**HTML**

* Hypertext markup language
* Used for creating structures for web pages.
* Building block for web development
* HTML elements are represented as tags. Each tag has a few attributes.
* A tag is enclosed in angular brackets <tagname>.
* HTML tags are not case sensitive however it is a good practice to use lowercase tags.
* As every structure has head and body, so does html.
* There are various elements such as "heading", "paragraph", "table", and so on..

**HTML elements:**

An HTML element generally consists of a start tag and end tag, with the content inserted in between:  
  
<tagname>tag body(content)</tagname>

Following is an example of paragraph tag.

<p>My first paragraph.</p>

**Empty HTML elements:**

HTML elements which has no body content are called empty elements.  
<br> is an empty element without a closing tag (the <br> tag defines a line break).  
Empty elements can be "closed" in the opening tag like this: <br />.  
In HTML5 it is not mandatory to close empty elements.

**The id Attribute:**

Id is used to identify and target a single element. You should not use same id for multiple elements. For same type of elements for which you need similar styling you can use class attribute as shown below.

For example:

<p id= “first-para”>My first paragraph.</p>

**Class Attribute**  
The class attribute specifies one or more class names for an HTML element.  
The class name can be used by CSS and JavaScript to perform certain tasks for elements with the specified class name.

For example:

<p class= “header-para”>My first paragraph.</p>

A simple example of HTML:

<!DOCTYPE html>

<html>

<head>

<title>This is a page title.</title> //you can see this in the browser’s tab.

</head>

<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

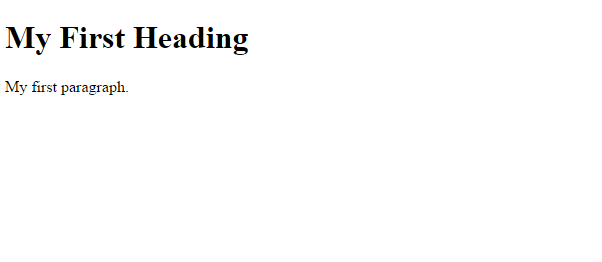
</body>

</html>

Explanation:

* The <!DOCTYPE html> declaration defines this document to be HTML5
* The <html> element is the root element of an HTML page
* The <head> element contains meta information about the document. Meta information is data about the data.
* The <title> element specifies a title for the document
* The <body> element contains the visible page content
* The <h1> element defines a large heading
* The <p> element defines a paragraph

Output of the above code snippet when running with browser:



* To write this code the suggested code editor is **sublime 3**
* You can download it from their official website.
* You must save all the html files with .html or .htm document so that your computer can identify and run the file appropriately.
* After installing the editor, feel free to write the above code by yourself. It will give you a better understanding of how tags work in a browser.
* In above example we saw the tag <h1>.
* Html provides many such headings to be used for different purpose and scenarios. There are six heading tags available in html.
* <h1> to <h6>- with descending font size

**HTML <blockquote> for Quotations**  
The HTML <blockquote> element defines a section that is quoted from another source.  
Browsers generally put an indentation on <blockquote> elements.

E.g <blockquote>What doesn’t kill you make you stronger.</blockquote>

**Links:**

You can add links to your webpage by using <a> tag.

Example:

<!DOCTYPE html>

<html>

<head>

<title>Adding link example.</title>

</head>

<body>

<a href="<https://www.google.com>">Visit Google!</a>

</body>

</html>

The link's destination is specified in the href attribute.   
Attributes are used to provide additional information about HTML elements.

**Images:**

Img is a self closing tag.

Example:

<!DOCTYPE html>

<html>

<head>

<title>Image example.</title>

</head>

<body>

<img src="img\_girl.jpg" alt="Girl in a jacket">

</body>

</html>

src attribute tells the path of the file. It is advised to always put a relative path in src so that your code works fine across all the devices.

The **alt** attribute specifies an alternative text to be used, when an image cannot be displayed.

From now onwards the code snippet will be shown of the main content only. But its implicit that those elements are written inside the body.

**Defining an HTML Table**  
An HTML table is defined with the <table> tag.  
Each table row is defined with the <tr> tag. A table header is defined with the <th> tag. By default, table headings are bold and centered. A table data/cell is defined with the <td> tag.

To add a caption you can use <caption>.

