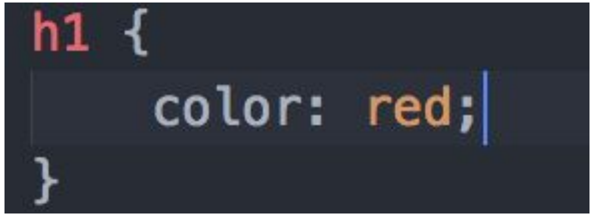
1) Inline styles (in HTML document)
2) Style blocks (in HTML document)
3) **External stylesheet (a linked CSS file)**

index.html



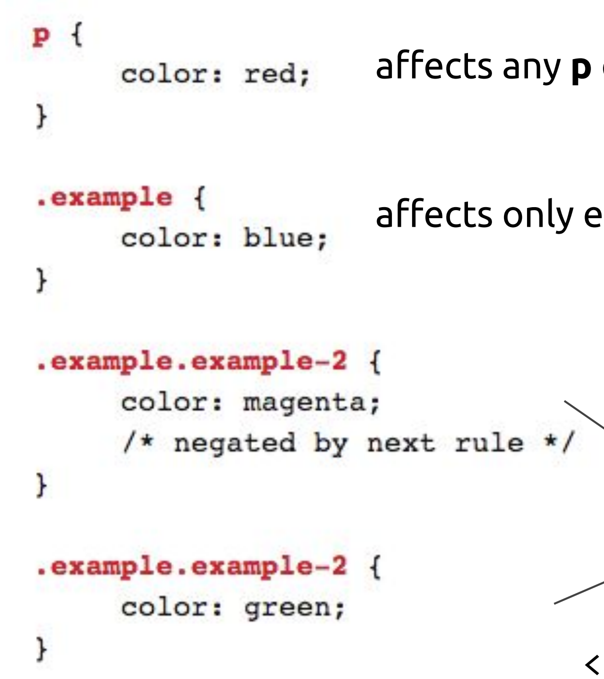
```
<head>
    <meta charset="utf-8">
    <title>styling</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>This is a heading!</h1>
</body>
```

styles.css



```
h1 {
    color: red;
}
```

The Cascade when rules collide



```
p {
        color: red;
}

.example {
        color: blue;
}

.example.example-2 {
        color: magenta;
        /* negated by next rule */
}

.example.example-2 {
        color: green;
}
```

affects any **p** element

affects only elements whose class is equal to **example**

affect only elements with both **example** AND **example-2** classes

`<div class=”example example-2”>...</div>`

The Cascade when rules collide

What happens when there is more than one style rule that applies to a given element?
If one rule defines an element’s **color**, and another rule defines its **width**, there is no conflict and those rules are combined.

However - suppose multiple CSS rules define the same property on an element…

Then CSS uses the principle of ***the cascade*** to determine which of a group of conflicting style declarations takes priority.

Three factors are considered - **specificity**, **order**, and **importance**

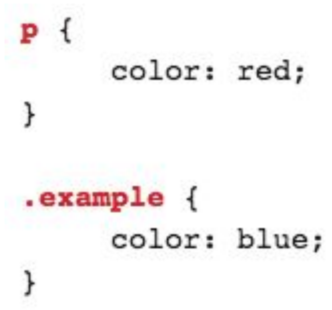
The Cascade when rules collide

Three factors are considered for resolving style rule conflicts:

1. **Specificity**
2. Order
3. importance

The law of *specificity* states that the more specific the selector, the stronger the rule.

Suppose we have a **p** element with class **example** applied:



```
p {
        color: red;
}

.example {
        color: blue;
}
```

The Cascade when rules collide

Three factors are considered for resolving style rule conflicts:

1. Specificity
2. **Order**
3. importance

Sometimes, specificity is the same for competing rules, so it can’t determine a ‘winner’.

In this case, the *order* breaks the tie; rules that are defined later will have greater priority.

