Front-End Development in 1 Hour

HTML5 | CSS3 | 960 Grid System

# 

[**Introduction**](#_d4c2l5xtaard) **2**

[**Tools Required**](#_d0so4jvs4rrf) **2**

[**Getting Started**](#_n63duupiypqn) **2**

[**HTML**](#_h75rl3ebt4aw) **2**

[Structure of the code](#_it253jvpmik) 4

[Lists](#_8rgh3t84pb4s) 6

[Other Tags](#_6u4rld71c9in) 7

[Parent and Child Tag](#_7fp8d02ov7sa) 8

[Notes](#_ko4icbemzu94) 8

[**CSS**](#_erl2fntjpl7e) **9**

[Linking of CSS and HTML file](#_8x5htfvglhn2) 9

[Properties List](#_dmnqen3h3ekt) 9

[Pseudo Class](#_6a4kf19sj1yz) 10

[Notes](#_cqzthikx6thb) 10

[**960 Grid**](#_bd0pbrq438wb) **11**

[Containers](#_k22sjd3kzty3) 12

[Grids](#_zea4cgovzda2) 13

[Notes](#_zi49gfftszge) 13

# 

# 

# 

# Introduction

The following tutorial will give a basic start to HTML5, CSS3 and 960 Grid to develop a knack of website building in beginners.

\**If you have some knowledge about any of the above topic then this tutorial is not for you.*\*

Ever wondered how to get started with developing websites, like what is the foremost step that you should take to start building one of those beauty. Well, if you are one of those who want to learn how to build a decent looking website then I request you to read on.

*Further note: Here I am assuming that the reader is familiar with the terms like website, webpage, language etc.*

The language that makes up the basic structure/skeleton of the webpages is HTML. It is used for structuring content of the webpage. When it comes to presentation, i.e. how the webpage should look like, CSS comes into picture. So these two languages combine to give a presentable website. 960 grid is a CSS framework that provides commonly used dimensions based on a certain width to elegantly structure and present the content of the website altogether.

# Tools Required

You will need a text editor to write the code (html and css file) and a web browser to render that code. Any basic text editor will do, even Notepad on Windows. But my recommendation is, you go for a text editor designed for coding specifically like Atom, Sublime Text, Notepad++ etc. For web browsers, you can use Google Chrome or Mozilla Firefox.

# Getting Started

So, we will be starting with HTML5, the fifth version of HTML. It stands for HyperText Markup Language. It is the language that we use for giving the basic structure to the web page. We can say that it decides what content should be present on the web page and what is the meaning associated to it. It does not style our page. Styling is the job of CSS and we will be learning it after HTML.

# HTML

Open a code editor and write something on it, anything. Now, save that file as “.html”. “.html” files are compatible with browsers and you can open one with Chrome. You will see the content written in the html file written on browser.

That is the basic functionality but that is not what we will be doing. We need to attach meaning to the content of the website.

Basics of HTML is plain text, though we need something more to apply *meaning* to what is written. **Tags** help us do that. Tags surround the content and apply meaning to it. They are of two types : opening and closing tag.

*Let us write a basic hello world program to understand more about structure of html page and about tags*

