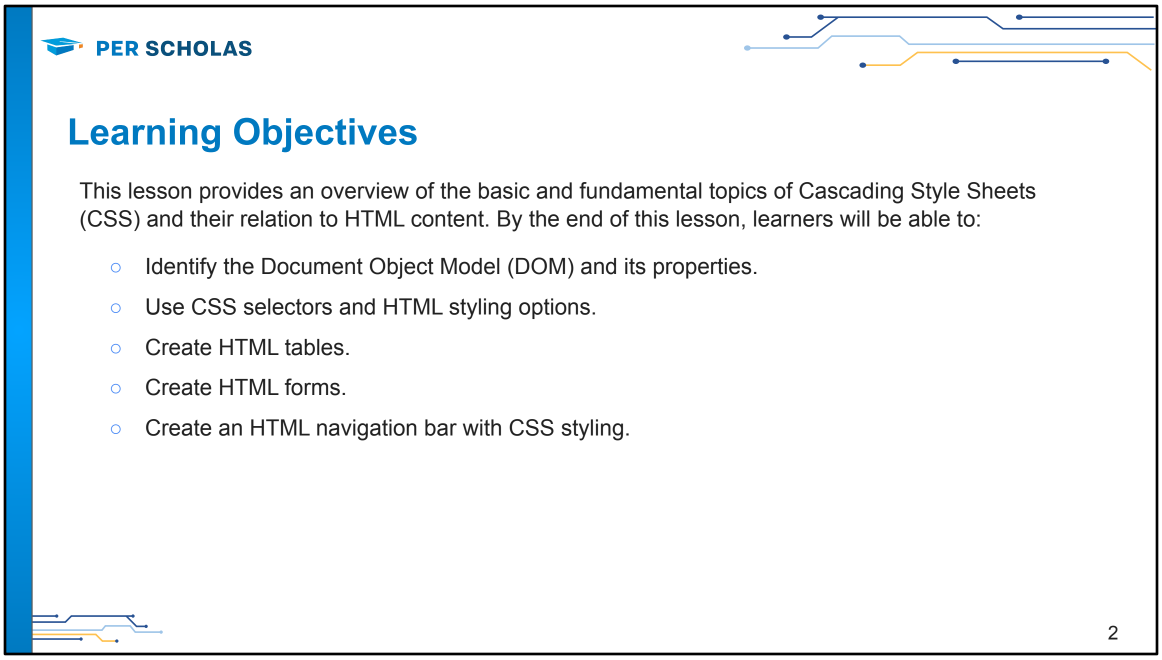
PER SCHOLAS

Fundamentals of HTML/CSS



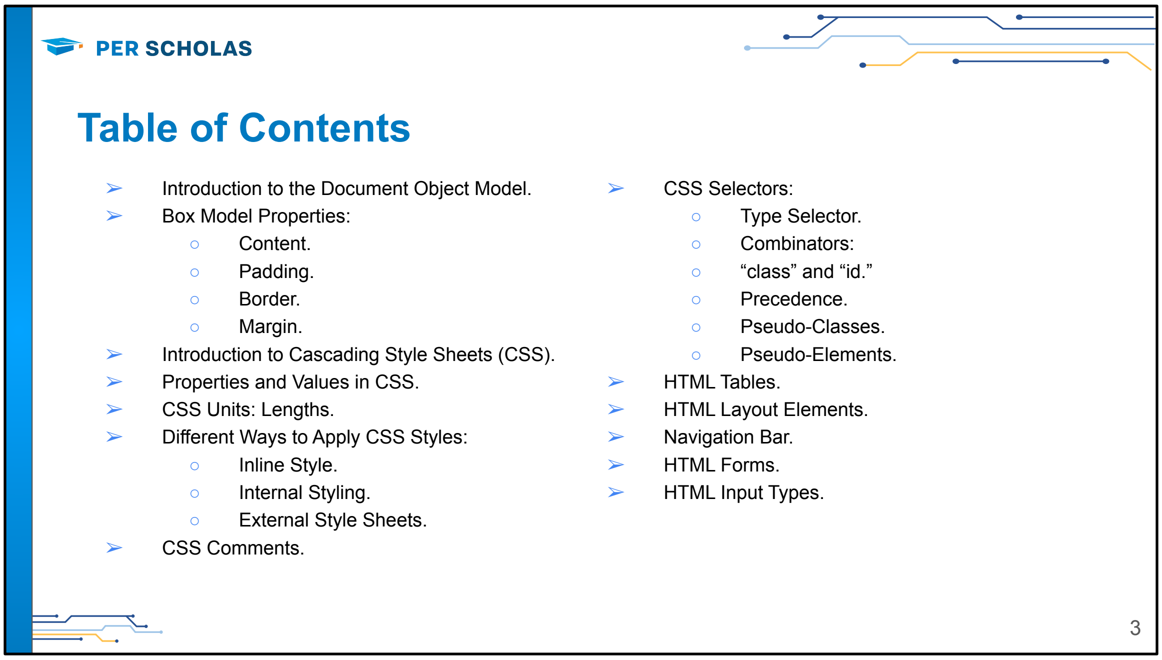
PER SCHOLAS

Learning Objectives

This lesson provides an overview of the basic and fundamental topics of Cascading Style Sheets (CSS) and their relation to HTML content. By the end of this lesson, learners will be able to:

- Identify the Document Object Model (DOM) and its properties.
- Use CSS selectors and HTML styling options.
- Create HTML tables.
- Create HTML forms.
- Create an HTML navigation bar with CSS styling.

2



PER SCHOLAS

Table of Contents

PER SCHOLAS

Introduction to the Document Object Model

The **Document Object Model (DOM)** is a programming interface for HTML and XML documents.

- A web page is a document. This document can be either displayed in the browser window or as the HTML source, but it is the same document in both cases. Likewise, the DOM is an object-oriented representation of the web page, which can be modified with a scripting language such as JavaScript.
- The DOM represents the web page as nodes and objects so that programs can change the document structure, style, and content.
- The W3C DOM and WHATWG DOM standards are implemented in most modern browsers. Many browsers extend the standard; therefore, care must be exercised when using them on the web where documents may be accessed by various browsers with different DOMs.

We will practice DOM manipulation in the JavaScript lessons.

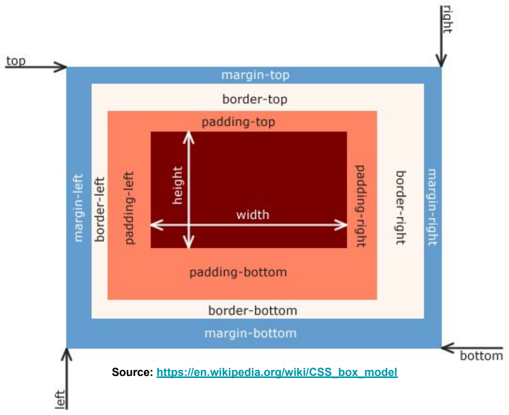
PER SCHOLAS

Box Model Properties

Before understanding how to create CSS layouts, you need to understand the Box Model. In CSS, the term "box model" is used when talking about design and layout.

Box Model:

- The Cascading Style Sheets (CSS) box model is the foundation of a layout on the Web.
- Each element is represented as a rectangular box, with the box's **`content`**, **`padding`**, **`border`**, and **`margin`** built up around one another like layers.
- As a browser renders a web page layout, it determines what styles are applied to the content of each box, how big the surrounding is, and where the boxes sit in relation to one another.
- **`Width`** and **`height`** represent the dimensions of the box. This includes the content inside the box.
    - The **`height`** and **`width`** describe dimensions of the actual content of the box (text, images, ...).

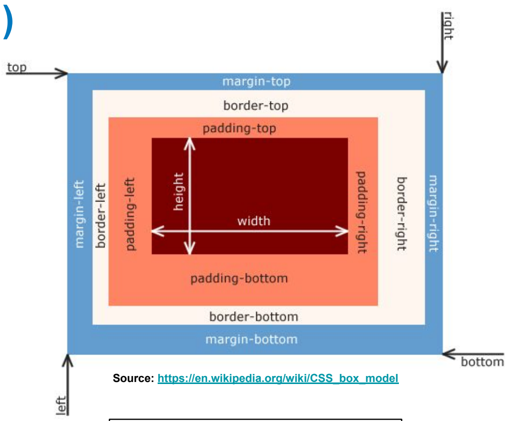


Source: https://en.wikipedia.org/wiki/CSS_box_model

PER SCHOLAS

Box Model Properties (continued)

- **`Padding`** is the space between the inner-edges of the box and the outer-edges of the content, including the border.
  - Padding can be applied as **`padding-left`**, **`padding-right`**, **`padding-top`**, and **`padding-bottom`**, or individually or all at once.
- The **`Border`** sits right on top of the outer edges of the padding and defines the thickness of the edges of the box.
  - Style (including color) can be applied to individual parts of the border.
- **`Margin`** is the space around the border. Think about margin like personal space.
  - Style can be applied to individual parts of the margin (**`margin-left`**, **`margin-right`**, **`margin-top`**, and **`margin-bottom`**).



Source: https://en.wikipedia.org/wiki/CSS_box_model

Note: Too much padding might cause the box to stretch more than its dimensions.

6

PER SCHOLAS

Box Model Properties (continued)

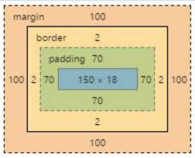
The following example shows the box model diagram on the top-right for the specified code on the left.

1. The top-left section has CSS properties of the **p** element.
2. The bottom-left section has HTML code of the **p** element.
3. The top-right section shows the box model diagram of the **p** element (copied from the browser inspector on Chrome).
4. The bottom-right section shows the output of this HTML code.

CSS

```
<style>
  p {
     padding:70px;
     border:2px solid green;
     margin:100px;
     width:150px;
   }
 </style>
```

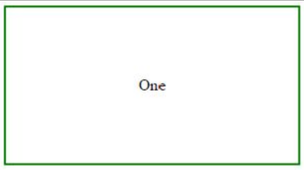
Box Model



HTML

```
<p> One </p>
```

Output



PER SCHOLAS

Introduction to Cascading Style Sheets

Cascading Style Sheets (CSS) is a *style sheet language,* which means that it lets you apply styles selectively to elements in HTML documents. For example, to select all of the paragraph elements on an HTML page and turn the text within them red, you would write this CSS:

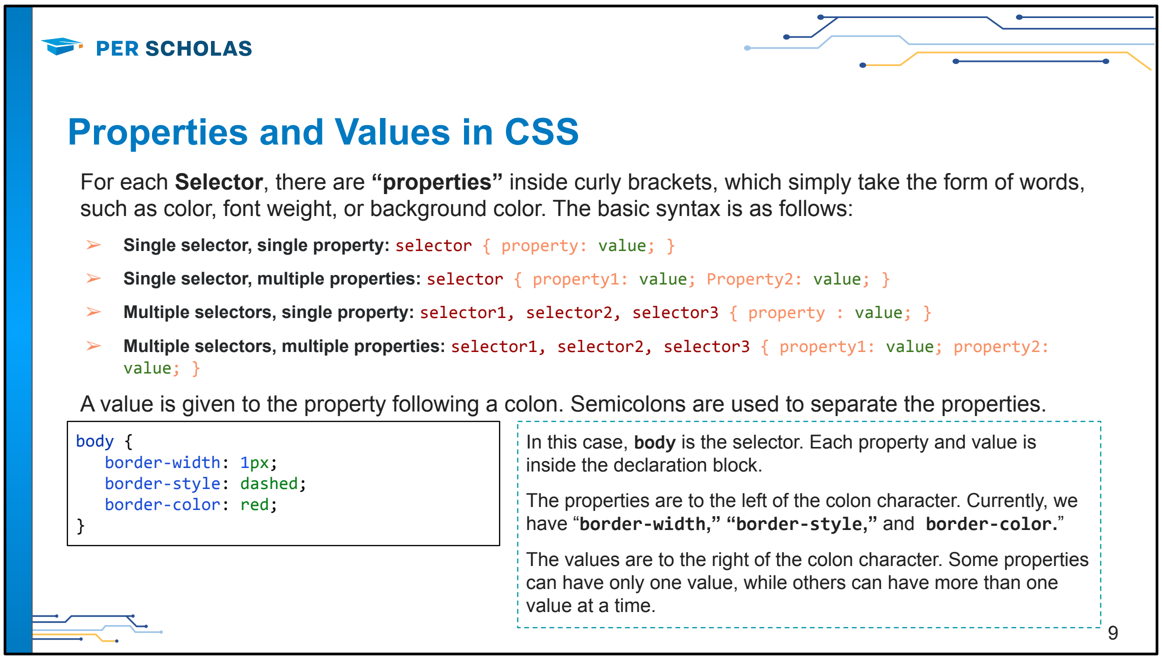
```
p {
   color: red;
}
```

CSS alone has no effect. Take this analogy to understand the difference between HTML and HTML with CSS:

- HTML is like a car, but without paint or polish. It can have content, like furnishings and accessories, but they are all bland and unfinished.
- HTML with CSS is that same car, but with color, style, and polish. The CSS adds the finishing touches to make the car look good as well as drive well.