A simple example(every content which is visible in web page is written inside body):

<table style="width:100%">

<caption>Monthly savings</caption>

<tr>

<th>Month</th>

<th>Savings</th>

</tr>

<tr>

<td>January</td>

<td>$100</td>

</tr>

<tr>

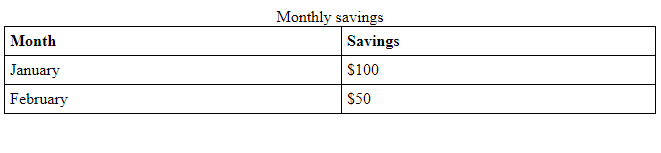
<td>February</td>

<td>$50</td>

</tr>

</table>

**Output**



**Lists in html:**

An unordered list

An unordered list starts with the **<ul>** tag. Each list item starts with the **<li>** tag.

The list items will be marked with bullets (small black circles) by default:

<ul>

<li>A</li>

<li>B</li>

<li>C</li>

<li>D</li>

</ul>

Output:

* A
* B
* C
* D

An ordered list

An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.  
The list items will be marked with numbers by default:

<ol>

<li>A</li>

<li>B</li>

<li>C</li>

<li>D</li>

</ol>

Output:

1. A
2. B
3. C
4. D

**Description Lists**  
A description list is a list of terms, with a description of each term.  
The <dl> tag defines the description list, the <dt> tag defines the term (name), and the <dd> tag describes each term.

Example:

<dl>

<dt>Coffee</dt>

<dd>black hot drink</dd>

<dt>Milk</dt>

<dd>white cold drink</dd>

</dl>

Output:



**Block-level Elements**  
A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

Block level elements in HTML:  
  
<address>,<article>,<aside>,<blockquote>,<canvas>,<dd>,<div>,<dl>,

<dt>,<fieldset>,<figcaption>,<figure>,<footer>,<form>,<h1>-<h6>,

<header>,<hr>,<li>,<main>,<nav>,<ol>,<output>,<p>,<pre>,<section>,

<table>,<tfoot>,<ul>,<video>.

**The <div> Element**  
The <div> element is often used as a container for other HTML elements.  
The <div> element has no required attributes.When used together with CSS, the <div> element can be used to style blocks of content.

**The <form> Element**  
The HTML <form> element defines a form that is used to collect user input.

An HTML form contains form elements.  
Form elements are different types of input elements, like text fields, checkboxes, radio buttons, submit buttons, and more.

**The <input> Element**  
The <input> element is very important form element.  
The <input> element can be displayed in several ways, depending on the type attribute.

Here are some examples:

<input type="text"> Defines a one-line text input field

<input type="radio"> Defines a radio button

<input type="submit"> Defines a submit button (for submitting the form)

Example:

<form action="/action\_page.php" method="get" >

First name:<br>

<input type="text" name="firstname" value="Mickey"><br>

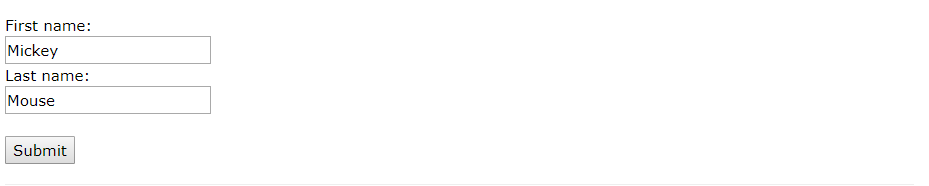
Last name:<br>

<input type="text" name="lastname" value="Mouse"><br><br>

<input type="submit" value="Submit">

</form>

Output:



**The Action Attribute**  
The action attribute defines the action to be performed when the form is submitted.  
Normally, the form data is sent to a web page on the server when the user clicks on the submit button.  
In the example above, the form data is sent to a page on the server called "/action\_page.php". This page contains a server-side script that handles the form data.

**The Target Attribute**  
The target attribute specifies if the submitted result will open in a new browser tab, a frame, or in the current window.

The default value is "\_self" which means the form will be submitted in the current window.  
To make the form result open in a new browser tab, use the value "\_blank".

**The Method Attribute**The method attribute specifies the HTTP method (GET or POST) to be used when submitting the form data.

When to Use GET?  
The default method when submitting form data is GET.  
However, when GET is used, the submitted form data will be visible in the page address field.

Notes on GET:  
Appends form-data into the URL in name/value pairs  
The length of a URL is limited (about 3000 characters)  
Never use GET to send sensitive data! (will be visible in the URL)  
Useful for form submissions where a user want to bookmark the result  
GET is better for non-secure data, like query strings in Google.

