

# HTML5 Training for Web Developers

Sections and Articles

## Lesson 1, Activity 2: The section Tag

HTML5 introduces the <section> tag to break up a page in a meaningful way, leaving <div> tags to be used only for structurally meaningless page sectioning. First, we will look at the HTML 4 way and then we'll see how to "fix" things with HTML5.

**The HTML 4 Way**

This following two demos show how pages can be "sectioned" in HTML 4.

**Code Sample:**

<html5-sections/Demos/html4-course-outline.html>

```

---- C O D E   O M I T T E D ----

CSS Training

Class Overview
This CSS training class teaches students to use Cascading Style Sheets to format HTML

Class Goals

    • Learn the benefits of CSS.

    • Learn to avoid using deprecated tags and attributes.

    • Learn CSS syntax.

    • Learn to use
      and tags appropriately.

    • Learn most of the common properties and their values.

    • Learn to create custom CSS cursors.

    • Learn to style links with CSS to create "CSS Buttons".

    • Learn to work with borders, margin, and padding (the box model).

    • Learn to style tables with CSS.

Class Outline

    • Crash Course in CSS

    • CSS Fonts

    • CSS Text

    • Colors and Backgrounds

    • Custom Cursors

    • CSS and Links

    • Borders, Margins and Padding

    • Styling Tables with CSS

Technical Requirements

    1. HTML or Text Editor

    2. Web Browser

```

We have used some CSS to make the page render as follows:

## CSS Training

### Class Overview

This CSS training class teaches students to use Cascading Style Sheets to format HTML pages.

### Class Goals

- Learn the benefits of CSS.
- Learn to avoid using deprecated tags and attributes.
- Learn CSS syntax.
- Learn to use <div> and <span> tags appropriately.
- Learn most of the common properties and their values.
- Learn to create custom CSS cursors.
- Learn to style links with CSS to create "CSS Buttons".
- Learn to work with borders, margin, and padding (the box model).
- Learn to style tables with CSS.

### Class Outline

1. Crash Course in CSS
2. CSS Fonts
3. CSS Text
4. Colors and Backgrounds
5. Custom Cursors
6. CSS and Links
7. Borders, Margins and Padding
8. Styling Tables with CSS

### Technical Requirements

1. HTML or Text Editor
2. Web Browser

Note that each "section" is separated visually, but without the CSS they would not be. Also note that the browser doesn't know that these divisions have structural meaning. To illustrate this point, we'll add a couple of meaningless <div> tags.

### Code Sample:

<html5.sections/Demos/html4-course-outline?html>

```

---- CODE OMITTED ----
CSS Training

---- CODE OMITTED ----

---- CODE OMITTED ----

---- CODE OMITTED ----

---- CODE OMITTED ----

```

With new "left" and "right" divs, we can now style the page with CSS to render as follows:

## CSS Training

### Class Overview

This CSS training class teaches students to use Cascading Style Sheets to format HTML pages.

### Class Goals

- Learn the benefits of CSS.
- Learn to avoid using deprecated tags and attributes.
- Learn CSS syntax.
- Learn to use <div> and <span> tags appropriately.
- Learn most of the common properties and their values.
- Learn to create custom CSS cursors.
- Learn to style links with CSS to create "CSS Buttons".
- Learn to work with borders, margin, and padding (the box model).
- Learn to style tables with CSS.

### Class Outline

1. Crash Course in CSS
2. CSS Fonts
3. CSS Text
4. Colors and Backgrounds
5. Custom Cursors
6. CSS and Links
7. Borders, Margins and Padding
8. Styling Tables with CSS

### Technical Requirements

1. HTML or Text Editor
2. Web Browser

But these two new <div> tags have not provided any new context to the page. They are purely there for formatting purposes.

## The HTML5 Way

Now let's look at the HTML5 way.

### Code Sample:

<html5.sections/Demos/html5-course-outline.html>

```

---- CODE OMITTED ----
CSS Training

  Class Overview
---- CODE OMITTED ----

  Class Goals
---- CODE OMITTED ----

  Class Outline
---- CODE OMITTED ----

  Technical Requirements

```

---- CODE OMITTED ----

The above code will render the following:

#### CSS Training

<p><b>Class Overview</b></p> <p>This CSS training class teaches students to use Cascading Style Sheets to format HTML pages.</p> <p><b>Class Goals</b></p> <ul style="list-style-type: none"> <li>Learn the benefits of CSS.</li> <li>Learn to avoid using deprecated tags and attributes.</li> <li>Learn CSS syntax.</li> <li>Learn to use &lt;div&gt; and &lt;span&gt; tags appropriately.</li> <li>Learn most of the common properties and their values.</li> <li>Learn to create custom CSS cursors.</li> <li>Learn to style links with CSS to create "CSS Buttons".</li> <li>Learn to work with borders, margin, and padding (the box model).</li> <li>Learn to style tables with CSS.</li> </ul>	<p><b>Class Outline</b></p> <ol style="list-style-type: none"> <li>Crash Course in CSS</li> <li>CSS Fonts</li> <li>CSS Text</li> <li>Colors and Backgrounds</li> <li>Custom Cursors</li> <li>CSS and Links</li> <li>Borders, Margins and Padding</li> <li>Styling Tables with CSS</li> </ol> <p><b>Technical Requirements</b></p> <ol style="list-style-type: none"> <li>HTML or Text Editor</li> <li>Web Browser</li> </ol>
--	--

Notice how we used <section> tags for the structurally meaningful parts:

- Overview
- Goals
- Outline
- Technical Requirements

And <div> tags for the structurally meaningless "left" and "right" divisions.

#### Display of HTML5 Structural Elements

By default, browsers treat the <div> tag as a block-level element as per [instructions provided by the W3C](#). However, the W3C has not specified anything for the new HTML5 structural tags (e.g., header, footer, aside, section, article, nav). As such, the different browser makers have had to decide individually whether to display these elements as blocks or inlines.

As the following shows, the browser makers initially differed in the way they handled the display of structural tags:

#### Code Sample:

[html5.sections/Demos/html5-structural-tags.html](#)

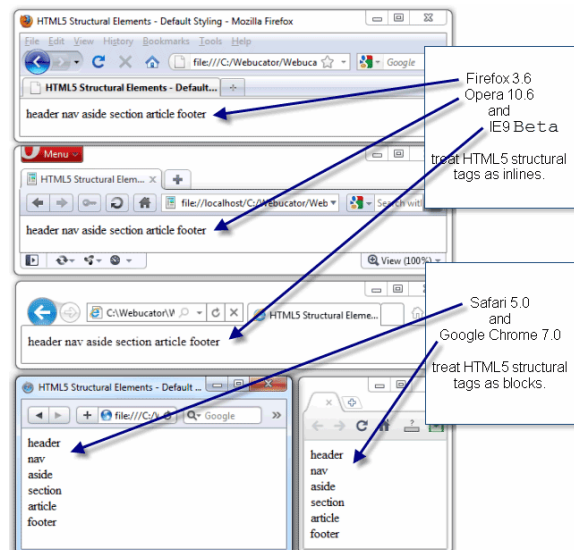
```

HTML5 Structural Elements - Default Styling

header
nav
aside
section
article
footer

```

Here is how different "early HTML5" browsers displayed this page:



As the image above shows, the following browsers treat HTML5 structural tags as inlines:

- Firefox 3.6
- Opera 10.6
- IE9 Beta

And the following browsers treat them as blocks:

- Safari 5.0

- Google Chrome 7.0

All the latest browsers correctly treat them as block-level elements; however, to be backward-compatible, you may want to include the following code in your CSS:

```
header, footer, aside, section, article, nav, hgroup {  
  display: block;  
}
```

#### Internet Explorer 8 and Earlier

Unfortunately, for Internet Explorer 8 (and earlier), we must take a further step. Because IE8 does not recognize these new tags, it ignores them unless they have been created programatically (i.e., with JavaScript). You can create each element using `document.createElement()` but someone has already done all this heavy lifting for you. To make sure IE8 treats these new elements properly, all you have to do is add the following code to the head:

```
<!--[if lt IE 9]>  
<script src="//html5shiv.googlecode.com/svn/trunk/html5.js"></script>  
<![endif]-->
```

The included JavaScript is called the *html5shim* or *html5shiv*. More information is available at <http://code.google.com/p/html5shiv>

Let's now take a look at the `article` element.

## Lesson 1, Activity 3: The article Tag

The W3C specification described the [article element as follows](#):

*The article element represents a self-contained composition in a document, page, application, or site and that is, in principle, independently distributable or reusable, e.g. in syndication. This could be a forum post, a magazine or newspaper article, a blog entry, a user-submitted comment, an interactive widget or gadget, or any other independent item of content.*

The major difference between the `article` element and the `section` element is that an `article` element encapsulates content that could stand alone and might be of interest outside the context of the page. The most obvious example is a blog entry; however, we could also apply this to the whole of our course description as you can imagine wanting to syndicate all of our outlines to a website that aggregates course information from different training companies.