Web browsers apply CSS rules to a document to affect how they are displayed. This is done by applying:

- **Selectors:** Element(s) you want to apply the updated property values to.
- **Properties:** Values set to update how the HTML content is displayed.



PER SCHOLAS

Properties and Values in CSS

For each **Selector**, there are **"properties"** inside curly brackets, which simply take the form of words, such as color, font weight, or background color. The basic syntax is as follows:

- **Single selector, single property:** `selector { property: value; }`
- **Single selector, multiple properties:** `selector { property1: value; Property2: value; }`
- **Multiple selectors, single property:** `selector1, selector2, selector3 { property : value; }`
- **Multiple selectors, multiple properties:** `selector1, selector2, selector3 { property1: value; property2: value; }`

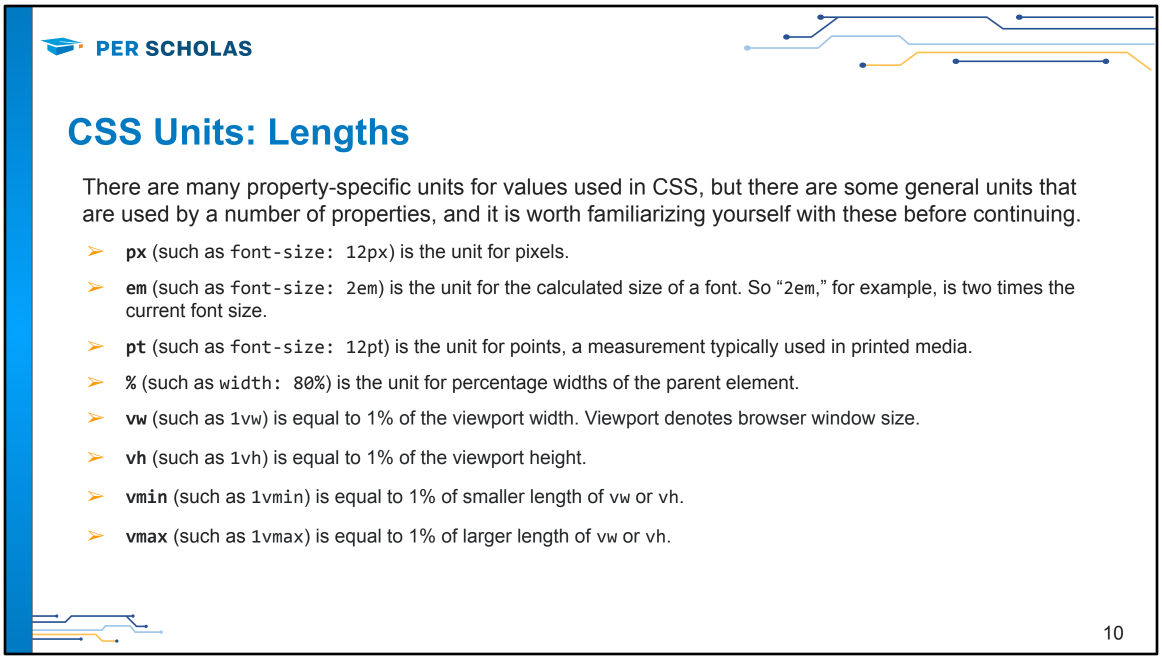
A value is given to the property following a colon. Semicolons are used to separate the properties.

```
body {
   border-width: 1px;
   border-style: dashed;
   border-color: red;
}
```

In this case, **body** is the selector. Each property and value is inside the declaration block.

The properties are to the left of the colon character. Currently, we have "**`border-width`**," "**`border-style`**," and **`border-color`**."

The values are to the right of the colon character. Some properties can have only one value, while others can have more than one value at a time.


PER SCHOLAS

CSS Units: Lengths

There are many property-specific units for values used in CSS, but there are some general units that are used by a number of properties, and it is worth familiarizing yourself with these before continuing.

- **px** (such as `font-size: 12px`) is the unit for pixels.
- **em** (such as `font-size: 2em`) is the unit for the calculated size of a font. So “`2em`,” for example, is two times the current font size.
- **pt** (such as `font-size: 12pt`) is the unit for points, a measurement typically used in printed media.
- **%** (such as `width: 80%`) is the unit for percentage widths of the parent element.
- **vw** (such as `1vw`) is equal to 1% of the viewport width. Viewport denotes browser window size.
- **vh** (such as `1vh`) is equal to 1% of the viewport height.
- **vmin** (such as `1vmin`) is equal to 1% of smaller length of `vw` or `vh`.
- **vmax** (such as `1vmax`) is equal to 1% of larger length of `vw` or `vh`.

10

Different types of CSS units are available for expressing lengths. Following table explains different types of CSS units.

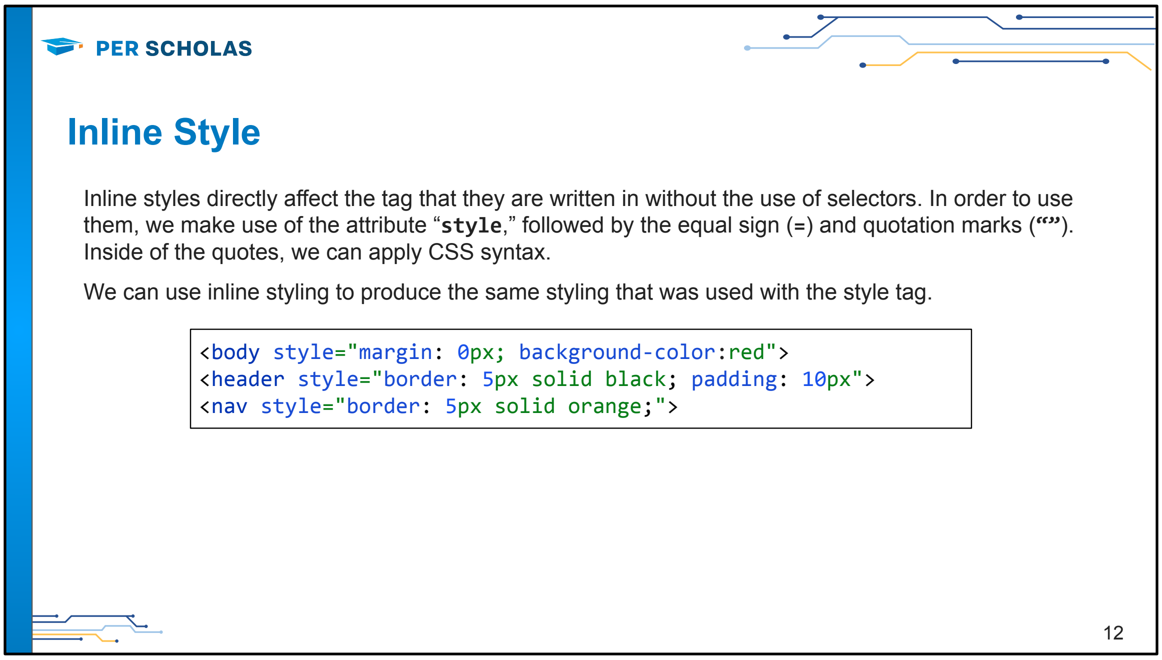
PER SCHOLAS

Different Ways to Apply CSS Styles

CSS can be added to HTML documents in three ways:

- **Inline -** by using the style attribute inside of HTML elements.
- **Internal** - by using a `<style>` element in the `<head>` section.
- **External** - by using a `<link>` element to link to an external CSS file.

The most common way to add CSS is to keep the styles in external CSS files.



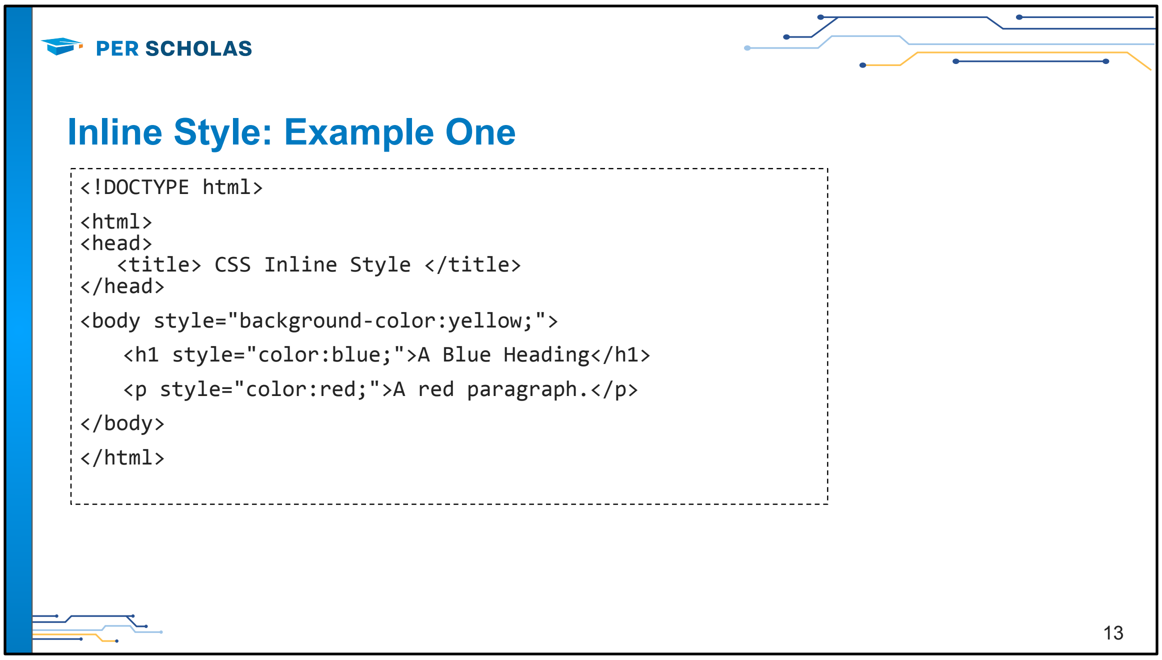
PER SCHOLAS

Inline Style

Inline styles directly affect the tag that they are written in without the use of selectors. In order to use them, we make use of the attribute “**`style`**,” followed by the equal sign (=) and quotation marks (**“”**). Inside of the quotes, we can apply CSS syntax.

We can use inline styling to produce the same styling that was used with the style tag.

```
<body style="margin: 0px; background-color:red">
<header style="border: 5px solid black; padding: 10px">
<nav style="border: 5px solid orange;">
```



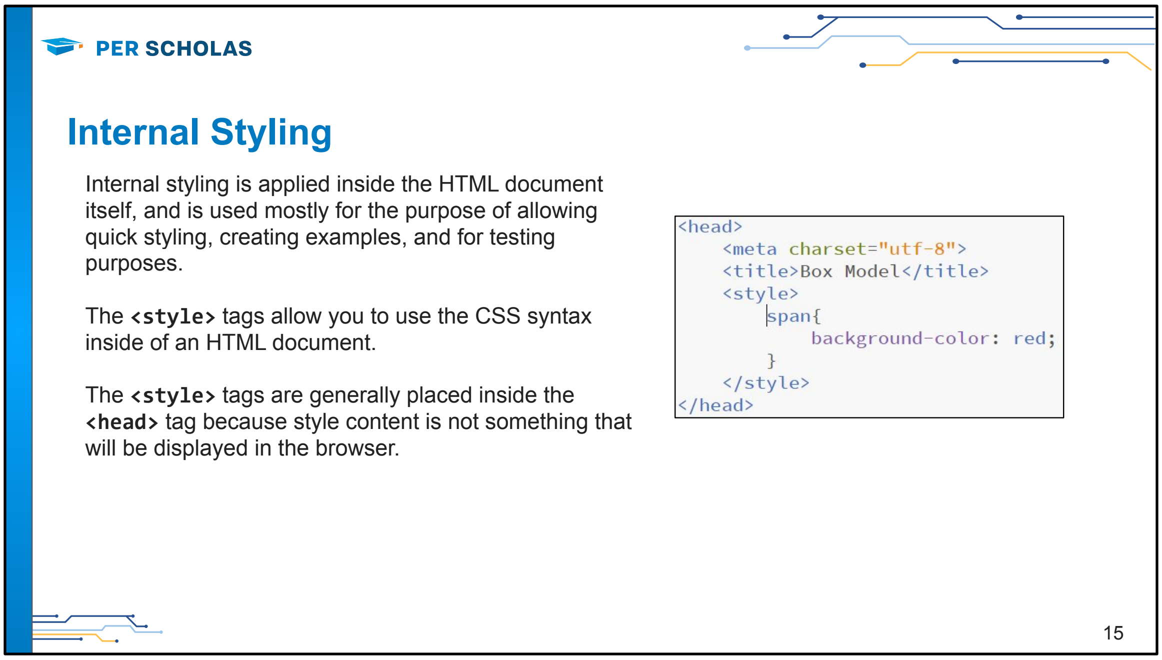
Inline Style: Example One

```
<!DOCTYPE html>
<html>
<head>
   <title> CSS Inline Style </title>
</head>
<body style="background-color:yellow;">
    <h1 style="color:blue;">A Blue Heading</h1>
    <p style="color:red;">A red paragraph.</p>
</body>
</html>
```

PER SCHOLAS

Inline Style: Example Two

```
<!DOCTYPE html>
<html>
<head>
   <title> CSS Inline Style </title>
</head>
<body style="margin: 1px; border:1px solid; background-color: red">
     <section>
         <h3> It is Header of the webpage</h3>     </section>
     <section style="border: 5px solid black; padding: 10px; background-color:#dbfffe">
     <h1 style="text-align:center;">CSS Styles</h1>
    <h2 style="background-color:green; color:white;">Different ways to apply CSS
styles</h2>
              <p style="font-family:arial; font-size:40px"> 1)    Inline style </p>
              <p style="font-family:arial; font-size:30px"> 2)    Internal style
sheet</p>
              <p style="font-family:arial; font-size:20px"> 3)    External style sheet
</p>
            </section>
</body>
</html>
```



PER SCHOLAS

Internal Styling

Internal styling is applied inside the HTML document itself, and is used mostly for the purpose of allowing quick styling, creating examples, and for testing purposes.

