

Chapter One Introduction

For you to be among the hackers community, which their main aim is to build and contribute effectively to the growth of the web and Internet, it is pertinent for you to know the rudiments of HTML, which is a MARKUP Language for creating web pages and css (Cascading Style Sheet) which is the presentation/beautification language of the Web.

This text is prepared to address in detail, the rudiment of website design and development. This comprises of the following programming tools; HTML, XHTML, HTML5, CSS, Fundamentals of JavaScript, Introduction to Bootstrap Framework for responsive web design, version control (Setting up of Git and Github), introduction to algorithm and more as well as prepares you for your ICT career in web development.

The goal of this book is to provide you with the fundamental building blocks of website designing and development that are very relevant for those interested in web development and programming as well as computer science students in higher institutions by giving you a solid working foundation from which you can grow your knowledge over time.

The production of this textbook based on intense research in many materials as well as experiences in executions of projects. It is just a fundamental collections for those interested in starting a career in computer and all computer science students who want to sharpen their understanding of core web development.

Web Design Fundamentals lead you into the detailed approach of creating a web page to a website. Technologies cover includes; XHTML, HTML5, CSS 2.0 and 3.0, JavaScript fundamentals, Bootstrap and fundamental of Git/GitHub version control with real life examples. The text is a great resource to all Computer Science Students especially beginners who want to make the best out of their career as well as advance students who can use it as a quick reference material.

If you are not a computer science student and want to venture into the art of web programming, grab a copy of this text to lay a great foundation in your new adventurous journey in the fantastic world of web programming.

The text is divided into the following sections; xhtml, CSS, JavaScript Fundaments, HTML5 and Introduction to Bootstrap and Version Control (Git/GitHub).

The text concludes with Appendices on HTML colour codes and HTML Entities for your quick references.

PINFOICT ACADEMY

1.2 Web Related Terms

The Internet

Since the Web Design, we want to discuss functions on the Internet, let us briefly discuss Internet. Internet is a worldwide collection of computer networks of **networks** sharing digital information via a common set of networking and software protocols - Internet protocol suite (Transmission Control Protocols (TCP)/Internet Protocols (IPs)) to link devices across the globe. It is a *large network* that consists of private, public, academic, business, and government networks of local to global scope, linked by a broad array of electronic, wireless, and optical networking technologies.

WWW

WWW stands for World Wide Web known as a web, which is a network of computers able to exchange text, graphics, and multimedia information via the Internet. It is the universe of network-accessible information, an embodiment of human knowledge. It uses Hypertext Transfer Protocol (HTTP), which allows for the retrieval of linked resources from across the web. An English scientist, Tim Berners-Lee in 1989, invented this technology.

The World Wide Web Consortium

The World Wide Web Consortium (W3C) was formed with the charter to define the standards for HTML, XHTML and, later, HTML5 and other web technologies. Members are responsible for drafting, circulating for review, and modifying the standard based on cross-Internet feedback to meet the needs of many using the Internet.

Beyond HTML, XHTML and HTML5, the W3C has the broader responsibility of standardizing any technology related to the Web; they manage the Hypertext Transfer Protocols (HTTP), Cascading Style Sheet (CSS), and Extensible Markup Language (XML) standards, as well as related standards for document addressing the Web. They also solicit draft standards for extensions to existing web technologies.

If you want to track HTML, XML, XHTML, HTML5, CSS, and other exciting web development and related technologies, visit W3C at <http://www.w3.org>.

Hypertext

You probably noticed the word “hypertext” in the spelled-out version of HTTP, Hypertext Transfer Protocol, originated by computing pioneer Theodore Nelson. Hypertext is a type of text that contains **hyperlinks** (or just links for short), which enable the reader to jump from one hypertext page to another. You may also hear the word **hypermedia**. A hypermedia system works just like hypertext, except that it includes graphics, sounds, videos, and animation as well as text. In contrast to ordinary text, hypertext gives readers the ability to choose their own path through the material that interests them. Computer-based hypertexts let readers jump around all they want. The computer part is important because it is hard to build a hypertext system out of physical media, such as index cards or pieces of paper.

The Web is a giant computer-based hypermedia system, and you have probably already done lots of jumping around from one page to another on the Web, this activity is call surfing or browsing. If one Web page does not seem all that interesting once you visit, you can click another link that seems more related to your needs (and so on).

