Lab 04 – CSS Layout

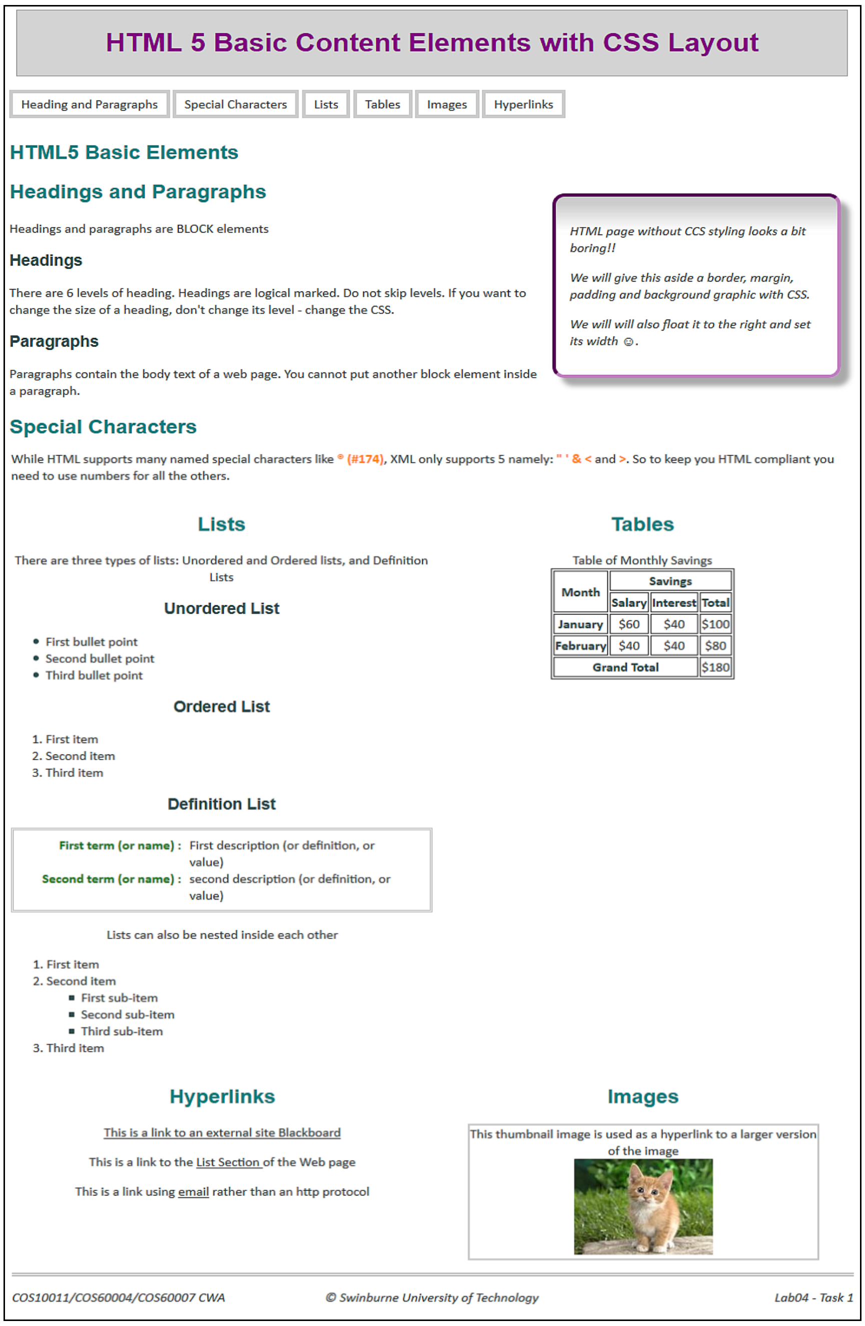
Aims:

- Understand and apply the CSS Box model.
- Apply page layout techniques to a Web page
- Implement a Web page that maintains legibility under scaling
- Implement conditional CSS stylesheets for responsive design (using CSS media queries)

Task 1: Applying CSS rules for Web page layout

In this task you will apply some simple layout CSS styling to a Web page. We will use a version of the Web page formatted with some typographic styles in Task 1. This HTML file is in `lab04.zip` and is called `lab04task1.html`. Put it in your `lab04` folder on your local machine.

Some extra <`section`> elements have been added to the html file to group together each heading with its associated text. These sections have been given `id` attributes so we can manipulate their layout with CSS. We will layout the HTML to look something like below:



HTML 5 Basic Content Elements with CSS Layout

Heading and Paragraphs | Special Characters | Lists | Tables | Images | Hyperlinks

HTML5 Basic Elements

Headings and Paragraphs

Headings and paragraphs are BLOCK elements

Headings

There are 6 levels of heading. Headings are logical marked. Do not skip levels. If you want to change the size of a heading, don't change its level - change the CSS.

Paragraphs

Paragraphs contain the body text of a web page. You cannot put another block element inside a paragraph.

HTML page without CCS styling looks a bit boring!!

We will give this aside a border, margin, padding and background graphic with CSS.

We will will also float it to the right and set its width ☺.

Special Characters

While HTML supports many named special characters like ® (#174), XML only supports 5 namely: " ' & < and >. So to keep you HTML compliant you need to use numbers for all the others.