When to Use POST?  
Always use POST if the form data contains sensitive or personal information. The POST method does not display the submitted form data in the page address field.  
  
Notes on POST:  
POST has no size limitations, and can be used to send large amounts of data.  
Form submissions with POST cannot be bookmarked.

**Grouping Form Data with <fieldset>**  
The <fieldset> element is used to group related data in a form.  
The <legend> element defines a caption for the <fieldset> element.  
Example

<form action="/action\_page.php">

<fieldset>

<legend>Personal information:</legend>

First name:<br>

<input type="text" name="firstname" value="Mickey"><br>

Last name:<br>

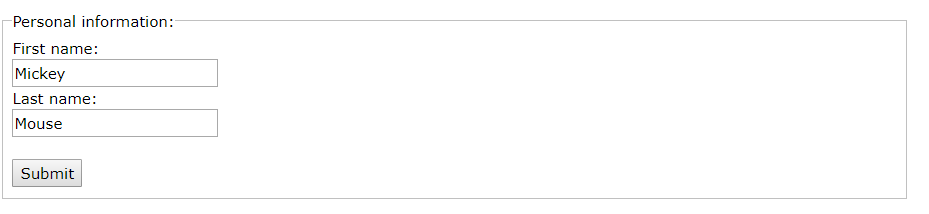
<input type="text" name="lastname" value="Mouse"><br><br>

<input type="submit" value="Submit">

</fieldset>

</form>

Output



**CSS**

* CSS stands for Cascading Style Sheets.
* CSS describes how HTML elements are to be displayed on screen, paper, or in other media.
* CSS can be added to HTML elements in 3 ways:
  + Inline - by using the style attribute in HTML elements
  + Internal - by using a <style> element in the <head> section
  + External - by using an external CSS file
* The preferred way is creating an external css file and then link it with the html file
* CSS files are stored with .css extension.
* You should keep all the css files in a seperate folder inside your main directory.

**Order of priority:**

The first priority is given to inline css then internal css and then least priority is given to external css. Inline and internal css are bad practice so avoid using it as much as you can.

First to apply some style to an html element, you need to select the element.

There are few ways to select an element as mentioned below.

**Universal selector:**

The universal selector selects all the elements. It is displayed by wildcard character.

Look at the following example:

\* {  
 color: green;  
 font-size: 20px;  
 line-height: 25px;  
}

**Element selector:**

It is referred as type selector. The styling provided using this method applies to all the similar element present in the page. Look at the following example:

ul {  
 list-style: none;

display: inline;  
 border: solid 1px #ccc;  
}

**ID selector:**

As we saw earlier you can give ID to a specific element. Now to style that element you can select that element using that ID. you have to write the id name after # symbol. Look at the following example.

Here container is the name of the id.

#container {  
 width: 960px;  
 margin: 0 auto;

padding-top: 10px;  
}

**Class selector:**

As we saw earlier you can keep similar elements in a same class. Now to style all those similar elements you can target it using class selector. You have to write a . symbol before the class name. Look at the following example.

Here box is the name of the class.

.box {  
 padding: 20px;  
 margin: 10px;  
 width: 240px;  
}

**Descendant combinator:**

It allows you to select two or more selectors so that you can be more specific and style a particular element.

Look at the following example:

Here container is id in which there is an element of class box.

#container .box {  
 float: left;  
 padding-bottom: 15px;  
}

**Pseudo class:**

It uses a colon : symbol to identify a pseudo state that an element might be in.

There could be many possible pseudo states such as hover, activated.

Look at the following example:

a:hover {  
 color: red;  
}

There are many more selectors available. Feel free to check it out. But these selectors cover most of the the portion which we will be using.

With css you can set many styling properties such as background color, font-size, font-family, border, underline, bullet points, margin, padding, height, width, float etc etc.

We will see the float property here. Rest you can check out on your own.

**Float:**

It is used for positioning and layout of the elements in web page.

Float property can have four values.

Left: element is floated on the left side of its container.

Right: element is floated on the right side of its container.

None: the element is not floated. It is shown as it is. This is set by default.

