BIN208 WEBTECHNOLOGIES FOR BIOINFORMATICS

11/9/2021
Exercise 1: CSS LAYOUTS

Aim : To create a two column and three column layout using CSS coding styles

Learning Outcomes:

- In this lab you will practice altering the default placement of HTML elements on the page, using Cascading Style Sheets (CSS).
- The student will be able to code any page layouts using CSS rules
- The student will be able use the concept of Box model approach using using ids and classes
- The student will be able to validate the CSS code

Box model calculation:

The **W3C box model** considers the width of the element to be equal to the content of the box *excluding* its padding and border. Padding and border are added *on top* of whatever dimensions you set for the element, which could have some unpredictable consequences on your page layout.

For example, let's consider a box with a width of 200px and a height of 200px, padding of 10px on all sides and a border of 2px all round. The browser does not see it simply as a 200px box. Rather, the browser calculates the horizontal space necessary to display the box as being 224px: 200 (width) + 2 (left border) + 10 (left padding) + 10 (right padding) + 2 (right border) = 224px. Given that the element is a perfect square, the height will also compute to 224px.

General Formulae:

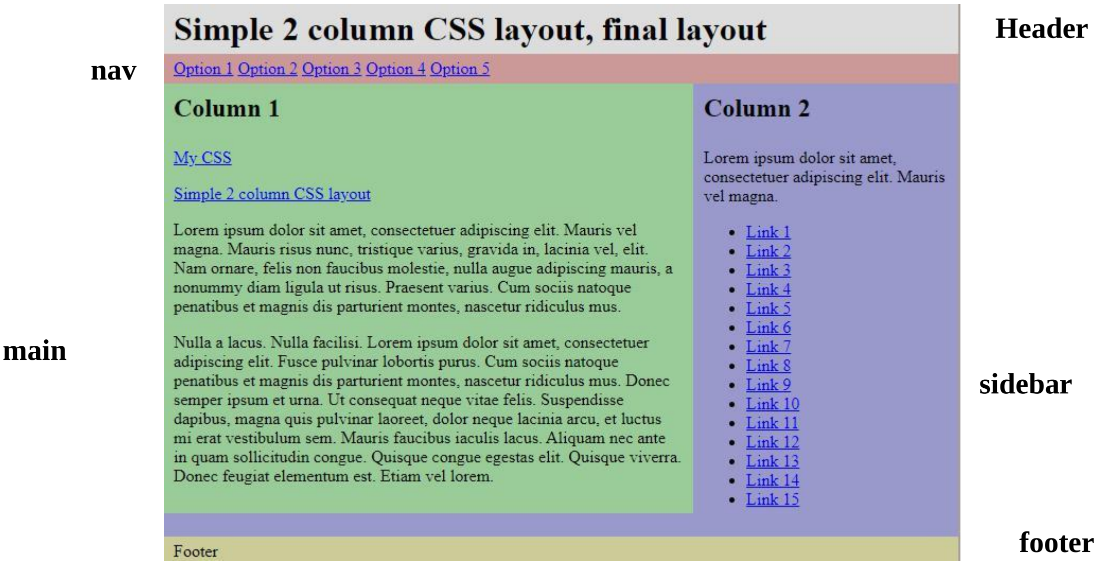
1.width= width+padding left+padding right+margin left+margin right+border left + border right
2. Height= Height+padding top+padding bottom+margin top+margin bottom +border top + border bottom

Task1:

Develop the CSS code for the divisions given below .The overall width of the design is 960px . The corresponding width of content-1 and content-2 is 240px and 720px and further the width of content-2-1 and content-2-2 is 480px and 240px.The padding space of 10px was allocated to header and footer sections.Use Box model approach and calculate the accurate width and height for each dvisions and record it in the observation notebook.The whole web layout is organized in the center of the web browser.

```
<div id="body" >
     <div id="header"  ></div>
     <div id="main">
          <div id="content-1" ></div>
          <div id="content-2" >
               <div id="content-2-1" ></div>
               <div id="content-2-2" ></div>
          </div>
     </div>
     <div id="footer" ></div>
</div>
```

Task2: Create the following two column layout using internal CSS coding



Simple 2 column CSS layout, final layout

Option 1 Option 2 Option 3 Option 4 Option 5

Column 1

My CSS

Simple 2 column CSS layout

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Mauris vel magna. Mauris risus nunc, tristique varius, gravida in, lacinia vel, elit. Nam ornare, felis non faucibus molestie, nulla augue adipiscing mauris, a nonummy diam ligula ut risus. Praesent varius. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

Nulla a lacus. Nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Fusce pulvinar lobortis purus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec semper ipsum et urna. Ut consequat neque vitae felis. Suspendisse dapibus, magna quis pulvinar laoreet, dolor neque lacinia arcu, et luctus mi erat vestibulum sem. Mauris faucibus iaculis lacus. Aliquam nec ante in quam sollicitudin congue. Quisque congue egestas elit. Quisque viverra. Donec feugiat elementum est. Etiam vel lorem.

Column 2

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Mauris vel magna.

- Link 1
- Link 2
- Link 3
- Link 4
- Link 5
- Link 6
- Link 7
- Link 8
- Link 9
- Link 10
- Link 11
- Link 12
- Link 13
- Link 14
- Link 15

Footer

Steps to be implemented in designing the CSS layout :

The layout consists of a header, a horizontal navigation bar, a main content column, a sidebar, and footer. It is also horizontally centered in the browser window.