(e.g., style rules written on line 11 will take precedence over rules written on line 5)

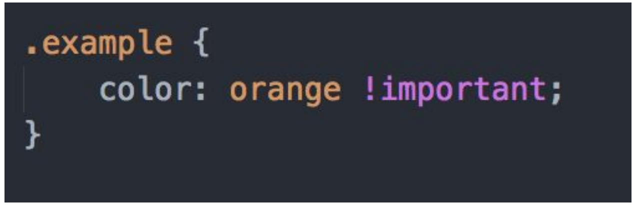
Also, **inline styles** will take priority over any **external styles**

The Cascade when rules collide

Three factors are considered for resolving style rule conflicts:

1. Specificity
2. Order
3. **importance**

The whole system can be overridden by declaring that a particular style should be more **important** than the others, regardless of specificity or order.



```css
.example {
    color: orange !important;
}
```

Properties and Values

Each CSS property has different rules about what values it can accept.

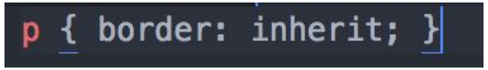
Some only accept predefined values such as **left** or **right**.

Some will accept any range of values we supply, such as **0px**, **500px**, or **50%**

Properties and Values

Use the **inherit** value when you want to explicitly specify that the value of a property should be the same as its parent element.

Example - suppose you have some **<p>** elements inside an **<article>** element that has a border applied to it. Borders aren't normally inherited, so the rule:

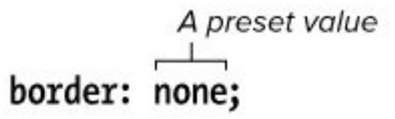


```
p { border: inherit; }
```

Will apply the same border effect to paragraphs; it tells paragraphs that they should inherit border properties from their parent element.

Properties and Values

Most CSS properties have a few **predefined** values that can be used. For example - the **float** property can be set to **left, right**, or **none**.

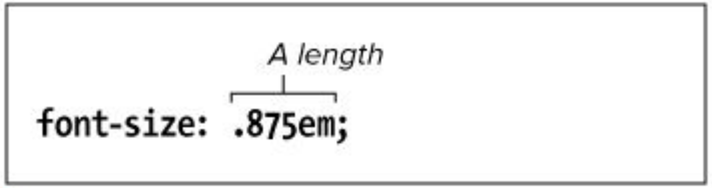


```
border: none;
```

Properties and Values

Many CSS properties take a length as their value. All length values must contain both **a quantity** and **a unit**.

Example:



```
font-size: .875em;
```

Properties and Values

There are length types that are relative to other values. For example an **em** is equal to the element’s font size.

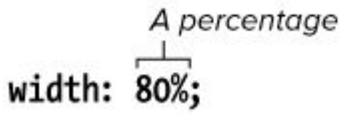
Setting **margin-left: 2em;** on an element would mean that the left margin will be double that element’s font size.

```
div {
    font-size: 16px;
    margin-left: 2em;
}
```

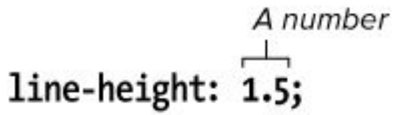
Properties and Values

The values you should be familiar with are:

- **px** - pixels
- **em** - equal to font size
- **%** - percentage - between 0 and 100
- **Bare numbers** *(e.g., 1.5)* no unit specified
- **URLs** - For example, setting a background image
- **CSS** colors *(e.g., green or blue)*



```
width: 80%;
```

