UNIT-I

Definition

- Open source software (OSS) is software that is distributed with its source code, making it available for use, modification, and distribution with its original rights.
- Source code is the part of software that most computer users don't ever see; it's the code computer programmers manipulate to control how a program or application behaves.
- Programmers who have access to source code can change a program by adding to it, changing it, or fixing parts of it that aren't working properly.
- OSS typically includes a license that allows programmers to modify the software to best fit their needs and control how the software can be distributed.

Why do people prefer using open source software?

People prefer open source software to proprietary software for a number of reasons, including:

Free redistribution: The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

Source code: Open source software (OSS) is software that is distributed with its source code, making it available for use, modification, and distribution with its original rights.

Control. Many people prefer open source software because they have more control over that kind of software. They can examine the code to make sure it's not doing anything they don't want it to do, and they can change parts of it they don't like. Users who aren't programmers also benefit from open source software, because they can use this software for any purpose they wish—not merely the way someone else thinks they should.

Training. Other people like open source software because it helps them become better programmers. Because open source code is publicly accessible, students can easily study it as

they learn to make better software. Students can also share their work with others, inviting comment and critique, as they develop their skills. When people discover mistakes in programs' source code, they can share those mistakes with others to help them avoid making those same mistakes themselves.

Security. Some people prefer open source software because they consider it more secure and stable than proprietary software. Because anyone can view and modify open source software, someone might spot and correct errors or omissions that a program's original authors might have missed. And because so many programmers can work on a piece of open source software without asking for permission from original authors, they can fix, update, and upgrade open source software more quickly than they can proprietary software.

Stability. Many users prefer open source software to proprietary software for important, long-term projects. Because programmers publicly distribute the source code for open source software, users relying on that software for critical tasks can be sure their tools won't disappear or fall into disrepair if their original creators stop working on them.

Community. Open source software often inspires a community of users and developers to form around it. That's not unique to open source; many popular applications are the subject of meetups and user groups.

Advantages of open-source software?

Absolute transparency

What you see is what you get: the point of open-source software is the transparency of its code. As a matter of fact, many other open-source advantages stem from the absolute visibility of its code.

Since users of open-source software can see the code, they are inclined to trust the software provider more. Furthermore, since the code is publicly available,

Better security

Many developers working on the source code means that they are likely to spot security problems quickly. Anyone can fix bugs or upgrade the code, without relying on a proprietary vendor.

Better quality

With more developers poring over the code, open source is typically considered to have fewer flaws and better quality than standard, commercially developed software.

More control

Many people prefer open source because they have more control over the code or the ways they can use it. Proprietary software is fully controlled by its developers. In contrast, all those who use it control open source software. This makes it easy to customise and tailor the product for your exact needs and purpose.

No vendor dependence

With open source, you are not locked into a relationship with a particular software company. As the code is out in the open, you are free to take your business elsewhere to find the support you need, when you need it.

Easier licence management

If you're using proprietary software, you must make sure that all your servers, desktops or laptops have the right type and number of commercial licences. With open source licences, you can generally use the software as many times, in as many places, as you wish. Depending on the licence type, different conditions may apply - read about open source licensing and legal issues.

Disadvantages

1. Complicated

It is not as user-friendly as the ones that are closed. To use this software, you must have a basic understanding of technology.

2, Security risk

Despite the defects having been detected, there is a risk of assaults because the attackers have access to the source code.

3. No support

If you run across an issue, there is no customer support available to assist you.

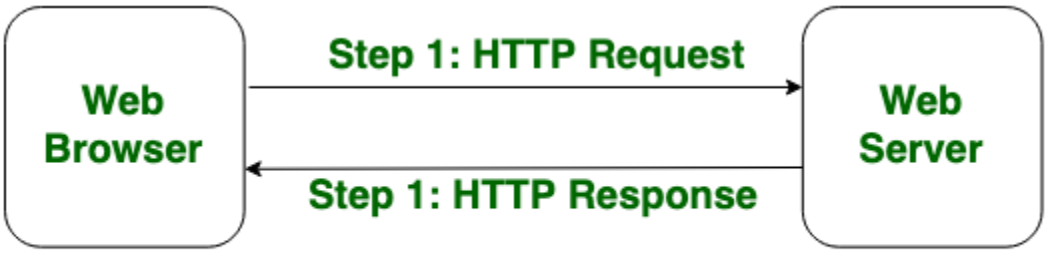
some examples of OSS?