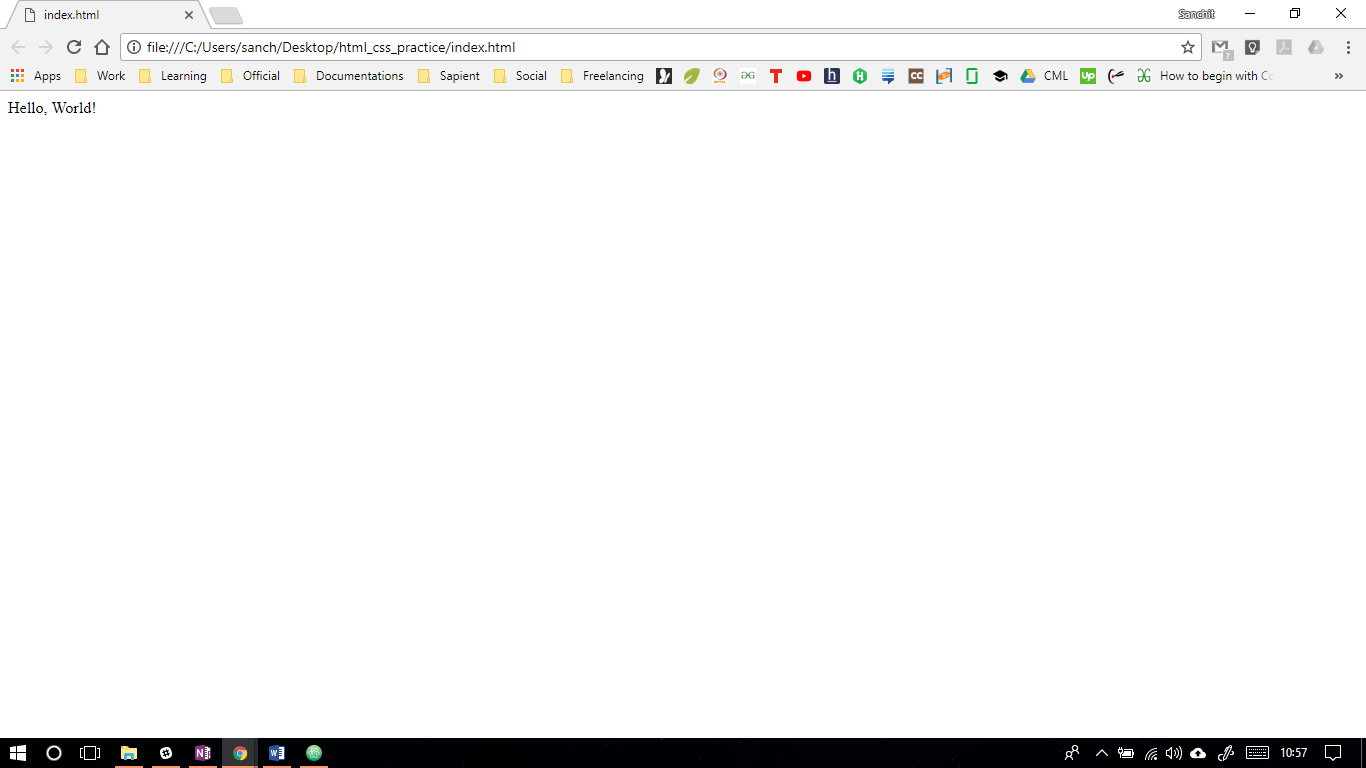
*<!doctype html>  
<html>  
  
<head>  
 <meta charset="utf-8">  
 <meta name="viewport" content="width = device-width, initial-scale = 1, shrink-to-fit=no">  
</head>  
  
<body>  
 Hello, World !  
</body>  
  
</html>*

Here, everything enclosed within < and > is a tag. It is not picked up “as it is” by the browser and is something that applies meaning to content of the webpage. For example, consider a small portion of the code above.

*<body>  
 Hello, World !  
</body>*

Body tag, having an opening and closing tag, signifies the content written within, “Hello, World” in this case is the body of the webpage. The thing here to note is, tags only apply semantic meaning to the content of the page.

The output of the above code is shown below:



## Structure of the code

An html document is basically divided into two parts : head and body . Head consists of all the tags that are used to represent the information about the web page. Body consists of content of the website.

Head tags wraps title, meta and associated tags. Title is used to define the head of the browser window. The other information like character set, responsiveness information, device width setting, etc comes under meta tag. Some description about the tags used in the above example is given below.

**<!doctype html>** : The very first thing in the html document, before the <html>. It is a instruction to the browser that the webpage has been written in html5.

**<html> </html>** : Web page’s root tag. It wraps everything around it.

**<head> </head>** : Meta-data about the page is placed here.

**<meta>** : information about the HTML document, that cannot be covered by title, style, base, link.

**<meta charset = “utf-8”> :** Defines the character set of the html page.

**<title> </title>** : Used to define title of web-page in browser window.

**<body> </body>** : Content of the website comes inside body tag.

While going through the tags, you will notice that they are actually of two types : the ones having an opening and closing tag, content being enclosed between the two; and the others having only one tag such as <br/>.