Lists

There are three types of lists: Unordered and Ordered lists, and Definition Lists

Unordered List

- First bullet point
- Second bullet point
- Third bullet point

Ordered List

1. First item
2. Second item
3. Third item

Definition List

First term (or name) :	First description (or definition, or value)
Second term (or name) :	second description (or definition, or value)

Lists can also be nested inside each other

1. First item
2. Second item
   - First sub-item
   - Second sub-item
   - Third sub-item
3. Third item

Tables

Table of Monthly Savings

Month	Savings		
	Salary	Interest	Total
January	$60	$40	$100
February	$40	$40	$80
Grand Total			$180

Hyperlinks

This is a link to an external site Blackboard

This is a link to the List Section of the Web page

This is a link using email rather than an http protocol

Images

This thumbnail image is used as a hyperlink to a larger version of the image

COS10011/COS60004/COS60007 CWA © Swinburne University of Technology Lab04 - Task 1

Step 1. Create another CSS file and link multiple stylesheets to the web page.

In this task we will reuse the CSS stylesheet **`lab03.css`** we created last week in Lab 3 as well as a new CSS style sheet we will create for this task. Create a new folder **`styles`** inside your **`lab04`** folder and copy your **`lab03.css`** into it.

It is common practice to keep images that are referenced by the CSS separate from any images referenced by the HTML. Inside the **`styles`** folder, create another directory called **`images`**. Place the file **`fade.jpg`** from Canvas in this new **`styles/images`** folder.

Using Notepad++ or similar editor, create a new text file called **`layout.css`** and save it in **`lab04/styles`**. Add a comment header to the top of your CSS file similar to the following, replacing the text in italics:

```
/*
filename: [your name]
author: [your name]
created: [enter date]
last modified: [enter date]
description: [html files it refers to (if known)]
*/
```

Create a link to this file from your HTML page by adding a reference to the external stylesheet in each of the <head> elements as follows:

```
<head>
      <!--  other meta stuff -->
      <link href= "styles/lab03.css" rel="stylesheet"/>
      <link href= "styles/layout.css" rel="stylesheet"/>

</head>
```

The stylesheet you created in the previous lab

Having multiple style sheets will very slightly slow the page load speed (because there are multiple HTTP requests), but can help the developer organise a complex site. For example, general site-wide CSS rules might be stored in one sheet, while page-specific CSS rules might be stored in another. All stylesheets will be loaded into the browser as if they were a single file. Inheritance and over-riding rules will still apply.

Step 2. Using multiple CSS files

Multiple CSS files are read into the browser in the order in which they are referenced. If there is any conflict between rules, the rule that is read in last will take precedence. In the previous Lab 4 we made the <aside> disappear when we moved the mouse over it. We can remove this effect without changing the original lab04.css file by redefining in **`layout.css`** the display attribute of the pseudo element `hover` to its default value as follows:

```
      aside:hover {display:block;}
```

In this week version of the HTML, we have changed the menu to a set of paragraphs each contain an anchor element (rather than a single paragraph). All these paragraphs have an attribute **`class="menu"`.** To make these paragraphs display on a single line we will format them all using the display property again, namely:

```
display: inline-block;
```

Add a rule to your CSS file that selects on the menu class to apply this property.

Another common way to create a menu is to use a list. For example

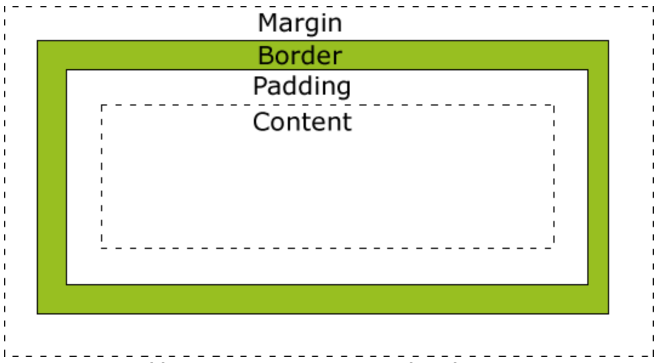
```
      <ul>
            <li><a …>Menu item 1</a></li>
            <li><a …>Menu item 2</a></li>
      </ul
```

List items can be in-lined as above. The dot points can be removed with the rule

```
      nav ul { list-style:none;}
```

Step 3. The CSS Box model

All HTML elements can be thought of as boxes. For these boxes we can create a border, margin outside the border, and padding between the border and the content inside the box. The relationship between these attributes is illustrated below.



Source: http://www.w3schools.com/css/css_boxmodel.asp

1. Let's set these parameters around our <aside> element.
For the **`border`** we will use the short form that set all **`top`**, **`right`**, **`bottom`** and **`left`** border properties in one statement. We have defined the border width in **`pixels`** rather than a measurement unit like **`cm`** which is more appropriate to the printed page. We will define the margin and padding in **`em`** units so there is a relationship to font size.

2. We will also set a background graphic to make the aside stand out from the main content a little.

```
aside {
        border: 4px inset purple;
        border-radius: 14px;
        margin: 1em;
        padding: 1em;
        background:url(images/fade.jpg) repeat-x top left;
}
```

Short form: width style color

Introduced with CSS3

In the folder used to store CSS images

What is this doing?