To do this, we would just wrap everything up in an `<article>` tag:

**Code Sample:**

[html5.sections/Demos/html5-course-outline-article.html](#)

```
<!DOCTYPE HTML>
---- C O D E   O M I T T E D ----
<article id="css-course">
  <h1>CSS Training</h1>
  <div id="left">
    <section id="overview">
      ---- C O D E   O M I T T E D ----
    </section>
    <section id="goals">
      ---- C O D E   O M I T T E D ----
    </section>
  </div>
  <div id="right">
    <section id="outline">
      ---- C O D E   O M I T T E D ----
    </section>
    <section id="tech-reqs">
      ---- C O D E   O M I T T E D ----
    </section>
  </div>
</article>
</body>
</html>
```

This would have no visual impact on the page.

Note that you can nest articles. The common example used is a comment in a blog article. However, we might want to change our **overview** from a `section` element to an `article` element recognizing that the course aggregation site might want to have a page aggregating the overviews:

**Code Sample:**

[html5.sections/Demos/html5-course-outline-article-nested.html](#)

```
<!DOCTYPE HTML>
---- C O D E   O M I T T E D ----
<article id="css-course">
  <h1>CSS Training</h1>
  <div id="left">
    <article id="overview" title="Webucator CSS Class Overview">
      <h2>Class Overview</h2>
      <p>This CSS training class teaches students to use Cascading Style Sheets to format HTML pages.</p>
    </article>
    <section id="goals">
      ---- C O D E   O M I T T E D ----
    </section>
  </div>
  <div id="right">
    <section id="outline">
      ---- C O D E   O M I T T E D ----
    </section>
    <section id="tech-reqs">
      ---- C O D E   O M I T T E D ----
    </section>
  </div>
</article>
</body>
</html>
```

In addition to nesting articles within articles, you can:

- Nest sections within sections.
- Nest sections within articles.
- Nest articles within sections.

## Lesson 1, Activity 5: Using section and article Elements

Duration: 15 to 25 minutes.

In this exercise, you will modify an HTML page we worked on earlier in the course to replace *meaningless* div elements with *meaningful* section and article elements.

1. Open [html5.sections/Exercises/html5-layout.html](#).
2. Replace *meaningless* div elements with *meaningful* section and article elements. Note that there is room for interpretation here, so there is no one correct solution.
3. To keep the page looking as it did before, you will also need to modify [html5.sections/Exercises/style-html5.css](#).

Be sure to try it on your own before looking at the solution below.

**Solution: The HTML**
[html5.sections/Solutions/html5-layout.html](#)

```
<!DOCTYPE HTML>
---- C O D E   O M I T T E D ----
<div id="content">
  <article id="services" title="HTML5 Solutions Services">
    <h1>Services</h1>
    <section id="training">

---- C O D E   O M I T T E D ----
  </section>
  <section id="consulting">

---- C O D E   O M I T T E D ----
  </section>
</article>
<article id="products" title="HTML5 Solutions Products">
  <h1>Products</h1>
  <section id="editor">

---- C O D E   O M I T T E D ----
  </section>
  <section id="templates">

---- C O D E   O M I T T E D ----
  </section>
</article>
</div>
---- C O D E   O M I T T E D ----
```

**Code Explanation**

An argument could be made for turning the "content" div into a section; however, the main reason for wrapping that content in a div tag is to be able to position the main content on the page, so we left it as a div.

Likewise, you could argue for making each partner div a section or even an article, but we wrapped them in div tags mostly so that we could float them left.

Finally, turning the "services" and "products" div elements into article elements is a bit arbitrary too. You could just as easily make them sections.

**Solution: The CSS**
[html5.sections/Solutions/style-html5.css](#)

```
body {
  font-family:tahoma;
}

header, footer, aside, section, article, nav, hgroup {
  display: block;
}
---- C O D E   O M I T T E D ----
#content section h2 {
  float:right;
  font-size:xx-large;
  color:#006;
}

#content>article {
  clear:both;
}

#content article section {
  float:left;
  padding:10px;
  width: 500px;
  border:1px solid #006;
  font-size:small;
}
---- C O D E   O M I T T E D ----
```

## Lesson 1, Activity 6: Outlining

The HTML5 specification describes an algorithm to determine the outline of a web page. This allows:

1. User agents (e.g., browsers and screenreaders) the ability to present data to the consumer in a logical way.
2. Syndicators to syndicate portions of a page's content in a structurally meaningful way.
3. Authors to create pages with any number of heading levels.