**The properties** of the tags are called attributes. They are the extra bits of information that the tags have. These properties can be set to different values.

For Example : meta is a tag and charset is its attribute.

The opening and closing tag along with its contents make an element.

For Example :

*<body>  
 Hello, World !  
</body>*

This is a body element.

Some of the other most used relevant tags are given below.  
What you need to do is apply all the following tags to the above template using some of the outside help, to get along with all the attributes. You can take the help of website http://www.htmldog.com . It has a handy reference to all the tags down there. You can see how the examples work out in the most exhaustive scenario.

|  |  |  |  |
| --- | --- | --- | --- |
| **Opening Tag** | **Closing Tag** | **Use** | **Attributes and possible values** |
| <h1> to  <h6> | </h1> to  </h6> | Heading Tags . h1 represents the largest length heading whereas h6, the smallest. |  |
| <p> | </p> | Paragraph Tag |  |
| <em> | </em> | To give emphasis to content inside it. |  |
| <i> | </i> | To make the text italics |  |
| <strong> | </strong> | To make the text strong |  |
| <b> | </b> | To make the text bold |  |
| <br> |  | Line-Break |  |
| <hr> |  | To apply horizontal ruler | **Height, Width, Color** |

In the above given table are some of the commonly used text-formatting tags. They are easy to use and will be fun to try out.

## Lists

Basically, there are three types of lists that can be represented using html.

