

CURRICULUM LAYOUT

Concept 1, Introduction to HTML and CSS, introduces you to the two foundation technologies of the web - HTML and CSS. We'll go over the syntax of both and look at how they are combined to make a web page.

Concept 2, Structure and Layout, introduces you to the structural elements in HTML, including header, footer, and section tags. You will also learn the three main CSS layout techniques - float, flex, and grid.

Before You Begin

Each great journey begins with a humble step. Our upcoming adventure in the land of HTML and CSS is no exception. Before we can do awesome things with data, we need to be prepared with a productive environment. In this section, we shall see how to do that.

Installing Visual Studio Code Here are the steps to install Visual Studio Code (VSCode):

1. Download the latest VSCode from <https://code.visualstudio.com/>

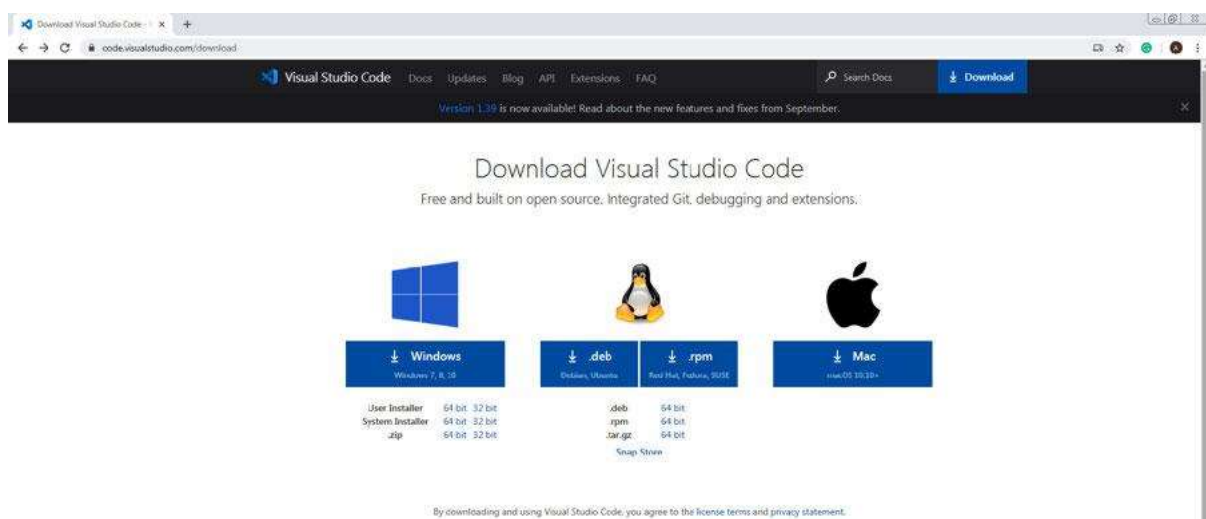


Figure 0.1: Downloading VSCode

2. Open the downloaded file, follow the installation steps, and complete the installation process.

Installing the "Open in Default Browser" Extension

1. Open your VSCode, click on the Extensions icon, and type in Open In Default Browser in the search bar, as shown in the following screenshot:

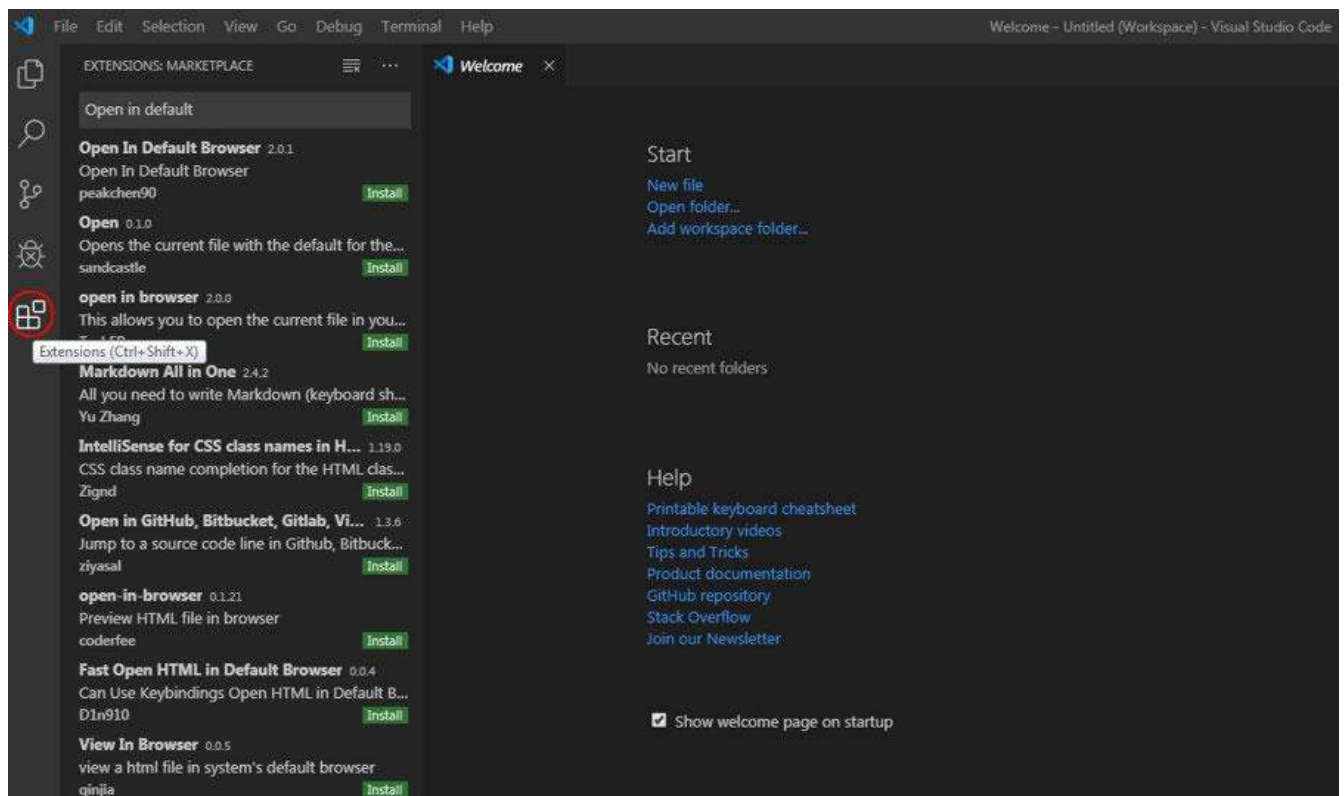


Figure 0.2: Open in Default Browser extension search

2. Click on Install to complete the installation process, as shown in the following screenshot:

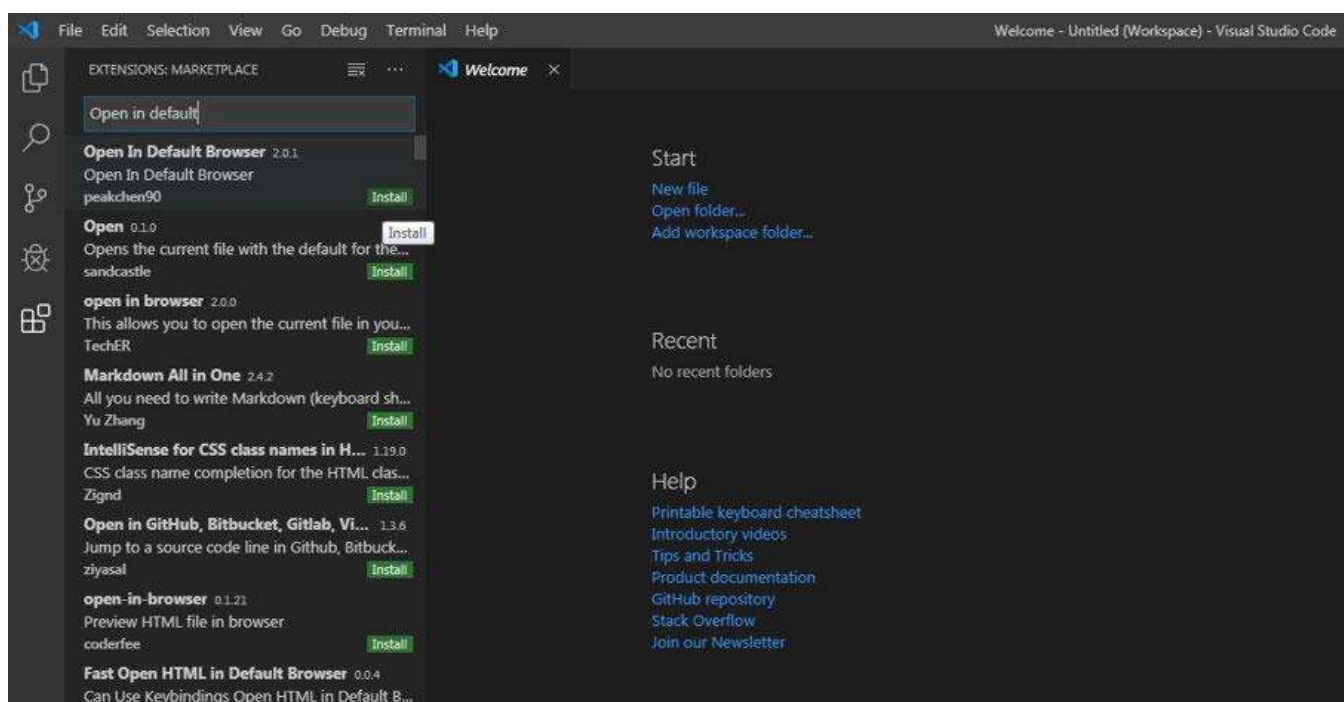


Figure 0.3: Installing the extension

1.Introduction to HTML and CSS *Overview*

By the end of this chapter, you will be able to describe the roles of HTML and CSS in a web page; explain the HTML DOM and CSSOM; explain how a web page renders; create a web page from scratch; and use CSS selectors and calculate CSS specificity. This chapter introduces two core technologies of the web - HTML and CSS. We will look at how they work, how we can write them, and how we can use them to build web pages.

HTML5

HyperText Markup Language (HTML) is a markup language used to describe the structure of a web page. Consider a snippet of text with no markup:

HTML HyperText Markup Language (HTML) is a markup language used to describe the structure of a web page.

Syntax

The syntax of HTML is made up of **tags** (with angle brackets, `<>`) and **attributes**. HTML provides a set of tags that can be used to mark the beginning and end of a bit of content.

The opening tag, closing tag, and all content within those bounds represent an HTML element. The following figure shows the HTML element representation

without attributes:

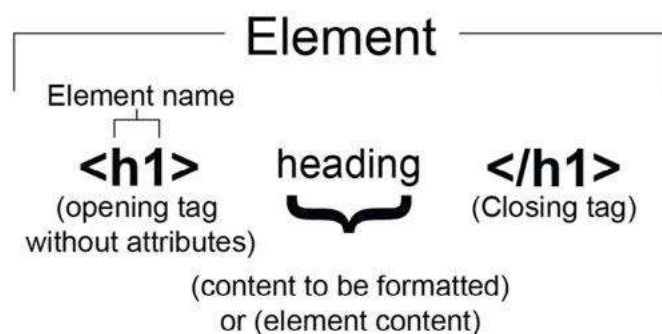


Figure 1.3: HTML element representation without tag attributes The following figure shows the HTML element representation with tag attributes:

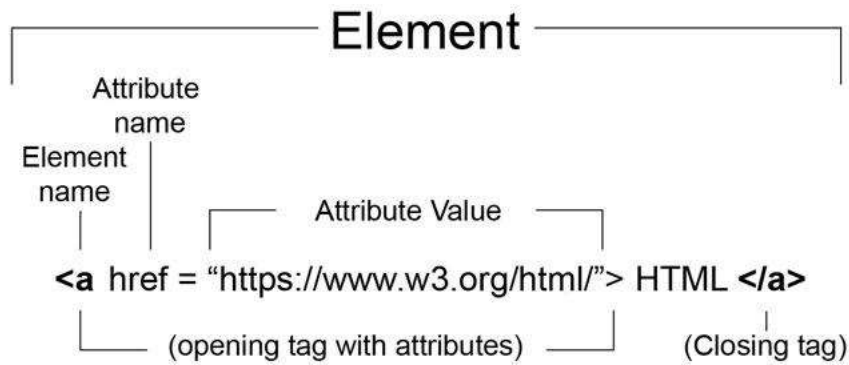


Figure 1.4: HTML element representation with tag attributes

A tag has a name (for instance, **p**, **img**, **h1**, **h2**, **h3**, **br**, or **hr**) and that name combined with attributes will describe how the browser should handle the content.