The Web makes surfing so easy that you will need to give some thought to keeping people on your sites, that is, keeping them engaged and interested so they would not surf away.

ISP

Internet Service Provider (ISP) is a company that provides Internet access (connection) and related services for a monthly fee, ISP comes with a software package, username, and password and access phone number. With a modem, one can then log on to the Internet and browse the World Wide.

BROWSER

Websites look very different in reality to what you see when you visit it. Everything is in fact encoded. A browser is the piece of software that decodes everything so that what you see is an attractive page rather than a lot of coding. Most people use the Microsoft Internet Explorer/Internet Edge browser, which comes with all Windows software. Take a browser to be an interface between your computer and the Internet.

URL

Uniform Resource Locator (URL) allows all resources on the Internet to be located in a uniform (unique) manner; a website address, which has all the pertinent information for finding the exact location attached to it. The first part of the address indicates what protocol to use, and the second part specifies the IP address or the domain name where the resource is located. The following are examples of domain names: protocol://domainname.organizarion/full_path_of_the_file.
<http://www.pinfotechnologies.net/tutorial.html>. http is the protocol, www.pinfotechnologies.net is the domain name or Internet address of the machine, and the path to the file is www.pinfotechnologies.net/tutorial.html .

ADDRESS BAR

This is a white bar towards the top of the web browser installed on your computer browser. This is where you type in the address of a website that you want to visit, example:
<http://www.eresearchplace.com>

BANDWIDTH

Bandwidth is the amount of data transmitted in a fixed amount of time. When a website gets many visitors, it uses a lot of bandwidth.

CACHE

Every time you do anything on your computer, it stores this in memory so that the next time you try to do the same thing, it happens quicker than having to wait from scratch. The place where it stores all this is the "*cache*". Note that if your cache gets too full, it makes your computer work very slow. Regularly empty your cache to keep your computer working optimally.

DOMAIN

A domain is the name of website on the Internet that is identified by number assigned to its unique space.

DOMAIN NAME

The number assigned to its unique space identifies a domain. To make it easier to use however, the number is given the name of your choice and this name is assigned to the number. In this way, people do not need to remember the number (IP) in order to visit a website, but can use the easier-to-remember domain name. This websites domain name is www.pinfotechnologies.net

MIME

Multipurpose Internet Mail Extensions (**MIME**) is an Internet standard that extends the **format** of email to support that lets people use the protocol to exchange different kinds of data files on the Internet: audio, video, images, application programs, and other kinds, as well as the ASCII text handled in the original protocol, the Simple Mail Transport Protocol (SMTP)

1.2.1 How Web Works

The computers that make all these Web pages available are called Web servers. On any computer connected to the Internet, you can run an application called a Web browser.

Technically, a Web browser is a Web client; a program that is able to contact a Web server and request information. When the Web server receives the requested information, it looks for this information within its file system, and sends out the requested information via the Internet.

They all speak a common “language,” called Hypertext Transfer Protocol (HTTP). (*HTTP is not really a language like the one people speak; it is a set of rules or procedures, called protocols, which enables computers to exchange information over the Internet.*)

Most Web pages contain hyperlinks, which are specially formatted words, or phrases that enable you to access another webpage on the Web. Although the hyperlink usually does not make the address of this page visible, it contains all the information needed for your computer to request a Web page from another computer. When you click the hyperlink, your computer sends a message called an HTTP request. This message says, in effect, **“Please send me the Web page that I want.”** The Web server receives the request, and looks within its stored files for the Web page you requested. When it finds the Web page, it sends it to your computer, and your Web browser displays it. If the requested page is not found, you see an error message, which probably includes the HTTP code for this error: **404, “Not Found.”**

1.3 A short history of HTML and CSS

To date, HTML has gone through five major standards, including the latest HTML 5.0. In addition to the HTML standards, Cascading Style Sheets and XML have also provided valuable contributions to Web standards.

The following sections provide a brief overview of the various versions and technologies.