Some famous examples of Open-source products are :

- **Operating systems –**
  Android, Ubuntu, Linux
- **Internet browsers –**
  Mozilla Firefox, Chromium
- **Integrated Development Environment (IDEs) –**
  Vs code (Visual Studio Code), Android Studio, PyCharm, Xcode

1.1Introduction to static and dynamic web content

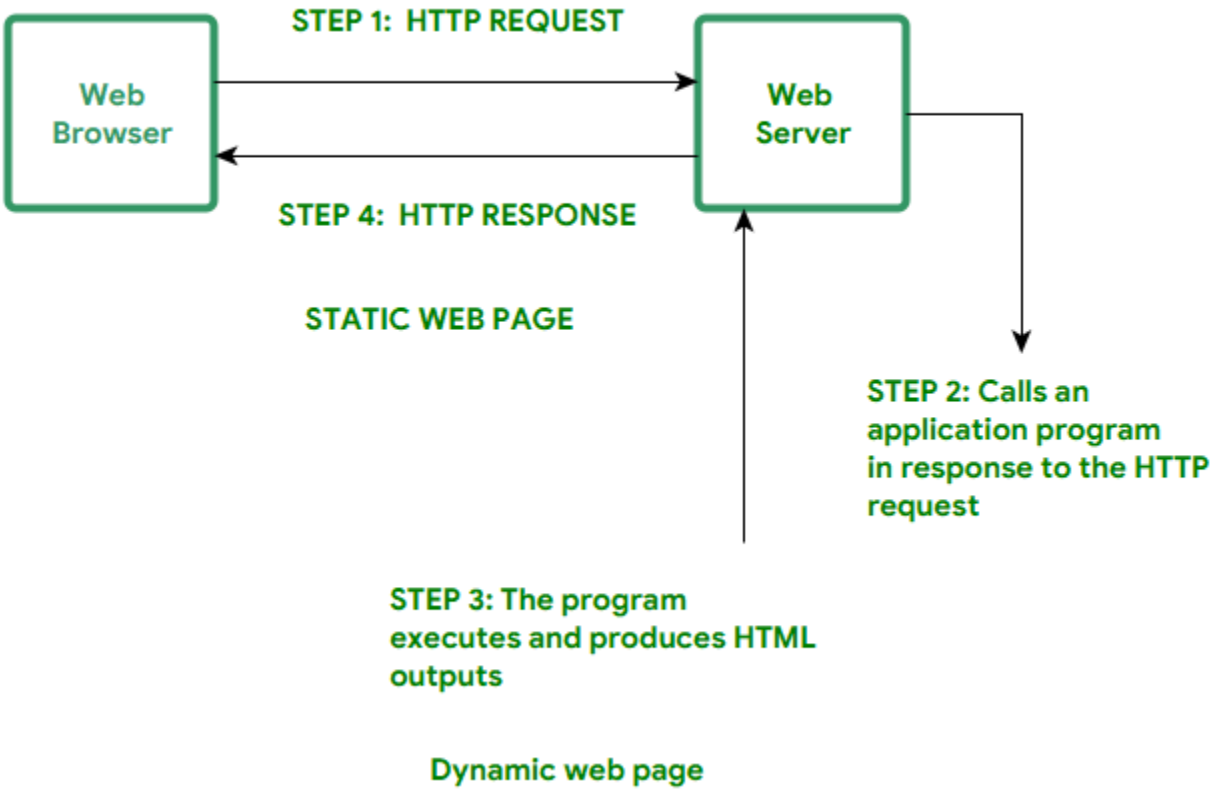
Static Web pages: Static Web pages are very simple. It is written in languages such as HTML, JavaScript, CSS, etc. For static web pages when a server receives a request for a web page, then the server sends the response to the client without doing any additional process. And these web pages are seen through a web browser. In static web pages, Pages will remain the same until someone changes it manually.



