

Structure and Semantics

In this lesson, we will delve into the very interesting concept of HTML5 semantics. Let's begin!

WE'LL COVER THE FOLLOWING

- Listing-3-2: The skeleton of an article with HTML5
- The beauty of semantics in our code
- The beauty of semantics in search engines and web page ranks
- Semantic elements of page structure



HTML5 *and the Beauty of Semantics*



HTML5 changes this ancient plight we discussed earlier with the `<div>` tag significantly; it adds new semantic elements to the markup, which provide the missing meaning of the embedded content.

Exercise 3-2 shows the skeleton of the same page of the article as drafted in Exercise 3-1, but it uses new semantic elements.

Listing-3-2: The skeleton of an article with HTML5 #

```

<!DOCTYPE html>
<html>
<head>
  <title>New Style Article</title>
<style>
  body {
    width: 720px;
    margin-left: 16px;
    font-family: Verdana, Arial, sans-serif;
  }
  header {
    background-color: deepskyblue;
    padding: 2px 16px;
  }
  h1 {
    color: white;
  }
  .byLine {
    color: white;
    font-style: italic;
  }
  .mainContent {
    background-color: aliceblue;
    padding: 4px 16px;
  }
  .abstract {
    font-size: 0.9em;
    font-style: italic;
    color: navy;
  }
  h2 {
    color: navy;
    border-bottom: 4px dotted cornflowerblue;
  }
  footer {
    background-color: cornflowerblue;
    padding: 1px 16px;
  }
  footer>p {
    color: white;
    font-size: 0.8em;
  }
</style>
</head>
<body>
  <article>
    <header>
      <h1>Visual Studio Platform and Extensibility</h1>
      <p class="byLine">by Istvan Novak</p>
    </header>
    <div class="mainContent">
      <p>
        <span class="abstract">
          Now there is a rapidly growing community,
          the Visual Studio Ecosystem behind Visual Studio.
          With any idea, imaginings, issues, or questions,
          you can turn to other community members and

```

```

        leverage the knowledge they have. Welcome to
        the world of VSX!
    </span>

</p>
<section>
    <h2>Motivation</h2>
    <p>
        As a .NET architect and developer I cannot imagine
        my everyday work without Visual Studio. I was
        always in a strange excitement when waiting for
        a new CTP, Beta or RTM of Visual Studio because
        I always expected some great new features with
        every release. During the years I have bought a
        few third-party add-ins and utilities for Visual
        Studio to make my development tasks easier and
        even created small add-ins to produce some useful
        piece of code. I knew that Visual Studio was
        extensible; I downloaded the SDKs and tried to
        get familiar with those hundreds of extensibility
        interfaces. However, due to lack of good
        documentation I often got frustrated.
    </p>
</section>
<section>
    <h2>Visual Studio: Extensible Platform</h2>
    <p>
        Hearing the word &quot;extensibility&quot;
        a developer always starts to think &quot;
        some API allowing extra functionality to be
        added to a product&quot;. More or less,
        I can agree with this definition in the case
        of Visual Studio. In this part I tell you how
        I understand Visual Studio extensibility, and
        what are the ways we can add functionality. Because
        I started learning this story with the release of
        Visual Studio 2008, I put the focus on that version.
    </p>
</section>
<section>
    <h2>Summary</h2>
    <p>
        Pellentesque porttitor, libero eu mattis pulvinar,
        urna dui scelerisque sapien, vitae varius sapien
        urna at odio. Fusce vel neque non massa varius
        mattis ut nec.
    </p>
</section>
</div>
</article>
<footer>
    <p>
        Full article published in CODE Magazine
        in April, 2008.</p>
</footer>
</body>
</html>

```

Without knowing anything about these new elements, you can intuitively understand the structure of the document: it contains an article that is composed from a `<header>` tag and a number of `<section>` tags; it also has a `<footer>` tag.

Comparing the two code listings (the old-school style with HTML5), you can easily state that the second one (HTML5 style) is easier to read, and is easier to maintain.

The beauty of semantics in search engines and web page ranks

If you think about how a **search engine** works, you can imagine that **this information helps a lot to separate the more important information in this page from the less important**. For example, an occurrence of a certain word in the *header* may have a *higher rank* than the same word in a section, or in a footer.

Semantic elements may empower tools that analyze or change the page on-the-fly. For example, you can generate an automatic navigation bar for the page that displays the structure of sections. The great thing is, that you can even create a library that prepares this structure and automatically attaches it to a page. Third parties can load your library and generate the content structure for their own pages.


HTML5 defines nine semantic elements that behave exactly like the good old `<div>` tag but add some extra meaning to your pages.

These elements are summarized in the table below:

Semantic elements of page structure

Element	Description
<code><article></code>	This element defines an independent, self-contained content, such as a blog entry, a newspaper article, a forum post, a CV, an author biography, a story, etc.—anything that you think of as an article.
<code><aside></code>	This element defines content that is separate from the other (surrounding) content of the page—aside from the content it is placed in. It is frequently used to create sidebars related to an article.
<code><figure></code>	Represents a figure that is—in contrast to traditional images—a self-contained content, such as an illustration, diagram, photo, etc. The <code><figure></code> element is a wrapper for this content, including the <code></code> for the figure, as well as the caption nested into a <code><figcaption></code> element. The aim is to indicate the relation between the image and its associated caption.
<code><figcaption></code>	This element defines a caption for the <code><figure></code> element.
<code><footer></code>	Defines a footer for a document or section, so this element should contain information about its container element. It can be a set of important links, a copyright notice, terms of use, contact information, etc.
<code><header></code>	This element represents an enhanced heading for a document or a section. It should be a container for introductory content, and may contain logo, byline, set of navigational links, etc.
<code><hgroup></code>	This element defines an enhanced heading that groups two or more heading elements without any additional content. Its purpose is to make a title and a subtitle (or subtitles) stand together.
<code><nav></code>	Defines a major block of links on a page. These links may point to topics on the current page, or to other pages on the website. Not all links of a document must be in a <code><nav></code> element. A page may have multiple <code><nav></code> sections.
<code><section></code>	This element defines logical sections in a document. These sections can be headers, footers, chapters, sections of chapters, etc. Use <code><section></code> only if other semantic elements do not apply. As a rule of thumb, the content <code><section></code> holds always should begin with a heading (<code><h1></code> , ..., <code><h6></code>)

Semantic elements of page structure

 **NOTE:** There are a few more semantic tags that are related to text elements. For example, the `<mark>` and `<wbr>` tags treated in [Chapter 3](#) are such ones. As you already learned, those are inline elements and are not the semantic extensions of `<div>` by any means.

These elements behave exactly like `<div>`. They only enclose a markup block and provide a logical container that can be used to apply formatting (through styling), or to add behavior by means of a script language (JavaScript).

It's time to see these semantic elements in action in the *next lesson*!