1. Unordered List
2. Ordered List
3. Definition List

Example of Unordered List :-

*<ul>*

*<li> Pen </li>*

*<li> Paper </li>*

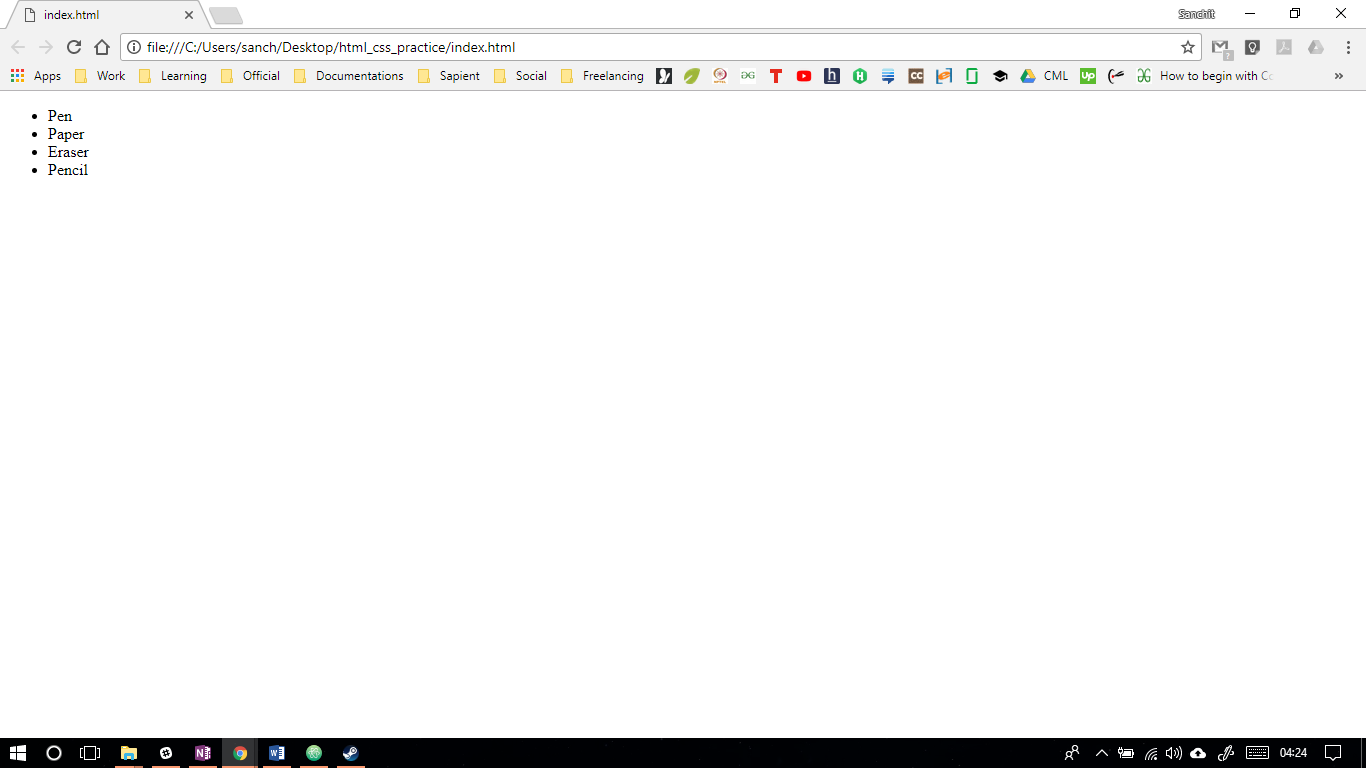
*<li> Eraser </li>*

*<li> Pencil </li>*

*</ul>*

<li> is used to represent each list item.

The output will look like this.



Similarly, for ordered lists, you can change the tag to <ol> and there will be numbering in place of bullets.

Definition lists are used in scenarios where you need to give a definition. Tag <dt> is used to signify definition term and <dd> is used to signify definition detail.

Example -

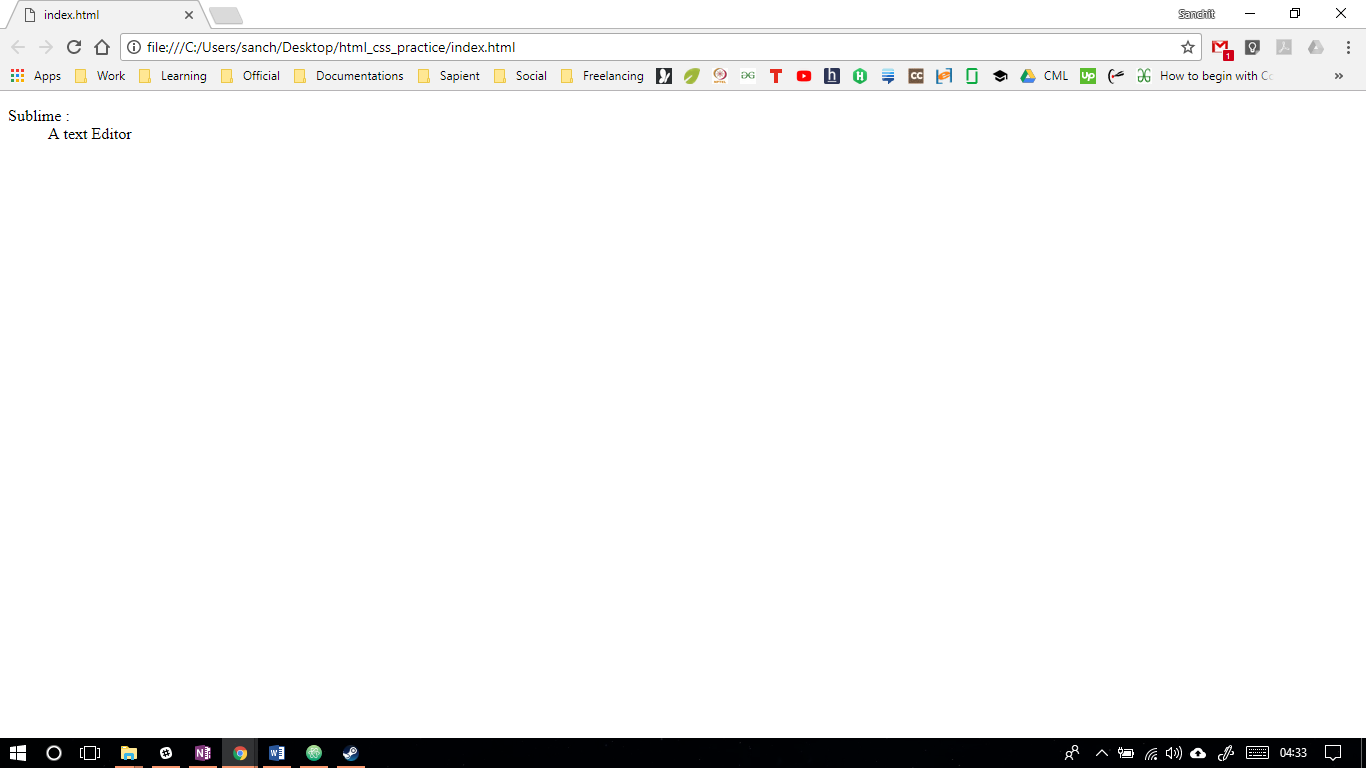
*<dl>*

*<dt> Sublime : </dt>*

*<dd> A text editor </dd>*

*</dl>*

The output will look like this.



