Web Engineering Lab

Lab 05

HTML Layout Tags
Responsive Web Design

Instructor: Hurmat Hidayat
Course Code: SL3003
Semester Spring 2023

Department of Computer Science,

National University of Computer and Emerging

Sciences Peshawar Campus

Lab Content

HTML5 Layout Tags

A web page being rendered in the browser consists of many things – logo, informative text, pictures, hyperlinks, navigational structure and more. HTML5 offers a set of markup elements that allow you to create a structured layout for web pages. These elements are often termed as Semantic Markup because they convey their meaning and purpose clearly to the developer and to the browser. Some of the important HTML5 elements that can contribute to the layout of a web page will be discussed here.

Semantic Markup

Semantic markup expresses its meaning and purpose clearly to the developers and to the browser. Web developers frequently use the element to layout their web pages. However, a element by itself doesn't convey what it is representing in a web page. A element may wrap a navigational menu or it may house a list of blog posts, but merely using doesn't convey much information to the developers or to the browsers. Usually CSS classes applied to elements reveal some information about their intended purpose. For example consider the following markup:

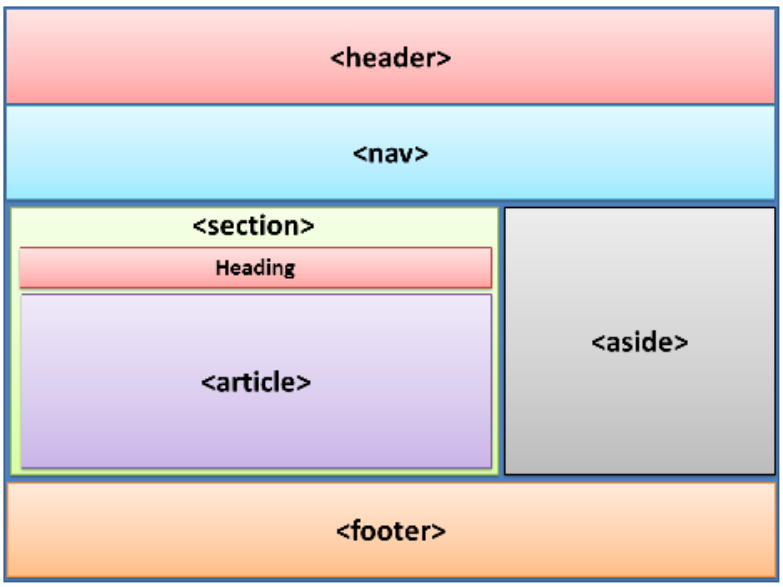
```
<div class="header">...</div>
```

In the above markup it is the header CSS class that gives you an idea as to what the might be doing. In the absence of this CSS class, however, there is no way to easily identify what the might be doing.

HTML5 includes a set of markup elements that overcome this difficulty. These new elements have meaningful names so that just by looking at these elements you get a clear idea about their content. These semantic elements of HTML5 are listed below:

- <header>
- <footer>
- <section>
- <article>
- <aside>
- <nav>

The following figure shows a sample page layout designed using these elements:



The above figure shows a typical arrangement of various elements. Note that their exact location on a page is purely dependent on the layout. For example, the <aside> element can be placed on the left hand side of the <section> or even above or below it.

Header Element

The <header> element can contain headings, logo images, supporting text and optionally a navigation structure. Most of the web pages have page headings in the form of logo, slogan and/or supportive text. The header element acts as a container for all these. Note that the header element can be used not only for the whole page but also for individual sections of a web page. For example, in addition to the page level header you may use the header element in the contact information section of the page.

```
<header>
  <h1>This is page heading</h1>
</header>
```

Nav Element

The <nav> section can contain links to the other pages from the website or to other parts of the same web page. It is recommended that you use <nav> only for the main navigational structures and not for minor set of hyperlinks (such as the ones that usually go in the footer of a web page).

```
<nav>
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">About Us</a></li>
    <li><a href="#">Contact Us</a></li>
  </ul>
</nav>
```

Section Element

The <section> element should not be confused with the element. The <section> element is a thematic grouping of content whereas the doesn't have any such restriction. The W3C specifications further clarifies – The section element is appropriate only if the contents would be listed explicitly in the document's outline. Some examples where a <section> element can be used include contact information section of a web page, announcement section of a web page and tabbed pages of a tabbed user interface. A section usually has some heading.

```
<section>
  <h1>This is a section heading</h1>
  <p>
    Hello world! Hello world! Hello world!
    Hello world! Hello world! Hello world!
    Hello world! Hello world! Hello world!
  </p>
</section>
```