The **`<style>`** tags allow you to use the CSS syntax inside of an HTML document.

The **`<style>`** tags are generally placed inside the **`<head>`** tag because style content is not something that will be displayed in the browser.

```
<head>
    <meta charset="utf-8">
    <title>Box Model</title>
    <style>
        span{
            background-color: red;
        }
    </style>
</head>
```

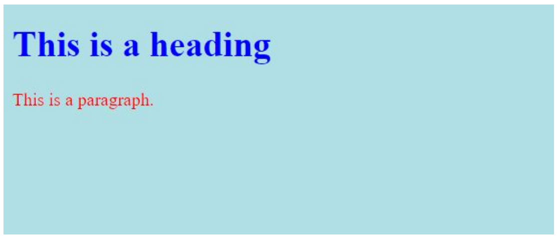
15

The style tags are generally placed inside the <head> tags because it is not something that will be displayed in the browser.

PER SCHOLAS

Internal Styling: Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
</style>
</head>
<body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
</body>
</html>
```



This is a heading

This is a paragraph.

PER SCHOLAS

External Style Sheets

External style sheets are created in separate documents with a **.css** extension. An external style sheet is simply a listing of CSS rules. It cannot contain HTML tags.

- The **`<link>`** tag, which goes in the **`<head>`** of and HTML page, specifies relationships between the current document and an external resource.
- The **`<link>`** element is most commonly used to link to stylesheets, but is also used to establish site icons (both "favicon" style icons and icons for the home screen and apps on mobile devices), among other things.
- When determining what styles to apply to elements, **external stylesheets have the lowest precedence, and inline styles have the highest precedence.** This means that a property written inline will override that same property in either internal styles or an external stylesheet.

To add a stylesheet to a file:

- In your styles folder, create a file name **index.css.**
- Go to your **index.html** file, and inside the **`<head>`** tag, add a **`<link>`** tag
- Use the **`path`** of your **`index.html`** file for the **`href`** attribute.
  - `<link rel="stylesheet" href="">`

PER SCHOLAS

Activity: External Style Sheets

Create html file and add the code below.

```html
<!DOCTYPE html>
<html>
    <head>
        <title>My Example</title>
        <link rel="stylesheet" href="style.css">
    </head>
    <body>
      <p id="intro">Allow you to define styles for the whole
website.</p>
      <p class="colorful">This has a style applied via a
class.</p>

        <section>
                 <h1>CSS Styles</h1>
                    <h2>Different ways to apply CSS styles</h2>
                    <p> 1)      Inline style </p>
                    <p> 2)      Internal style sheet</p>
                    <p> 3)      External style sheet </p>
       </section>
    </body>
</html>
```

Create a CSS file named style.css and add the code below.

```css
body {
    background-color: darkslategrey;
    color: black;
    font-size: 1.1em;
}
h1 {
    color: coral;
}
#intro {
    font-size: 1.3em;
}
.colorful {
    color: orange;
}

section { border: 5px solid black; padding: 10px;
background-color:#dbfffe; }
```

PER SCHOLAS

CSS Comments

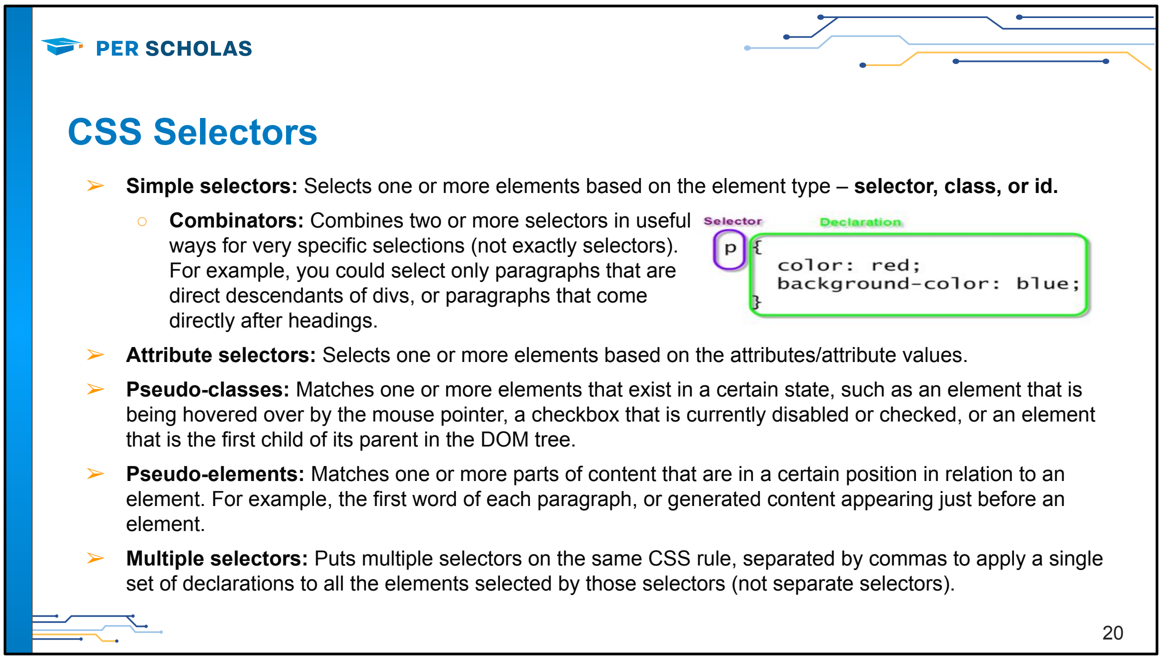
CSS comments are not displayed in the browser, but they can help document and organize your source code, as well as make it more readable.

A CSS comment is placed inside the `<style>` element, and starts with `/*` and ends with `*/`.

In CSS, for both single and multiples line comments, you only have to use `/*` at the beginning and `*/` at the end of the text you would like to make into a comment; you do not need to add comment indicators on every line.

Example

```
/* This is a single-line comment */
/* This is a
multi-line
comment */
p {
  color: red;
}
```


PER SCHOLAS

CSS Selectors

- **Simple selectors:** Selects one or more elements based on the element type – **selector, class, or id.**
  - **Combinators:** Combines two or more selectors in useful ways for very specific selections (not exactly selectors). For example, you could select only paragraphs that are direct descendants of divs, or paragraphs that come directly after headings.

Selector

Declaration

```
p {
    color: red;
    background-color: blue;
}
```

- **Attribute selectors:** Selects one or more elements based on the attributes/attribute values.
- **Pseudo-classes:** Matches one or more elements that exist in a certain state, such as an element that is being hovered over by the mouse pointer, a checkbox that is currently disabled or checked, or an element that is the first child of its parent in the DOM tree.
- **Pseudo-elements:** Matches one or more parts of content that are in a certain position in relation to an element. For example, the first word of each paragraph, or generated content appearing just before an element.
- **Multiple selectors:** Puts multiple selectors on the same CSS rule, separated by commas to apply a single set of declarations to all the elements selected by those selectors (not separate selectors).

20

In CSS, selectors are used to target a group or individual elements in a web page that we want to style. There are a wide variety of CSS selectors available, allowing for fine, grained precision when selecting elements to style.

PER SCHOLAS

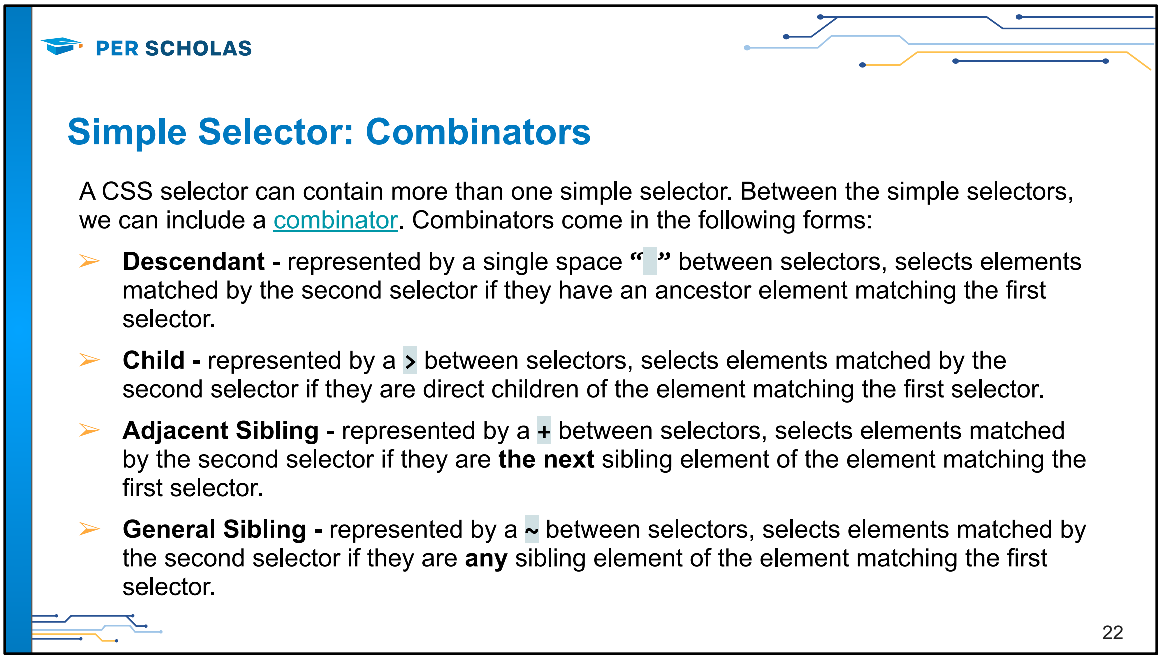
Element Type Selectors

A type selector is sometimes referred to as a tag name selector or element selector because it selects an HTML tag/element in your document.

The CSS type selector matches elements by node name. In other words, it selects all elements of the given type within a document.