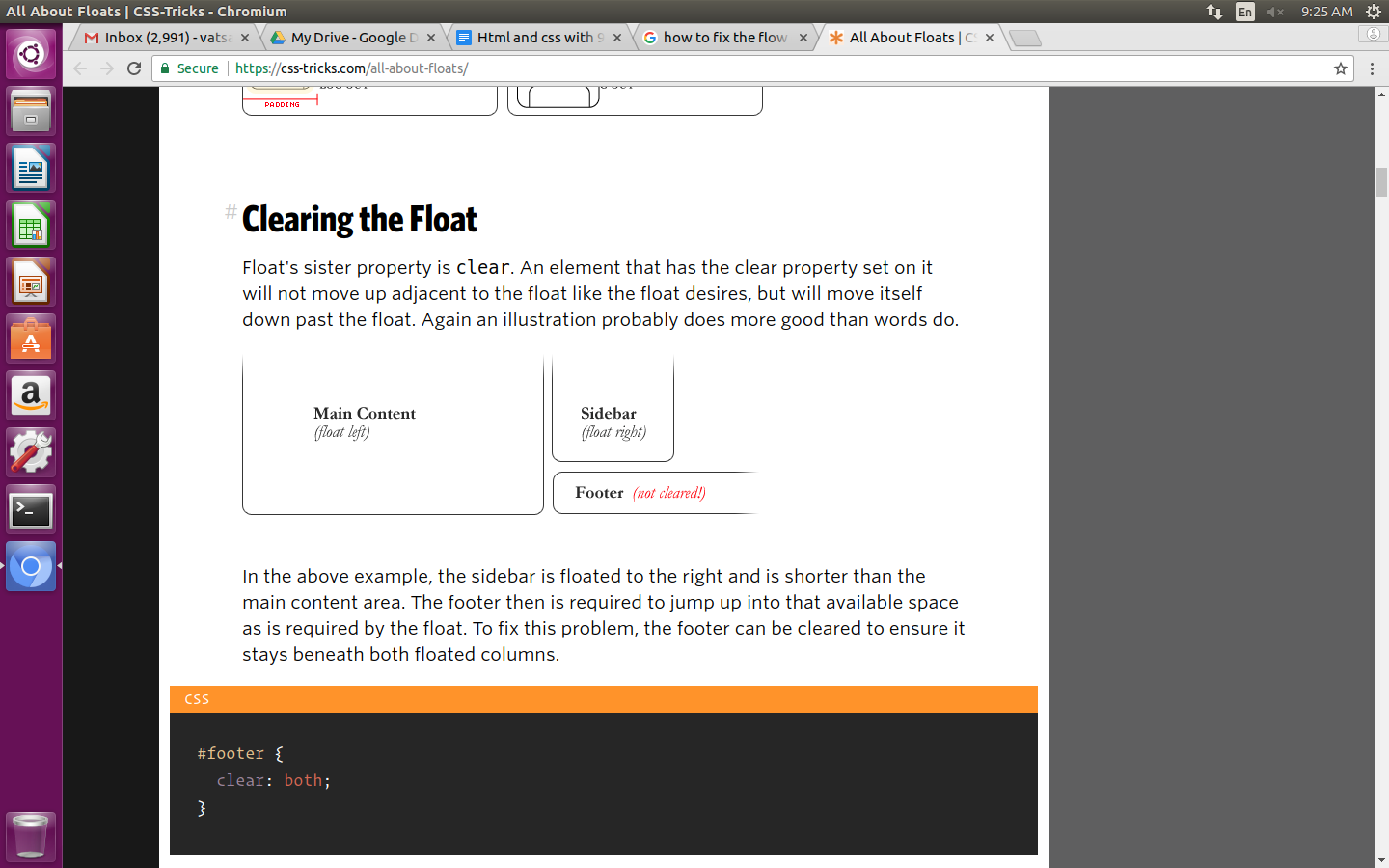
Inherit: the element gets the float property of its parent.

If you apply float to an element then that element will no longer be in the flow of web page. You can see this effect when you apply background color.

**Clearing the Float**  
Float's sister property is clear. An element that has the clear property set on it will not move up adjacent to the float like the float desires, but will move itself down past the float.

