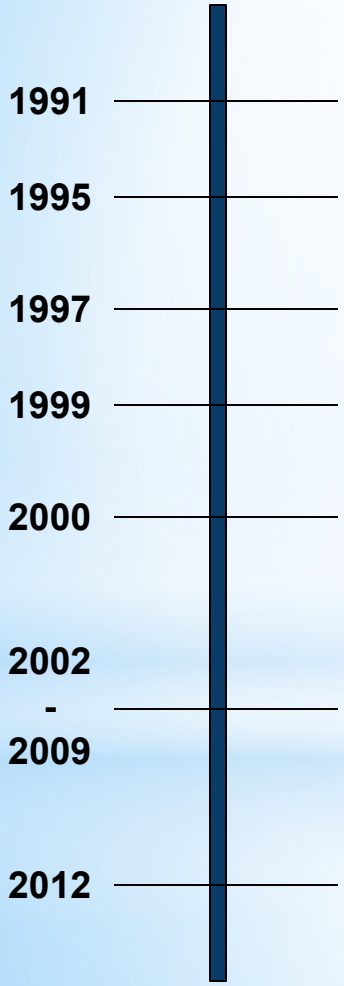
Introduction to HTML5

*History of HTML



After HTML 4.01 was released, focus shifted to XHTML and its stricter standards.

XHTML 2.0 had even stricter standards than 1.0, rejecting web pages that did not comply. It fell out of favor gradually and was abandoned completely in 2009.

HTML5 is much more tolerant and can handle markup from all the prior versions.

Though HTML5 was published officially in 2012, it has been in development since 2004.

*What is HTML5?

- HTML5 is the newest version of HTML, only recently gaining partial support by the makers of web browsers.
- It incorporates all features from earlier versions of HTML, including the stricter XHTML.
- It adds a diverse set of new tools for the web developer to use.

*Goals of HTML5

- Support all existing web pages. With HTML5, there is no requirement to go back and revise older websites.
- Reduce the need for external plugins and scripts to show website content.
- Improve the semantic definition (i.e. meaning and purpose) of page elements.
- Make the rendering of web content universal and independent of the device being used.
- Handle web documents errors in a better and more consistent fashion.

About HTML

- HTML5 is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG).
- WHATWG was working with web forms and applications, and W3C was working with XHTML 2.0. In 2006, they decided to cooperate and create a new version of HTML.

*New Elements in HTML5

<article>
<aside>
<audio>
<canvas>
<datalist>
<figure>
<figcaption>
<footer>
<header>
<hgroup>
<mark>
<nav>
<progress>
<section>
<source>
<svg>
<time>
<video>

These are just some of the new elements introduced in HTML5. We will be exploring each of these during this course.

*Other New Features in HTML5

- Built-in audio and video support (without plugins)
- Enhanced form controls and attributes
- The Canvas (a way to draw directly on a web page)
- Drag and Drop functionality
- Support for CSS3 (the newer and more powerful version of CSS)
- More advanced features for web developers, such as data storage and offline applications.

*First Look at HTML5

Remember the DOCTYPE declaration from XHTML?

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

In HTML5, there is just one possible DOCTYPE declaration and it is simpler:

```
<!DOCTYPE html>
```

Just 15 characters!

The DOCTYPE tells the browser which type and version of document to expect. This should be the last time the DOCTYPE is ever changed. From now on, all future versions of HTML will use this same simplified declaration.

HTML5 HTML

```
<!DOCTYPE html>
```

HTML 4

```
<!DOCTYPE html PUBLIC
  "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
```

Transitional XHTML 1.0

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

Strict XHTML 1.0

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

XML Declaration

```
<?xml version="1.0" ?>
```

*DOCTYPEs

*The <html> Element

This is what the <html> element looked like in XHTML:

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

Again, HTML5 simplifies this line:

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

The **lang** attribute in the <html> element declares which language the page content is in. Though not strictly required, it should always be specified, as it can assist search engines and screen readers.

Each of the world's major languages has a two-character code, e.g. Spanish = "es", French = "fr", German = "de", Chinese = "zh", Arabic = "ar".

The <head> Section

Here is a typical XHTML <head> section:

```
<head>
  <meta http-equiv="Content-type" content="text/html; charset=UTF-8" />
  <title>My First XHTML Page</title>
  <link rel="stylesheet" type="text/css" href="style.css" />
</head>
```

And the HTML5 version:

```
<head>
  <meta charset="utf-8">
  <title>My First HTML5 Page</title>
  <link rel="stylesheet" href="style.css">
</head>
```

Notice the simplified character set declaration, the shorter CSS stylesheet link text, and the removal of the trailing slashes for these two lines.