Save your CSS and reload the HTML

3. Let's format the <header> in a similar way. Define properties for the header element so that:

- it has a solid border **`1px`** wide coloured **`darkgray`**
- has a **`0.5em`** margin and padding
- has a background colour set using the property **`background-color:lightgray;`**
- it is centred using the appropriate value for the **`text-align`** property.

4. Around the footer element define a border in your choice of colour. The border should be **`4px`** thick, **`double`** and *only* visible on **`top`** of the footer.

Step 4. Sizing and Positioning Boxes

1. Notice that the aside takes up the whole width of the web page (minus its margins). Using percentages is a good way to maintain the relative layout of your design. Let's make it take up 30% of the width of its parent element by adding to the aside rule you created in the previous step the property

```
  width: 30%;
```

*What is its **parent element**?*
*What is the **width** of the parent?*

Now notice that the aside shrinks and expands as you resize the viewport.

2. Asides typically sit to the right of the page out of the main flow of the text. Let's move the aside to the right of the page so that the other content "floats" up around it. We will use the **`float:right;`** property to do this. Your CSS rule for **`aside`** should now look like:

```
aside {
    border: 4px inset purple;
    border-radius: 14px;
    margin: 1em;
    padding: 1em;
    width: 30%;
    background:url(images/fade.jpg) repeat-x top left;
    float: right; }
```

Want to make it 'lift' off the page even more? Add a **`box-shadow: 10px 10px 5px #aaaaaa;`**

3. Create a style rule that floats the **`lists`** section to the **left**. For this we will use an **`id`** selector.
*What is the effect? Notice the subsequent sections float up to the right of the **`lists`** section, while maintaining their normal flow.*

4. There is still a problem as the table is hard against the side of the lists.
Set the margin of the **`table`** element to **`auto`** to centre it in the column.

5. Create a rule that floats the **`tables`** section to the **right** using the appropriate **`id`** selector.

Notice the subsequent sections now float up between the lists and the **`tables`** sections.
Reduce and enlarge the size of the viewport to observe what happens.

6. If we want to stop subsequent elements floating up, we need to use the **`clear`** property. Create the following rule for the **`hyperlinks`** section:

```
#hyperlinks {clear:right;}
```

Notice the **`hyperlinks`** section now sits under the tables sections but still floats to the right of the **`lists`** section. Now try:

```
#hyperlinks {clear:both;}
```

The hyperlinks and subsequent sections now sit below the **`lists`** section.

7. If we want the **`images`** section to sit under the **`tables`** section we could try:

```
#hyperlinks {   clear:both;
                float: left;}
```

What's wrong? The page looks a bit of a mess because the **`images`** section and **`footer`** have moved up.

8. Force the **`footer`** to the bottom of the page by using a **`clear:both;`** property on the **`footer`** element.

9. Let's float the **`images`** section to the right so that it sits under the **`tables`** section, and add a **`border`** property for the **`figure`** element of `2px solid #ccc`

10. Let's even up the widths of our sections in the two columns we have created by using a grouping selector as follows:

```
#lists, #images, #tables, #hyperlinks{
            width :50%;
            text-align:center;  }
```

Careful

Another approach would be to give all these elements a class e.g. `class="column"` in the HTML and then select on that class

Remember that the **`width`** property is the *width of the **content*** of the CSS box. If we are going to have **padding**, **borders** or **margins** around that content, we must reduce the width accordingly, i.e. less than 50%. In this we don't need to because we have no border etc.

Notice that when we apply the `text-align:center;` property to these sections above *all* their textual *ancestor elements* also end up centre aligned through inheritance. The lists look messy, fix this by adding a grouping selector **`ol, ul, dl {text-align:left;}`**
Your CSS rules for the last sections of the page should now look something like this:

```
#lists      {float:left;}
table       {margin: auto;}
#tables     {float:right; margin: auto;}
#hyperlinks {clear:left; float:left;}
#images     {float:right;}
figure      {border: 2px solid #ccc;}
footer      {border-top: 4px double darkgray;   clear: both;}
#lists,#images,#tables,#hyperlinks {width: 50%; text-align: center;}
ol, ul, dl  {text-align: left;}
```

11. As the width of a viewport is reduced but resizing the browser window, the layout and text can become hard to read. Observe what happens when you resize the window to as narrow as it will go. Expand the window again.

Let's set a minimum width on the **`article`** element to help preserve legibility. Observe the difference.

```
article { min-width:30em;}
```

Note that because some of the elements have been taken out of the regular flow by floating them to the right they now disappear off the edge of the page. Ideally the CSS should be able to detect and handle this. To find out more about how to do this look up resources on CSS3 Media Queries.

Step 5. More Pseudo Class Selectors and a Pseudo Element Selector

Another way to select HTML elements is to refer to their relative position in the hierarchy of HTML elements on the Web page. First we will align the paragraphs in the footer next to each other, and then use pseudo selectors to format each of the paragraphs.