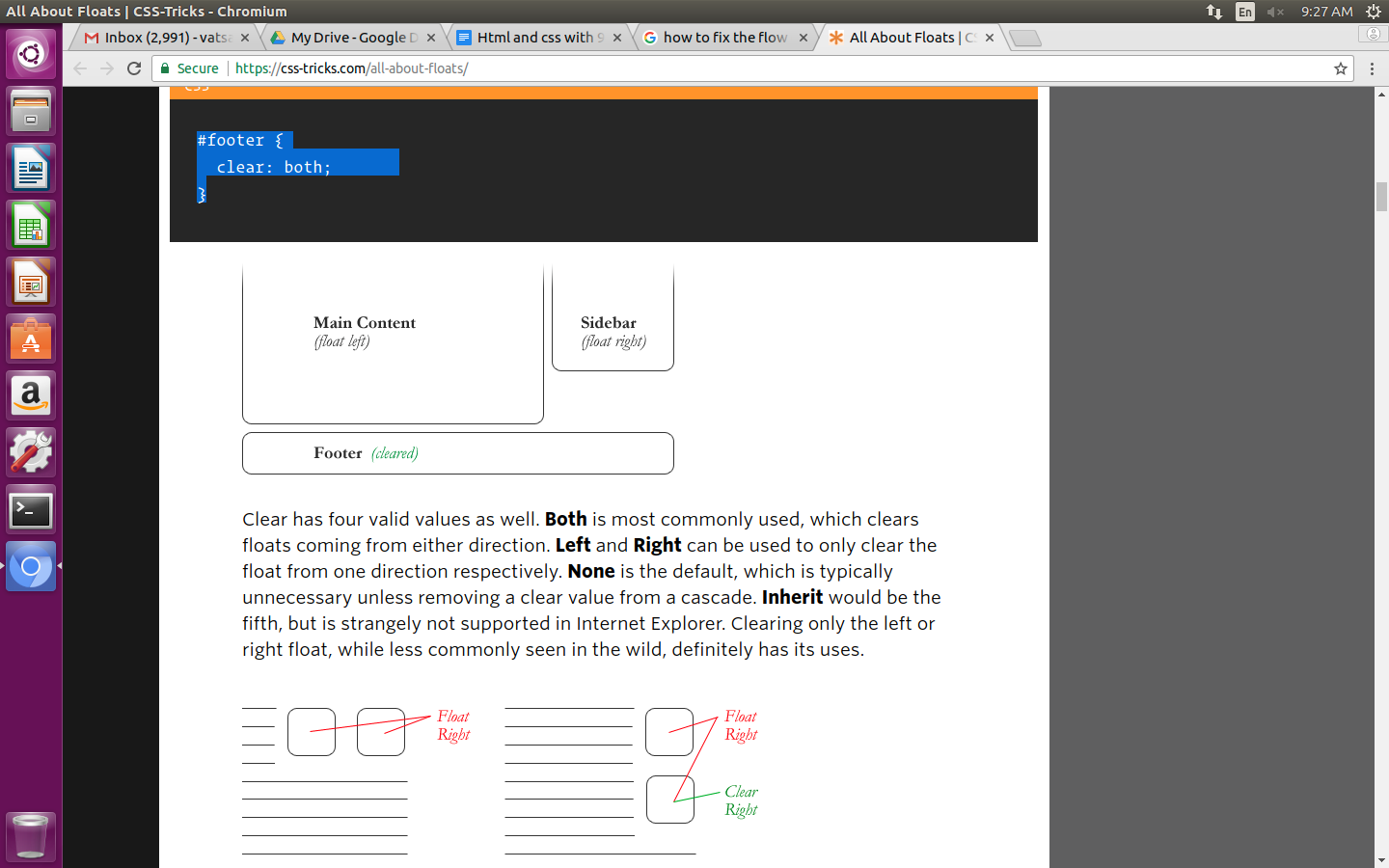
Now it's up to you to check out the rest of the attributes as and when needed.

Now



Now look at the following code:

#footer {  
 clear: both;   
}



**960 grid system to reduce your work:**

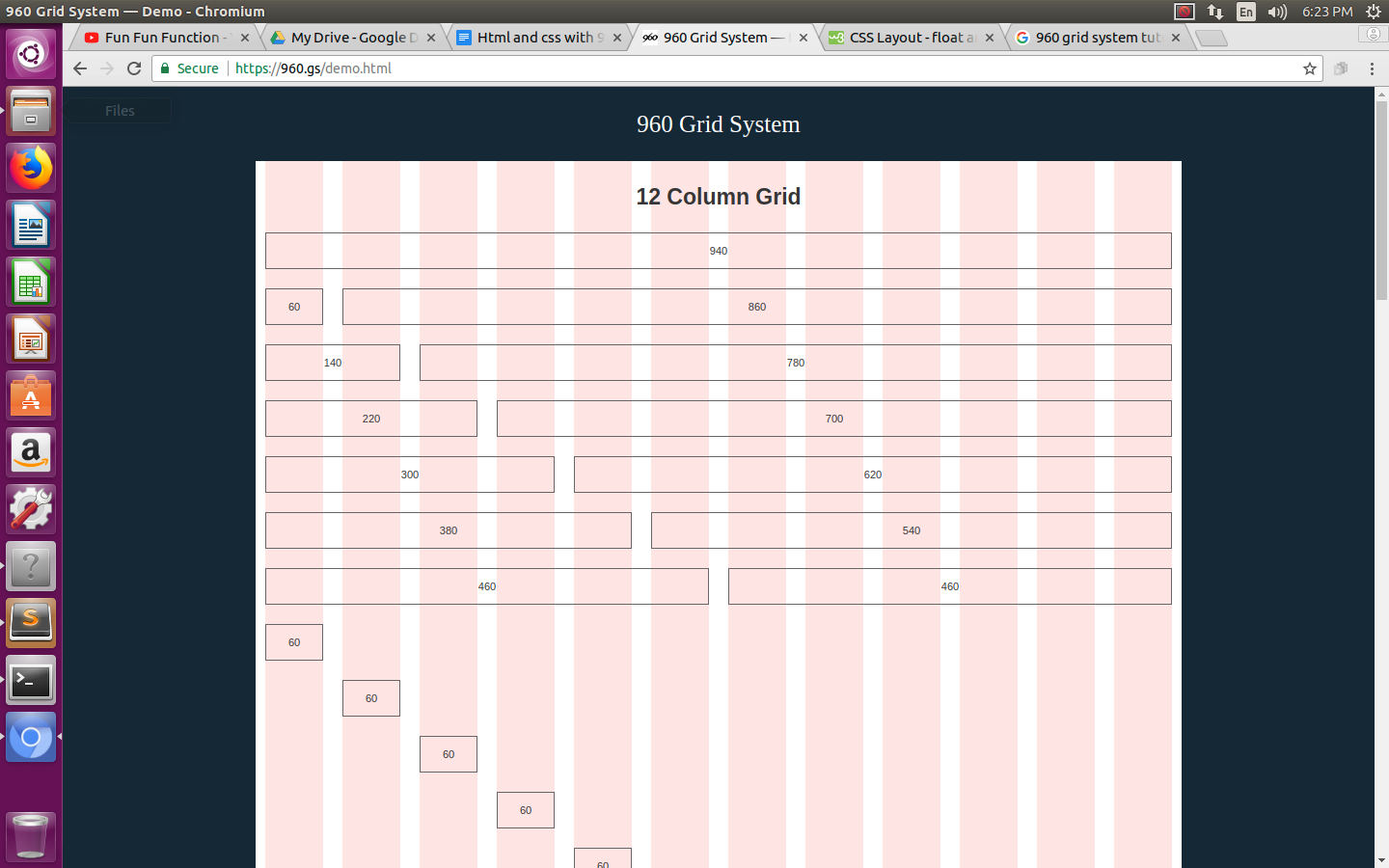
The 960 grid system is a framework which is used to position the elements on a web page.

It helps in reducing the variations in page layout and tries to standardize the layouts.