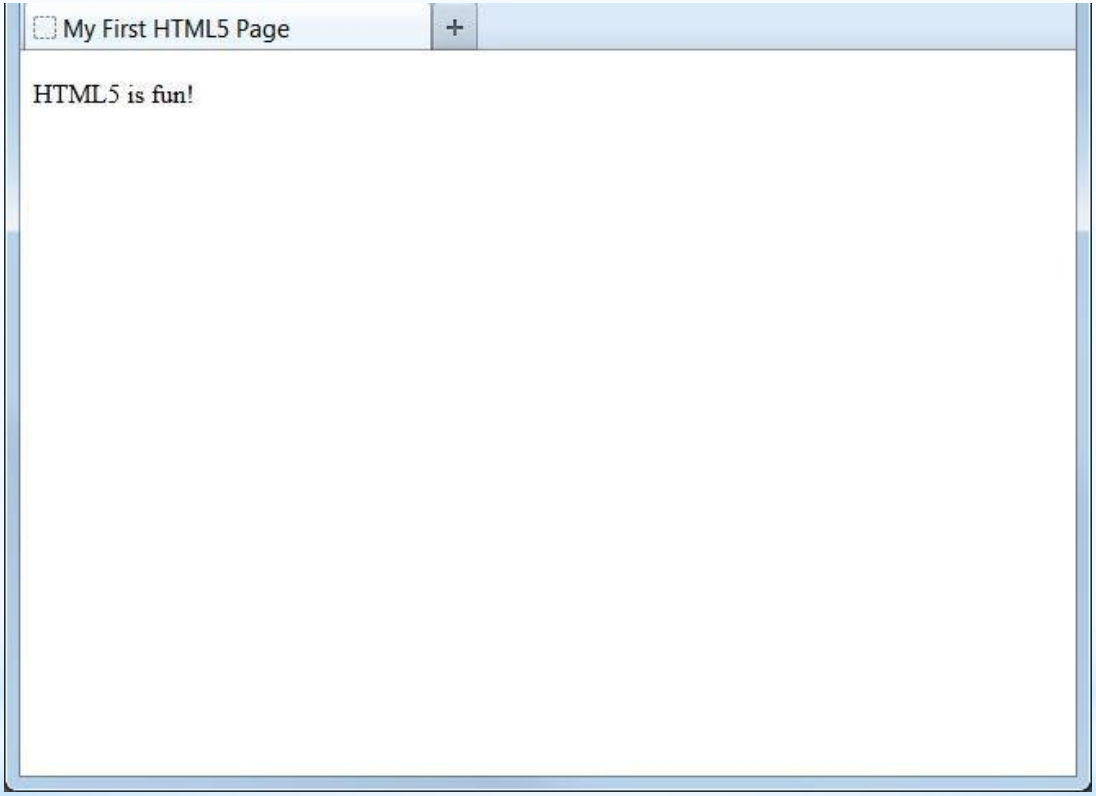
*Basic HTML5 Web Page

Putting the prior sections together, and now adding the <body> section and closing tags, we have our first complete web page in HTML5:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>My First HTML5 Page</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <p>HTML5 is fun!</p>
</body>
</html>
```

Let's open this page in a web browser to see how it looks…

Viewing the HTML5 Web Page



Even though we used HTML5, the page looks exactly the same in a web browser as it would in XHTML. Without looking at the source code, web visitors will not know which version of HTML the page was created with.

HTML5 Syntax

*Looser Syntax Rules

- In an effort to make HTML5 accommodate older web pages, numerous things that would be invalid in XHTML are now permitted as valid.
- To web designers accustomed to writing strict code in XHTML, these looser rules might appear sloppy or unprofessional.
- These relaxed rules are not suggestions or guidelines for modern web programmers. Rather, they were put in place to be more accepting of older websites.

Though we will cover the new, looser syntax rules, we will be adopting just a few of them for this course. We will avoid the rest in favor of cleaner and more readable syntax, very close to what we learned in XHTML.

*Capitalization

In XHTML, all elements and attributes are required to be in lower case:

```
<body>
  <p class="intro">This is the first paragraph</p>
</body>
```

While the following code will not validate as error-free in XHTML, it is now considered valid in HTML5:

```
<BODY>
  <p Class="intro">This is the first paragraph</P>
</body>
```

Inconsistent capitalization makes reading code more difficult and serves no useful purpose. For these reasons, we will continue to use all lower-case in our pages.

*Closing Tags

In XHTML, elements such as <p> and <li> require closing tags:

```
<p>This is the first paragraph.</p>
<p>This is the second paragraph.</p>
<ul>
  <li>First list item</li>
  <li>Second list item</li>
</ul>
```

But in HTML5, the following is considered valid:

```
<p>This is the first paragraph.
<p>This is the second paragraph.</p>
<ul>
  <li>First list item
  <li>Second list item
</ul>
```

Omitting closing tags makes reading code more difficult and can lead to unexpected problems when going back to edit pages. We will continue to close all such elements.

*Self-Closing Elements

In XHTML, self-closing elements require closing slashes:

```
<br />
<img src="image1.jpg" alt="Description" />
<hr />
```

In HTML5, these closing slashed can now be omitted:

```
<br>
<img src="image1.jpg" alt="Description">
<hr>
```

These closing slashes were required by XHTML but served no useful purpose and did not improve the readability of code. For these reasons, we will adopt this relaxed rule and omit them in all our new HTML5 documents.

Other self-closing elements include <meta> and <link>.

Attribute Quotes

In XHTML, all element attributes must be enclosed in quotation marks:

```
<div id="header">
<img src="image1.jpg" alt="Description" />
<a href="page2.html">Click Here</a>
```

In HTML5, it's permissible to drop these quotes:

```
<div id=header>
<img src=image1.jpg alt=Description>
<a href=page2.html>Click Here</a>
```

Nearly all web programmers agree that omitting attribute quotes is a bad idea. It reduces readability of code and **will result in major errors if the attribute value contains special characters or spaces**. We will continue using the quotes.

*Form Attributes

In XHTML, in order to "pre-select" default choices on forms, we used this syntax:

```
<input type="radio" name="gender" value="male" />Male
<input type="radio" name="gender" value="female"
   checked="checked" />Female
```

In HTML5, we can shorten this a bit:

```
<input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="female" checked>Female
```

Like omitting self-closing tags, this change does not affect readability and also saves us a little typing, so we'll use the shorter syntax.

*Optional Page Elements

By now, we have become accustomed to seeing three basic elements in every HTML page: the <html>, <head>, and <body> elements.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>My First HTML5 Page</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <p>HTML5 is fun!</p>
</body>
</html>
```

Optional Page Elements Omitted

HTML5 now makes these three elements optional:

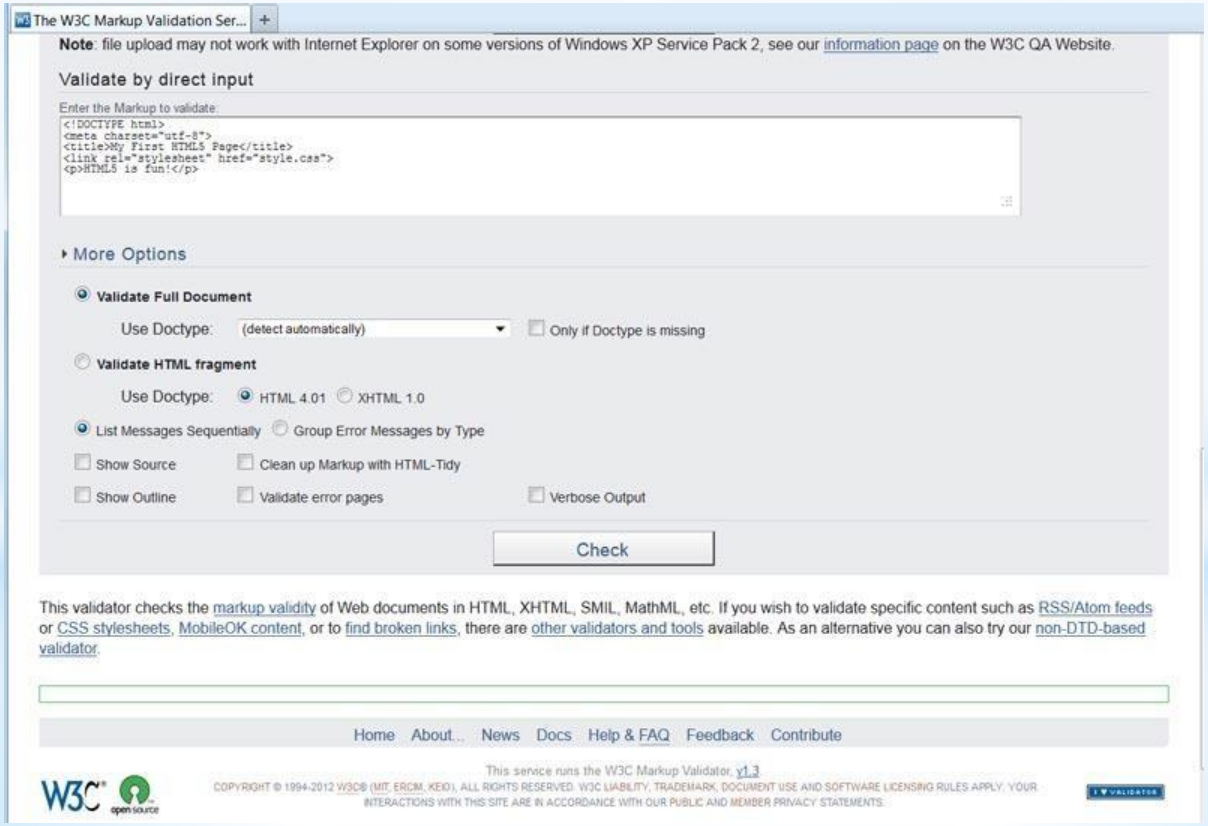
```
<!DOCTYPE html>
<meta charset="utf-8">
<title>My First HTML5 Page</title>
<link rel="stylesheet" href="style.css">
<p>HTML5 is fun!</p>
```

Hard to believe, but this is a completely valid web document in HTML5!

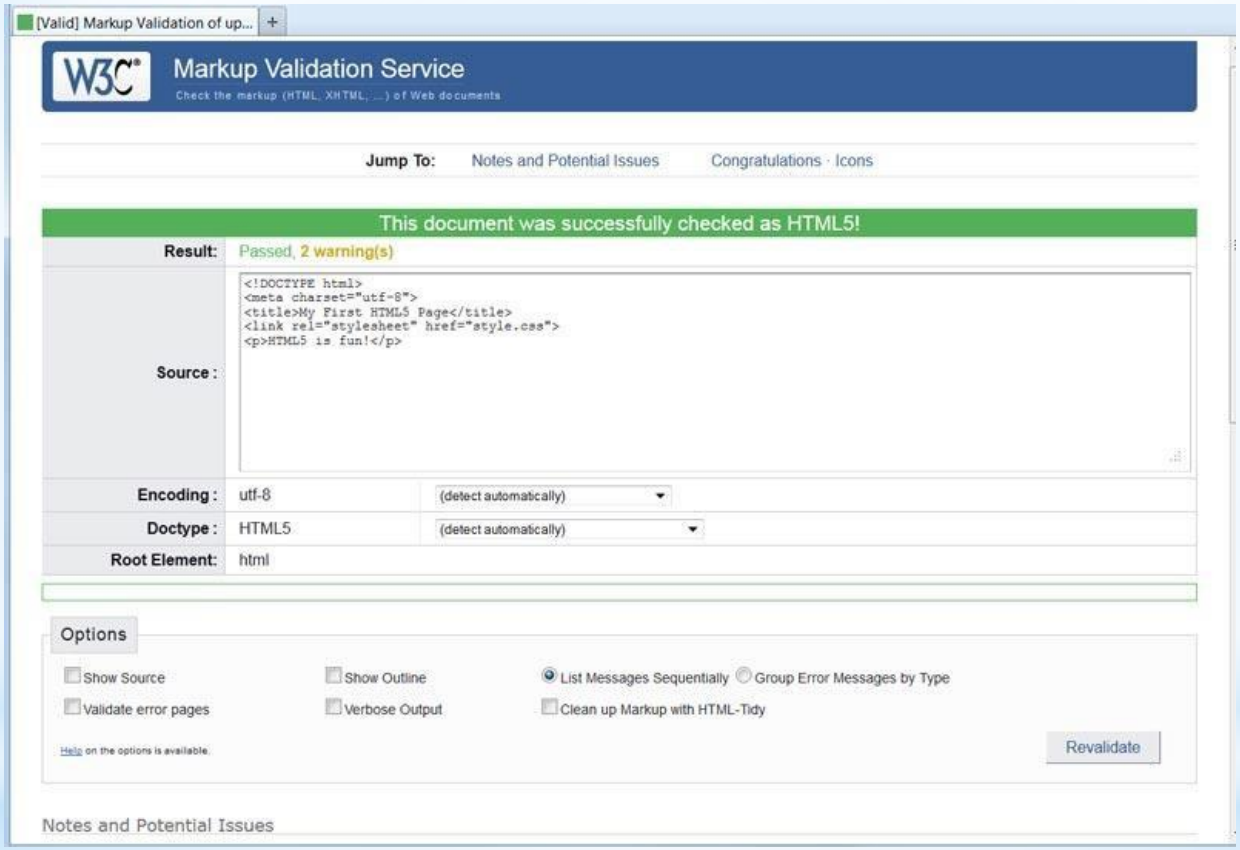
Very few web programmers choose to omit these traditional elements, which logically organize the document and make it easier to understand the flow of the page content, especially when reviewed by others. We will continue to include them in every document we create.