Many tags have a start and an end tag with some content in between, but there are also tags that don't expect any content, and these can be self-closing.

A **comment** begins with `<!--` and ends with `-->`.

COMMON TAGS

- `<h[1-6]>` Heading
- `<div>` Page section
- `` Inline section
- `<p>` Paragraph
- `
` Line break

The HTML Document

An HTML document represents a hierarchical tree structure, rather like a family tree. Starting from a root element, the relationship between an element and its contents can be seen as that of a parent element and a child element.

An element that is at the same level of the hierarchy as another element can be considered a sibling to that element, and we can describe elements within a branch of the tree as ancestors and descendants. This structure can be represented as a tree diagram to get a better idea of the relationship between elements.

Take, for example, this simple HTML document:

```
<html>

<head>

<title>HTML Document structure</title>

</head>

<body>

  <div>

    <h1>Heading</h1>

    <p>First paragraph of text.</p>

    <p>Second paragraph of text.</p>

  </div>

</body>

</html>
```

Here, we have an HTML element, the root of the document, which hosts a head element containing a title element, and a body element, containing some content including a div element with an h1 heading element and some paragraph elements.

It can be represented as a tree diagram as follows:

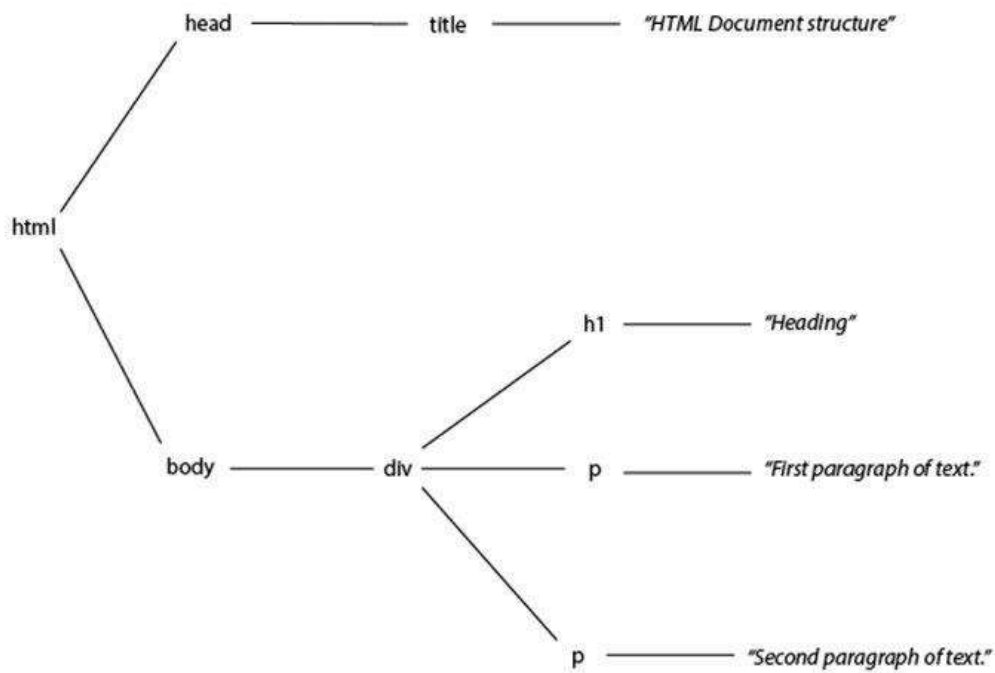


Figure 1.8: A representation of the HTML document as a tree diagram In the browser, this code would render the following web page:

```
<!DOCTYPE html>
```

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

```
<head>
```

```
<meta charset="UTF-8">
```

```
<meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>Document</title>
```

```
</head>
```

```
<body>
```

```
</body>
```

```
</html>
```

CSS

Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation of a web page. The language is designed to separate concerns. It allows the design, layout, and presentation of a web page to be defined separately from content semantics and structure. This separation helps keep source code readable and it is important because a designer can update styles separately from a developer who is creating the page structure or a web editor who is changing content on a page.