```
<body>
   <h1>External Styles</h1>
   <p>Allow you to define styles for the whole
website.</p>
    <p>This has a style applied via a class.</p>
    <section>
              <h1>CSS Styles</h1>
                 <h2>Different ways to apply CSS
styles</h2>
                 <p> 1)   Inline style </p>
                 <p> 2)   Internal style sheet</p>
                 <p> 3)   External style sheet </p>
   </section>
    </body>
```

```
<style>

body{
   margin: 0;
}

h1 {
    color: red;
    font-size: 20px;
}

h2 {
    color: green;
}

p {
    color: yellow;
    background-color: gray;
}

</style>
```



PER SCHOLAS

Simple Selector: Combinators

A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator. Combinators come in the following forms:

- **Descendant -** represented by a single space **“ ”** between selectors, selects elements matched by the second selector if they have an ancestor element matching the first selector.
- **Child -** represented by a **>** between selectors, selects elements matched by the second selector if they are direct children of the element matching the first selector.
- **Adjacent Sibling -** represented by a **+** between selectors, selects elements matched by the second selector if they are **the next** sibling element of the element matching the first selector.
- **General Sibling -** represented by a **~** between selectors, selects elements matched by the second selector if they are **any** sibling element of the element matching the first selector.

22

PER SCHOLAS

Combinators: Descendant Example

Descendant Combinator ( )

The descendant combinator ( ) separates two selectors, and matches the second element only if it is a descendant of the first element, regardless of depth.

In the example, `<section>` is the parent element. The `<p>` elements are descendants of the `<section>` element. All `<p>` elements will be affected.

CSS Block

```
section p {
   color: grey;
}
```

Html Block

```
<section>
   <h2>Bootstrap</h2>
   <div>
      <p>Lorem ipsum dolor sit amet,
consectetur adipisicing elit.</p>
   </div>
   <p>Doloribus explicabo incidunt magnam
magni nobis.</p>
</section>
```

PER SCHOLAS

Combinators: Child Example

Child Combinator (>)

The child combinator (>) separates two selectors, and matches the second element only if it is a child of the first element.

In the example, `<section>` is the parent element. The `<p>` elements are children of the `<section>` element. Both `<p>` elements will be affected.

The **child** selector may seem similar to the **descendant** selector, but note that the **child** selector will only affect **direct children** of the parent element, while the **descendant** selector affects **all descendants** of the element, regardless of nesting.

CSS Block

```
section > p {
   color: grey;
}
```

Html Block

```
<section>
   <h2>Bootstrap</h2>
   <p>Lorem ipsum dolor sit amet,
consectetur adipisicing elit.</p>
   <p>Doloribus explicabo incidunt magnam
magni nobis.</p>
</section>
```

PER SCHOLAS

Combinators: Adjacent Sibling Example

Adjacent Sibling Combinator (+)

The adjacent sibling combinator (+) separates two selectors, and matches the second element only if it immediately follows the first element and both are children of the same parent element.

In the example, `<section>` is the parent element. `<h2>` and `<p>` elements are children of the same parent element. `<h2>` immediately precedes the first `<p>` element, but not the second, so only the first `<p>` element will be affected.

CSS Block

```
h2 + p {
   color: grey;
}
```

Html Block

```
<section>
   <h2>Bootstrap</h2>
   <p>Lorem ipsum dolor sit amet,
consectetur adipisicing elit.</p>
   <div>Separator div does affect adjacent
siblings.</div>
   <p>Doloribus explicabo incidunt magnam
magni nobis.</p>
</section>
```

Combinators: General Sibling Example

General Sibling Combinator (~)

The general sibling combinator **(~)** separates two selectors and matches the second element only if it follows the first element (though not necessarily immediately), and both are children of the same parent element.

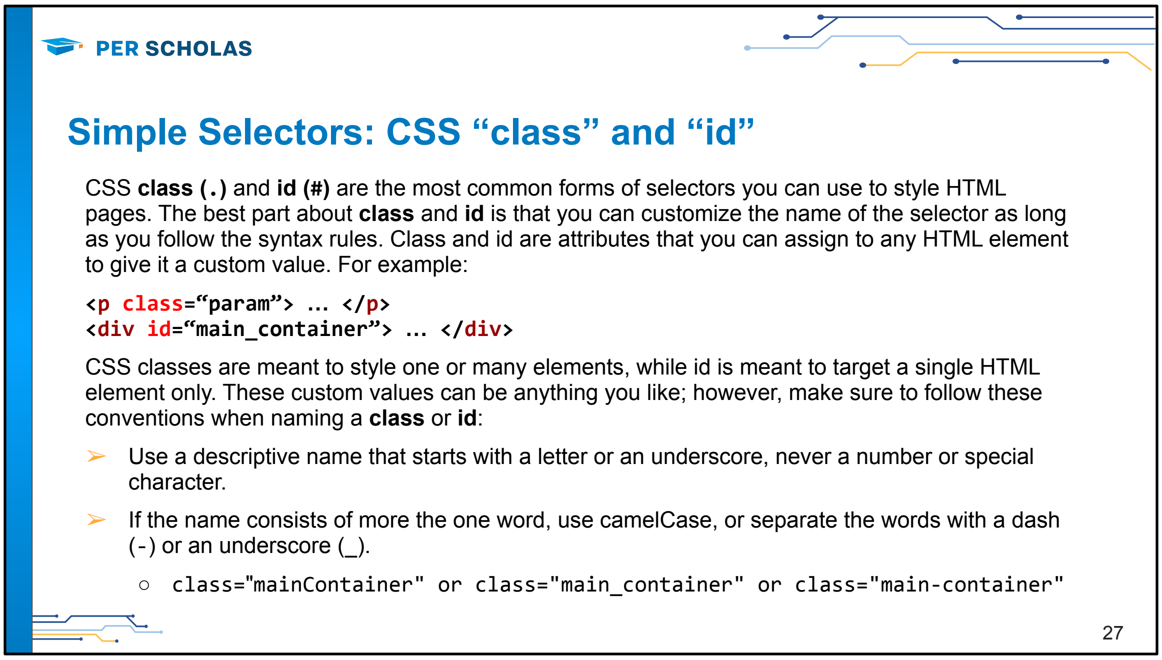
In the example, `<section>` is the parent element. `<h2>` and `<p>` elements are children of the same parent element. `<h2>` precedes both of the `<p>` elements, so they will both be affected.

CSS Block

```css
h2 ~ p {
   color: grey;
}
```

Html Block

```html
<section>
   <h2>Bootstrap</h2>
   <p>Lorem ipsum dolor sit amet,
consectetur adipisicing elit.</p>
   <div>Separator div does not affect
general siblings.</div>
   <p>Doloribus explicabo incidunt magnam
magni nobis.</p>
</section>
```



PER SCHOLAS

Simple Selectors: CSS “class” and “id”

CSS **class (.)** and **id (#)** are the most common forms of selectors you can use to style HTML pages. The best part about **class** and **id** is that you can customize the name of the selector as long as you follow the syntax rules. Class and id are attributes that you can assign to any HTML element to give it a custom value. For example:

```
<p class=“param”> … </p>
<div id=“main_container”> … </div>
```

CSS classes are meant to style one or many elements, while id is meant to target a single HTML element only. These custom values can be anything you like; however, make sure to follow these conventions when naming a **class** or **id**:

- ➢ Use a descriptive name that starts with a letter or an underscore, never a number or special character.
- ➢ If the name consists of more the one word, use camelCase, or separate the words with a dash (-) or an underscore (_).
    - ○ `class="mainContainer" or class="main_container" or class="main-container"`

27

Selectors: Class Example

Class Selector (`.`)

The class selector (**`.`**) matches elements based on their class attributes. The selector will choose all elements with the associated class, regardless of location within the DOM.

In the example, the `.red-text` class has been assigned to both a `<h2>` element and one of the `<p>` elements, and will affect both of those elements.

CSS Block

```css
.red-text {
   color: red;
}
```

Html Block

```html
<section>
   <h2 class="red-text">Bootstrap</h2>
   <p>Lorem ipsum dolor sit amet,
consectetur adipisicing elit.</p>
   <div>Separator div does not affect
general siblings.</div>
   <p class="red-text">Doloribus explicabo
incidunt magnam magni nobis.</p>
</section>
```

PER SCHOLAS

Selectors: id Example

id Selector (#)

The id selector **(#)** matches elements based on their id attribute. The selector will choose an element with the associated id, regardless of location within the DOM.

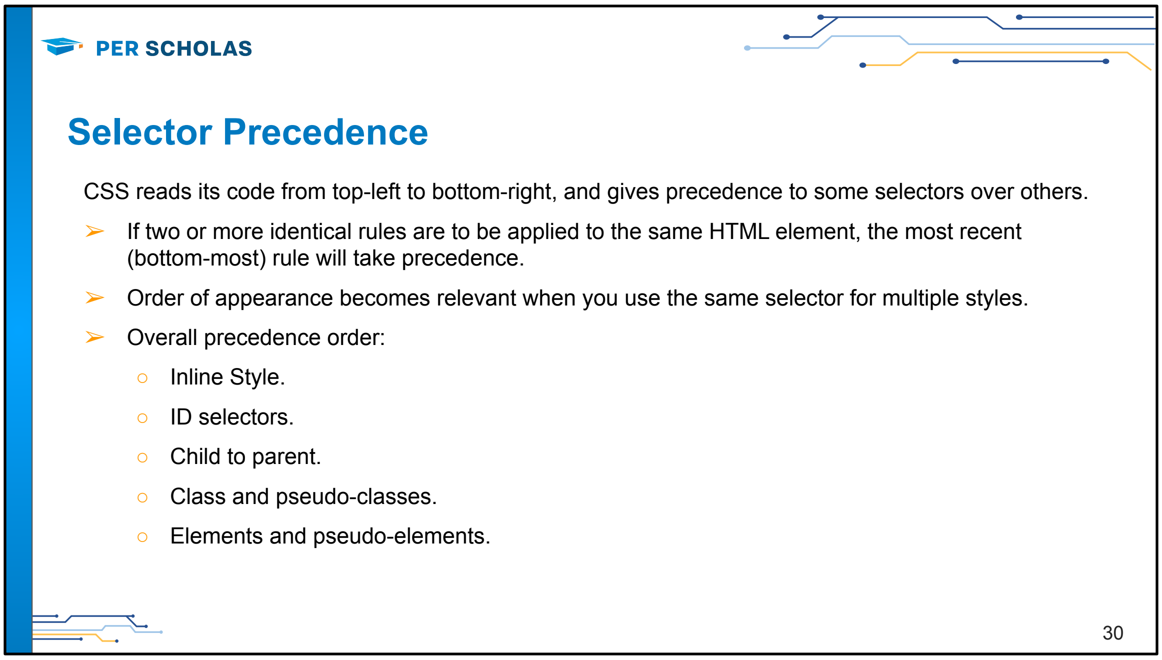
In the example, the `#mySection` id has been assigned to a `<section>` element, and will affect that entire `<section>`.

CSS Block