*HTML5 Page Validation

- Just as we did with XHTML, we can submit our pages to the validator at http://validator.w3.org/
- Let's try it now with the basic page that omits the <html>, <head>, and <body> tags:



*Validation Results



Even with the relaxed rules, running pages through the validator can be helpful to catch typos, missing tags, improperly nested elements, and other common errors.

* https://developer.mozilla.org/en-US/docs/Web/HTML/Element#obsolete_and_deprecated_elements

* Obsolete and deprecated elements

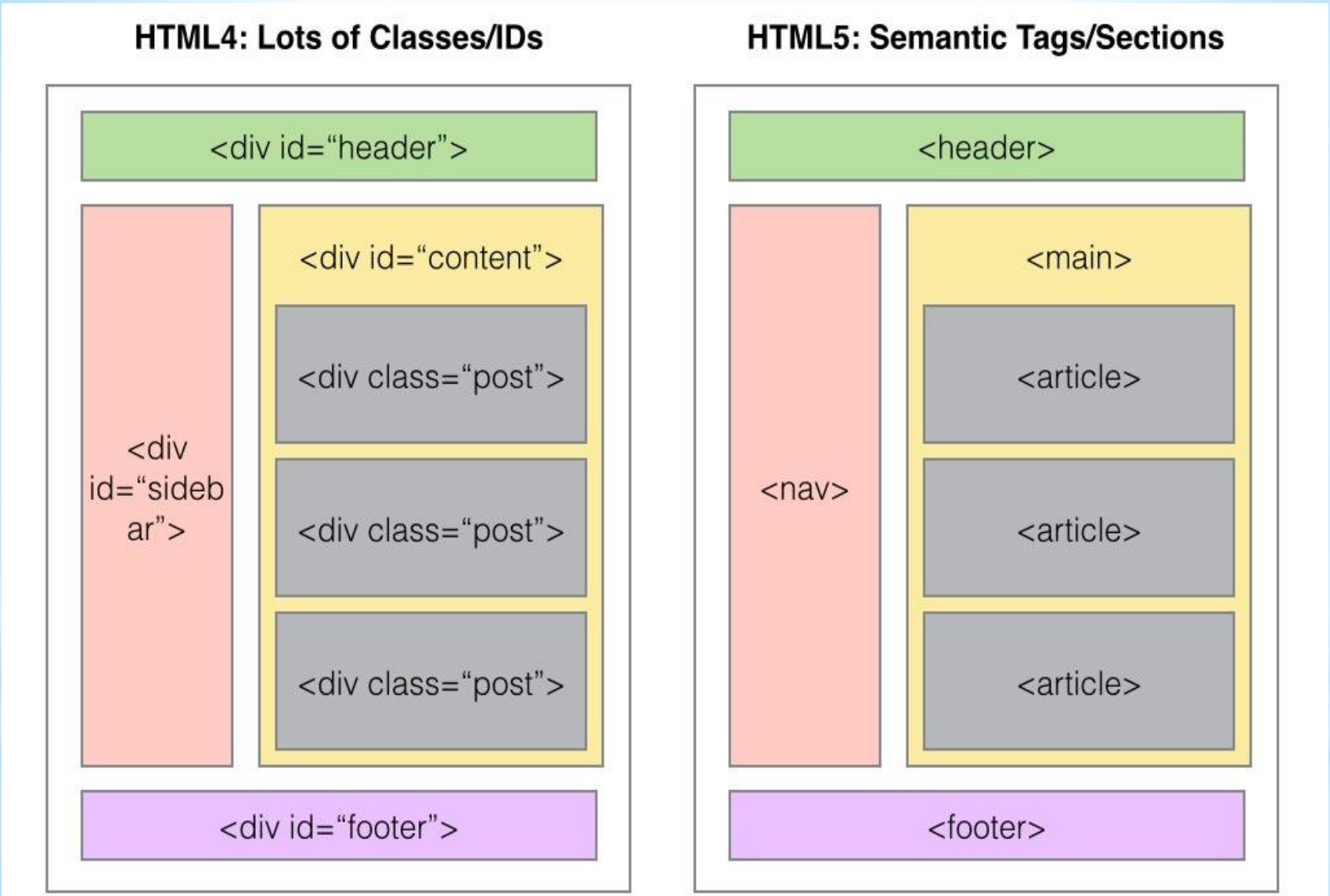
New Semantic Elements (Part 1)

*Semantics Explained

- The textbook definition of "semantics" is the study of the relationship between words and their meanings.
- In the context of HTML, a semantic element is one whose name describes its actual purpose.
- Take, for example, the element that we used extensively in XHTML. It's useful for defining a section of a web page but it does not indicate anything about the nature of the section itself.
- HTML5 introduces new semantic elements that better define and organize web documents.