Article Element

The <article> element represents an independent item section of content such as a blog post, forum post or a comment. It is expected that the content in the article element should be independently distributable or reusable as in the case of content syndication.

```
<article>
  <h1>This is article heading</h1>
  <p>
    Hello world! Hello world! Hello world!
    Hello world! Hello world! Hello world!
    Hello world! Hello world! Hello world!
  </p>
</article>
```

Aside Element

The <aside> element is intended to house content that is related to the surrounding content but at the same time is a standalone piece of content in itself. If you take out the <aside> from the page it shouldn't change or alter the meaning or clarity of the main page content. Think of it as a sidebar that gives some extra, related yet standalone information about the topic being discussed. Some examples of <aside> include – extra information, related links and contextual advertisements. The <aside> element can house any type of content including textual information and images. For example, the <aside> used in the example web page looks like this:

```
<aside>
  <figure>
    <img src="images/laptop.png" height="100px" width="100px" />
    <figcaption>Figure caption goes here</figcaption>
  </figure>
  <p>
    Hello world! Hello world! Hello world!
    Hello world! Hello world! Hello world!
  </p>
</aside>
```

Footer Element

The <footer> element represents the footer of the whole page or a <section> element. It is intended to contain footer information such as copyright notice. The <footer> element of the example web page looks like this:

```
<footer>
  <hr />
  Copyright (C) 2013. All rights reserved.
</footer>
```

Example

~/Desktop/Spring2023/WebEngineering/S23_WELab_05/htmlFiveStruct.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>Sample HTML5 Layout</title>
    <link href="StyleSheet.css" rel="stylesheet" />
</head>
<body>
    <header>
        <h1>This is page heading</h1>
    </header>
    <nav>
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">About Us</a></li>
            <li><a href="#">Contact Us</a></li>
        </ul>
    </nav>
    <article>
        <h1>This is article heading</h1>
        <p>
            Hello world! Hello world! Hello world!
            Hello world! Hello world! Hello world!
            Hello world! Hello world! Hello world!
        </p>
    </article>
    <aside>
        <figure>
            <img src="images/laptop.png" height="100px" width="100px" />
            <figcaption>Figure caption goes here</figcaption>
        </figure>
        <p>
            Hello world! Hello world! Hello world!
            Hello world! Hello world! Hello world!
        </p>
    </aside>
    <section>
        <h1>This is a section heading</h1>
        <p>
            Hello world! Hello world! Hello world!
            Hello world! Hello world! Hello world!
            Hello world! Hello world! Hello world!
        </p>
    </section>
    <footer>
        <hr />
        Copyright (C) 2013. All rights reserved.
    </footer>
</body>
</html>
```

~/Desktop/Spring2023/WebEngineering/S23_WELab_05/StyleSheet.css

```css
article
{
    padding:5px;
    border:dotted 3px #ff006e;
    margin-top:5px;
}

header
{
    padding:0px;
    text-align:center;
}

aside
{
    margin-top:5px;
    background-color:#f0eaea;
    padding:5px;
    text-align:center;
    font-style:italic;
    border:double 3px #b200ff;
}