HTML 1.0

HTML 1.0 was never formally specified by the W3C because the W3C came pretty too late to specify standards. HTML 1.0 was the original specification Mosaic 1.0 used, and it supported few elements. What you could not do on a page is more interesting than what you could do. You could not set the background colour or background image of the page. There were no tables or frames. You could not dictate the font. All inline images had to be GIFs; JPEGs used for out-of-line images, and there were no forms. Every page looked pretty much the same: grey background and Times Roman font. Links indicated in blue until you have visited them, and then they changed to red. Because scanners and image-manipulation software were not available

then, the image limitation was not a huge problem. HTML 1.0 was implemented only in Mosaic and Lynx (a text-only browser that runs under UNIX).

HTML 2.0

Huge strides forward were recorded between HTML 1.0 and HTML 2.0. An HTML 1.1 actually did exist, created by Netscape to support what its first browser could do. Because only Netscape and Mosaic were available at the time (both written under the leadership of Marc Andreessen), browser makers were in the habit of adding their own new features and creating names for HTML elements to use those features. Between HTML 1.0 and HTML 2.0, the W3C also came into being, under the leadership of Tim Berners-Lee. HTML 2.0 was a huge improvement over HTML 1.0. Background colours and images could be set. Forms became available with a limited set of fields, but for the first time, visitors to a Web page could submit information. Tables also became possible.

HTML 3.2

HTML 3.2 was vastly richer than HTML 2.0. It included support for style sheets (CSS level 1). Even though CSS was supported in the 3.2 specification, the browser manufacturers did not support CSS well enough for a designer to make much use of it. HTML 3.2 expanded the number of attributes that enabled designers to customize the look of a page. HTML 3.2 did not include support for frames, but the browser makers implemented them anyway.

Note: A page with two frames is actually process like three separate pages within your browser. The outer page is the frameset. The frameset indicates to the browser, which pages go where in the browser window. Implementing frames can be tricky, but frames can also be an effective way to implement a Web site. A common use for frames is navigation in the left pane and content in the right. Detailed implementation of frames discuss later in this material.

HTML 4.0

HTML 4.0 clearly deprecates any use of HTML that relates to forcing a browser to format an element a certain way, and offered an improved accessibility for people with disabilities. All formatting moved into the style sheets from this version of HTML. This movement of presentation out of the document finally facilitates the continued rapid growth of the Web.

HTML 4.01

HTML 4.01 is a minor revision of the HTML 4.0 standard. In addition to fixing errors identified since the inception of 4.0, HTML 4.01 also provides the basis for meanings of XHTML elements and attributes, reducing the size of the XHTML 1.0 specification.

XHTML 1.0

Extensible HyperText Markup Language (XHTML) is the first specification for the HTML and XML cross-breed. XHTML was created to be the next generation of markup languages, infusing the standard of HTML with the extensibility of XML; designed to be used in XML-compliant environments, yet compatible with standard HTML 4.01 user agents.

XML 1.0

Extensible Markup Language (XML) originally designed to meet the needs of large-scale electronic publishing. As such, help to separate structure from presentation and provide enough power and flexibility to be applicable in a variety of publishing applications. In fact, many

modern word processing programs contain XML components or even export their documents in XML-compliant formats.

HTML 5

HTML5 is the fifth and current version of the HTML standard, published in October 2014 by the World Wide Web Consortium (W3C) to improve the language with support for the latest multimedia, while keeping it both easily readable by humans and consistently understood by computers and devices such as web browsers, parsers, mobile, etc.

HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalizes the mark-ups available for documents, and introduces mark-up and application programming interfaces (APIs) for complex web applications. HTML5 enables cross-platform mobile applications, because it includes features designed with low-powered devices in mind.

Many new syntactic features are included to natively include and handle multimedia and graphical content, the new `<video>`, `<audio>` and `<canvas>` elements were added, and support for scalable vector graphics (SVG) content and MathML for mathematical formulas. To enrich the semantic content of documents, new page structure elements such as `<main>`, `<section>`, `<article>`, `<header>`, `<footer>`, `<aside>`, `<nav>` and `<figure>`, are added. New attributes are introduced, some elements and attributes have been removed, and others such as `<a>`, `<cite>` and `<menu>` have been changed, redefined or standardized.

The APIs and Document Object Model (DOM) are now fundamental parts of the HTML5 specification-HTML5 also better defines the processing for any invalid documents.