Static Web Page

Dynamic Web Pages: Dynamic Web Pages are written in languages such as CGI, AJAX, ASP, ASP.NET, etc. In dynamic web pages, the Content of pages is different for different visitors. It takes more time to load than the static web page. Dynamic web pages are used where the information is changed frequently,

for example, stock prices, weather information, etc.



1.2 HTTP and HTML:

- HTTP, on the other hand, stands for Hypertext Transfer Protocol.
- It is a means of data communication for the World Wide Web.
- It is an application protocol for distributed, collaborative, hypermedia information systems.
- **HTTP** is a communication standard governing the requests and responses that take place between the browser running on the end user's computer and the web server.
- The server's job is to accept a request from the client and attempt to reply to it in a meaningful way, usually by serving up a requested web page—that's why the term *server* is used.
- The natural counterpart to a server is a *client*, so that term is applied both to the web browser and the computer on which it's running.
- They serve different roles in ensuring that the requests and responses are correctly transferred between the client and server. Typically, they use the Internet to send this information.
- HTML stands for HyperText Markup Language. It is a well known markup language used to develop web pages. It has been around for a long time and is commonly used in webpage design..

1.3 The Request/Response Procedure

At its most basic level, the request/response process consists of a web browser asking the web server to send it a web page and the server sending back the page. The browser then takes care of displaying the page (see Figure 1-1).

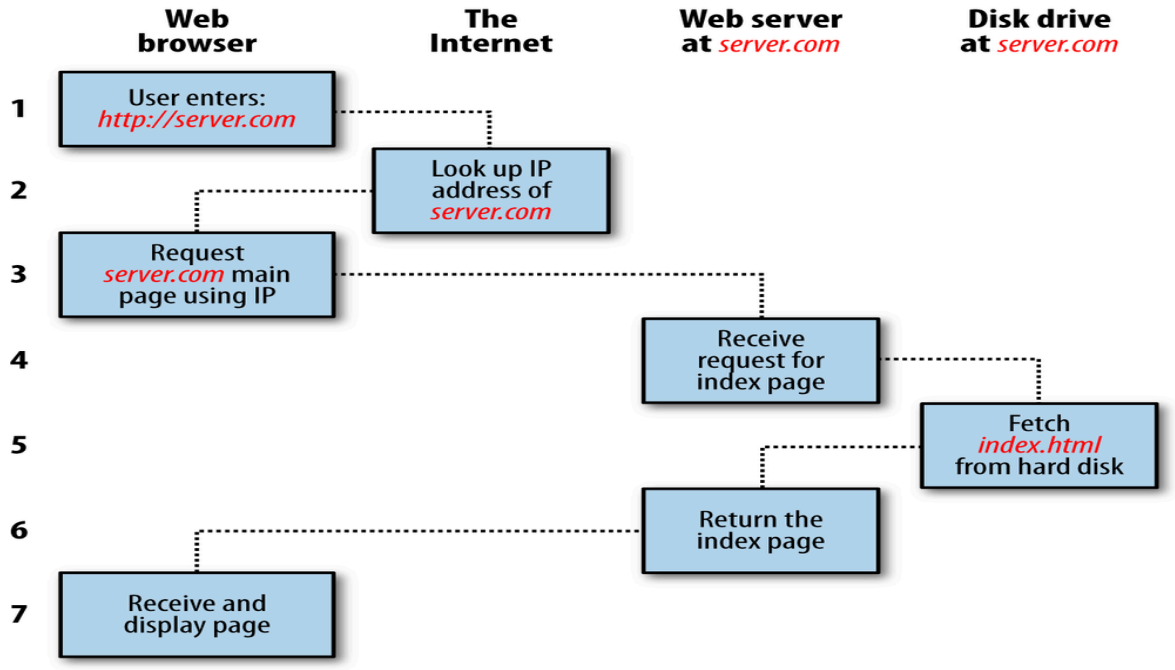


Figure 1-1. The basic client/server request/response sequence

Each step in the request and response sequence is as follows:

1. You enter *http://server.com* into your browser's address bar.
2. Your browser looks up the IP address for *server.com*.
3. Your browser issues a request for the home page at *server.com*.
4. The request crosses the Internet and arrives at the *server.com* web server.
5. The web server, having received the request, looks for the web page on its hard disk.
6. The web page is retrieved by the server and returned to the browser.
7. Your browser displays the web page.