*Benefits of Semantic Elements

- Pages become much easier to understand when it comes time to edit them in the future. This is especially true if someone other than the original programmer is doing the editing.
- Search engines can better understand our website's content. The better Google, Yahoo, and Bing can pinpoint what our pages are all about, the higher our pages will likely appear in the search results.
- Screen readers and assistive devices can more easily interpret the organization of a page, therefore presenting site content more effectively to disabled visitors.


HTML4: Lots of Classes/IDs

HTML5: Semantic Tags/Sections
<header>
<nav>
<main>
<article>
<article>
<article>
<footer>

The Header Section in XHTML

Remember how a typical XHTML web page defined the header section?

```
<div class="header">
  <h1>My Super Cool Website</h1>
</div>
```

And here's the CSS styling to define the height, width, and background color of the header section, as well as the text color and size of the <h1> heading:

```
.header {
  height: 100px;
  width: 800px;
  background-color: #0033FF;
}
.header h1 {
  text-size: 24px;
  color: #CCCCCC;
}
```

The HTML5 <header> Element

HTML5 now gives us a new semantic element for the header section:

```
<header>
  <h1>My Super Cool Website</h1>
</header>
```

In the CSS style sheet, adding styling to the header section is done using the element itself, rather than via a class. Notice there is no preceding dot:

```
header {
  height: 100px;
  width: 800px;
  background-color: #0033FF;
}
header h1 {
  text-size: 24px;
  color: #CCCCCC;
}
```

The HTML5 page still looks identical to the XHTML page.

The Footer Section in XHTML

In XHTML, defining the footer section was done in a similar fashion:

```
<div class="footer">
  <p>&copy; 2013 SuperCool LLC</p>
</div>
```

Here is the corresponding CSS styling to the footer section and text:

```
.footer {
  height: 40px;
  width: 800px;
  background-color: #0033FF;
}
.footer p {
  text-size: 16px;
  color: #CCCCCC;
  text-align: center;
}
```

*The HTML5 <footer> Element

HTML5 now provides us with a dedicated <footer> element:

```
<footer>
  <p>&copy; 2013 SuperCool LLC</p>
</footer>
```

The CSS styling remains the same, but we are now styling the element directly, rather than a class:

```
footer {
  height: 40px;
  width: 800px;
  background-color: #0033FF;
}
footer p {
  text-size: 16px;
  color: #CCCCCC;
  text-align: center;
}
```

Again, the two pages (XHTML and HTML5) look the same when rendered by a browser.

*Navigation in XHTML

In XHTML, defining the navigation menu followed a similar path:

```
<div class="nav">
  <div class="navlink">
    <a href="index.html">Home</a>
  </div>
  <div class="navlink">
    <a href="page2.html">Page 2</a>
  ...
</div>
```

The CSS styled the <nav> class:

```
.nav {
  border: 1px solid black;
  width: 798px;
  height: 35px;
...
}
.navlink {
  width: 199px;
...
}
```

*The HTML5 <nav> Element

HTML5 now provides us with the semantic <nav> element:

```
<nav>
  <div class="navlink">
    <a href="index.html">Home</a>
  </div>
  <div class="navlink">
    <a href="page2.html">Page 2</a>
  ...
</nav>
```

The CSS now styles the <nav> element:

```
nav {
  border: 1px solid black;
  width: 798px;
  height: 35px;
...
}
.navlink {
  width: 199px;
...
}
```

*Revisiting the Header Element

- The official W3C specification for the <header> element says that it "represents a group of introductory or navigational aids."
- The specification further states that, "A header element is intended to usually contain the section's heading (an h1–h6 element) but this is not required. The header element can also be used to wrap a section's table of contents, a search form, or any relevant logos."

Since the header section often includes site navigation menus, let's go back to our simple site and move the <nav> within the <header>.

Nesting <nav> Within <header>

Here we have moved the navigation menu to be inside the <header> section:

```
<header>
  <h1>My Super Cool Website</h1>
  <nav>
    <div class="navlink">
      <a href="index.html">Home</a>
    </div>
    <div class="navlink">
      <a href="page2.html">Page 2</a>
    ...
  </nav>
</header>
```

Many HTML5 web designers prefer to place the main site menu within the top header section, as we just did.

*Summary of Semantic Elements

- The new semantic elements in HTML5 add meaning to sections of a web document, assisting other programmers, search engines, and screen readers for the visually impaired.
- Just like the elements they replace, the new semantic elements do not actually do anything that is visible on the web page. That is accomplished completely with CSS styling.

In the coming days, we will be learning about and using several more of the new semantic elements in HTML5.

New Semantic Elements (Part 2)

Three More New HTML5 Elements

- The <article> element represents web content that could stand by itself, even if separated from the surrounding page information.
- The <aside> element represents content that is visually set apart from the main content of the page, yet is still somewhat related.
- The <section> element is a generic area of a web document that groups related content.

Along with <header>, <footer>, <nav>, and the all-purpose , these three elements make up the basic building blocks when designing site layouts. Let's look at each one in turn.