## Other Tags

**Anchor tag**

<a> </a> : Anchor tag is used to give a link to another html page.  
Attribute list : href, target

Example : <a href = “<https://www.google.com>” target = “\_blank”> Google </a>

**Image Tag**  
<img> : To add an image  
Attributes : src, alt, height, width

Example : <img src = “image.png” alt = “Image” height = “300px” width = “200px”>

**<blockquotes> </blockquotes>** : To enclose quotes

**<div> </div>** : A generic container for block in HTML  
Attributes : id , class

The main difference between id and class is we use id to reference to an element that we will be using one time. We use classes if we will be referencing that element multiple times in future to give all of them same styling.

**<span> </span>** : A generic container for an element or inline element in HTML  
Attributes : id , class

The main difference between div and span is that we use div to contain a block of elements while we use span to reference a single elements.

**<style> </style>** : To give internal styling to text document

**<table> </table>** : To make a table   
**<tr> </tr>** : To make a row  
**<td> </td>** : Table data  
**<th> </th>** : Table heading

**<form> </form>** : To create a form  
Attributes : action, method  
**<label> </label>** : To write a label to input type  
Attributes : for  
**<input> </input>** : To make an element that takes some kind of input. We can self-close it.  
Attributes : type, value, id, name  
**<textarea> </textarea>** : To enter a long passage in the input  
Attributes : rows, cols

**<header> </header>** : To apply header  
**<footer </footer>** : To apply footer

## Parent and Child Tag

You will notice that some tags are applied inside other opened tags.

For example : *<html>*

*<head>*

*<body>*

*</body>*

*</head>*

*</html>*

Here, <html> is the parent tag and <head> is its child tag. Same is the relation in html and body. But <body> and <head> are siblings.

## Notes

One thing to note is that always name the first file in the website folder index.html as web servers by default access index.html

The other thing that you should keep in mind is to always maintain a directory structure. Keep relevant html, css and image files in relevant folders.

# CSS

When it comes to formatting the written text, CSS is the language. It is only used for styling the content of the document.

CSS can be applied in 3 ways :-

1. Inline CSS :-

Example - < a href = “#” style = “color : red ; underline : none;”>Link </a>

2) Internal CSS :-

In head tag give those styling elements as

<style>

a{

Color : red;

Underline : none;

}

</style>

3) External CSS :-

Make an css file and give its link to the head of the html file so that html file can render it.

## Linking of CSS and HTML file

In head section of html file, you need to add

*<link type = “text/css” rel = "stylesheet" href="style.css">*

## Properties List

CSS takes the shape of property name and property value.

Some common examples are :

**Color** : red or #236589 or rgb(100,100,100) or [rgba()--?];

**Text-decoration** : underline or none;

**Font-family** : (use html safe fonts) Helvetica, Arial, sans-serif;

**Margin-/left/right/top/bottom/** : px or % or auto;

**Padding-/left/right/top/bottom/** : px or % or auto;

**Width** : px or %

**Height** : px or % : Don't use height generally, let the content decide height

**Text-align** : center or left or right

Example :- h1,a { } : means : apply property to h1 and a

Example :- h1 a { } : means : apply property to a of h1

Example :-

Find a list item , go to its parent and then find the first item of the sect id.