```
#mySection {
   font-weight: bold;
}
```

HTML Block

```
<section id="mySection">
   <h2 class="red-text">Bootstrap</h2>
   <p>Lorem ipsum dolor sit amet,
consectetur adipisicing elit.</p>
   <div>Separator div does not affect
general siblings.</div>
   <p class="red-text">Doloribus explicabo
incidunt magnam magni nobis.</p>
</section>
```



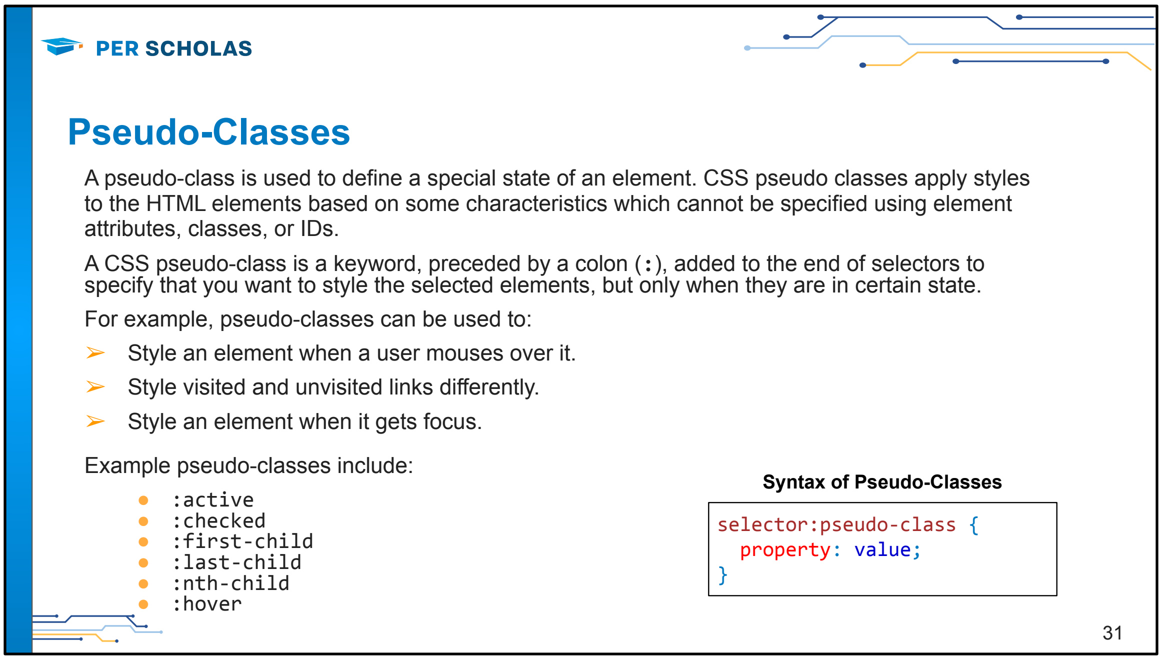
PER SCHOLAS

Selector Precedence

CSS reads its code from top-left to bottom-right, and gives precedence to some selectors over others.

- If two or more identical rules are to be applied to the same HTML element, the most recent (bottom-most) rule will take precedence.
- Order of appearance becomes relevant when you use the same selector for multiple styles.
- Overall precedence order:
  - Inline Style.
  - ID selectors.
  - Child to parent.
  - Class and pseudo-classes.
  - Elements and pseudo-elements.

30



PER SCHOLAS

Pseudo-Classes

A pseudo-class is used to define a special state of an element. CSS pseudo classes apply styles to the HTML elements based on some characteristics which cannot be specified using element attributes, classes, or IDs.

A CSS pseudo-class is a keyword, preceded by a colon (`:`), added to the end of selectors to specify that you want to style the selected elements, but only when they are in certain state.

For example, pseudo-classes can be used to:

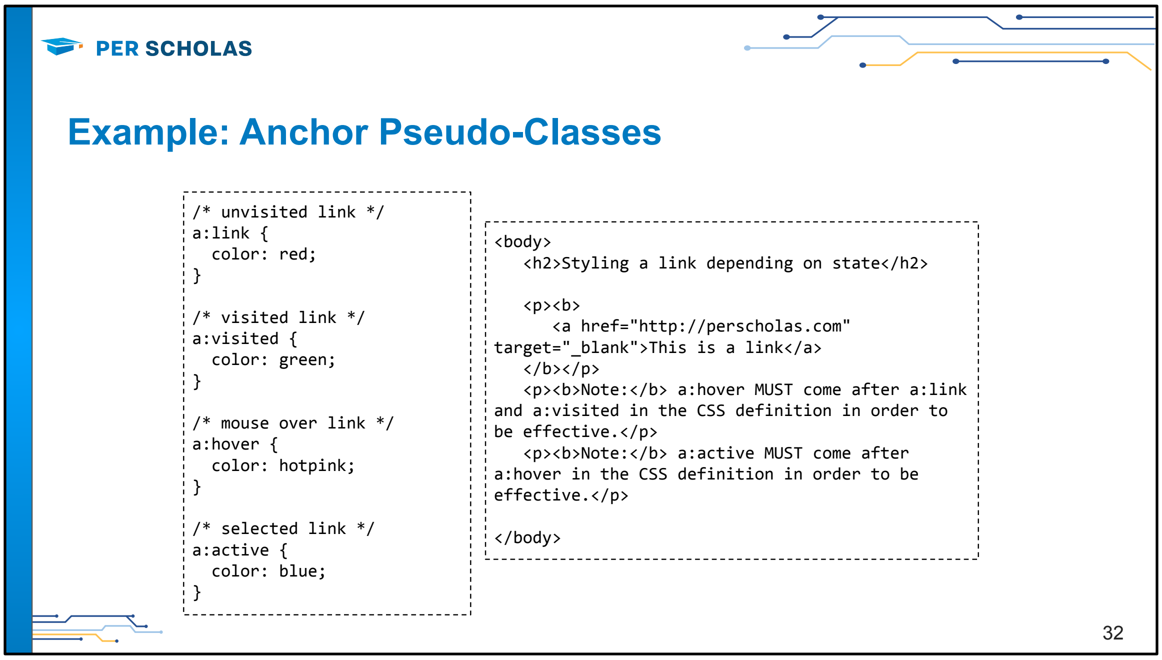
- ➢ Style an element when a user mouses over it.
- ➢ Style visited and unvisited links differently.
- ➢ Style an element when it gets focus.

Example pseudo-classes include:

- `:active`
- `:checked`
- `:first-child`
- `:last-child`
- `:nth-child`
- `:hover`

Syntax of Pseudo-Classes

```
selector:pseudo-class {
  property: value;
}
```



PER SCHOLAS

Example: Anchor Pseudo-Classes

```
/* unvisited link */
a:link {
  color: red;
}

/* visited link */
a:visited {
  color: green;
}

/* mouse over link */
a:hover {
  color: hotpink;
}

/* selected link */
a:active {
  color: blue;
}
```

```
<body>
   <h2>Styling a link depending on state</h2>

   <p><b>
      <a href="http://perscholas.com"
target="_blank">This is a link</a>
   </b></p>
   <p><b>Note:</b> a:hover MUST come after a:link
and a:visited in the CSS definition in order to
be effective.</p>
   <p><b>Note:</b> a:active MUST come after
a:hover in the CSS definition in order to be
effective.</p>

</body>
```

32

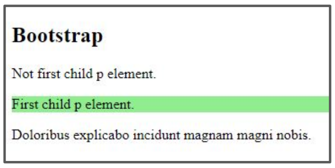
PER SCHOLAS

Pseudo-Classes: :first-child Example

`:first-child`

The **`:first-child`** pseudo-class matches elements when they are the first child of their parent element.

In the example, `p:first-child` matches every `<p>` element that is the first child of its parent, and no other `<p>` elements.



Bootstrap

Not first child p element.

First child p element.

Doloribus explicabo incidunt magnam magni nobis.

CSS Block

```
p:first-child {
   background: lightgreen;
}
```

Html Block

```
<section>
   <h2>Bootstrap</h2>
   <p>Not first child p element.</p>
   <div>
      <p>First child p element.</p>
   </div>
   <p>Doloribus explicabo incidunt magnam
magni nobis.</p>
</section>
```

33

PER SCHOLAS

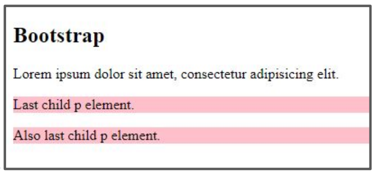
Pseudo-Classes: :last-child Example

`:last-child`

The **`:last-child`** pseudo-class matches elements when they are the last child of their parent element.

In the example, `p:last-child` matches every `<p>` element that is the last child of its parent, and no other `<p>` elements.

Result


Bootstrap

Lorem ipsum dolor sit amet, consectetur adipisicing elit.

Last child p element.

Also last child p element.

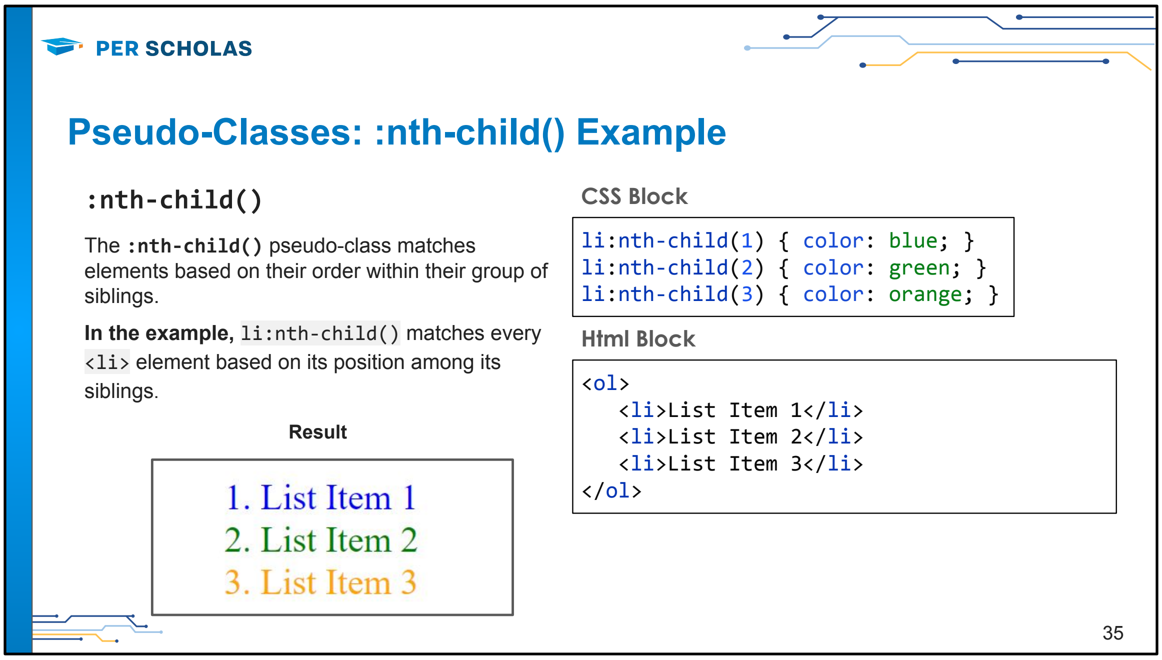
CSS Block

```
p:last-child {
   background: pink;
}
```

Html Block

```
<section>
   <h2>Bootstrap</h2>
   <p>Lorem ipsum dolor sit amet,
consectetur adipisicing elit.</p>
   <div>
      <p>Last child p element.</p>
   </div>
   <p>Also last child p element.</p>
</section>
```

34



PER SCHOLAS

Pseudo-Classes: :nth-child() Example

`:nth-child()`

The **`:nth-child()`** pseudo-class matches elements based on their order within their group of siblings.

In the example, `li:nth-child()` matches every `<li>` element based on its position among its siblings.

Result

1. List Item 1
2. List Item 2
3. List Item 3

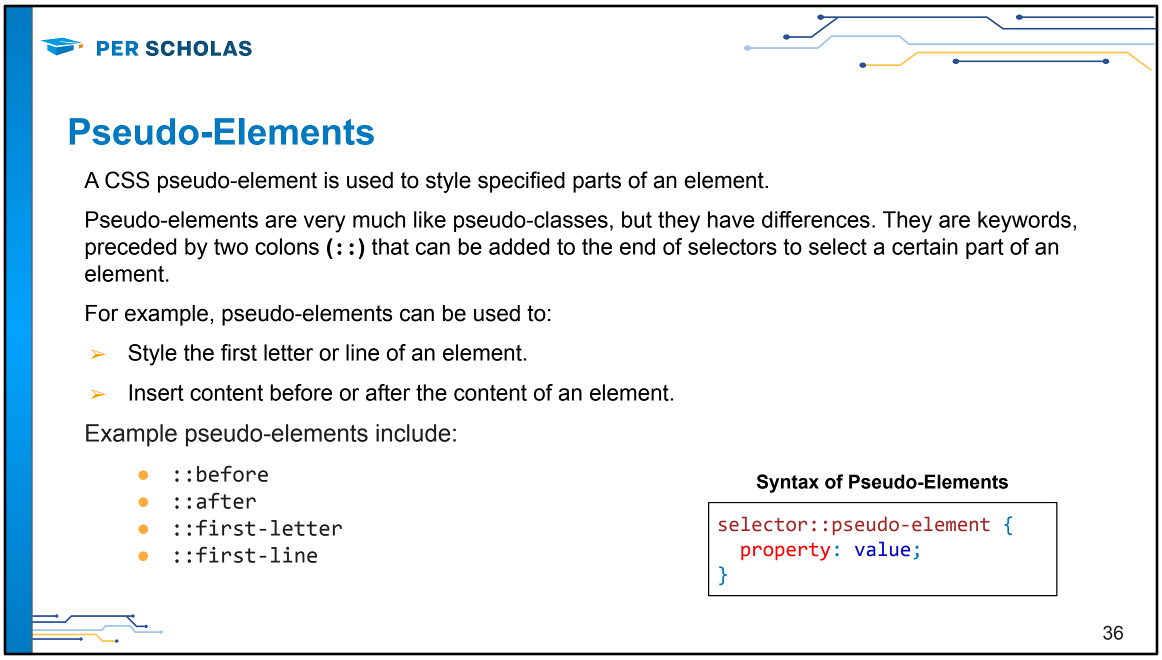
CSS Block

```css
li:nth-child(1) { color: blue; }
li:nth-child(2) { color: green; }
li:nth-child(3) { color: orange; }
```

Html Block

```html
<ol>
   <li>List Item 1</li>
   <li>List Item 2</li>
   <li>List Item 3</li>
</ol>
```

35


PER SCHOLAS

Pseudo-Elements

A CSS pseudo-element is used to style specified parts of an element.

Pseudo-elements are very much like pseudo-classes, but they have differences. They are keywords, preceded by two colons **(::)** that can be added to the end of selectors to select a certain part of an element.

For example, pseudo-elements can be used to:

- ➢ Style the first letter or line of an element.
- ➢ Insert content before or after the content of an element.

Example pseudo-elements include:

- `::before`
- `::after`
- `::first-letter`
- `::first-line`

Syntax of Pseudo-Elements

```
selector::pseudo-element {
  property: value;
}
```

36

PER SCHOLAS

Pseudo-Elements: Examples

The following table shows different pseudo-elements and their details.

Pseudo-Element	Description	Example
`::before`	Inserts content beginning of the targeted element content.	`p::before {` `color: red;` `}`
`::after`	Inserts content end of the targeted element content.	`div::after {` `background-color: yellow;` `}`
`::first-line`	Selects first line of the targeted block level element	`div::first-line {` `color: blue;` `}`
`::first-letter`	Selects first letter of the targeted block level element.	`div::first-letter {` `color: red;` `}`

PER SCHOLAS

HTML Tables

In HTML, you create tables using the `<table>` element in conjunction with the `<tr>` (table row), `<th>` (table header), and `<td>` (table data) elements.

Each `<tr>` element represents a row within the table that it is nested in, and each `<td>` element represents a table data cell within the row that it is nested in. You can also add table headers using the `<th>` element.