*The <article> Element

- The official specification for <article> states that it is “a self-contained composition in a page that is independently distributable or reusable, e.g. in syndication. This could be a forum post, a magazine or newspaper article, a blog entry, or any independent item of content.”
- The key concept in the above definition is that it is self-contained. The <article> element was designed for content that can be extracted from its containing page and still retain its full value.
- In HTML5, there can be multiple <article> elements on a web page. In fact, this is fairly common. For example, a typical blog has several different blog posts visible on the home page, with the most recent post at the top.

Let's now add an <article> element to our simple web page.

*Using the <article> Element

Here we have added a short article to the main content area of our page:

```
<article>
  <h2>Why This Website is Cooler
    Than Yours</h2>
  <p>While the rest of the world
    is stuck in the last decade,
    this site is built using
    HTML5.</p>
...
</article>
```

And just a little CSS styling:

```
article {
  margin: 15px;
  padding: 10px;
  background-color: #EEEEEE;
}
```

*The <aside> Element

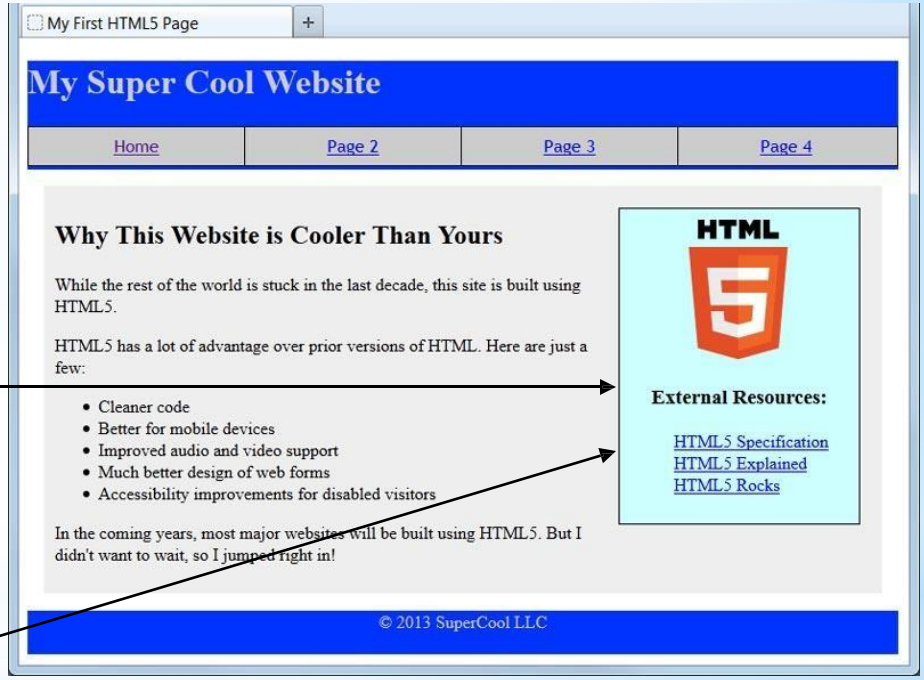
- The official specification for <aside> is “a section of a page that consists of content that is tangentially related to the content around it, and which could be considered separate from that content. Such sections are often represented as sidebars in printed typography.”
- It goes on to state that <aside> “can be used for effects like sidebars, for advertising, for groups of nav elements, and for other content that is considered separate from the main content of the page.”
- The most common uses of <aside> are for links to external websites, for advertising, and for "About Us" or "Contact Us" sidebars.
- An <aside> can be nested within an <article> if its content applies specifically to that article.

We'll now add an <aside> to our sample web page.

*Using the <aside> Element

We've added an <aside> box inside our article and inserted an image and external links:

```
<aside>
  <img src="images/html5-logo.png"
    alt="HTML5 Logo">
  <h3>External Resources:</h3>
  <ul>
    <li>
      <a href="http://dev.w3.org...">
      HTML5 Specification</a>
    </li>
...
</aside>
```



And used CSS to float the box to the right:

```
aside {
  width: 200px;
  float: right;
  background-color: #CCFFFF;
...
}
```

Don't confuse a list of external links as a <nav> section. The <nav> element is for navigating within our own site.

*The <section> Element

- The official specification for <section> is “generic section of a document. A section, in this context, is a thematic grouping of content, typically with a heading.”
- It continues to say that “a web site's home page could be split into sections for an introduction, news items, and contact information.”
- A common use of <section> is to contain a group of articles.
- The <section> element is an element of last resort, to be used only if the other semantic options (<header>, <footer>, <nav>, <article>, <aside>) are not appropriate.
- If the sole purpose is for styling, it is improper to use the <section> element, as there is no semantic value. Use the generic instead.

Let's add a <section> element to our existing page.

*Using the <section> Element

We've inserted a <section> element to contain our article:

```
<section>
  <h2>Home Page</h2>
  <article>
    <aside>
    ...
    </aside>
  ...
  </article>
</section>
```