## Sectioning

To understand how the outlining algorithm works, you need to know about **sectioning roots** and **sectioning content**.

## Sectioning Roots

Sectioning roots are elements that have their own outline and are not included in the ancestral sectioning roots and sectioning content. The following elements are sectioning roots:

1. body
2. blockquote
3. td
4. details
5. fieldset
6. figure

So each of the above elements has its own outline.

## Sectioning Content

Sectioning content is defined by elements that can (but don't necessarily) have their own headers and footers. The following elements define sectioning content:

1. section
2. article
3. aside
4. nav

While each sectioning content element has its own outline, it also contributes to the outline of ancestral sectioning roots and sectioning content.

So, in considering an outline, first start with the sectioning root and then drill down from there.

## The Outline's "List Items"

It helps to think of an outline as a numbered list with the nesting of list items determined by the sectioning content elements and the content of the list items determined by heading elements.

- h1
- h2
- h3
- h4
- h5
- h6
- hgroup

Notice the hgroup element is included in the list of heading elements. Only the highest-level h# element nested within the hgroup element is considered in the outline.

Two factors play into the outline level:

1. The nested level of the sectioning content element.
2. The *relative* level of heading within a sectioning content element.

Consider the following demo:

## Code Sample:

[html5-sections/Demos/html5-outlining.html](#)

```
HTML5 Outlining

Fight for Fruit
We hate vegetables

• Apples

• Bananas

• Oranges

Apples
Why do you think the expression is an apple of your eye" and not the zucchini of your eye. It's because apples are good and zucchinis aren't, that's why!.
Doctor Bias towards Veggies
An apple a day keeps the doctor away and we like going to the doctor almost as little as we like eating vegetables. But doctors want your business, that's why they often

Peaches
If someone tells you that you're a peach, that's a good thing, right? But who wants to be told that they're a squash? Nobody.

Cherries
Remember the line from the musical Oklahoma in the song "I can't say No:"

S'posin' 'at he says 'at yer lips're "like cherries"

Having lips like cherries is flattering, right? Would you like lips like eggplant? Of course, not. Cherries = good. Eggplant = bad.

Vegetables to Avoid
Really you should avoid all veggies as they are likely to make your taste buds squirm, but there are a couple you should avoid at all costs.

Tomatoes are Traitors
Tomatoes are the Benedict of all "veggies" and should never be eaten.
The Benedict Arnold Fruit
Nobody likes a traitor...
```



```

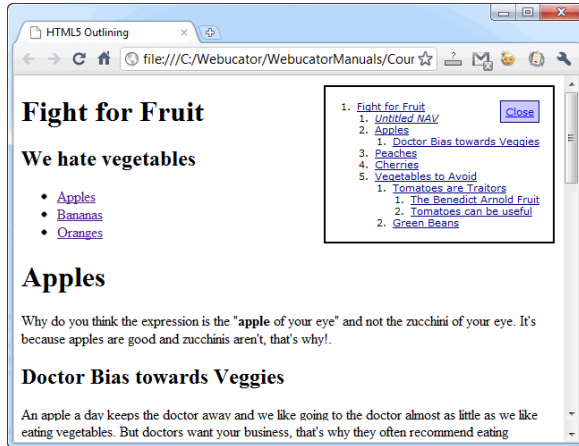
Tomatoes can be useful
We're not saying never to buy tomatoes. Just don't eat them. They can be useful for other things, such as throwing at bad actors.

Green Beans
If it's green, it ain't ripe yet. Muff said.

© Fruit Lobby. All rights reserved.

```

Using the HTML5 outliner bookmarklet discussed earlier, we can see the resulting outline:



You'll notice that the "Doctor Bias towards Veggies" heading, which is an `h2` element, is at the same level as the "Tomatoes are Traitors" heading, which is an `h1` element. This is because the "Tomatoes are Traitors" `h1` is nested three levels deep in sectioning content elements (body -> aside -> article), while the "Doctor Bias towards Veggies" `h2` is nested only two levels deep ((body -> section).

The screenshot in the example above shows the interpreted outline in the upper-right corner. As modern browsers don't yet support the outlining algorithm, we've used the [h5o HTML 5 Outliner bookmarklet](#) to display the outline. Follow these instructions to get the bookmarklet:

1. Go to <http://code.google.com/p/h5o/>.
2. Click on the Bookmarklet link.
3. Download the Bookmarklet HTML ([.html](#) extension) file. (Make sure to get the IE version if you're using Internet Explorer.)
4. You may need to change the extension from [.txt](#) to [.html](#).
5. Open the file in your HTML5-compliant browser.
6. Drag the link on the page into your "Favorites" or "Bookmark" toolbar.