1. Create a CSS rule using a contextual selector **`footer p`** to select all those paragraphs in the footer element. In the rule, set the property **`display: inline-block`** to make the blocks sit next to each other on the one line. Define another property for the rule that sets the width of all the paragraphs in the footer to **`33%`**.

```
footer p {display:inline-block; width:33%}
```

2. Create the following rule using the **`:firstchild`**, **`:nth-child(n)`**, and **`:last child`** pseudo class selectors. These will select the sibling paragraphs according to the position in which they sit under their parent element. We then use the properties we used in the previous step to format the paragraphs across the footer:

```
footer p:first-child {float:left;}
footer p:nth-child(2) {text-align: center;}
footer p:last-child {float:right; text-align: right;}
```

3. Add the additional provided stylesheet **`dl.css`** to your **`style`** folder and add a `link` in the `head`. Inspect the CSS code. This uses many of the styles we have used previously but also provides a simple example of styling a **`dl`**, and the use of a pseudo element **`::after`**.

Validate your HTML and CSS using the appropriate validators.

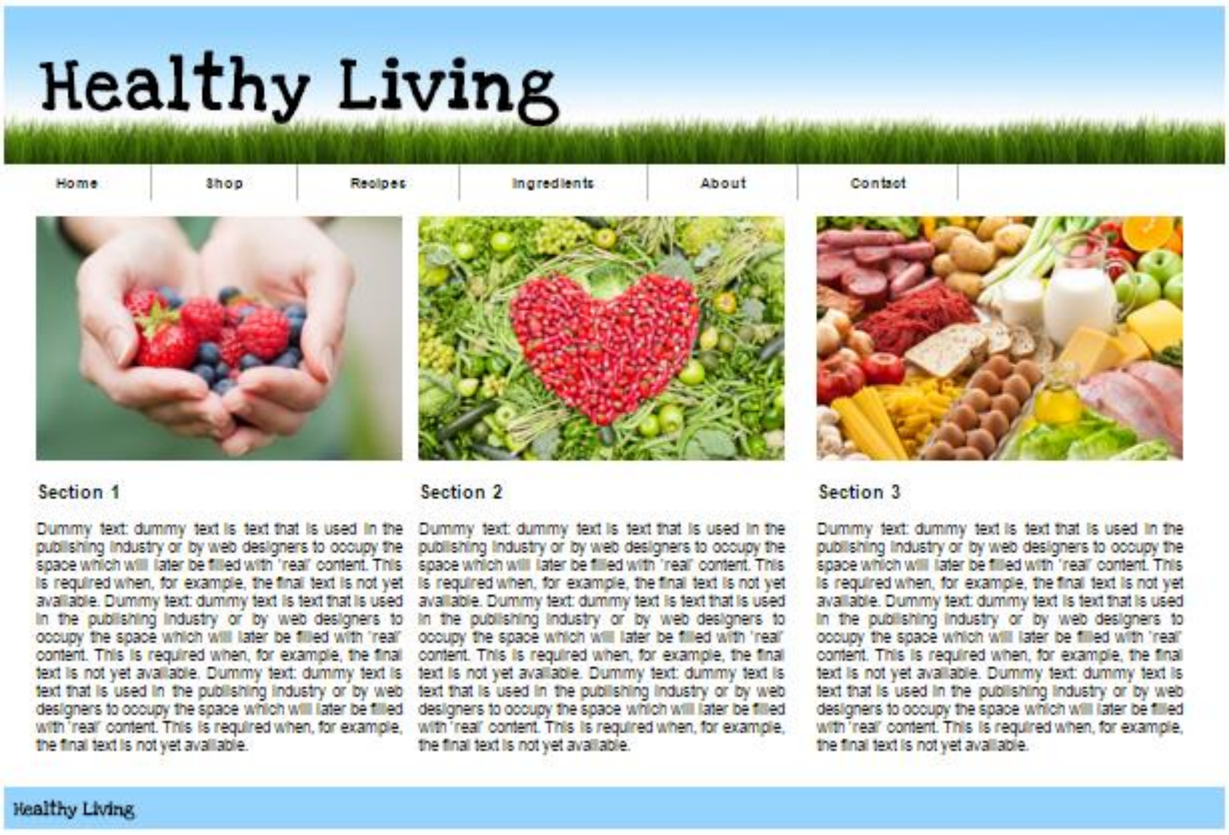
[IMPORTANT] Send your tutor the link to your web page running on the Mercury server to be marked off.

Task 2: Create a more advanced layout with CSS file (Optional)

This task will not be assessed by your tutor but you may find it useful for your assignment.

In this task you will apply some simple CSS styling to a Web page. This HTML file is in **`Lab04.zip`** and is called **`lab04task2.html`**. Put it in your **`lab04`** folder on your local machine.

With CSS, we will layout the HTML (**`lab04task2.html)`** to look something like below:



Healthy Living

Home | Shop | Recipes | Ingredients | About | Contact

Section 1

Dummy text: dummy text is text that is used in the publishing industry or by web designers to occupy the space which will later be filled with 'real' content. This is required when, for example, the final text is not yet available. Dummy text: dummy text is text that is used in the publishing industry or by web designers to occupy the space which will later be filled with 'real' content. This is required when, for example, the final text is not yet available. Dummy text: dummy text is text that is used in the publishing industry or by web designers to occupy the space which will later be filled with 'real' content. This is required when, for example, the final text is not yet available.