CSS 1.0, 2.0 and 3.0

Cascading Style Sheets (CSS) helps move formatting out of the HTML specification. Much like styles in a word processing program, CSS provides a mechanism to easily specify and change formatting without changing the underlying code. The “cascade” in the name comes from the fact that specification allows for multiple style sheets to interact; allowing individual Web documents to be formatted slightly different from their kin (following department document guidelines, but still adhering to the company standards for example). The second version of CSS (2.0) builds on the capabilities of the first version, adding more attributes and properties for a Web designer to draw upon as well as the third version, which more features have been introduce such as animation and 2D and 3D design.

1.4 Tools for Web Designer

Before you embark on web design, there are necessary tools you need to have in place. The following are the essential tools;

1. **Text editor:** This is an environment you do your programming and save your file. Please always save your file with .html or .htm extension in order for the system to see it as a web page file. There are many types of text editor, but as a beginner, you are encouraged to use notepad text editor that came with your windows operating system, if you are using Linux operating system, you can use gedit from the terminal environment. This is beyond the scope of this manual. Other examples of text editors are notepad++, Edit plus, komodoedit, sublime, atom, visual studio, etc. As you advance in your programming skills, you will be introduced to Integrated Development Environment (IDE) such as Netbeans, Spring source, Eclipse, etc.

2. **Web browser:** Web Browser is the primary software used by designers and site visitors to access web pages. You need a browser to always test your work and modify it. Please note that when you save your file with .html or .htm, it automatically takes the icon of the browser that is set as a default browser in your system, by double clicking on the icon, it loads the saved file to the browser as a web page. You can use different types of web browsers such as Internet Explorer (this browser comes with windows operating system from windows 8 below), Microsoft edge(this comes with windows 10 operating system), Mozilla Firefox, Google Chrome, Opera, Safari, CometBird, etc. to run your designed web pages. It is important to test your web pages with many web browsers to see how they appear when visitors visit them.
3. **Internet Connectivity (optional):** This is for you to make references and researches in areas that you do not really understand. However, this is optional; but is necessary for the validation of your web pages according the W3C Standards.

Beyond the essentials are some specialized software tools for developing and preparing HTML documents and accessory multimedia files such as Dreamweaver.

1.5 The Overall Structure: HTML, Head, and Body

All HTML documents have three document-level tags in common. These tags are <html>, <head>, and <body> that delimit certain sections of the HTML document.

In HTML, there are two tags; the **open tag** and the **close tag**. Whenever a tag is opened, always make sure that it is closed with its close tag, as illustrated in all the examples in this text.

In this text, we will be using XHTML and HTML5 in our examples. In XHTML, all the elements are written in lower cases, this means small letters and the opening and closing tags rules are enforced, though HTML5 does not have such strict rule but it is a good practice to be consistent in your code by using lowercases (small letters).

The following are the basic HTML structure.

```
<html>
  <head>
    <title></title>
  </head>
  <body>
    </body>
</html>
```

The <html> tag

The <html> tag surrounds the entire HTML document. This tag tells the client browser where the document begins and ends.

```
<html>
  ...
  ... document contents ...
  ...
</html>
```

Additional language options were declared within the <html> tag in previous versions of HTML. However, those options (notably **lang** and **dir**) have been deprecated in HTML version 4.0, but the **lang** introduced in HTML 5 as it is shown below;

```
<!DOCTYPE HTML>
<html lang="en">
```

... document contents ...

</html>

The <head> tag

The <head> tag delimits the heading of the HTML document. The heading section of the document contains certain heading information for the document. The document's title, Meta information, and, in most cases, document scripts are all contained in the <head> section. A typical <head> section could resemble the following:

```
<html>
  <head>
    <title>Federal ICT Scheme, Jos</title> } Title of the page
    <link rel="stylesheet" type="text/css" href="/styles.css"> } CSS external file inclusion
    <meta name="description" content="Sample Page">
    <meta name="keywords" content="sample, heading, page"> } Page Meta
      Description
  </head>
  <body>
    This is a page to develop a site for the Federal Information System
  </body>
</html>
```

It is only the title information is displayed on the web page browser, other information is not displaying.



Fig. 1

The <head> section of your document can also include <meta> tags which are not rendered as visible text in the document—they are used to pass information and commands to the client browser.