A set of CSS rules in a style sheet determines how an HTML document is displayed to the user. It can determine whether elements in the document are rendered at all, whether they appear in some context but not others, how they are laid out on the web page, whether they are rendered in a different order to the order in which they appear within a document, and their aesthetic appearance. We will begin by looking at the syntax of CSS.

Syntax

A CSS declaration is made of two parts:

a property and a value.

The property is the name for some aspect of style you want to change; the value is what you want to set it to.

Here is an example of a CSS declaration:

color: red;

The property is color and the value is red. In CSS, color is the property name for the foreground color value of an element. That essentially means the color of the text and any text decoration (such as underline or strikethrough). It also sets a current color value. For this declaration to have any effect on an HTML document, it must be applied to one or more elements in the document. We do this with a selector. For example, you can select all the `<p>` elements in a web page with the `p` selector. So, if you wanted to make the color of all text in all paragraph elements red, you would use the following CSS ruleset:

`p`


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

The result of this CSS ruleset applied to an HTML document can be seen in the following figure:

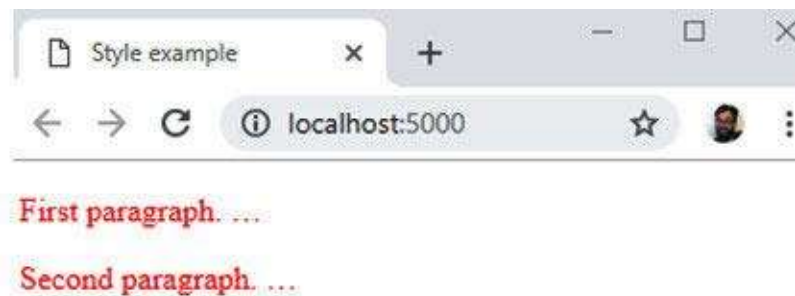


Figure 1.18: Result of a CSS rule applied to <p> elements in HTML

The curly braces represent a declaration block and that means more than one CSS declaration can be added to this block.

Multiple selectors can share a CSS ruleset. We can target these with a comma-separated list. For example, to apply the color red to p elements, h1 elements, and h2 elements, we could use the following ruleset:

p, h1, h2 { color: red; } Multiple CSS rulesets form a style sheet. The order of these CSS rules in a style sheet is very important as this is partly how the cascade or specificity of a rule is determined. A more specific rule will be ranked higher than a less specific rule and a higher-ranked rule will be the style shown to the end user. We will look at cascade and specificity later in this session:

CSS Rule Set

Selector or Selector List

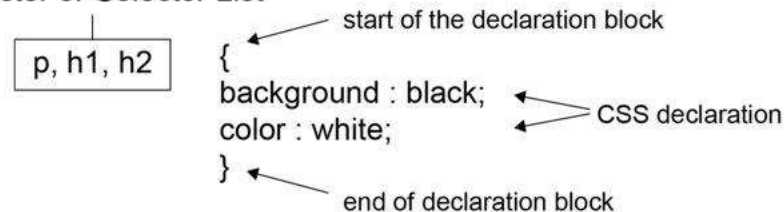


Figure 1.20: A CSS ruleset explained

Element, ID, and Class

Three commonly used selectors are:

- **Element type:** For example, to select all p elements in an HTML document, we use the p selector in a CSS ruleset. Other examples are **h1, ul, and div**.
- **A class attribute:** The class selector starts with a **dot**. For example, given the HTML snippet
`<h1 class="heading"> Heading</h1>`,
you could target that element with the `.heading` selector.
- **An id attribute:** The id selector starts with a **hash symbol**. For example, given the HTML snippet

```
<div id="login">  
<!-- login content -->  
</div>
```

you could target this element with the `#login` selector.