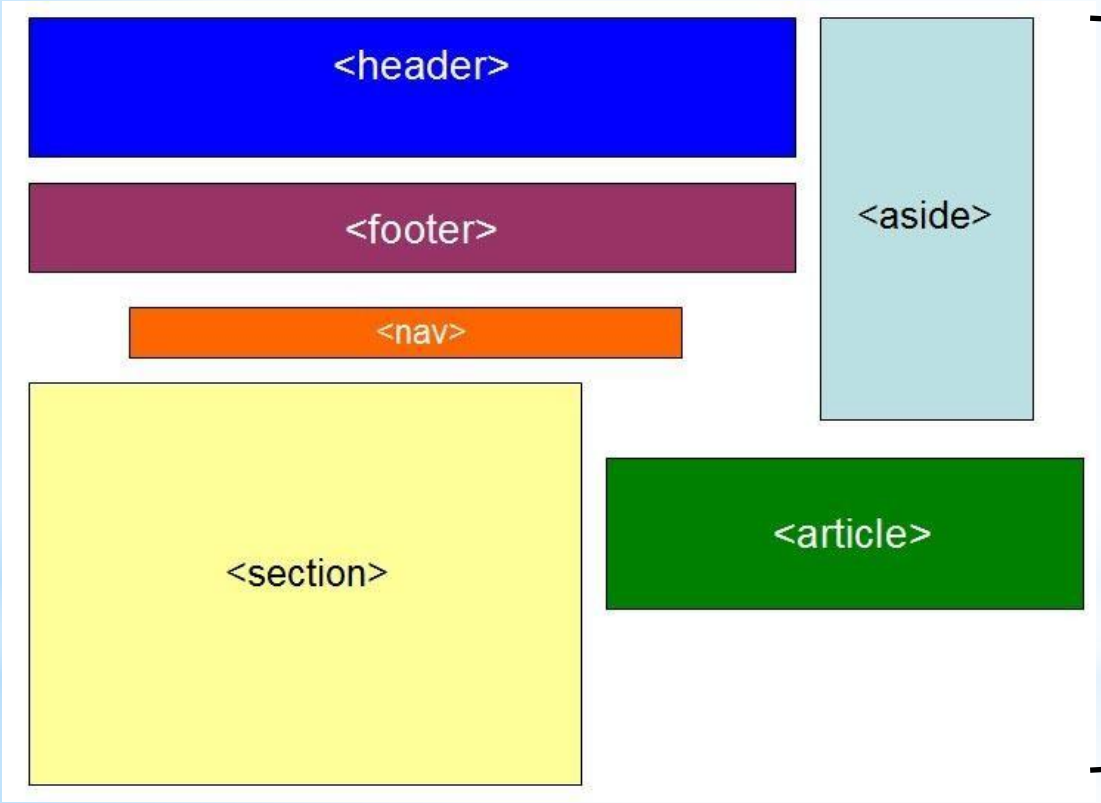
And CSS to change the background color:

```
section {
  background-color: #FFFFCC;
  padding: 5px;
}
```

We could now add more articles to be contained within the main <section>.

*Site Layouts Using HTML5

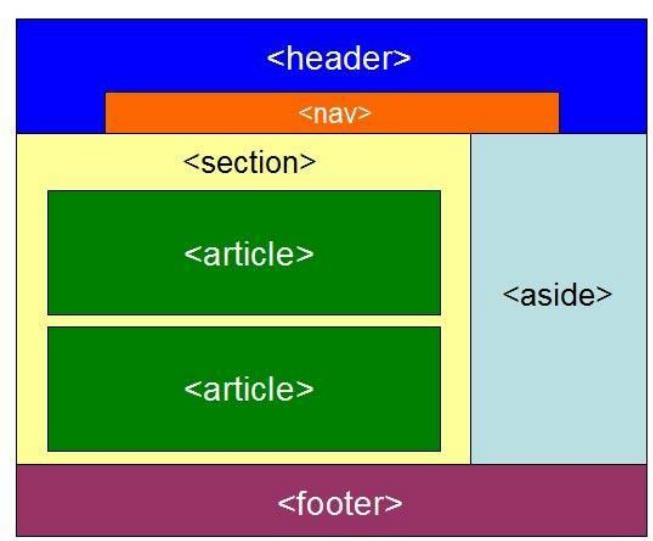
We have now covered all the basic building blocks for laying out HTML5-based websites:



Let's experiment now with these building blocks to explore different ways to create site layouts.

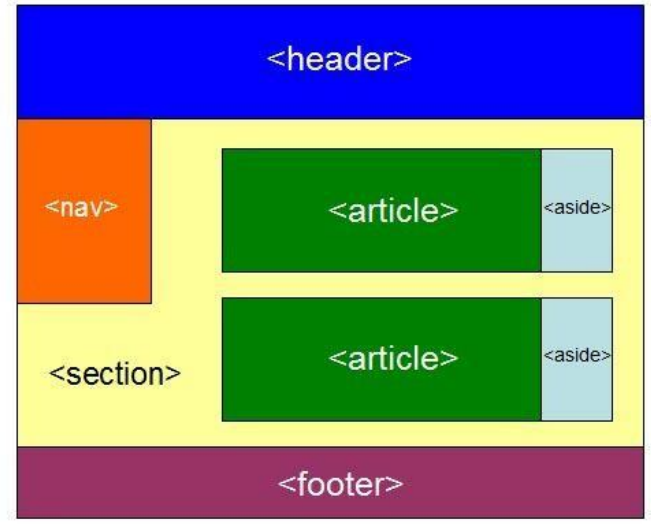
Don't forget that the element is still a common component when designing web page layouts in HTML5. For this lesson, though, we are focusing on the new semantic elements.

Example Site Layouts



This layout closely resembles the sample page we just created during this lesson.