```
<table>
   <tr>
       <th>Table Header</th>
   </tr>
   <tr>
       <td>Table cell</td>
       <td>Table cell</td>
       <td>Table cell</td>
   </tr>
   <tr>
       <td>Table cell</td>
       <td>Table cell</td>
       <td>Table cell</td>
   </tr>
</table>
```

Click here for a reference on HTML Tables.

PER SCHOLAS

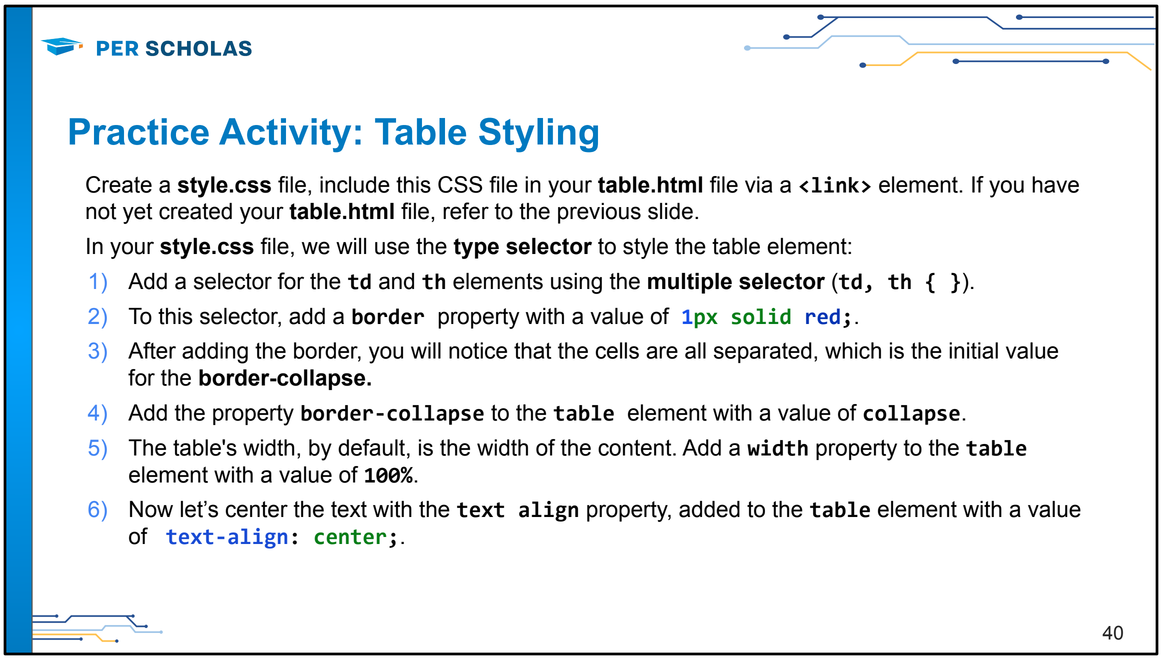
HTML Table Sections

Tables can be organized into sections for further clarity.

The **`<thead>`**, **`<tbody>`**, and **`<tfoot>`** elements represent the different sections of a table. The **`<caption>`** element specifies the caption (or title) of a table.

Copy the HTML to the right, into a file named **table.html**.
We will use this file as part of a practice activity beginning on the next slide.

```
<table>
  <caption>Practice Table</caption>
   <thead>
      <tr>
         <th>Header1</th>
         <th>Header2</th>
         <th>Header3</th>
      </tr>
   </thead>
   <tbody>
      <tr>
         <td>col 1 cell 1</td>
         <td>col 2 cell 2</td>
         <td>col 3 cell 3</td>
      </tr>
      <tr>
         <td>col 1 cell 1</td>
         <td>col 2 cell 2</td>
         <td>col 3 cell 3</td>
      </tr>
   </tbody>
   <tfoot>
      <tr>
         <td>footer1</td>
         <td>footer2</td>
         <td>footer3</td>
      </tr>
   </tfoot>
</table>
```



PER SCHOLAS

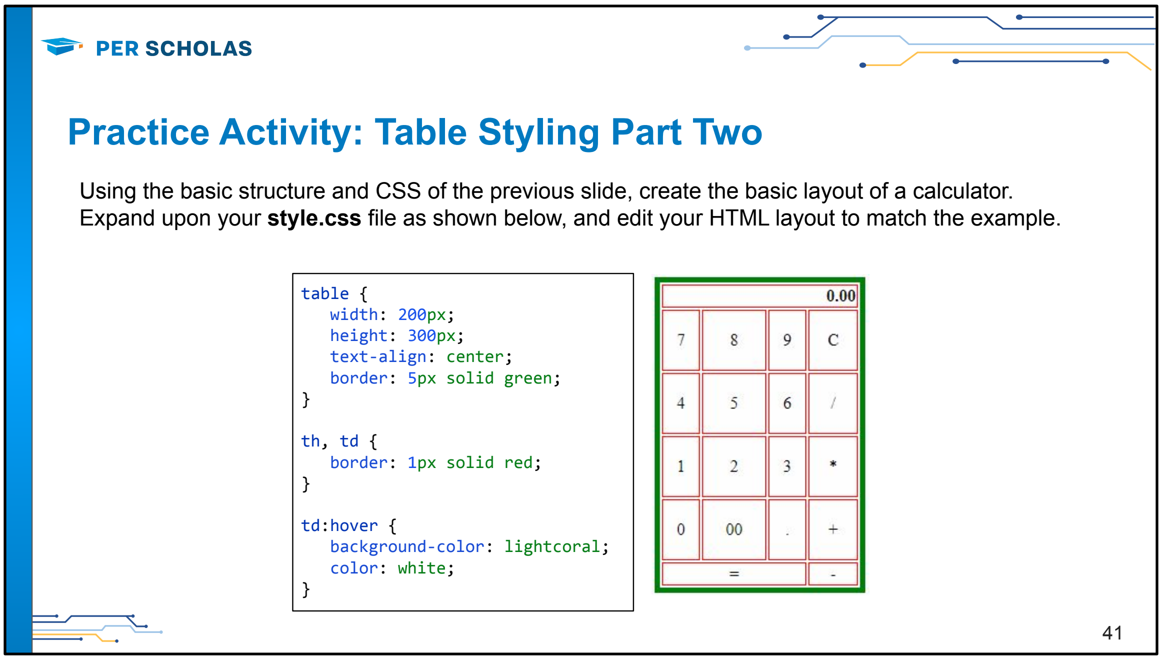
Practice Activity: Table Styling

Create a **style.css** file, include this CSS file in your **table.html** file via a **`<link>`** element. If you have not yet created your **table.html** file, refer to the previous slide.

In your **style.css** file, we will use the **type selector** to style the table element:

1) Add a selector for the **`td`** and **`th`** elements using the **multiple selector** (**`td, th { }`**).
2) To this selector, add a **`border`** property with a value of **`1px solid red;`**.
3) After adding the border, you will notice that the cells are all separated, which is the initial value for the **border-collapse.**
4) Add the property **`border-collapse`** to the **`table`** element with a value of **`collapse`**.
5) The table's width, by default, is the width of the content. Add a **`width`** property to the **`table`** element with a value of **`100%`**.
6) Now let's center the text with the **`text align`** property, added to the **`table`** element with a value of **`text-align: center;`**.

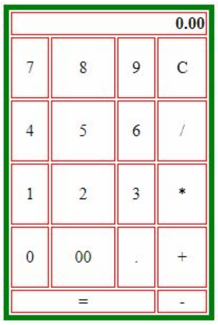
40


PER SCHOLAS

Practice Activity: Table Styling Part Two

Using the basic structure and CSS of the previous slide, create the basic layout of a calculator. Expand upon your **style.css** file as shown below, and edit your HTML layout to match the example.

```
table {
   width: 200px;
   height: 300px;
   text-align: center;
   border: 5px solid green;
}

th, td {
   border: 1px solid red;
}

td:hover {
   background-color: lightcoral;
   color: white;
}
```



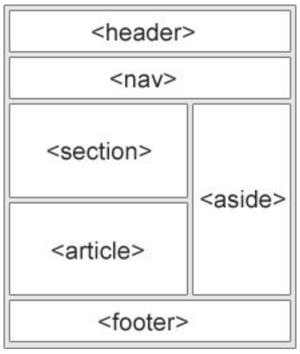
			0.00
7	8	9	C
4	5	6	/
1	2	3	*
0	00	.	+
=			-

41

PER SCHOLAS

HTML Layout Elements

You have seen most websites on the internet usually display their content in multiple rows and columns, formatted like a magazine or newspaper to provide the users a better experience. This can be easily achieved by using HTML tags such as `<table>`, `<div>`, `<header>`, `<footer>`, `<section>`, etc., and adding some CSS styles to them.



- `<header>` - Defines a header for a document or a section.
- `<nav>` - Defines a set of navigation links.
- `<section>` - Defines a section in a document.
- `<article>` - Defines an independent, self-contained content.
- `<aside>` - Defines content aside from the content (like a sidebar).
- `<footer>` - Defines a footer for a document or a section.
- `<details>` - Defines additional details that the user can open and close on demand.
- `<summary>` - Defines a heading for the `<details>` element.

PER SCHOLAS

HTML Content Division Element

`<div>`

The `<div>` HTML element is a generic content container that divides content into workable sections. The `<div>` tag is the most usable tag in web development because it helps to separate data in the web page and create particular sections for particular data or functions.

Inside a `<div>` tag, we can put more than one HTML element, group them together, and apply CSS to them. The `<div>` tag is a block-level tag; every `<div>` tag will start from a new line.

```
<html>
    <body>
        <div>Lorem ipsum dolor sit amet, consectetur
adipisicing elit.</div>
        <div>Doloribus explicabo incidunt magnam magni nobis
pariatur quia, rerum sint tempora vero.</div>
    </body>
</html>
```

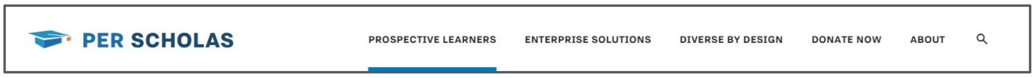
Lorem ipsum dolor sit amet, consectetur adipisicing elit.
Doloribus explicabo incidunt magnam magni nobis pariatur quia, rerum int tempora vero.

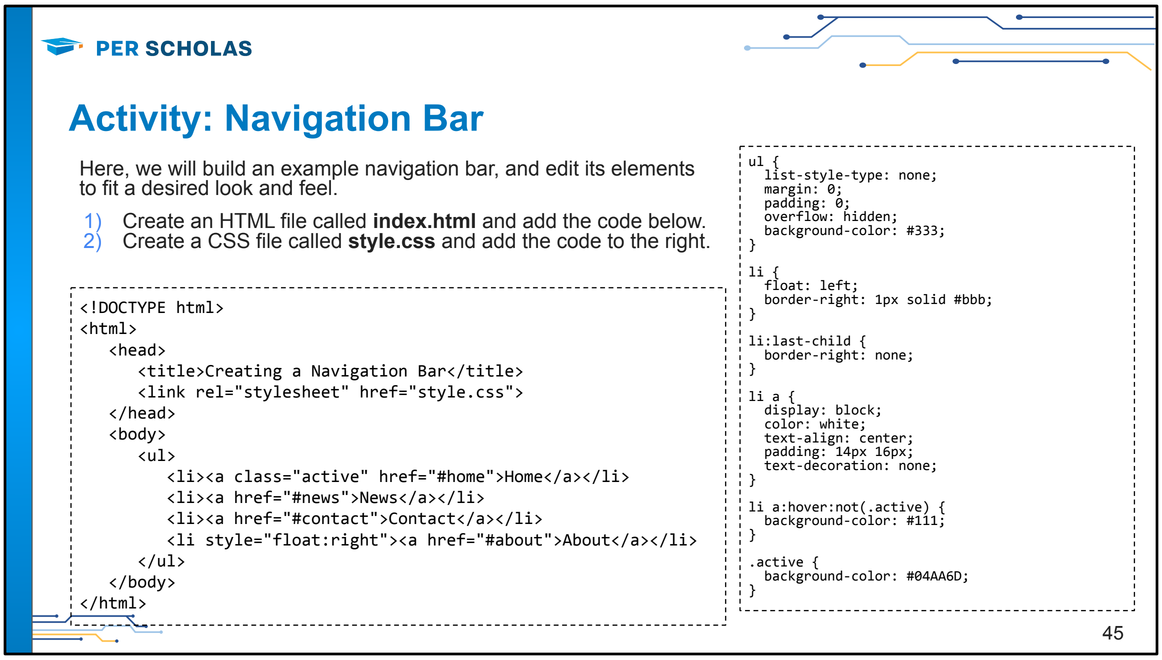
Result

PER SCHOLAS

Navigation Bar

- A navigation bar consists of a number of links that are called upon based on user actions.
- A navigation bar helps readers in selecting topics, sub-topics, or links of their interest. Using a navigation bar, users need not enter the URL of the specific webpage, as this is automatically taken care of by the navigation bar. The navigation sections have the necessary links of the webpage embedded within them.
- For Web developers, creating a navigation bar helps in separating content from structure.





PER SCHOLAS

Activity: Navigation Bar

Here, we will build an example navigation bar, and edit its elements to fit a desired look and feel.

1) Create an HTML file called **index.html** and add the code below.
2) Create a CSS file called **style.css** and add the code to the right.