For an average web page, this process takes place once for each object within the page: a graphic, an embedded video or Flash file, and even a CSS template.

For dynamic web pages, the procedure is a little more involved, because it may bring both PHP and MySQL into the mix (see Figure 1-2).

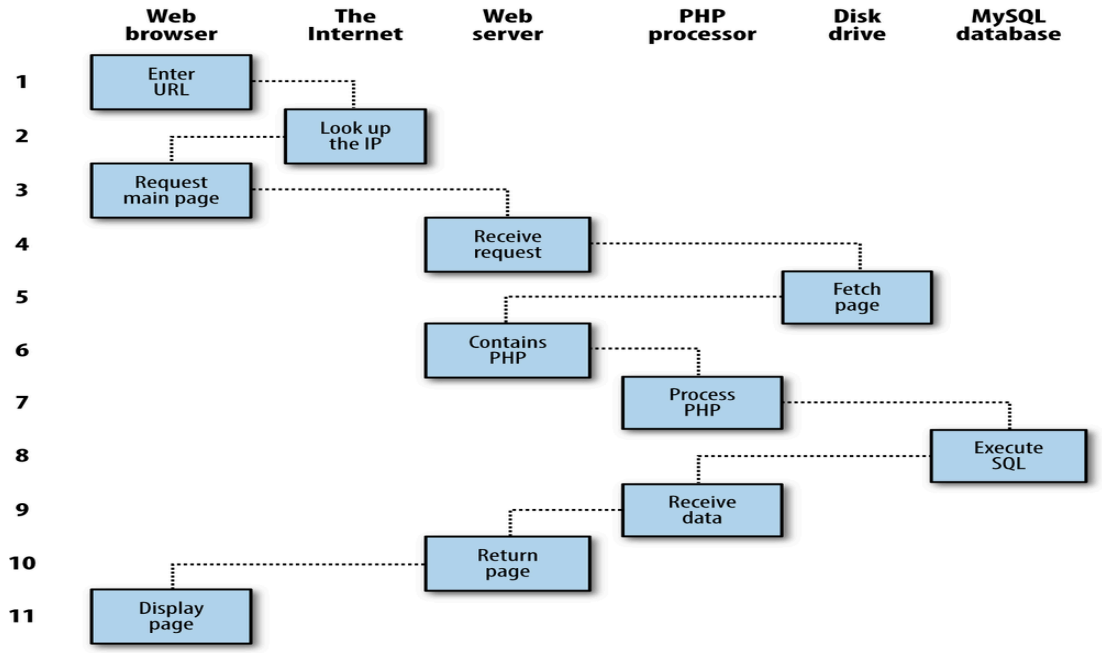


Figure 1-2. A dynamic client/server request/response sequence

Here are the steps for a dynamic client/server request/response sequence:

1. You enter *http://server.com* into your browser's address bar.
2. Your browser looks up the IP address for *server.com*.
3. Your browser issues a request to that address for the web server's home page.
4. The request crosses the Internet and arrives at the *server.com* web server.
5. The web server, having received the request, fetches the home page from its hard disk.
6. With the home page now in memory, the web server notices that it is a file incorporating PHP scripting and passes the page to the PHP interpreter.
7. The PHP interpreter executes the PHP code.
8. Some of the PHP contains MySQL statements, which the PHP interpreter now passes to the MySQL database engine.
9. The MySQL database returns the results of the statements back to the PHP interpreter.
10. The PHP interpreter returns the results of the executed PHP code, along with the results from the MySQL database, to the web server.
11. The web server returns the page to the requesting client, which displays it.