The Universal Selector (*)

To select all elements throughout an HTML document, you can use the universal selector, which is the asterisk symbol (*). Here is an example snippet of CSS that is often added to web pages; a value is set on the html element and then inherited by all descendant elements:

html

```
{ box-sizing: border-box; }
```

Dev Tools

Most browsers come with some tools to help web developers create and change web pages. One of the most useful of these tools is the built-in developer tools that come with the Chrome browser. You can access the developer tools on any web page with the **Command + Option + I** keyboard shortcut (on Mac) and **F12 or Control + Shift + I** (on Linux and Windows). On opening the developer tools, you should see something similar to the following figure:

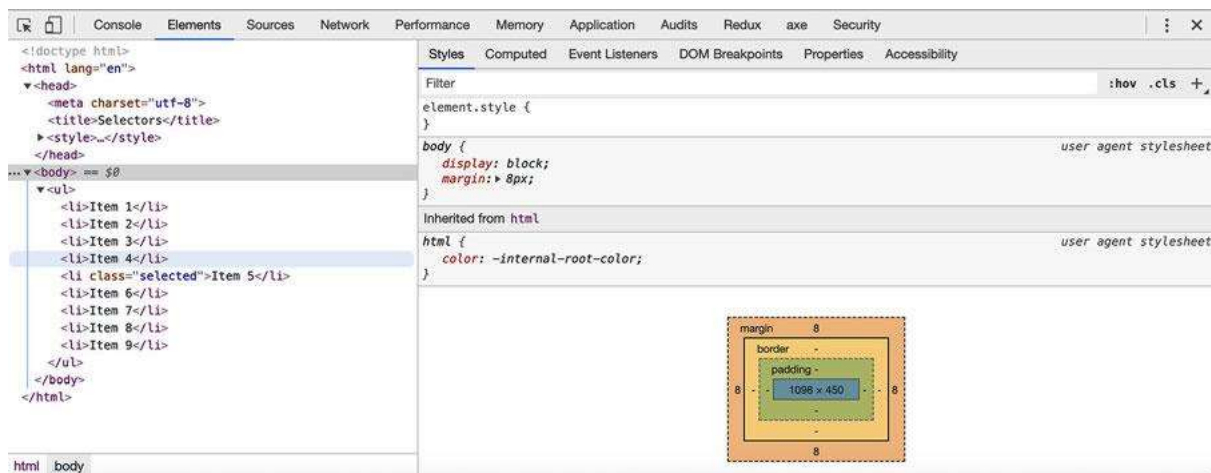


Figure 1.33: Chrome developer tools (Elements tab)

There are a lot of options available in the developer tools. We will not cover all of them here but will focus on the top bar and the Elements tab in this session.

The Top Bar The top bar gives you access to several options and tools, including access to all the other sections of the developer tools via the tabs:





Figure 1.34: Chrome DevTools top bar The top bar has the following tools and options:

Select tool - You can use this tool to select an element from within the web

page.  **Devices toolbar** - Changes the view so you can select the view size of various devices.

Tabs - We can access various tools from the top bar menu such as

Console, Elements, Sources, Network, and Performance. We will focus on the Elements tab.  **Configuration** - Gives you access to various settings for the developer tools.

 **Close** - Closes the developer tools. While developing the HTML and CSS for a web page, one of the tabs we use the most is the Elements tab.

2. Structure and Layout Overview

By the end of this session you will be able to use the correct HTML5 elements to markup a web page; style a web page using float, flex, and grid layouts; describe how the box model works; and build a home page and a product page layout. This chapter introduces the essential HTML elements that are required in order to build a web page. Finally, this knowledge will be utilized by carrying out a number of exercises to create a few well-structured web pages.

Structural Elements

HTML5 provides us with a variety of tags that we can use when dividing our page into different parts. When browsing the web, you would have noticed that web pages typically have a few common things to them.

