Old Style: The Omnipotent

In this lesson, we'll see the old way to define logical blocks and section divisions in a webpage.
Let's begin!

WE'LL COVER THE FOLLOWING

It’s amazing how much the web changed in the last decade!

In contrast to this, it’s surprising how prehistoric HTML remained for a long time, how verbose and ambiguous markup was created by developers in response to new challenges.

Undoubtedly, the reason for this ancient style was the semantic weakness of HTML. With the emergence of CSS, the conciseness of webpages has improved by styling attributes and you can leave the content and moved to the style section of the page, but the HTML content of complex pages stayed messier by the sheer page structure.

Olden times: `<div>` to define division sections in webpages

The ultimate weapon of web designers was the `<div>` tag, which defines a **division** or **section** in HTML. If they had any part of the content that constituted a logical block with their own visual properties, they used the following pattern:

```
<!-- … -->
<head>
  <title>Sample</title>
  <style>
    #myBlock {
      <!-- Style description of the block -->
    }
  </style>
</head>
<body>
  <div id="myBlock">
    <!-- Content of the block -->
  </div>
</body>
```

 tag in action: defining logical blocks in webpages

This pattern assigns an identifier (myBlock) to the section, and a style (here #myBlock) that describes the appearance of the `<div>`.

It's simple, but the `<div>` itself does not help to understand the meaning of the section, nor does it tell the intention of the designer.

> 📜 **NOTE:** In Chapter 2, you already met exercises using the `<div>` tag.

The problem with the `<div>` tag

While there are only a few `<div>` tags in our code, those may be easily understandable and maintainable.

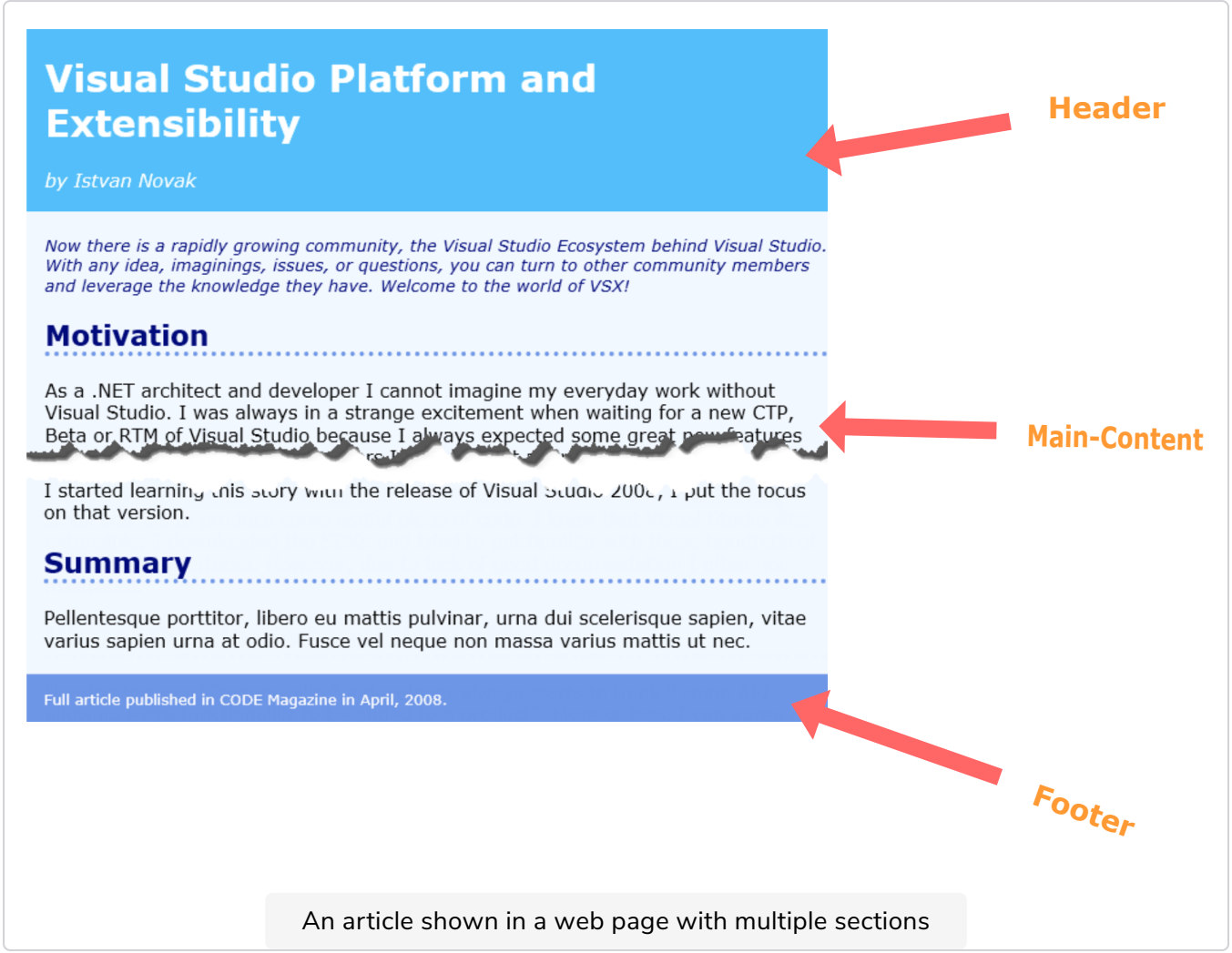
However, as the size of the markup grows, designers and developers feel lost in the complexity of the page.

It's not easy to find whether a `<div>` is a header, a section of the main document, a sidebar, a part of the navigation structure, or something else. It's because `<div>` has poor semantics as it's just a generic block.

> *** is just a generic block with poor semantic***

For example, the image below shows an article in a web page.

The structure of the article is typical: it contains a *header*, the *main content*, and a *footer*; the main content is divided further into sections.



An article shown in a web page with multiple sections

According to the listings and exercises you already went through, you can easily draft the skeleton of such a webpage. One example is shown in Listing 3-1 below:

Listing-03-01: Defining the `<div>` tag in a webpage

```
<html>
<head>
    <title>Old 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;
    }
    div.footer>p {
      color: white;
      font-size: 0.8em;
   }
  </style>
</head>
<body>
  <div class="header">
    <h1>Visual Studio Platform and Extensibility</h1>
    <p class="byLine">by Istvan Novak</p>
  </div>
  <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>
    <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>
    <h2>Visual Studio: Extensible Platform</h2>
    <p>
```

```
      Hearing the word &amp;quot;extensibility&amp;quot;
      a developer always starts to think &amp;quot;
      some API allowing extra functionality to be
      added to a product&amp;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>
    <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>
  </div>
  <div class="footer">
    <p>
      Full article published in CODE Magazine
      in April, 2008.</p>
  </div>
</body>
</html>
```

Although this skeleton uses `<div>` tags to define document sections, it's pretty easy to find the header (**line 45**), footer (**line 98**), and the main content (**line 49**) due to deliberate naming conventions. The class attributes adorning the `<div>` tags exactly tell the roles of the corresponding sections.

It's easy for a human reader, but what about search engines and other tools that analyze the structure of the page?

If I used "fejléc", "lábléc", and "tartalom" for class names instead of "header", "footer", and "mainContent", would a human reader know their role? Not unless they speak Hungarian!

> ***It's a big issue with the good old `<div>` tag that it misses any extra meaning that would help in analyzing and understanding the role of page sections.***

In the *next lesson*, we'll study the structure and semantics of an HTML page.