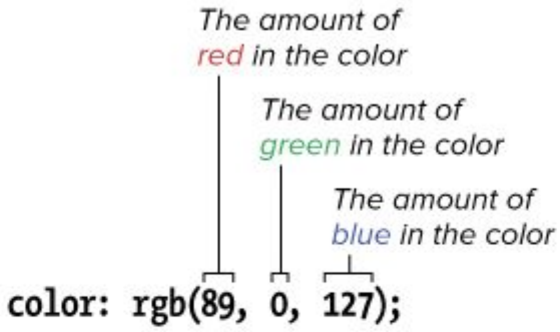


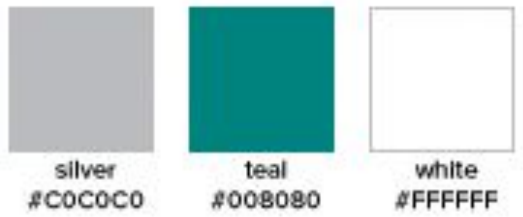
```
line-height: 1.5;
```

Regarding colors

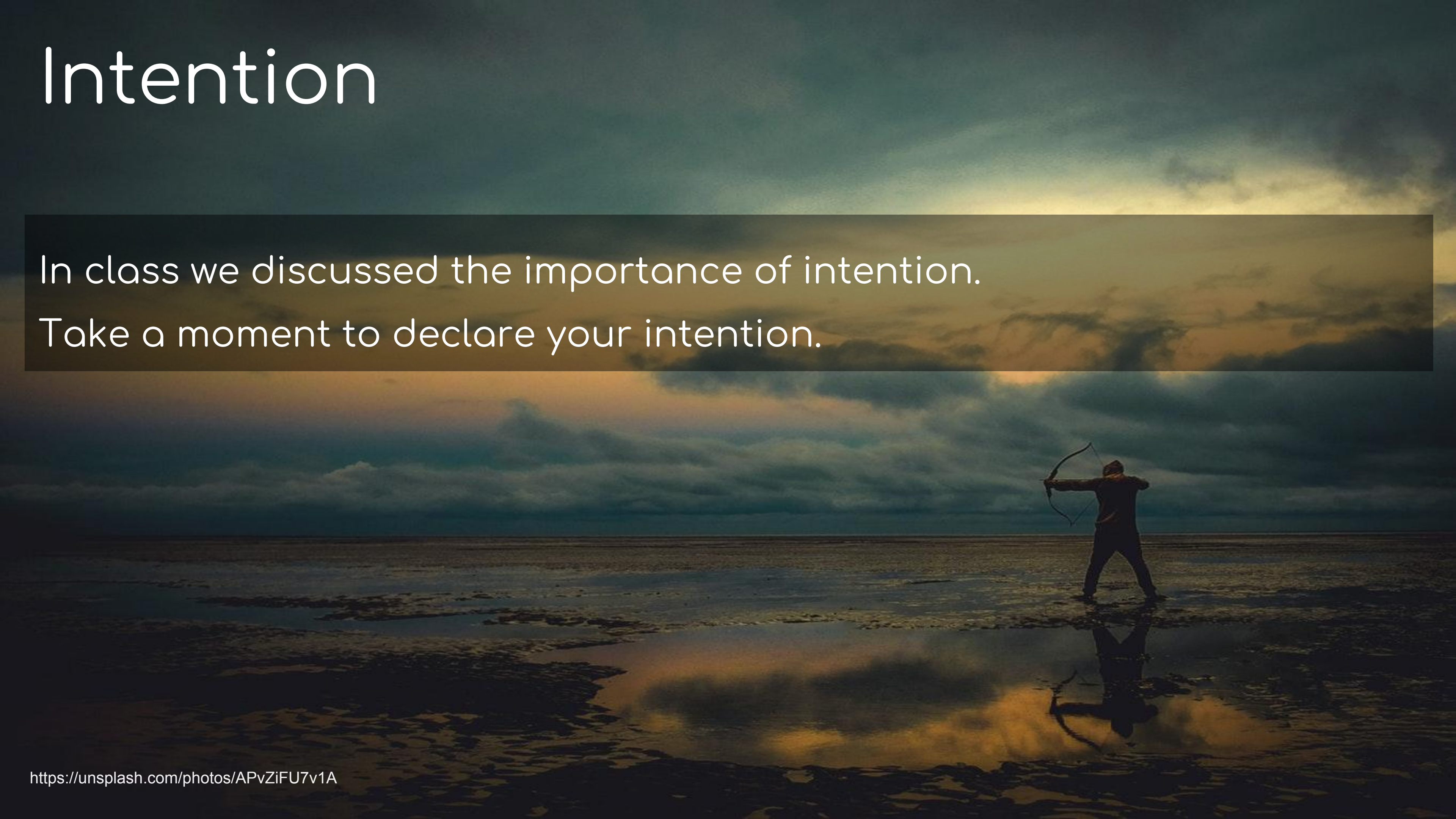
We have several ways of defining colors:

- Named CSS colors such as “green” or “blue”
- RGB - specifying the amount of red/green/blue
- Hexadecimal - six digit code that begins with **#**


The amount of red in the color
The amount of green in the color
The amount of blue in the color
color: rgb(89, 0, 127);



Intention



In class we discussed the importance of intention.

Take a moment to declare your intention.

https://unsplash.com/photos/APvZiFU7v1A