Section 2

Dummy text: dummy text is text that is used in the publishing industry or by web designers to occupy the space which will later be filled with 'real' content. This is required when, for example, the final text is not yet available. Dummy text: dummy text is text that is used in the publishing industry or by web designers to occupy the space which will later be filled with 'real' content. This is required when, for example, the final text is not yet available. Dummy text: dummy text is text that is used in the publishing industry or by web designers to occupy the space which will later be filled with 'real' content. This is required when, for example, the final text is not yet available.

Section 3

Dummy text: dummy text is text that is used in the publishing industry or by web designers to occupy the space which will later be filled with 'real' content. This is required when, for example, the final text is not yet available. Dummy text: dummy text is text that is used in the publishing industry or by web designers to occupy the space which will later be filled with 'real' content. This is required when, for example, the final text is not yet available. Dummy text: dummy text is text that is used in the publishing industry or by web designers to occupy the space which will later be filled with 'real' content. This is required when, for example, the final text is not yet available.

Healthy Living

Using Notepad++ or similar editor, create a new text file called **`main.css`** and save it in **`lab04/styles`**. Add a comment header to the top of your CSS file similar to the following, replacing the text in italics:

```
/*
filename: your name
author: your name
created: enter date
last modified: enter date
description: html files it refers to (if known)
*/
```

Create a link to this file from your web page **`lab04task2.html`** by adding a reference to the external stylesheet ***main.css*** in the <head> element.

Notice the files also reference an external font which will be imported.

Below is a CSS files using some of the techniques we coved in the Task 1. Review the CSS rules below to make sure you understand what they are doing, and then type or cut-paste them into your **`main.css`** file.

```
/* main.css */
body {font-family: arial;}
article{
    max-width: 1220px;
    margin: 0 auto;
}
header {
    background-image: url(images/background.png);
    background-repeat: repeat-x;
    background-position: bottom;
    height: 158px;
    width: 100%;
}
header h1 {
    font-family: 'Love Ya Like A Sister', arial;
```

Font imported in HTML file

```
    letter-spacing: 4px;
    font-size: 72px;
    padding-top: .5em;
    padding-left: .5em;
}
nav ul {
    margin: 0;
    padding: 0;
    }
nav ul li {
    list-style: none;
    float: left;
    font-size: 93%;
}
nav ul li a {
    display: block;
```

What is this doing?

```
    font-weight: bold;
    padding: .625em 3.5em; /* 10px 15px */
    text-decoration: none;
    color: #000;
    border-right: 1px solid #bababa;
}
img { max-width: 100%}
#section1, #section2, #section3 {
    float: left;
    width: 30%;
    text-align: justify;
    margin: 1em 1em 1em 1em;
}
footer { clear: both; }
footer h3 {
    font-family: 'Love Ya Like A Sister', arial;
    background-color: #94d4ff;
    padding: .5em 0 .5em .5em;
    background-repeat: repeat-x;
    background-position: bottom;
}
```

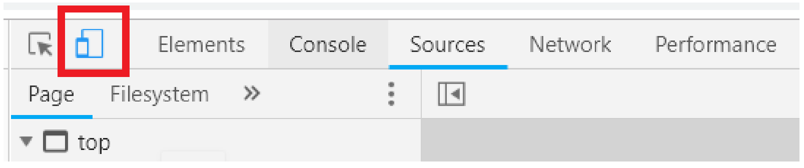
Task 3: Create a responsive layout (Optional)

In this task you will look at some of the problems that can occur when pages design for a desktop are displayed on smaller screens. You will then create some responsive CSS.

How can I test my webpage designs to see how responsive they are, without switching back and forth across devices?

Use the ***"Developer tools"*** available in Firefox and Chrome. Then each of these tools has a mode that shows how a web page will look on different size screens.

- In Firefox menu, select "Web Developer" then select the *"Responsive Design Mode" item*.
- In Google Chrome menu, select "More tools" then select "Developer tools". To activate device mode, click the device icon in the top left corner of the Developer Tools window.