1.5 Introduction to HTML 5:

- HTML stands for Hyper Text Markup Language. It is used to design web pages using markup language.
- HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. Markup language is used to define the text document within tag which defines the structure of web pages.
- HTML 5 is the fifth and current version of HTML. It has improved the markup available for documents and has introduced application programming interfaces(API) and Document Object Model(DOM).

The HTML 5 canvas:

- The HTML <canvas> element is used to draw graphics, on the fly, via JavaScript.
- The <canvas> element is only a container for graphics. You must use JavaScript to actually draw the graphics.
- Canvas has several methods for drawing paths, boxes, circles, text, and adding images.
- The HTML <canvas> element is used to draw graphics on a web page.
- The graphic to the left is created with <canvas>. It shows four elements: a red rectangle, a gradient rectangle, a multicolor rectangle, and a multicolor text.

Browser Support

- The numbers in the table specify the first browser version that fully supports the <canvas> element.

Element					
<canvas>	4.0	9.0	2.0	3.1	9.0

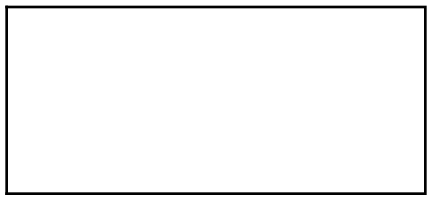
Canvas Examples

- A canvas is a rectangular area on an HTML page. By default, a canvas has no border and no content.

Here is an example of a basic, empty canvas:

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">
</canvas>
```

Output:



1.6 HTML5 AUDIO AND VIDEO:

Audio on the Web

- Before HTML5, audio files could only be played in a browser with a plug-in (like flash).
- The HTML5 <audio> element specifies a standard way to embed audio in a web page.

Browser Support

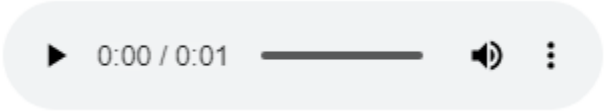
- The numbers in the table specify the first browser version that fully supports the <audio> element.

Element					
<audio>	4.0	9.0	3.5	4.0	10.5

The HTML <audio> ElementTo play an audio file in HTML, use the <audio> element:

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

Output:



Playing Videos in HTML

- Before HTML5, a video could only be played in a browser with a plug-in (like flash).
- The HTML5 <video> element specifies a standard way to embed a video in a web page.

Browser Support

- The numbers in the table specify the first browser version that fully supports the <audio> element.

Element					
<audio>	4.0	9.0	3.5	4.0	10.5

The HTML <video> ElementTo show a video in HTML, use the **<video>** element:

```
<html>
<body>
<video width="320" height="240" controls>
<source src="movie.mp4" type="video/mp4">
<source src="movie.ogg" type="video/ogg">
  Your browser does not support the video tag.
</video>
</body>
</html>
```

Output:

1.7 Introduction to CSS:

- CSS stands for **"Cascading style sheets"**.It enables you to define rules on how an element should **appear**.
- HTML defines the **structure** of a webpage while CSS defines the **appearance** and style of a webpage.
- CSS is easy to learn and understood but it provides powerful control over the presentation of an HTML document.

CSS Rules

- A *CSS rule* is a grouping of one or more CSS properties which are to be applied to one or more target HTML elements.
- A CSS rule consists of a CSS selector and a set of CSS properties.
- The CSS selector determines what HTML elements to target with the CSS rule. The CSS properties specifies what to style of the targeted HTML elements.

Here is a CSS rule example:

```
div {
   border    : 1px solid black;
   font-size : 18px;
}
```

- This example creates a CSS rule that targets all div elements, and the set the CSS properties border and font-size for the targeted elements.
- The CSS selector part a CSS rule is the part before the first {. In the example above it is the div part of the CSS rule. The CSS properties are listed inside the { ... } block.
- CSS rules have to be specified inside either a style element or inside an external CSS file.

Style types:

CSS in three ways in your HTML document:

- **External Style Sheet** − Define style sheet rules in a separate .css file and then include that file in your HTML document using HTML <link> tag.

- **Internal Style Sheet** − Define style sheet rules in header section of the HTML document using <style> tag.
- **Inline Style Sheet** − Define style sheet rules directly along-with the HTML elements using **style** attribute.

External Style Sheet

- If you need to use your style sheet to various pages, then its always recommended to define a common style sheet in a separate file.
- A cascading style sheet file will have extension as **.css** and it will be included in HTML files using <link> tag.

Example

Consider we define a style sheet file **style.css** which has following rules:

```
.red {
  color: red;
}
.thick {
  font-size:20px;
}
.green {
  color:green;
}
```

External CSS file in our following HTML document:

```
<html>
<head>
<title>HTML External CSS</title>
<link rel = "stylesheet" type = "text/css" href = "/html/style.css">
</head>
<body>
<p class = "red">This is red</p>
<p class = "thick">This is thick</p>
<p class = "green">This is green</p>
<p class = "thick green">This is thick and green</p>
</body>
</html>
```

Result:

This is red

This is thick

This is green

This is thick and green

Internal Style Sheet

- If you want to apply Style Sheet rules to a single document only, then you can include those rules in header section of the HTML document using <style> tag.
- Rules defined in internal style sheet overrides the rules defined in an external CSS file.

Example

```
<html>
<head>
<title>HTML Internal CSS</title>

<style type = "text/css">
        .red {
           color: red;
        }
        .thick{
           font-size:20px;
        }
        .green {
           color:green;
        }
</style>
</head>

<body>
<p class = "red">This is red</p>
<p class = "thick">This is thick</p>
<p class = "green">This is green</p>
<p class = "thick green">This is thick and green</p>
</body>

</html>
```

This will produce the following result:

This is red
This is thick
This is green
This is thick and green

Inline Style Sheet

- We apply style sheet rules directly to any HTML element using **style** attribute of the relevant tag.
- This should be done only when you are interested to make a particular change in any HTML element only.

Example

The HTML elements using **style** attribute of those elements.

```
<html>
<head>
<title>HTML Inline CSS</title>
</head>
<body>
<p style = "color:red;">This is red</p>
<p style = "font-size:20px;">This is thick</p>
<p style = "color:green;">This is green</p>
<p style = "color:green;font-size:20px;">This is thick and green</p>
</body>
</html>
```

Output:

This is red

This is thick

This is green

This is thick and green

CSS Selector

- **CSS selectors** are used *to select the content you want to style*. Selectors are the part of CSS rule set.
- CSS selectors select HTML elements according to its id, class, type, attribute etc.

There are several different types of selectors in CSS.

1. CSS Element Selector
2. CSS Id Selector
3. CSS Class Selector
4. CSS Universal Selector

5. CSS Group Selector

1) CSS Element Selector

The element selector selects the HTML element by name.

```
<html>
<head>
<style>
p{
   text-align: center;
   color: blue;
}
</style>
</head>
<body>
<p>This style will be applied on every paragraph.</p>
<p id="para1">Me too!</p>
<p>And me!</p>
</body>
</html>
```

Output:

This style will be applied on every paragraph.
Me too!
And me!

2. CSS Id Selector

- The id selector selects the id attribute of an HTML element to select a specific element.
- An id is always unique within the page so it is chosen to select a single, unique element.It is written with the hash character (#), followed by the id of the element.

Now take an example with the id **"para1".**

```
<html>
<head>
<style>
#para1 {
   text-align: center;
   color: blue;
}
</style>
```

```
</head>
<body>
<p id="para1">Hello World</p>
<p>This paragraph will not be affected.</p>
</body>
</html>
```

Output:

Hello World

3. CSS Class Selector

- The class selector selects HTML elements with a specific class attribute. It is used with a period character . (full stop symbol) followed by the class name.

  Now take an example with a class **"center".**

```
<html>
<head>
<style>
.center {
   text-align: center;
   color: blue;
}
</style>
</head>
<body>
<h1 class="center">This heading is blue and center-aligned.</h1>
<p class="center">This paragraph is blue and center-aligned.</p>
</body>
</html>
```

Output:

This heading is blue and center-aligned.

This paragraph is blue and center-aligned.

4. CSS Universal Selector

The universal selector is used as a wildcard character. It selects all the elements on the pages.