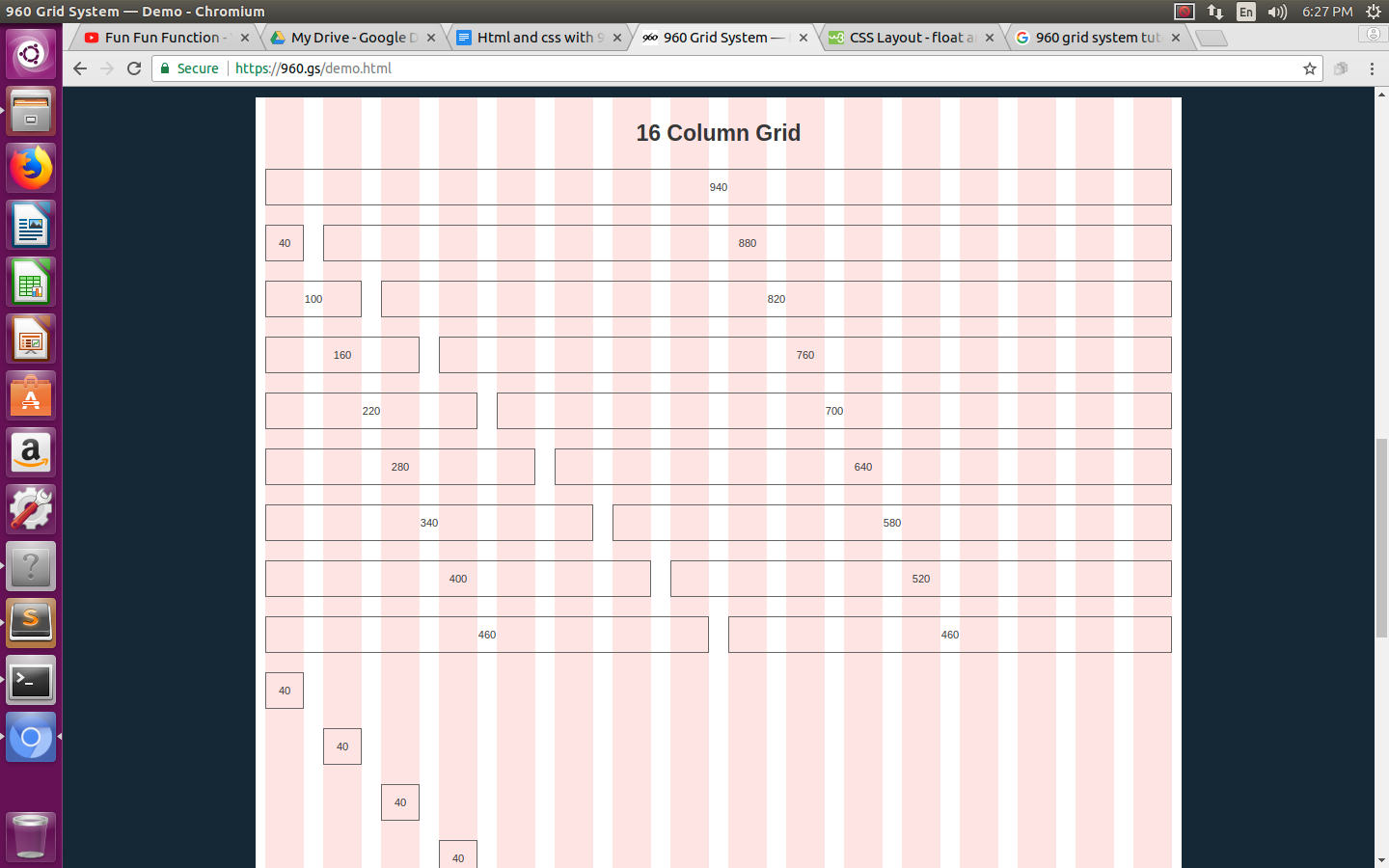
It is based on 960 pixels wide layout. There are two variants available one is 12 columns grid and other is 16 columns grid.

The 12-column grid is divided into portions that are 60 pixels wide. The 16-column grid consists of 40 pixel increments. Each column has 10 pixels of margin on the left and right, which create 20 pixel wide margin between columns.

The following is an image of 960 grid system with 12 columns layout.



The following is an image of 960 grid system with 16 columns layout.



Now there are various classes which are available using which you can position your elements accordingly.

**Classes in 960 grid css:**

Wrap all your contents inside div and give class as container\_12 for 12 column layout or container\_16 for 16 columns layout.

Then for the elements inside the wrapper div you can give classes as per the width an element requires. For example if your h1 tag requires to cover all horizontal width then give class name as grid\_12 or grid\_16 as per your layout. It means that the element will take all 12 or 16 columns required.