1. Basic structure

First of all, we create the basic HTML structure:

```
<div id="wrap">
<div id="header"></div>
<div id="nav"></div>
<div id="main"></div>
<div id="sidebar"></div>
<div id="footer"></div>
```

2. After that, we put some content in the different sections:

```
<div id="wrap">
<div id="header"><h1>Document Heading</h1></div>
<div id="nav">
<ul>
<li><a href="#">Option 1</a></li>
<li><a href="#">Option 2</a></li>
...
</ul>
</div>
<div id="main">
<h2>Column 1</h2>
<p>--------------------------------------------------</p>
</div>
<div id="sidebar">
<h2>Column 2</h2>
<p>----------------------------------------------------------</p>
<ul>
<li><a href="#">Link 1</a></li>
<li><a href="#">Link 2</a></li>
...
</ul>
</div>
<div id="footer">
<p>Footer</p>
</div>
</div>
```

Now we have a completely unstyled HTML document which is structured in a way that lets us use CSS to control its layout.

3. Adjust the body and html elements

To make the content reach the edges of the browser window, we set the margin and padding of the body and html elements to zero. We also specify colors for text and background.

```
body,
html {
margin:0;
padding:0;
color:#000;
background:#a7a09a;
}
```

4. On to the main containers

After that it’s time to give the content area a width and center it horizontally. We do that by specifying the width and margins of the main container, #wrap. We also give it a background colour to make it show up on the page.

The method we use to center the content is based on the fact that when an element’s left and right margins are set to auto, they will share whatever is left when the element’s width has been subtracted from that of its container. In this case the width of #wrap will be subtracted from the width of the browser window**.**

```
#wrap {
width:750px;
margin:0 auto;
background:#99c;
}
```

5. After that, we give the different sections of the document different background colours to make them show up.

```
#header {
background:#ddd;
}
#nav {
background:#c99;
}
#main {
background:#9c9;
}
#sidebar {
background:#c9c;
}
#footer {
background:#cc9;
}
```

6. Place the columns side by side

To make the two columns (#main and #sidebar) display side by side we float them, one to the left and the other to the right. We also specify the widths of the columns.

```
#main {
float:left;
width:500px;
background:#9c9;
}
#sidebar {
float:right;
width:250px;
background:#c9c;
}
```

7. Push the footer down

The footer doesn’t get pushed down to the bottom of the content because of the way float works. When you float an element, it is removed from the document flow and doesn’t push elements that follow it down. In this case #footer will start right below #sidebar.

To avoid this we use the clear property to tell the footer that it can’t have any elements next to it.

```
#footer {
clear:both;
background:#cc9;
}
```

8. Make the navigation bar horizontal

#nav contains a regular unordered list of links. Since we don’t want it to look like an unordered list we restyle it.

```
#nav ul {
margin:0;
padding:0;
list-style:none;
}
#nav li {
display:inline;
margin:0;
padding:0;
}
```

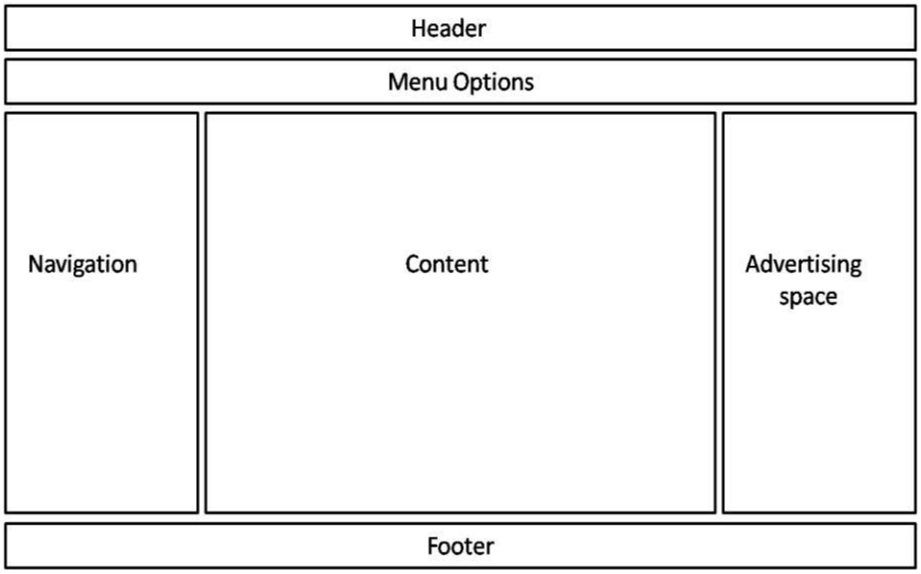
9. Adjust margins and paddings, and make IE 6 cooperate

Almost done, Time to adjust the margin and padding of some elements to make the layout a little less cramped.

```
header {
padding:5px
10px;background:#ddd;}
h1 {
margin:0;
}
#nav {
```

```
padding:5px 10px;
background:#c99;
}
#main {
float:left;
width:480px;
padding:10px;
background:#9c9;
}
h2 {
margin:0 0 1em;
}
#sidebar {
float:right;
width:230px;
padding:10px;
background:#99c;
}
#footer {
clear:both;
padding:5px 10px;
background:#cc9;
}
#footer p {
margin:0;
}
```

Task3: Create the following three column layout using external CSS coding


Header
Menu Options
Navigation
Content
Advertising space
Footer