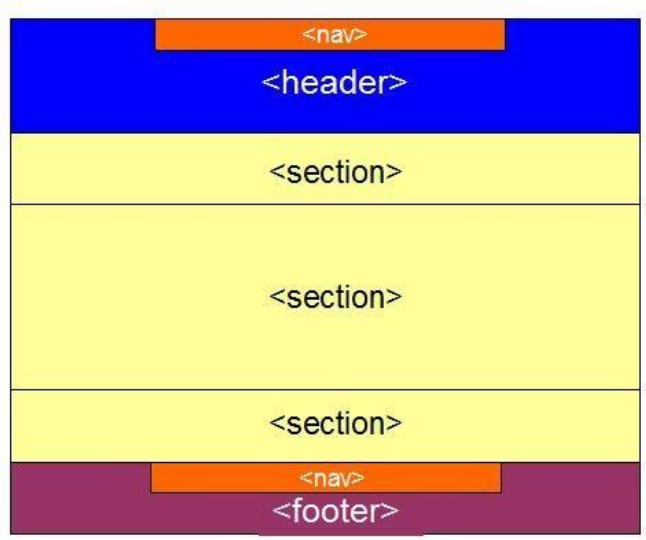
However, there is one difference. Note that the <aside> applies to the entirety of the main content, not to a specific article. This <aside> might contain "About Us" or "Contact Us" content, for example.



This layout is similar to the first, but moves the navigation menu to the left column.

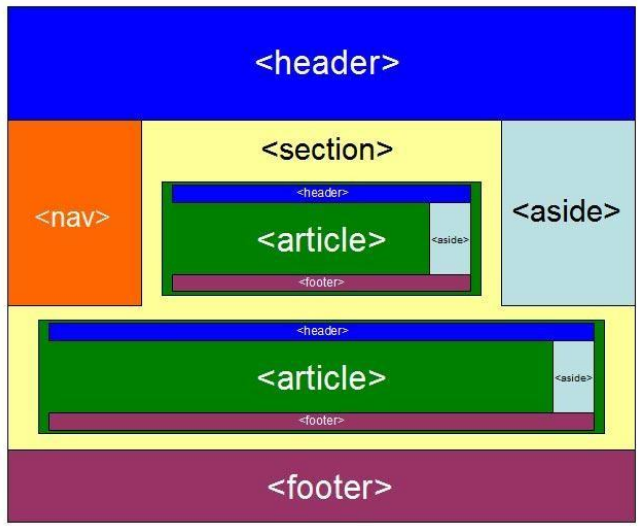
Also, each article has its own aside, which might contain links, advertisements, or a quote pulled from the main article.

More Site Layouts



This layout has two navigation menus, one at the very top of the page, and one in the footer section. This is not at all unusual, as many sites give visitors more than one way to click to other areas of the site.

Also, notice that this page contains no articles or asides. Instead, there are three <section> elements. These might be an introduction, a main content area, and a conclusion.



Some layouts can be much more complex and use multiple instances of elements.

Notice on this layout that each article has its own header, footer, and aside. This is a fairly common practice, as articles are designed to be syndicated to other sites, and the original creators of the article want to include such things as company logos, contact info, and copyright details.

*Semantic Elements in Site Layout

- There is no "right way" to create a website layout in HTML5. The best approach is to use the semantic elements, within their intended purposes, in the way that presents our page content most effectively to our audience.
- Any of the semantic elements - <header>, <footer>, <nav>, <article>, <aside>, and <section> - can be used multiple times within the same page. They can also be nested one within another.
- We can still use elements in HTML5 but only when they are being used solely for styling purposes or when there is no better choice among the semantic elements.

*Web Forms 2.0

*Semantic Elements in Site Layout

- Web Forms 2.0 is an extension to the forms features found in HTML4. Form elements and attributes in HTML5 provide a greater degree of semantic mark-up than HTML4 and free us from a great deal of tedious scripting and styling that was required in HTML4.

*Type attribute new values

- datetime
- datetime-local
- date
- month
- week
- time
- number
- range
- email
- url

*New attributes

New attributes for <form>:
autocomplete novalidate

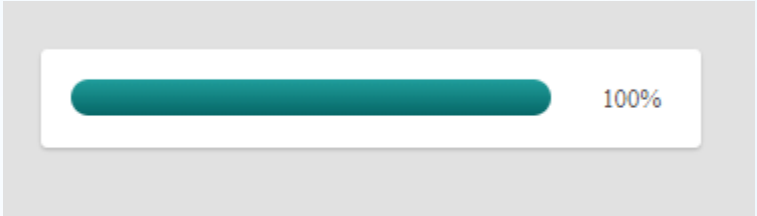
- placeholder
- autofocus
- required
- output tag
- min and max
- pattern (regexp)
- required
- step

* Representing scalar measurements or fractional values.
* Meter is also known as a gauge.
* It should not be used to indicate progress.
* Attributes:
  * 1) value 2) min
  * 3) max 4) high
  * 5) low 6) optimum

<meter value="0.6“ min=“” max=“” high ="0.6“ > Medium </meter>

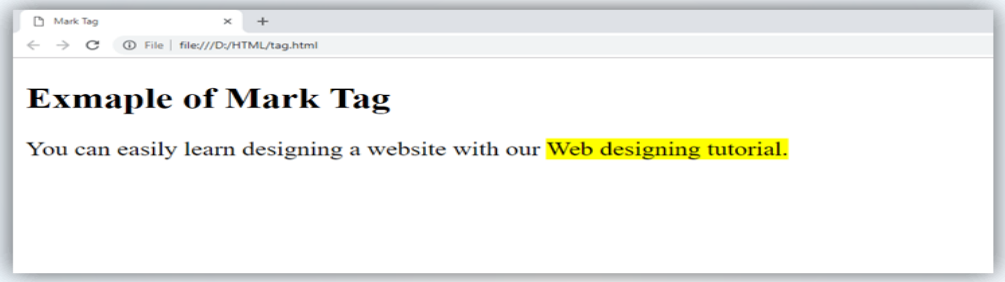
* Meter

* **<progress id=“pBar" max="100" value=“10"> </progress>**



* Progress

* Marked or Highlighted text
* Indicates point of interest or relevance
* Useful for:
  * Highlighting relevant code in a code sample
  * Highlighting search keywords in a document (e.g. in Google Cache)



* Mark