```
<html>
<head>
<style>
* {
  color: green;
```

```
  font-size: 20px;
}
</style>
</head>
<body>
<h2>This is heading</h2>
<p>This style will be applied on every paragraph.</p>
<p id="para1">Me too!</p>
<p>And me!</p>
</body>
</html>
```

Output:

This is heading

This style will be applied on every paragraph.

Me too!

And me!

5. CSS Group Selector

- The grouping selector is used to select all the elements with the same style definitions.
- Grouping selector is used to minimize the code. Commas are used to separate each selector in grouping.

Now CSS code without group selector.

```
h1 {
   text-align: center;
   color: blue;
}
h2 {
   text-align: center;
   color: blue;
}
p {
   text-align: center;
```

```
   color: blue;
}
```

NOW you need to define CSS properties for all the elements. It can be **grouped** in following ways:

```
h1,h2,p {
   text-align: center;
   color: blue;
}
```

Now see the full example of CSS **group** selector.

```
<html>
<head>
<style>
h1, h2, p {
   text-align: center;
   color: blue;
}
</style>
</head>
<body>
<h1>Hello World</h1>
<h2>Hello World (In smaller font)</h2>
<p>This is a paragraph.</p>
</body>
</html>
```

Output:

Hello World

Hello World (In smaller font)

This is a paragraph.

CSS COLORS

- CSS uses color values to specify a color. Typically, these are used to set a color either for the foreground of an element (i.e., its text) or else for the background of the element.
- They can also be used to affect the color of borders and other decorative effects.

Format	Syntax	Example
Hex Code	#RRGGBB	p{color:#FF0000;}
Short Hex Code	#RGB	p{color:#6A7;}
RGB %	rgb(rrr%,ggg%,bbb%)	p{color:rgb(50%,50%,50%);}
RGB Absolute	rgb(rrr,ggg,bbb)	p{color:rgb(0,0,255);}
keyword	aqua, black, etc.	p{color:teal;}

CSS Colors - Hex Codes

- A hexadecimal is a 6 digit representation of a color. The first two digits(RR) represent a red value, the next two are a green value(GG), and the last are the blue value(BB).
- A hexadecimal value can be taken from any graphics software like Adobe Photoshop.
- Each hexadecimal code will be preceded by a pound or hash sign '#'. Following are the examples to use Hexadecimal notation.

Color	Color HEX
	#000000(black)
	#FF0000(Red)
	#00FF00(Green)

CSS Colors - Short Hex Codes

- This is a shorter form of the six-digit notation. In this format, each digit is replicated to arrive at an equivalent six-digit value. For example: #6A7 becomes #66AA77.

Color	Color HEX

	#000(black)
	#F0(Red)
	#0F0(Green)

CSS Colors - RGB Values

- This color value is specified using the **rgb( )** property. This property takes three values, one each for red, green, and blue. The value can be an integer between 0 and 255 or a percentage.
- All the browsers does not support rgb() property of color so it is recommended not to use it.

Following is the example to show few colors using RGB values.

Color	Color RGB
	rgb(0,0,0)(Black)
	rgb(255,0,0)(red)
	rgb(0,255,0)(Green)
	rgb(0,0,255)(Blue)

Building Color Codes

- We can build millions of color codes using our Color Code Builder. Check our **HTML Color Code Builder**. To use this tool, you would need a Java Enabled Browser.

Browser Safe Colors

- Here is the list of 216 colors which are supposed to be most safe and computer independent colors. These colors vary from hexa code 000000 to FFFFFF.
- These colors are safe to use because they ensure that all computers would display the colors correctly when running a 256 color palette