```html
<!DOCTYPE html>
<html>
   <head>
      <title>Creating a Navigation Bar</title>
      <link rel="stylesheet" href="style.css">
   </head>
   <body>
      <ul>
         <li><a class="active" href="#home">Home</a></li>
         <li><a href="#news">News</a></li>
         <li><a href="#contact">Contact</a></li>
         <li style="float:right"><a href="#about">About</a></li>
      </ul>
   </body>
</html>
```

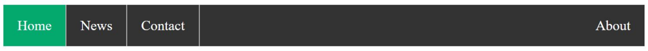
```css
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333;
}

li {
  float: left;
  border-right: 1px solid #bbb;
}

li:last-child {
  border-right: none;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

li a:hover:not(.active) {
  background-color: #111;
}

.active {
  background-color: #04AA6D;
}
```

45

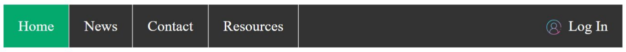
PER SCHOLAS

Activity: Navigation Bar (continued)

3) Navigate to your **index.html** file in your web browser and view the contents. You can quickly achieve this by right-clicking on the file in your folder and selecting "Open With" and your browser of choice. You should see the following:



4) Change the sections of the navigation bar to something that suits you. You can add new sections, rename existing sections, and change the layout or styling components.
5) Find some images to accompany your sections, and add them to the navigation bar via <img> elements. An example has been provided to the right, including CSS styling. This example produces the following result when added to our navigation bar:



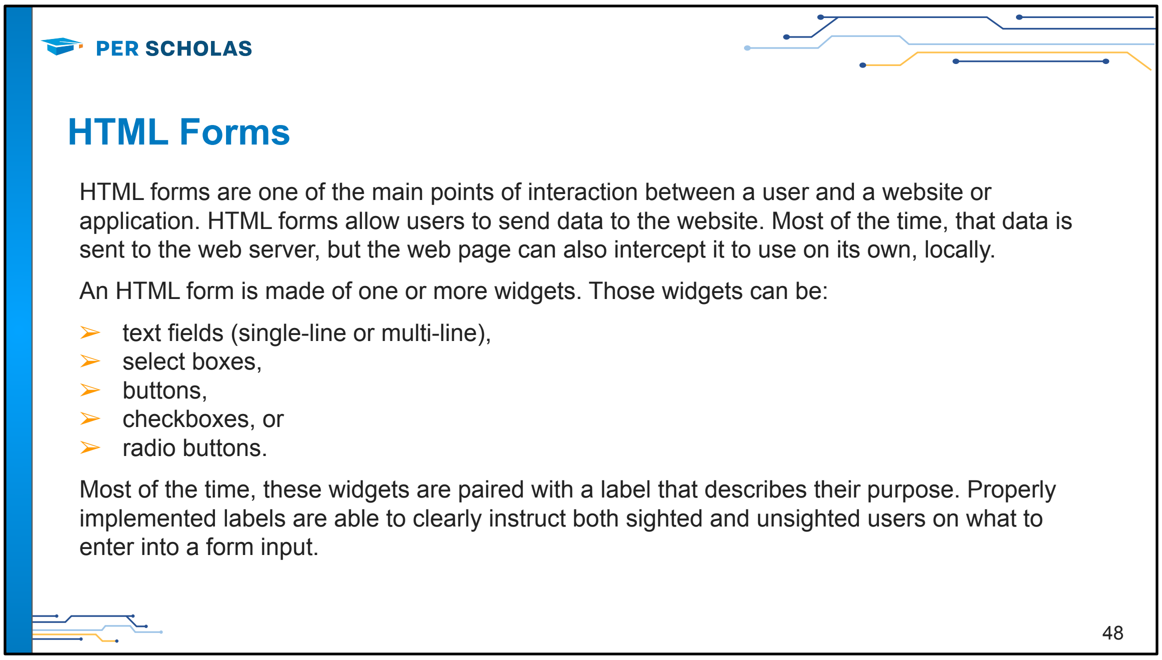
```
.login-user {
   height: 1em;
   margin-right: 0.5em;
   vertical-align: bottom;
}
```

```
<ul>
   <li><a class="active"
href="#home">Home</a></li>
   <li><a href="#news">News</a></li>
   <li><a href="#contact">Contact</a></li>
   <li><a href="#resources">Resources</a></li>
   <li style="float:right"><a href="#login">
      <img src="login-user.png" alt="Login
Icon" class="login-user"/>Log In
   </a></li>
</ul>
```

PER SCHOLAS

Navigation Bar Activity Code Break-Down

- `display`: The **`display`** CSS property sets whether an element is treated as a **`block`** or an **`inline`** element, and the layout used for its children, such as **`flow`** layout, **`grid`**,or **`flex`**. We will cover this property with more detail in later slides.
- `list-style-type`: The **`list-style-type`** CSS property sets the marker (such as a **`disc`**, **`character`**, or **`custom`** counter style) of a list item element.
- `text-decoration`: This shorthand CSS property sets the appearance of decorative lines on text. It is a shorthand for **`text-decoration-line`**; **`text-decoration-color`**; **`text-decoration-style`**; and the newer, **`text-decoration-thickness`** property.
- `margin-right`: The **`margin-right`** CSS property sets the margin area on the right side of an element. A positive value places it farther from its neighbors, while a negative value places it closer to its neighbors.
- `background-color`: The **`background-color`** CSS property sets the background color of an element.



PER SCHOLAS

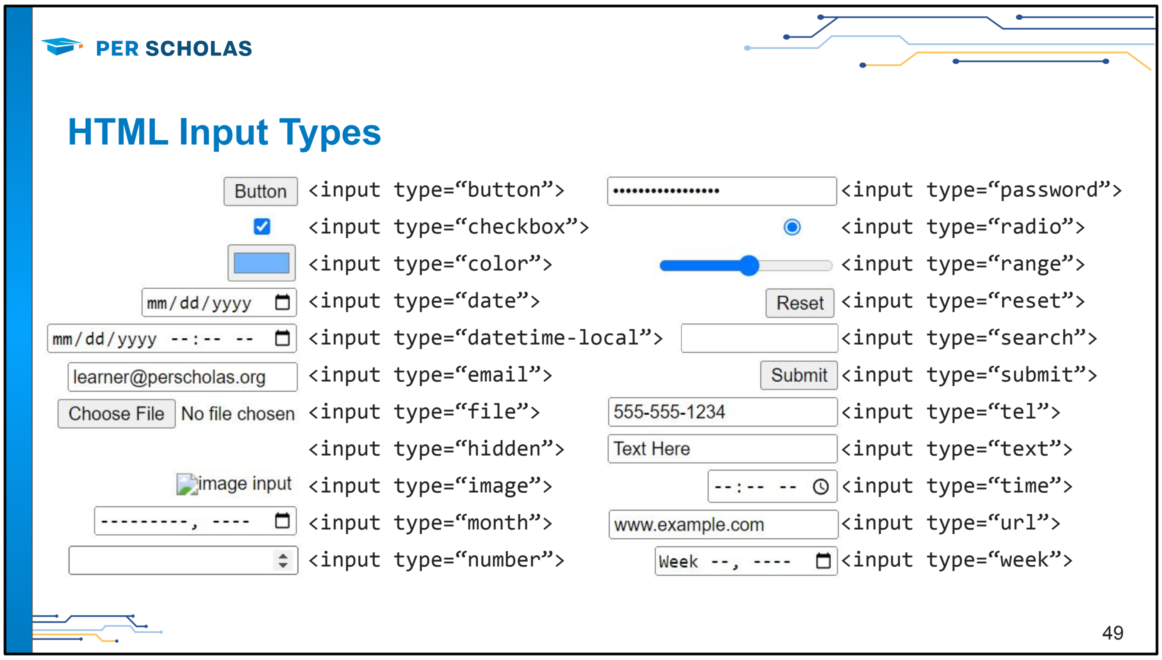
HTML Forms

HTML forms are one of the main points of interaction between a user and a website or application. HTML forms allow users to send data to the website. Most of the time, that data is sent to the web server, but the web page can also intercept it to use on its own, locally.

An HTML form is made of one or more widgets. Those widgets can be:

- text fields (single-line or multi-line),
- select boxes,
- buttons,
- checkboxes, or
- radio buttons.

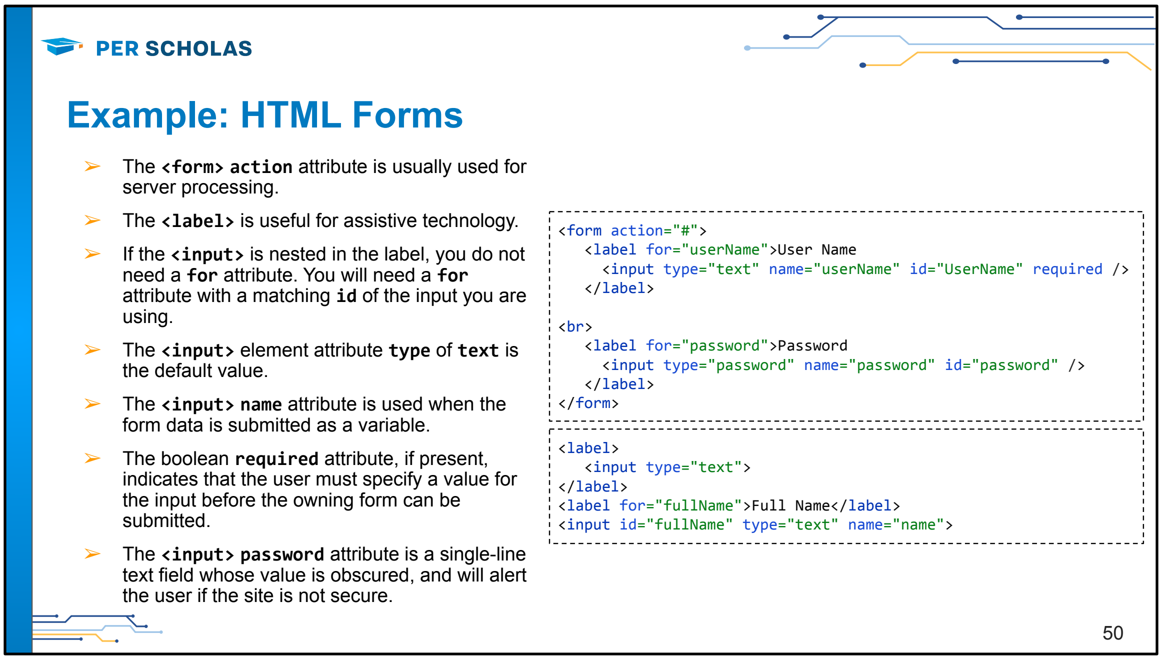
Most of the time, these widgets are paired with a label that describes their purpose. Properly implemented labels are able to clearly instruct both sighted and unsighted users on what to enter into a form input.


PER SCHOLAS

HTML Input Types

Button	`<input type="button">`	••••••••••••••••••	`<input type="password">`
✓	`<input type="checkbox">`	●	`<input type="radio">`
🟦	`<input type="color">`		`<input type="range">`
mm/dd/yyyy	`<input type="date">`	Reset	`<input type="reset">`
mm/dd/yyyy --:-- --	`<input type="datetime-local">`		`<input type="search">`
learner@perscholas.org	`<input type="email">`	Submit	`<input type="submit">`
Choose File No file chosen	`<input type="file">`	555-555-1234	`<input type="tel">`
	`<input type="hidden">`	Text Here	`<input type="text">`
image input	`<input type="image">`	--:-- --	`<input type="time">`
---------, ----	`<input type="month">`	www.example.com	`<input type="url">`
	`<input type="number">`	Week --, ----	`<input type="week">`

49



PER SCHOLAS

Example: HTML Forms

- The **`<form>` `action`** attribute is usually used for server processing.
- The **`<label>`** is useful for assistive technology.
- If the **`<input>`** is nested in the label, you do not need a **`for`** attribute. You will need a **`for`** attribute with a matching **`id`** of the input you are using.
- The **`<input>`** element attribute **`type`** of **`text`** is the default value.
- The **`<input>` `name`** attribute is used when the form data is submitted as a variable.
- The boolean **`required`** attribute, if present, indicates that the user must specify a value for the input before the owning form can be submitted.
- The **`<input>` `password`** attribute is a single-line text field whose value is obscured, and will alert the user if the site is not secure.