As the name implies, the <meta> tag contains meta information for the document. Meta information is information about the document itself, instead of information about the document's contents. The Web server that delivers the document generates most of the

document's Meta information. However, by using `<meta>` tags, you can supply a different or additional information about the document. The amount of information you can specify with `<meta>` tags is quite extensive.

If you use the `HTTP-EQUIV` parameter in the `<meta>` tag, you can supply or replace HTTP header information. For example, the following `<meta>` tag defines the content type of the document as HTML with the Latin character set (ISO-8859-1):

```
<meta http-equiv="Content Type" content="text/html; charset=ISO-8859-1">
<meta http-equiv="pragma" content="no-cache">
<meta http-equiv="refresh"
content="3;URL=http://www.example.com/newpage.html">
```

1.5.1 Creating the Basic Structure

The basic structure for all HTML documents is the same, which include the following minimum elements and tags:

- ◆ `<DOCTYPE>`—the declared type of the document
- ◆ `<html>`—the main container for HTML pages
- ◆ `<head>`—the container for page header information
- ◆ `<title>`—the title of the page
- ◆ `<body>`—the main body of the page

These elements fit together in the following template format:

```
<!DOCTYPE html>
<html>
<head>
<meta ... meta tags go here ... >
<title>title of the page/document goes here</title>
<LINK rel="stylesheet" href="external style sheet name"
type="text/css">
<style>
... any document specific styles go here ...
</style>
<script>
... client-side scripts go here ...
</script>
</head>
<body>
... body of document goes here, paragraphs modified by
block elements, characters, words and sentences modified by
in line elements ...
</body>
</html>
```

1.5.2 HTML Elements Declaring the Document Type

The <DOCTYPE> declaration defines what format your page is in and what standard it follows, by specifying what DTD the document adheres to. For example, the following <DOCTYPE> definition specifies the strict XHTML DTD:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

There are three XHTML 1.0 doctypes available, plus one XHTML 1.1 doctype. The XHTML 1.0 doctypes are Strict, Transitional, and Frameset (to be used only when you are using frames to lay out your documents);

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

The XHTML 1.1 doctype comes in only one variant rather than Strict and Transitional versions:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"  
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

Note: In HTML5, the doctype is declared as follows;

```
<!doctype html >
```

and this is the current doctype use in web pages though older web pages still carry doctype for XHTML.

There is also a doctype for XHTML Basic , a stripped-down version of XHTML, an XML-based structured markup language used for pages designed for mobile devices such as mobile phones, PDAs, etc.:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"  
"http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
```

1.5.3 The Basic Html Tags/Attributes

The most important tags in HTML are tags that define headings, paragraphs and line breaks while attributes provide additional information about html tags on your document page.

<body> Tag:

This tag defines the body element of your HTML page. With an added bgcolor attribute, you can tell the browser that the background colour of your page should be red, like this: <body bgcolor="red">.

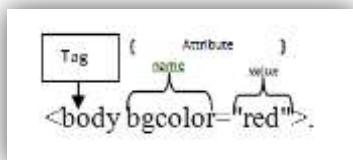


Fig.1.1

<table> Tag:

This tag defines an HTML table. With an added border attribute, you can tell the browser that the table should have no borders: `<table border="0">`. Attributes always come in name/value pairs like this: - `name="value"`.

Headings

Headings are defined with the `<h1>` to `<h6>` tags. `<h1>` defines the largest heading. `<h6>` defines the smallest heading as shown below;

```
<h1>This is a heading</h1>
<h2>This is a heading</h2>
<h3>This is a heading</h3>
<h4>This is a heading</h4>
<h5>This is a heading</h5>
<h6>This is a heading</h6>
```

This Code:

```
<html>
<head>
<title>Federal ICT Scheme,Jos</title>
</head>
<body>
<h1>Federal ICT Literacy Scheme,
Jos</h1><h2>Federal ICT Literacy Scheme,
Jos</h2><h3>Federal ICT Literacy Scheme,
Jos</h3><h4>Federal ICT Literacy Scheme,
Jos</h4><h5>Federal ICT Literacy Scheme,
Jos</h5><h6>Federal ICT Literacy Scheme,
Jos</h6></body>
</html>
```



Fig. 1.2

Paragraphs

Paragraphs are defined with the `<p>` tag, as it is shown below;