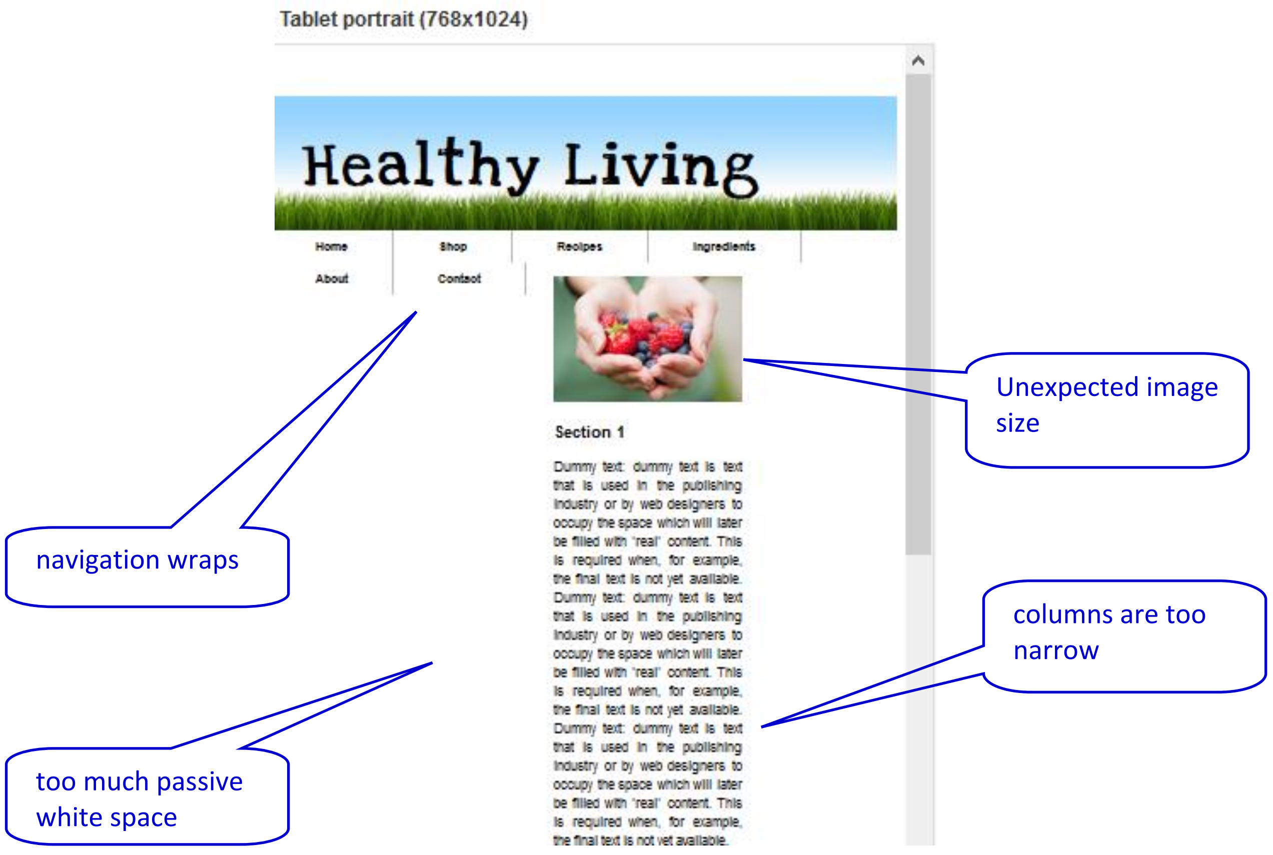


Configure the device resolution and test the result.

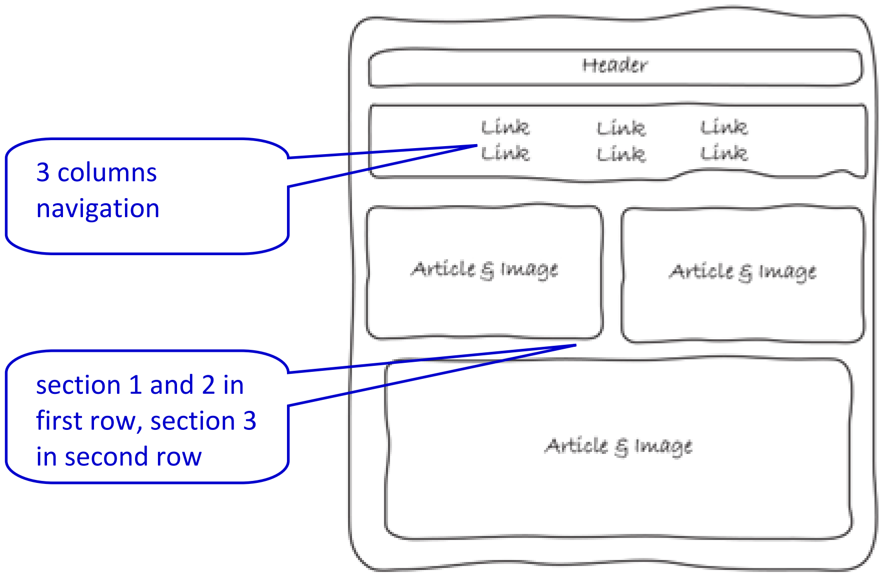
It is recommended that you use developer tools in Firefox or Chrome. The device emulation tool in Internet Explorer 11's developer tools window is more difficult to use for this purpose than either Firefox or Chrome.

Let's first check the Web page using a Responsive Web Developer Tool. We can identify the problems regarding responsiveness when we select various screen sizes as follows:
Select ***<u>Medium Display (maximum display width is 1024 px, like iPad, Tablet, etc.)</u>***:

The Web page will be something like below:



We can identify some problems here like navigation wraps, section columns are too narrow and misplaced, too much passive (unintentional) white space, and unexpected image size. We could design a mock-up for the above identified problems. The design could be something like below:



To implement a responsive design that can change to the above layout when the screen sizes changes we should follow the following steps:

Step 1: Setting the viewport scale to 1.0

The viewport is equal to the size of the browser window. The viewport on handheld devices is much narrower than a desktop/laptop browser window. The narrower viewport causes problems with responsive Web pages. Thus, you should include the following meta tag in the <head> element to the HTML file **`lab04task2.html`**, in order to set the viewport scale 1.0.