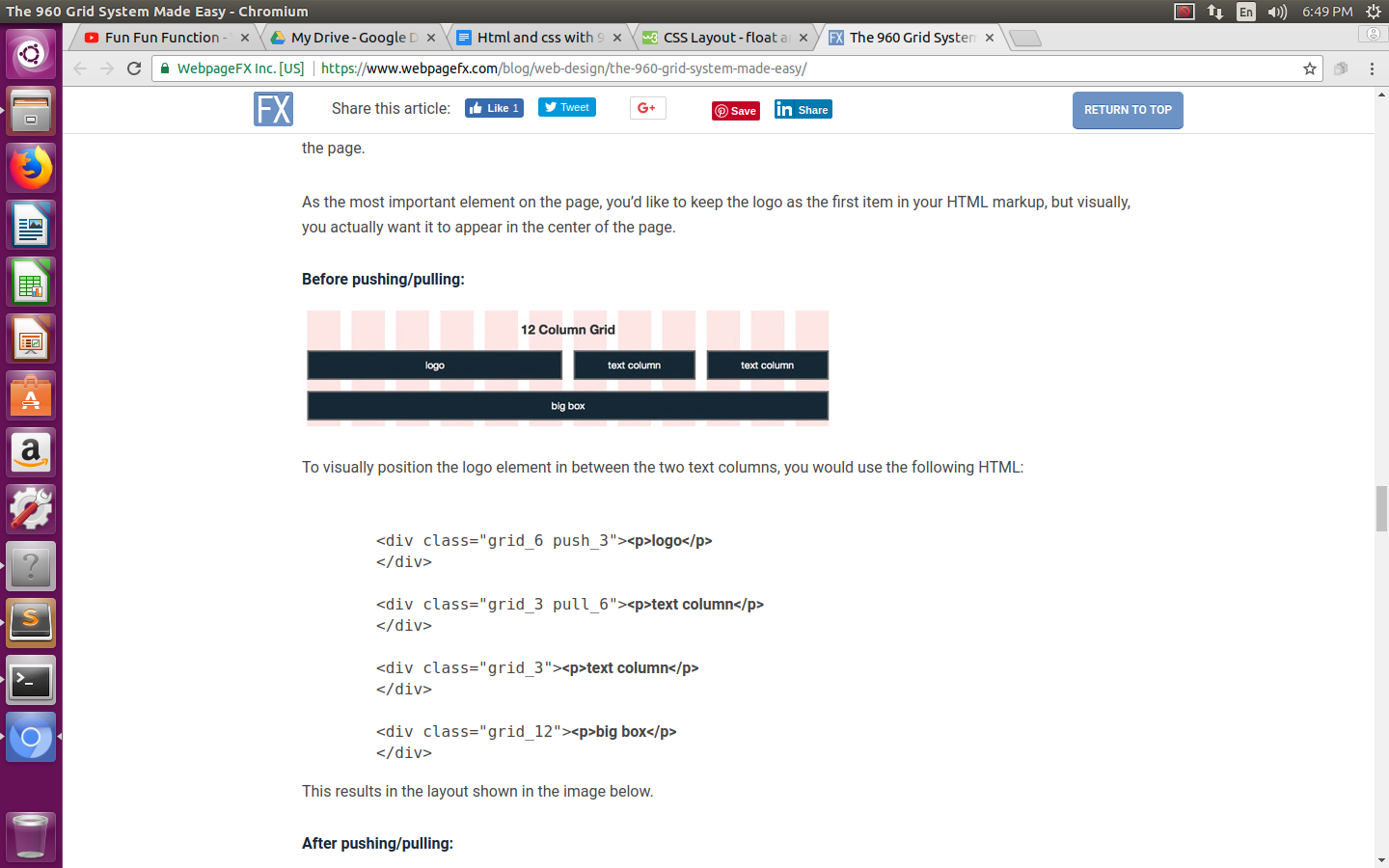
There are many other classes based on your width/column requirements such as grid\_1, grid\_2, grid\_3 etc. etc.

If you want to fill a page horizontally, make sure the numbers add up to 12 or 16 (i.e. grid\_4 + grid\_2 + grid\_6 = 12).

Now if you need to move the elements independently then you can use pull\_1, pull\_2, pull\_3 etc and push\_1, push\_2, push\_3 etc classes. It will move that element accordingly regardless the markup of your page.

For example look at the following code snippet:

<div class="grid\_6 push\_3"> <p>logo</p> </div>  
 <div class="grid\_3 pull\_6"> <p> text column</p> </div>  
 <div class="grid\_3"><p> text column</p> </div>  
 <div class="grid\_12"> <p>big box</p> </div>





Also, there may be a case in which you need open space in your layout. For this purpose you can use suffix\_1,suffix\_2, suffix\_3 etc. or suffix\_1,suffix\_2, suffix\_3 etc. classes to give that much column wide open space.

The difference between suffix and prefix class can be understood from the class name itself.

Suffix class will add the white space after the element and prefix class will give white space before the element.

**Alpha and omega:**

And the remaining two classes are alpha(first) and omega(last) classes.

You must apply this class to a grid which is a child of/nested inside another grid.

The alpha class will be applied to the first child and the omega class to the last child.

In a nutshell, these classes provides margin fix so you can nest grids inside other grids.