```
<form action="#">
   <label for="userName">User Name
     <input type="text" name="userName" id="UserName" required />
   </label>

<br>
   <label for="password">Password
     <input type="password" name="password" id="password" />
   </label>
</form>
```

```
<label>
   <input type="text">
</label>
<label for="fullName">Full Name</label>
<input id="fullName" type="text" name="name">
```

PER SCHOLAS

Example: HTML Forms (continued)

- The **`<fieldset>`** HTML element is used to group several controls, as well as labels (**`<label>`**) within a web form.
- The **`<legend>`** HTML element represents a caption for the content of its parent.
- The **`<input> checkbox`** attribute is rendered by default as boxes that are checked (ticked) when activated.
- The **`<input> radio`** attribute creates radio buttons. Radio buttons are typically rendered as small circles, which are filled or highlighted when selected. Only one radio button in a given group can be selected at the same time.

```
<fieldset>
   <legend>Checkboxes</legend>
   <label for="checkbox">Checkbox1
      <input type="checkbox" name="checkbox" id="checkbox" />
   </label>
   <label for="checkbox2">Checkbox2
      <input type="checkbox" name="checkbox2" id="checkbox2" />
   </label>
   <label for="checkbox3">Checkbox3
      <input type="checkbox" name="checkbox3" id="checkbox3" />
   </label>
</fieldset>
```

```
<fieldset>
   <legend>Radio Buttons</legend>
   <label>Radio1
      <input type="radio" name="radio" id="radio1" />
   </label><br>
   <label>Radio2
      <input type="radio" name="radio" id="radio2" />
   </label><br>
   <label>Radio3
      <input type="radio" name="radio" id="radio3" />
   </label>
</fieldset>
```

PER SCHOLAS

Example: Complete HTML Form (continued)

```
<form action="#">
   <label for="UserName">User Name</label>
   <input type="text" name="userName" id="UserName" required><br>
   <label for="password">Password</label>
   <input type="password" name="password" id="password"><br><br>
   <fieldset>
       <legend>Checkboxes</legend>
       <label for="checkbox">Checkbox1</label><input type="checkbox" name="checkbox" id="checkbox">
       <label for="checkbox2">Checkbox2</label><input type="checkbox" name="checkbox2" id="checkbox2">
       <label for="checkbox3">Checkbox3</label><input type="checkbox" name="checkbox3" id="checkbox3">
   </fieldset><br>
   <fieldset>
       <legend>Radio Buttons</legend>
       <label for="radio1">radio1</label>
       <input type="radio" name="radio" id="radio1">
       <label for="radio2">radio2</label>
       <input type="radio" name="radio" id="radio2">
       <label for="radio3">radio3</label>
       <input type="radio" name="radio" id="radio3">
   </fieldset>
</form>
```

For a reference on attributes for input fields, click here.

PER SCHOLAS

Dropdown Menus in HTML Forms

A dropdown menu allows users to select from a list of predefined values when inputting data into a form.

- The **`<select>`** element defines a drop-down list.
- The **`<option>`** elements defines an option that can be selected.
- By default, the first item in the drop-down list is selected.
- To define a pre-selected option other than the first item in the list, add the **`selected`** attribute to the option.

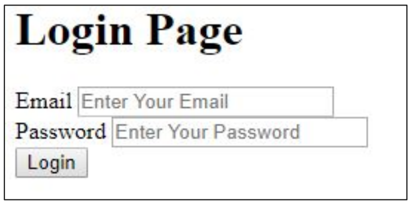
```
<!DOCTYPE html>
<html>
<body>

<h2>The select Element</h2>

<p>The select element defines a drop-down list:</p>

<form action="#">
  <label>Choose a car:</label>
  <select name="cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi" selected>Audi</option>
  </select>

  <input type="submit">
</form>

</body>
</html>
```

PER SCHOLAS

Practice Activity: HTML Forms

Using HTML, create the following pages:

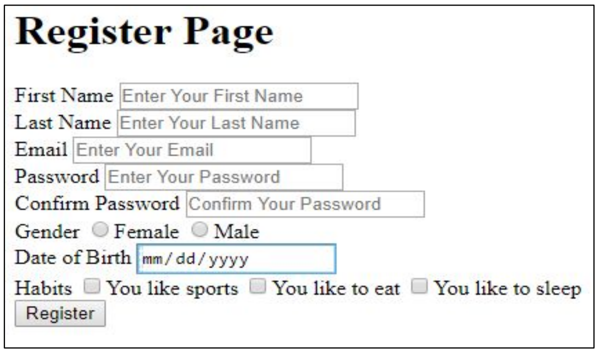
1) **Create a Login Page**



Login Page

Email [Enter Your Email]
Password [Enter Your Password]
[Login]

2) **Create a Register Page**



Register Page

First Name [Enter Your First Name]
Last Name [Enter Your Last Name]
Email [Enter Your Email]
Password [Enter Your Password]
Confirm Password [Confirm Your Password]
Gender ○ Female ○ Male
Date of Birth [mm/dd/yyyy]
Habits ☐ You like sports ☐ You like to eat ☐ You like to sleep
[Register]

PER SCHOLAS

Practice Assignment: Page Wireframes

Please follow the links below to the practice assignments for creating page wireframes:

- PA - 307.2.1 - Create a Wireframe for your Landing Page
- PA - 307.2.2 - Create a Wireframe for your Login and Registration Pages

You can also find these practice assignments on Canvas under the Assignments section.

If you have questions while performing the activity, ask your instructors for assistance.

PER SCHOLAS

Practice Activity: Web Page

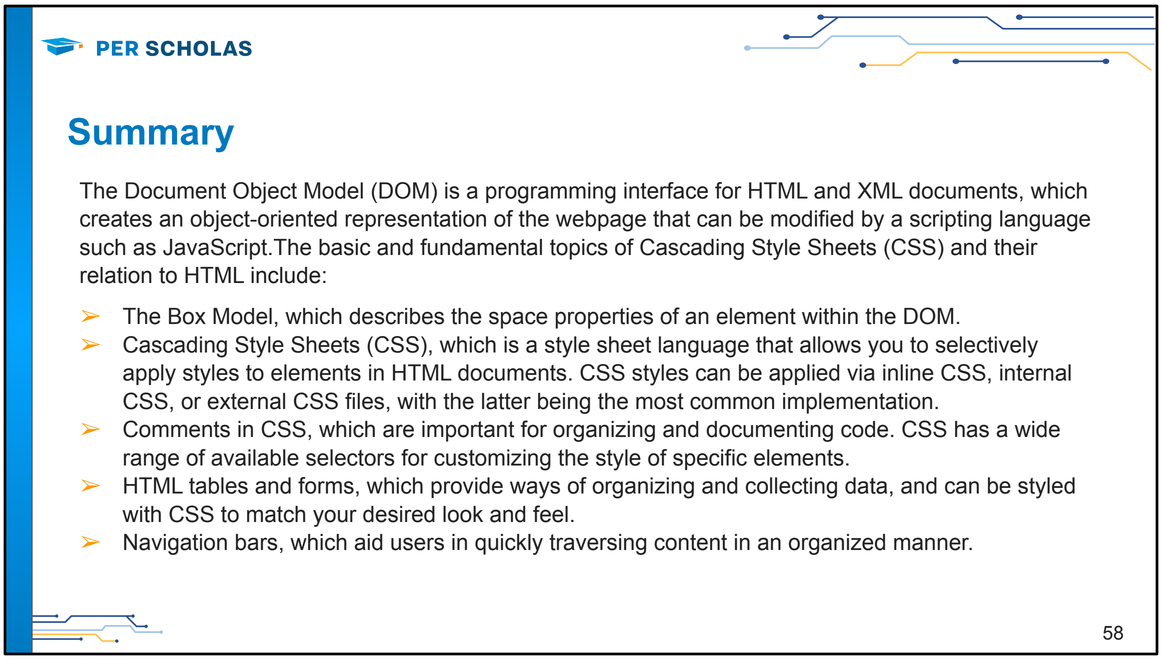
Create a simple single web page. Choose any industry, such as family, movies, books, bootique, construction, event organization, or any other idea you like for web page.

- Try to use a large combination of the HTML and CSS tools that you have learned so far.
- Provide as much information as you can about the topic, and be creative in your styling.
- Provide at least one hyperlink to an actual website.
- Add at least one HTML form, such as a registration form, “get a quote” form, etc.
- After you are finished, take three minutes to present your page.
- Once presentations are done, spend the rest of the time working on your Capstone Project, using the tools and techniques you have learned so far.

PER SCHOLAS

Knowledge Check

- What is the Document Object Model?
- What are the four properties that make up the CSS box model?
- What is a CSS property, and how is it assigned a value?
- What is a CSS selector, and how can they be used to apply style to elements?
- What is the difference between inline style, internal style, and external style? Which takes precedence?
- How do you write a comment in CSS?
- What are combinators, and how are they used to select elements in CSS?
- How do you select a class in CSS? How do you select an id?
- What is a pseudo-class and what is it used for?
- What is a pseudo-element and what is it used for?



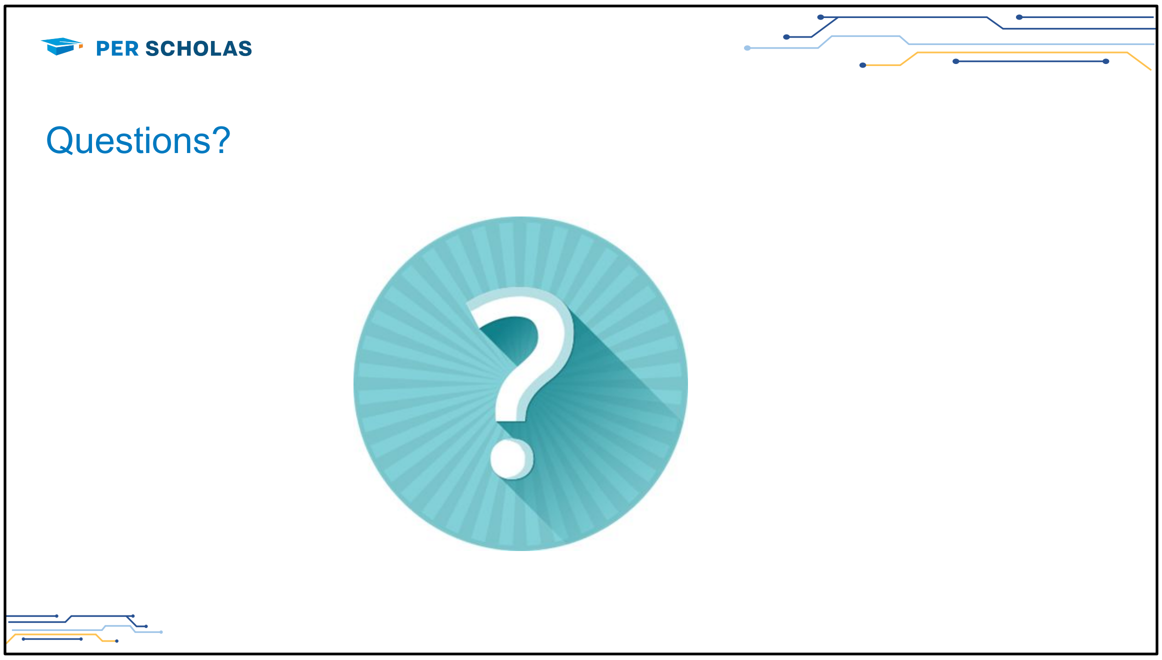
PER SCHOLAS

Summary

The Document Object Model (DOM) is a programming interface for HTML and XML documents, which creates an object-oriented representation of the webpage that can be modified by a scripting language such as JavaScript.The basic and fundamental topics of Cascading Style Sheets (CSS) and their relation to HTML include:

- The Box Model, which describes the space properties of an element within the DOM.
- Cascading Style Sheets (CSS), which is a style sheet language that allows you to selectively apply styles to elements in HTML documents. CSS styles can be applied via inline CSS, internal CSS, or external CSS files, with the latter being the most common implementation.
- Comments in CSS, which are important for organizing and documenting code. CSS has a wide range of available selectors for customizing the style of specific elements.
- HTML tables and forms, which provide ways of organizing and collecting data, and can be styled with CSS to match your desired look and feel.
- Navigation bars, which aid users in quickly traversing content in an organized manner.

58



PER SCHOLAS

Questions?