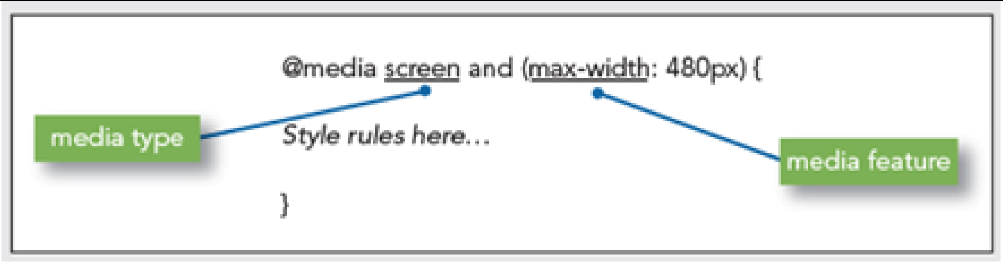
```
<head>
     <!-- Viewport set to scale 1.0 -->
     <meta name="viewport" content="width=device-width, initial-
      scale=1.0"/>

     <!-- References to external basic CSS file -->
     <link href= "styles/main.css" rel="stylesheet"/>
</head>
```

Step 2: Apply CSS Media Query

You should apply styles based on display device characteristics using CSS media query. A media query is an expression that lets you create precise rules for destination media/device. It contains both a ***media type*** and optional expressions that check conditions called ***media features***. Media features include ***variables*** such as the width or height of the destination device.

An example media query is shown below where the media type is ***screen*** and the media feature is ***max-width*** (the max-width value is set to 480 px). The max-width value is called a ***breakpoint*** and it is best measured in ems, because they are flexible (1 ***em*** equals 16 ***px,*** in most browsers when the default medium font is selected.).



```
@media screen and (max-width: 480px) {
Style rules here…
}
```

Create a link to the HTML file **lab04task2.html** by adding a reference to the external responsive stylesheet ***responsive.css*** in the <head> element as follows:

```
<head>
     <!-- Viewport set to scale 1.0 -->
     <meta name="viewport" content="width=device-width, initial-
      scale=1.0"/>

     <!-- References to external basic CSS file -->
     <link href= "styles/main.css" rel="stylesheet"/>

     <!-- References to external responsive CSS file -->
     <link href="responsive.css" rel="stylesheet" media="screen and (max-
     width: 1024px)"/>
</head>
```

Create ***responsive.css*** file with the appropriate header in your **lab04/styles** folder then add the following CSS rules.

```
/* responsive.css  */
/* media query for display: over 500 */

@media screen and (min-width: 31.25em) {

            nav ul li {
                  width: 33%;
            }

            #section1 {
                  width: 43%;
            }

            #section2 {
                  width: 43%;
            }

            #section3 {
                  clear: both;
                  float: none;
                  width: 90%;
                  margin: 0 1em 1em 1.5em;
            }

            #section3 img {
                        float: left;
                        width: 30%;
                        margin: .5em 1em 1em 0;
            }

            article {
                        min-width: 600px;
            }
}
```

media query

3 columns navigation

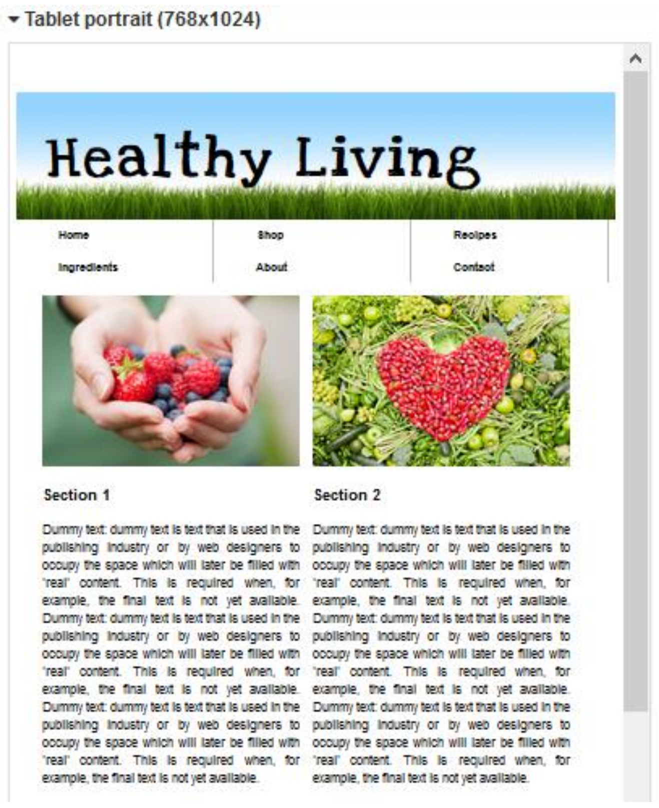
section 1 and 2 set to 43%, i.e., in first row

section 3 sets to 90%, i.e., in second row

section 3 image wraps with content

entire article sets to display window

Medium display:
Your responsive HTML should
now look something like:



Tablet portrait (768x1024)