section
{
    padding:5px;
    border:dashed 3px #0026ff;
    margin-top:5px;
}

footer
{
    padding:5px;
    text-align:center;
    font-weight:bold;
}

nav
{
    text-align:center;
}
ul li
{
    display:inline;
    padding-left:5px;
    padding-right:5px;
    text-align:left;
    font-size:16px;
    font-weight:bold;
}
```

This is page heading

Home About Us Contact Us

This is article heading

Hello world! Hello world! Hello world! Hello world! Hello world! Hello world! Hello world! Hello world! Hello world!

Figure caption goes here

Hello world! Hello world! Hello world! Hello world! Hello world! Hello world!

This is a section heading

Hello world! Hello world! Hello world! Hello world! Hello world! Hello world! Hello world! Hello world! Hello world!

Copyright (C) 2013. All rights reserved.

Responsive Web Design

In responsive web design, the layout and content of the website automatically adjust and adapt to the user's screen size, without the need for a separate mobile site or app. This is achieved through the use of flexible grids, images, and CSS media queries, which enable the site to detect and respond to changes in the screen size and device orientation.

The key benefits of responsive web design include improved accessibility, increased mobile traffic and engagement, and reduced development and maintenance costs, as there is only one site to build and maintain instead of multiple sites for different devices.

Figure 1: Desktop, tablet, mobile

It is called responsive web design when you use CSS and HTML to resize, hide, shrink, enlarge, or move the content to make it look good on any screen.

The Viewport

The viewport is the user's visible area of a web page. The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen. HTML5 introduced a method to let web designers take control over the viewport, through the <meta> tag. You should include the following <meta> viewport element in all your web pages:

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

This gives the browser instructions on how to control the page's dimensions and scaling.

The **width=device-width** part sets the width of the page to follow the screen-width of the device (which will vary depending on the device). The **initial-scale=1.0** part sets the initial zoom level when the page is first loaded by the browser. Here is an example of a web page *without* the viewport meta tag, and the same web page *with* the viewport meta tag:

Without the viewport meta tag **With the viewport meta tag**

Size Content to The Viewport

Users are used to scroll websites vertically on both desktop and mobile devices - but not horizontally!

So, if the user is forced to scroll horizontally, or zoom out, to see the whole web page it results in a poor user experience. Some additional rules to follow:

1. **Do NOT use large fixed width elements -** For example, if an image is displayed at a width wider than the viewport it can cause the viewport to scroll horizontally. Remember to adjust this content to fit within the width of the viewport.

2. **Do NOT let the content rely on a particular viewport width to render well** - Since screen dimensions and width in CSS pixels vary widely between devices, content should not rely on a particular viewport width to render well.

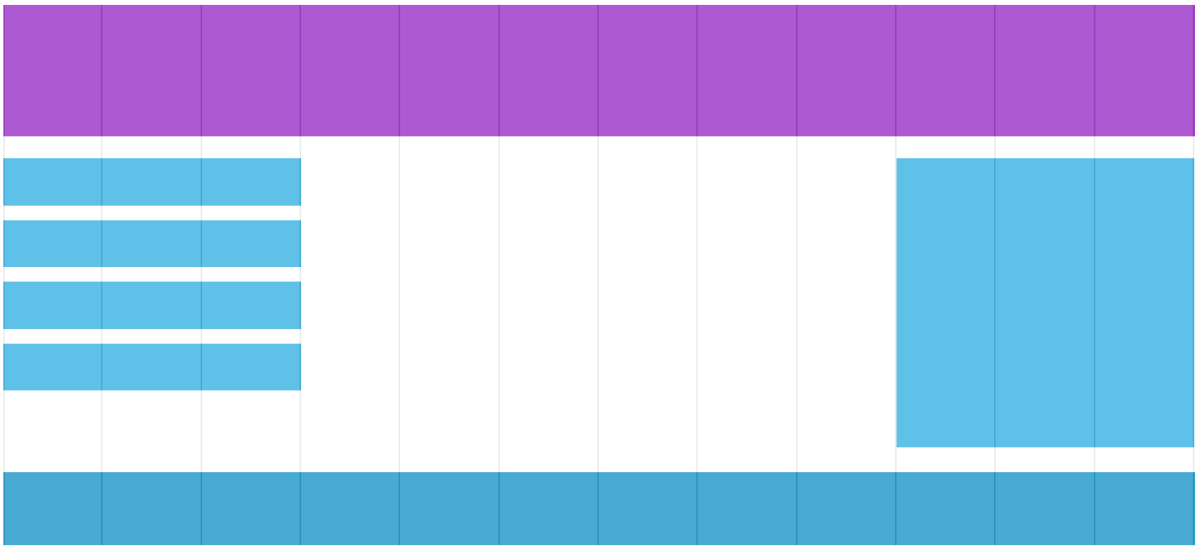
3. **Use CSS media queries to apply different styling for small and large screens** - Setting large absolute CSS widths for page elements will cause the element to be too wide for the viewport on a smaller device. Instead, consider using relative width values, such as width: 100%. Also, be careful of using large absolute positioning values. It may cause the element to fall outside the viewport on small devices.

Grid-View

A grid is a structure formed by invisible lines that divide a page into columns or modules. These columns help designers align and organize page elements through their designs.

Many web pages are based on a grid-view, which means that the page is divided into columns.

Using a grid-view is very helpful when designing web pages. It makes it easier to place elements on the page.



A responsive grid-view often has 12 columns, and has a total width of 100%, and will shrink and expand as you resize the browser window.

Lets start building a responsive grid-view. First ensure that all HTML elements have the box-sizing property set to border-box. This makes sure that the padding and border are included in the total width and height of the elements.

Add the following code in your CSS:

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

The following example shows a simple responsive web page, with two columns:

25%	75%

```
.menu {
  width: 25%;
  float: left;
}
.main {
  width: 75%;
```

```
  float: left;
}
```

The example above is fine if the web page only contains two columns. However, we want to use a responsive grid-view with 12 columns, to have more control over the web page. First we must calculate the percentage for one column: 100% / 12 columns = 8.33%. Then we make one class for each of the 12 columns, class="col-" and a number defining how many columns the section should span:

```
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```

All these columns should be floating to the left, and have a padding of 15px:

```
[class*="col-"] {
  float: left;
  padding: 15px;
  border: 1px solid red;
}
```

Each row should be wrapped in a . The number of columns inside a row should always add up to 12:

```
<div class="row">
  <div class="col-3">...</div> <!-- 25% -->
  <div class="col-9">...</div> <!-- 75% -->
</div>
```

The columns inside a row are all floating to the left, and are therefore taken out of the flow of the page, and other elements will be placed as if the columns do not exist. To prevent this, we will add a style that clears the flow:

```
.row::after {
  content: "";
  clear: both;
  display: table;
}
```

Media Query

Media query is a CSS technique introduced in CSS3. It uses the @media rule to include a block of CSS properties only if a certain condition is true.