For example, a web page will typically have a logo and page navigation area at the top of the page. We would call this area of the page the header. You may also have noticed that the bottom of the page may include a list of links and copyright information. We would call this area the footer. The following diagram shows the representation of a few of the main elements of a web

page:

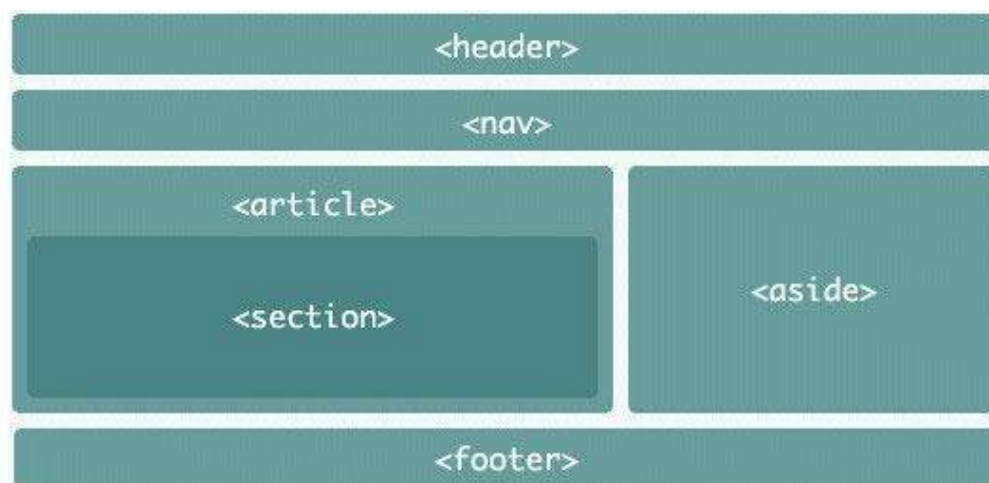


Figure 2.1: HTML5 page elements In this topic, we will be looking at the following HTML5 page elements:

- header
- footer
- section
- article
- nav
- aside
- div

Before we begin ,We need to understand some concepts:

- **Ids and Classes and Div**
- **Box Model**
- **Inline and Block elements**
- **Positioning**

IDs and Classes in HTML

The **ID** is an identifier which must be unique in the whole HTML document. It is used to find an element while linking, scripting, or styling. Whereas, **Classes** allow CSS and JavaScript to select and access specific elements. Let us start by making a new file as *idsandclasses.html* and adding an instant boilerplate to get the basic HTML template. Name the <title> tag as **Ids and Classes in HTML** to give the title of the website.

When a new child is born, we urge to give him a name or his identity by which he will be known further. Or if you are having a pet, you must have given him some name to call. In the same way, IDs refer to giving a name to any particular element for its identity. It simply refers giving an identity to an element. We know, no two names can be given to any of the two members of the family. In the same way, one ID can be given to only one element on a website. Therefore, in the below example, the id **mainBox** cannot be given to any other element.

```
<div id= "mainBox" class= "redBG">
```

Now the question arises what is the need for an ID in HTML? The answer is, while using JavaScript or CSS, we can target one full element and can make the necessary changes in it. In the same way, we can grab the full element and change the border or width or many more things through CSS.

Let us now understand what are classes with an example. Assume that I am having 100 elements in my HTML and I want to give a red background to all the 100 elements. To do this, we have two options. Either we have to select each element and assign a red background to it or we can create a class **redBG** and assign a red background to it. Then we can give this class to the elements in which we want a red background color. To avoid confusion, I am assuming that the class redBG is already defined.

One point to note here is we can assign only one ID to a particular element but it is not so in the case of classes. An element can have more than one class in itself. The more classes we add in an element, the more property will get added to it.

Classes are denoted by a dot `'.'` and ID is denoted by hash `'#'`. For example, to get a redBG class in an element we can simply write that element name followed by `.redBG`.

CSS Box Model, Margin and Padding

The box model is a very important topic of CSS and if not understood properly, it can create a lot of confusion in the future. It is the basic framework of web development whether you are making a website using any other language such as Angular or React. The box model helps us to define the **padding**, **border**, and **margin** around an element. So from the below diagram we can see where all these things lie around the element. The element is in the center surrounded by padding, border and margin.

These parts can be explained as-

- **Content-** The content of the box, where text and images appear.
- **Padding-** It clears an area around the content. The padding is transparent.
- **Border-** A border is one that covers the padding and content.
- **Margin-** It clears an area outside the border. The margin is also transparent.