#### Accessibility

Again, it is the **relative** heading level within a sectioning content element that determines the outline level. So, from an outlining point of view, the following two code samples are the same:

```

Heading
Content
Heading
Content>

```

```

Heading
Content
Heading
Content>

```

In a couple of ways, the first sample is nicer. It allows us to think about each sectioning content element individually, starting each with an `h1` element. In this way, we also get an unlimited number of heading levels as we can create lower-level headings simply by nesting sectioning content elements.

However, there are two big downsides to the first sample:

1. **It creates a CSS nightmare.** You have to write rules for every possible nesting level of the sectioning content elements:

```

section>section>h1 {}
section>section>h2 {}
section>article>h1 {}
section>article>h2 {}
article>section>h1 {}
article>section>h2 {}
section>aside>h1 {}
section>aside>h2 {}
aside>section>h1 {}
aside>section>h2 {}
...

```

Yikes!

2. **Modern browsers do not currently support the HTML5 outlining algorithm**, which means that accessibility tools such as [JAWS Screen Reader](#) have to rely solely on the `h#` elements to determine the heading levels. Because these tools use heading levels as a way of providing simple page navigation (e.g., jumping from `h1` to `h1`), people using screen readers will not be able to experience your page correctly if you restart the heading level numbers with each new sectioning content element.

## Lesson 1, Activity 7: Determining the Outline

Duration: 10 to 20 minutes.

In this exercise, you will try to determine the outline of an HTML page.

1. Review the code below.
2. Create a list either on paper, in a text editor, or in a word processor that shows the HTML outline as specified by the HTML5 specification.

**Code Sample:**[html5:sections/Exercises/html5-outlining.html](#)

```

<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
<title>HTML5 Outlining</title>
<link href="style-html5-outlining.css" rel="stylesheet">
</head>
<body>
<header>
<hgroup>
<h1>HTML5 Solutions</h1>
<h2>HTML5 Products and Services</h2>
</hgroup>

</header>
<nav id="mainnav">
<ul>
<li>Home</li>
<li><a href="">Products</a></li>
<li><a href="">Services</a></li>
<li><a href="">About</a></li>
</ul>
</nav>
<div id="content">
<article id="services" title="HTML5 Solutions Services">
<h1>Services</h1>
<section id="training">
<h2>HTML5 Training</h2>

<p>Lorem ipsum dolor sit amet....</p>
</section>
<section id="consulting">
<h2>HTML5 Consulting</h2>

<p>Lorem ipsum dolor sit amet....</p>
</section>
</article>
<article id="products" title="HTML5 Solutions Products">
<h1>Products</h1>
<section id="editor">
<h2>HTML5 Editor</h2>

<p>Lorem ipsum dolor sit amet....</p>
</section>
<section id="templates">
<h2>HTML5 Templates</h2>

<p>Lorem ipsum dolor sit amet....</p>
</section>
</article>
</div>
<aside id="partners">
<div>
<h3>Partner 1</h3>
<p>Lorem ipsum dolor sit amet....</p>
</div>
<div>
<h3>Partner 2</h3>
<p>Lorem ipsum dolor sit amet....</p>
</div>
<div>
<h3>Partner 3</h3>
<p>Lorem ipsum dolor sit amet....</p>
</div>
</aside>
<footer>
<p>&copy; Webucator, Inc. All rights reserved.</p>
</footer>
</body>
</html>

```

**Challenge**

Open [html5:sections/Demos/html5-outlining.html](#) in your editor. Add an embedded style sheet so that only the headings are shown and that they are sized and tabbed according to their outline level as shown in the screenshot below: If you use the HTML5 outliner bookmarklet, don't worry about the "Untitled Nav" that it shows. That's just a warning to indicate that there is a sectioning content element without a heading, but you don't have to have a heading in every sectioning content element.

The solution is shown below:

**Challenge Solution:**[html5:sections/Solutions/right-for-fruit.html](#)

```

---- CODE OMITTED ----
<style>
h1, h2, h3 {

```

```
background-color:#eee;
}

h1 {
  font-size:x-large;
}

h2, aside>article>h1 {
  font-size:large;
  margin-left:50px;
}

aside>article>h2 {
  font-size:medium;
  margin-left:100px;
}

ul, p {
  display:none;
}
</style>
---- C O D E   O M I T T E D ----
```