Example

If the browser window is 600px or smaller, the background color will be lightblue:

```
@media only screen and (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```

Design for Mobile First

Mobile First means designing for mobile before designing for desktop or any other device (This will make the page display faster on smaller devices). This means that we must make some changes in our CSS. Instead of changing styles when the width gets *smaller* than 768px, we should change the design when the width gets *larger* than 768px. This will make our design Mobile First:

```
/* For mobile phones: */
[class*="col-"] {
  width: 100%;
}

@media only screen and (min-width: 768px) {
  /* For desktop: */
  .col-1 {width: 8.33%;}
  .col-2 {width: 16.66%;}
  .col-3 {width: 25%;}
  .col-4 {width: 33.33%;}
  .col-5 {width: 41.66%;}
  .col-6 {width: 50%;}
  .col-7 {width: 58.33%;}
  .col-8 {width: 66.66%;}
  .col-9 {width: 75%;}
  .col-10 {width: 83.33%;}
  .col-11 {width: 91.66%;}
  .col-12 {width: 100%;}
}
```

HTML Example

For desktop: The first and the third section will both span 3 columns each. The middle section will span 6 columns.

For tablets: The first section will span 3 columns, the second will span 9, and the third section will be displayed below the first two sections, and it will span 12 columns:

```
<div class="row">
  <div class="col-3 col-s-3">...</div>
  <div class="col-6 col-s-9">...</div>
  <div class="col-3 col-s-12">...</div>
</div>
```

Device Breakpoints

There are tons of screens and devices with different heights and widths, so it is hard to create an exact breakpoint for each device. To keep things simple you could target five groups:

```
/* Extra small devices (phones, 600px and down) */
@media only screen and (max-width: 600px) {...}

/* Small devices (portrait tablets and large phones, 600px and up) */
@media only screen and (min-width: 600px) {...}

/* Medium devices (landscape tablets, 768px and up) */
@media only screen and (min-width: 768px) {...}

/* Large devices (laptops/desktops, 992px and up) */
@media only screen and (min-width: 992px) {...}

/* Extra large devices (large laptops and desktops, 1200px and up) */
@media only screen and (min-width: 1200px) {...}
```

Orientation: Portrait / Landscape

Media queries can also be used to change layout of a page depending on the orientation of the browser. You can have a set of CSS properties that will only apply when the browser window is wider than its height, a so called "Landscape" orientation:

The web page will have a lightblue background if the orientation is in landscape mode:

```
@media only screen and (orientation: landscape) {
  body {
    background-color: lightblue;
  }
}
```

Hide Elements With Media Queries

Another common use of media queries, is to hide elements on different screen sizes:

I will be hidden on small screens.

```css
/* If the screen size is 600px wide or less, hide the element */
@media only screen and (max-width: 600px) {
  div.example {
    display: none;
  }
}
```

You can also use media queries to change the font size of an element on different screen sizes.

Lab Tasks:

Task 1. Semantic Markup and Responsive Design

In this task, you will be creating a responsive web layout using semantic markup. Follow the steps below:

1. Create a new HTML file with appropriate markup for a webpage, including the following elements: header, nav, section, article, aside, and footer.
2. Use CSS grid to create a two-column layout for the main content section. The first column should be 70% of the width, while the second column should be 30%.
3. Use media queries to make the layout responsive. At smaller device widths, the two-column layout should switch to a single column, with the content sections stacked on top of each other.
4. Apply appropriate styles to the header, nav, section, article, aside, and footer elements to make the content visually appealing and easy to read.

Task 2. Semantic Markup and Responsive Design (Bonus Task)

Create a simple web page that displays a list of products. The page should have the following elements:

1. A header with a logo and a navigation menu.
2. A section that displays a list of products in a grid layout.
3. Each product should have an image, a title, a brief description, and a price.
4. When a user hovers over a product, the image should zoom in slightly.
5. When a user clicks on a product, they should be taken to a new page with more information about that product.

Use HTML and CSS to create the web page, and make sure to follow best practices for semantic markup and responsive design.