#sect li:nth-child(1){

color : yellow;

}

ol:first-of-type li:nth-child(1){

color : yellow;

}

**Float** : left or right

**Background** : red or URL(/path/) (no-repeat/repeat-x/repeat-y);

**List-style** : none;

**Min-width** : px;

**Max-width** : px;

**Min : height** : px;

**Max: height** : px;

**Text-indent** : px;

**Display** : inline or block;

**Font-size** : 30px;

**Text-transform** : uppercase / lowercase;

**Text-shadow** : (x-offset y-offset blur color) - accepts comma separated values for multiple shadows

**Font-variant** : small-caps / normal;

## Pseudo Class

Example : a:hover{

}

H1:after{ }

Here hover and after are pseudo classes that determine styling on mouse hovering and content to display after an element respectively.

## Notes

When you use percentages, it decides the percentage of window rather than the length in pixels.

When we apply floats, it takes those elements out of the flow of the document. Block display is not honored here. Fix it by using a footer so that it triggers the wrapper (as seen in video) and use the property clear.

Example : footer{

Clear : both;

}

Or you can fix it by using overflow : hidden in main wrapper class

Example : .wrap{

Overflow : hidden;

}

By default list items have a display of block. If we do inline, they will come in a line as long as width is less than the width of wrapper class.

By default label have a display of inline.

Cursor : pointer; (//indication to make it clickable - not actually clickable - just changes the pointer to finger)

By default, heading elements have a display of block. If we change it to inline, width of background will be equal to width of text.

Monospace font is good for embedding code. Other are cursive, Arial, Helvetica, etc.

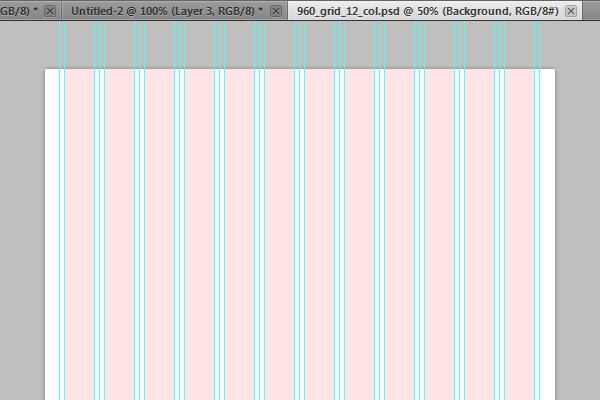
Letter-spacing : px;

# 960 Grid

960 grid is an effort to streamline web development by providing commonly used dimensions based on width of 960 pixels. 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 gutters between columns.

960 Grid is a CSS framework (predefined classes ready to be used) which makes it easy to position your elements because all the structure of the page is pre defined. You just need to have the vision and you can place it instantly and easily.

Here's how the structure looks like.



Now, this is one of 12 column layouts that 960 grid framework provides.

The package comes with a lot of css files, you can choose any of your liking for eg

12 column grid, 16 column and 24 column.

We also recommend to other files to be linked :

1. 960.css : It defines the basic container and grid layout
2. reset.css : This css files reset all the margins, padding and positions with which it gets same for every browser be it firefox, safari, chrome . So that you don't have to worry about every browser.
3. text.css : It is the built in typography which has decent fonts already applied. You can override those with your own css sheet.

Never, change the css files of the framework. They are tried and tested. If you want to apply your own styles, override that particular element in your own stylesheet.

## **Containers**

Now, it is up to you if you want 16 columns, 24 or 12.

So, to apply a 12 column grid, this is how we do it

For Example :-

In order to choose the container we use :-

*<div class class = "container\_12">*

*This is how you apply 12 grid system.*

*</div>*

## **Grids**

That's the beauty of frameworks, it makes everything so easy. Let's say you are making a huge header and you want it to take the whole width.

All you have to do is :

*<div class = "container">*

*<div class = "grid\_12">*

*<!-- insert header here-->*

*</div>*

*</div>*

If you want 2 paragraphs taking half of page. Then use grid\_6 for both of them in the same container. This way, they will take up exactly half of the page. Now, if you for instance give grid\_6 to one and grid\_ 7 to the other, then it will automatically move to the next line. This is a very nice way to create even pages.

## Notes

960 Grid is basically a framework which eases our work. Some like o use it and some don’t. It basically eases the task to choose the grids and containers using predefined classes. All of this makes positioning a hell lot easier.

That is all about HTML, CSS and 960 Grid. Hope you enjoyed it!