Healthy Living

Home Shop Recipes
Ingredients About Contact

Section 1

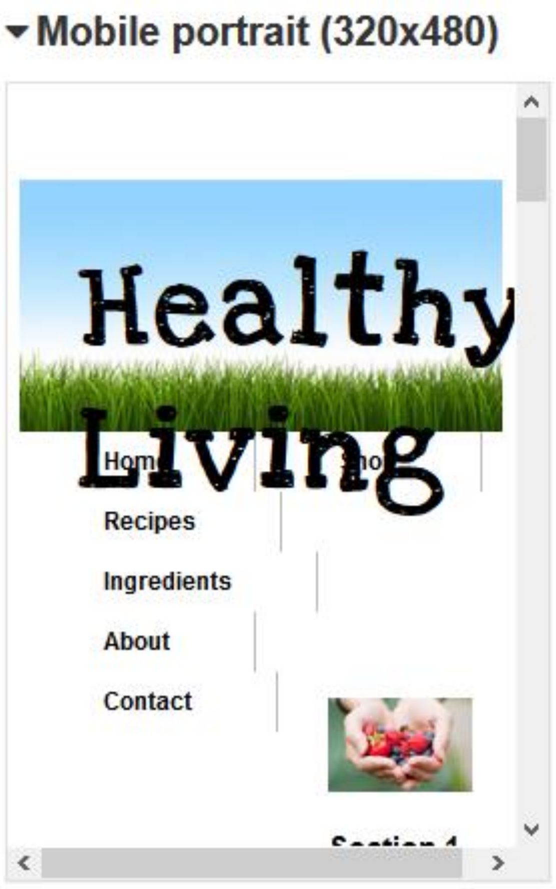
Dummy text: dummy text is text that is used in the publishing industry or by web designers to occupy the space which will later be filled with 'real' content. This is required when, for example, the final text is not yet available. Dummy text: dummy text is text that is used in the publishing industry or by web designers to occupy the space which will later be filled with 'real' content. This is required when, for example, the final text is not yet available. Dummy text: dummy text is text that is used in the publishing industry or by web designers to occupy the space which will later be filled with 'real' content. This is required when, for example, the final text is not yet available.

Section 2

Dummy text: dummy text is text that is used in the publishing industry or by web designers to occupy the space which will later be filled with 'real' content. This is required when, for example, the final text is not yet available. Dummy text: dummy text is text that is used in the publishing industry or by web designers to occupy the space which will later be filled with 'real' content. This is required when, for example, the final text is not yet available. Dummy text: dummy text is text that is used in the publishing industry or by web designers to occupy the space which will later be filled with 'real' content. This is required when, for example, the final text is not yet available.

Now let's create a design for a ***Small Display (maximum display width is 568 px)***:

The Web page will be something like below if we resize it to mobile size (:



Mobile portrait (320x480)

Healthy Living

Hom ho

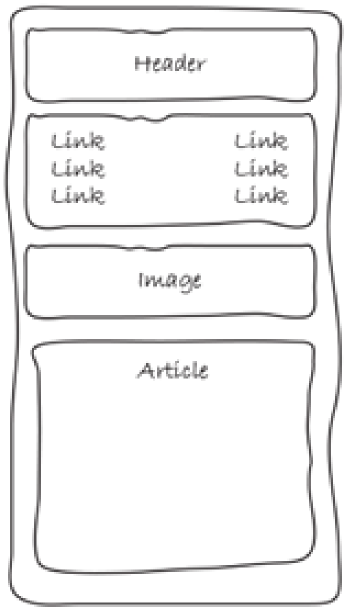
Recipes

Ingredients

About

Contact

The mock-up design and some hints have been given. The mock-up design will be something like below (***2 columns navigation and 1 column section***):



Create a separate CSS file for a small display like Apple iPhone 5.

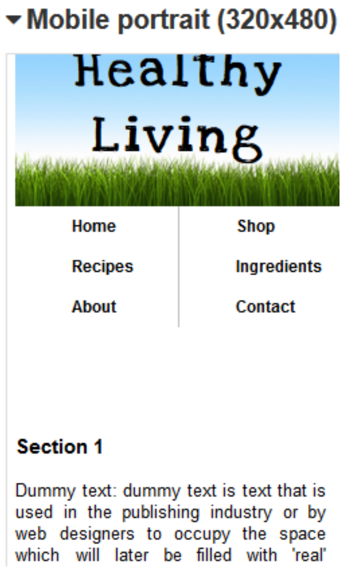
Add the following conditional media query in your ***`responsive.css`*** file (complete the following CSS file and check your responsive Web page using Firefox Responsive Developer Tool).

```
/*  media query for display: under 500 px  */
@media screen and (max-width: 31.25em ) {
                header h1 {
                        font-size: 48px;
                        padding: 0;
                        text-align: center;
                }
                nav ul li {
                        width: ?
                }
                #section1, #section2, #section3 {
                        margin: 0;
                        float: none;
                        width: ?
                }
                img {
                  display: block;
                  margin: 0 auto;
                }
                article{
                                min-width: ?
                }
}
```

responsive content

What should this value be?

Small display (Apply iPhone 5):



Section 1

Dummy text: dummy text is text that is used in the publishing industry or by web designers to occupy the space which will later be filled with 'real'

Validate your HTML and CSS using the appropriate validators.

Want some more practice with CSS layout? If you are interested in exploring further here are some other tutorials:

http://learnlayout.com/
https://www.codecademy.com/courses/web-beginner-en-6merh/4/1?curriculum_id=50579fb998b470000202dc8