```
<body>
<h1>Federal Secretariat</h1>
<p>Tundu Wada</p>
<p>Jos North</p>
<p>Plateau State</p>
<p>informationcentre@fed.sec.ng</p>
<p>Bring ICT to your doorsteps.</p>
</body>
```



Fig. 1.3

Adding line break to our document

Most of the time, we might be in a situation we do not want to use `<p>` to break out of a particular line, that is, not to start a new paragraph but to start a new line. In this case, line break `
` tag is the best option.

Example;

```
<p>This <br /> is a para<br />graph with line  
breaks</p>
```

When added to the above code, it will produce the line break as below;



Fig. 1.4

Comments in HTML

The comment tag inserts into HTML source code explains your code, which can help you during source code editing later. The browser ignores any line of codes commented.

Example;

```
<!-- This is a comment -->
```

Note that you need an exclamation point after the opening bracket, but not before the closing bracket.

Hint: It is a good programming style to comment your codes for clarity during editing by another programmer or yourself. As you advance in your programming career, give attention to this.

1.6 HTML Text Formatting

Formatting simply mean changing the appearance of your document. Examples of tags we apply formatting attributes to are ***bold***, ***italic***, ***superscripted***, and ***subscripted text***.

1. **The `` Tag:** This tag boldfaces a character or segment of text enclosed between its opening tag (``) and its corresponding end tag (``).
2. **The `<i>` Tag:** This tag is like the ``, it tells the browser to render the enclosed text in italic font style which is accompanied with its corresponding end tag `</i>`.

3. **The <sub> Tag:** The text contain between the _{tag and its and its corresponding end tag} displays half a character's height lower, but in the same font and size as the current text flow. Both <sub> and its <sup> counterpart is useful for math equations, scientific notation, and chemical formulae.
4. **The ^{Tag:** This tag and its corresponding end tag} superscripts the enclosed text; it displays half a character's height higher, but in the same font and size as the current text flow. This tag is useful for adding footnotes to your documents and exponential values in equations. When you use it in combination with the <a> tag, you can create nice, hyperlinked footnotes:

Example;

```
<html>
<head>
<title>Federal ICT Scheme, Jos</title>
</head>
<body>
<i> This is an example of Italic text style</i><br />
<b> This is an example of bold face text style</b><br />
10<sub>3</sub> this is an example of subscript tag<br />
X<sup>3</sup> this is an example of superscript.
</body>
</html>
```

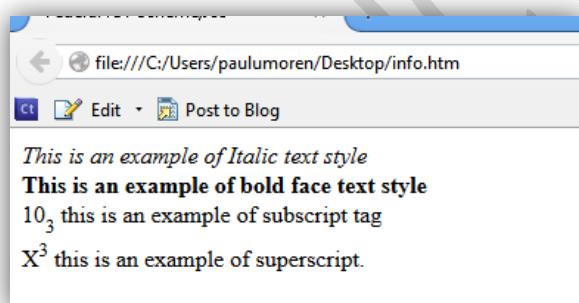


Fig. 1.5

The clear attribute(deprecated)

Normally, the
 tag tells the browser to stop the current flow of text immediately and resume at the left margin of the next line or against the right border of a left-justified inline graphic or table. Sometimes, you would rather want the current text flow resume below any tables or images currently blocking the left or right margin.

HTML 4 and XHTML provide that capability with the clear attribute for the
 tag. The clear attribute can have one of these values; left, right, or both- each related to one or both of the

margins. When the specified margin or margins are clear of images, the browser resumes the text flow

Example:

```
<html>
<head>
<title>Federal ICT Scheme, Jos</title>
</head>
<body>

This text should wrap around the image, flowing between the<br />
image and the right margin of the document.<br />
This text will flow as well, but will be below the image,<br />
extending across the full width of the page. There will be<br />
whitespace above this text and to the right of the image.<br />
this is kind of cool, but later we will look at how we can use <br />
CSS to achieve this effect anywhere.<br />
it is actually a good starting for beginning programmers<br />
Let see if this will flow out of the line. I think this is great and nice.
<br clear=left>
Normal text can continue from here anywhere let see what this will
produce on a browser. This seems great however.
</body>
</html>
```

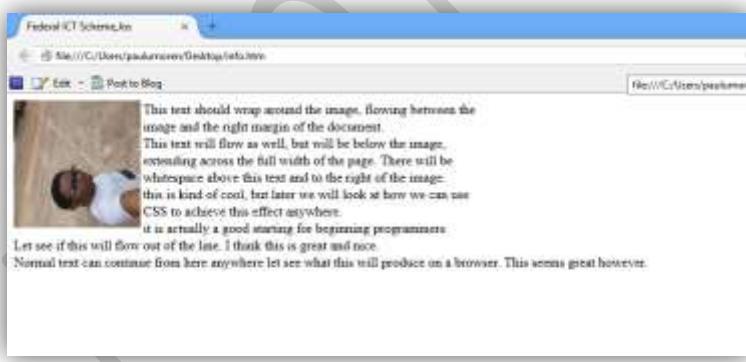


Fig.1.6

<br clear="left"/>: the old-fashioned HTML way. The new way of implementing this is using CSS as shown below;
<br style="clear:left;" /> : the CSS way of implementing clear attribute.

The <pre> Tag

The HTML standards' <pre> tag and its required end tag (</pre>) defines a segment inside which the browser renders text in exactly the character and line spacing written in the source document. Normal word wrapping and paragraph filling are disabled, and extraneous leading and trailing spaces are honoured. Browsers display all text between the <pre> and </pre> tags in a mono-spaced font.

Example;

```
<body>
The processing program is:
<pre>
main(int argc, char **argv)

{
FILE *f;
int i;

if(argc != 2)
    fprintf(stderr, "usage: %s <file>\n",
            argv[0]);
<a href="http:process.c">process</a>(argv[1]);
exit(0);
}
</pre>
</body>
```

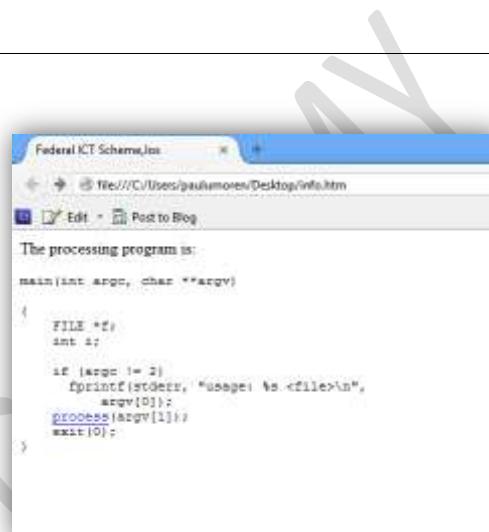


Fig. 1.7

The <center> Tag (deprecated)

This tag is used to centered web page contents such as text, graphics, tables, and so on inside the browser's window. The <center> alignment remains in effect until it is cancelled with its corresponding end tag </center>

Example:

Applying <center> tag in the above example, this is what it produces on the browser:



```
The processing program is:  
main(int argc, char **argv)  
{  
    FILE *f;  
    int i;  
  
    if (argc != 2)  
        fprintf(stderr, "usage: %s <file>\n",  
                argv[0]);  
    process(argv[1]);  
    exit(0);  
}
```

Fig. 1.8

Note: The center tag is deprecated in html4, xhtml and html5 ; it has been replaced by CSS attributes as shown below, so you should stop using it in your web pages.

<div style="text-align:center">This text is centered.
<p> this paragraph is centered.</p></div>

Implementing the
center tag using
CSS

The <address> Tag

This tag tells browser that the enclosed text is a contact address, typically for snail mail or email. The address may include other contact information too. The browser formats the text in a different manner from the rest of the document text or use the address in some special way.

Example:

```
<html>  
<head>  
<title>Federal ICT Scheme, Jos</title>  
</head>  
<body>  
<body>  
<address>Federal Secretariat<br />  
Tundu Wada<br />  
Jos North<br />  
Plateau State<br />  
informationcentre@fed.sec.ng<br />  
Bring ICT to your doorsteps.<br />  
</address>  
</body>  
</body>  
</html>
```

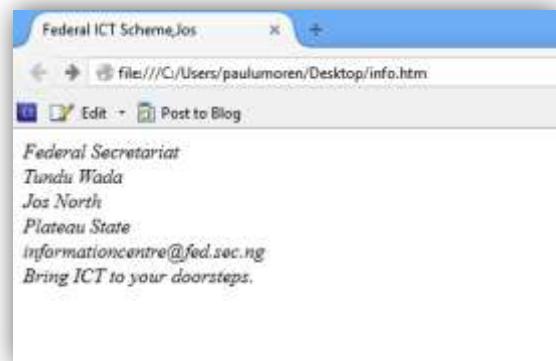


Fig. 1.9

Chapter Two Working with Lists

2.1 Listing

In the case of lists, HTML defines three different types of lists: ordered, commonly known as numbered lists, unordered, commonly known as bulleted lists, and definition lists for term and definition pairs.

2.1.1 Ordered Lists

Ordered lists use the ordered list tag (``) to define the entire list and the list item tag (``) to define each individual list item. Ordered list displays items using number format.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Example Ordered List</title>
</head>
<body>
<h1>Example of Ordered List</h1>
<ol>
<li>Microsoft Office Packages</li>
<li>Autocad Packages</li>
<li>Web Design/Development</li>
<li>Software Development</li>
<li>Java Standard Edition Programming</li>
<li>Mysql Database Management System Programming</li>
<li>Web Development Using PHP</li>
</ol>
</body>
</html>
```

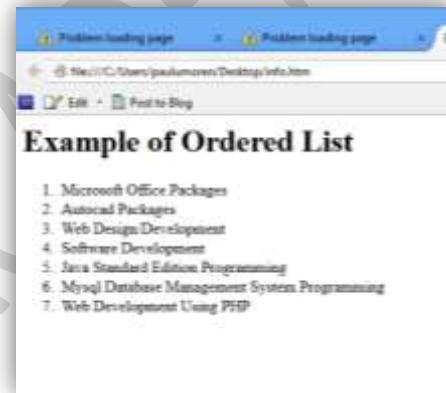


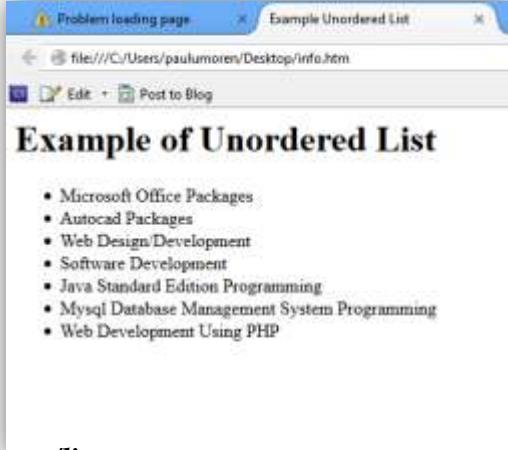
Fig. 2.1

2.1.2 Unordered List

`` Tag is used for unordered list. This tag signals to the browser that the following content, between the opening tag `` and the corresponding end tag `` is an unordered list of items. Inside, a leading `` tag identifies each item in the unordered list.

Unordered list displays items in bulleted format as shown in the example and output below;

```
<!DOCTYPE html>
<html>
<head>
<title>Example Unordered List</title>
</head>
<body>
<h1> Example of Unordered List</h1>
<ul>
<li>Microsoft Office Packages</li>
<li>Autocad Packages</li>
<li>Web Design/Development</li>
<li>Software Development</li>
<li>Java Standard Edition Programming</li>
<li>Mysql Database Management System Programming</li>
<li>Web Development Using PHP</li>
</ul>
</body>
</body>
</html>
```



The screenshot shows a web browser window with the title bar "Example Unordered List". The address bar displays "file:///C:/Users/paulumoren/Desktop/info.htm". Below the address bar are standard browser controls: back, forward, stop, refresh, and a "Post to Blog" button. The main content area contains the heading "Example of Unordered List" followed by an ordered list of items:

- Microsoft Office Packages
- Autocad Packages
- Web Design/Development
- Software Development
- Java Standard Edition Programming
- Mysql Database Management System Programming
- Web Development Using PHP

Fig. 2.2

2.2 Nesting Lists

When lists are placed inside another list, this process is called nesting lists.

2.2.1 Nested Unordered Lists

A different bullet character at the discretion of the browser may precede the items in each nested unordered list. For example, Internet Explorer/Microsoft Edge displays an alternating series of hollow, solid circular and square bullets for the various nests as shown below;

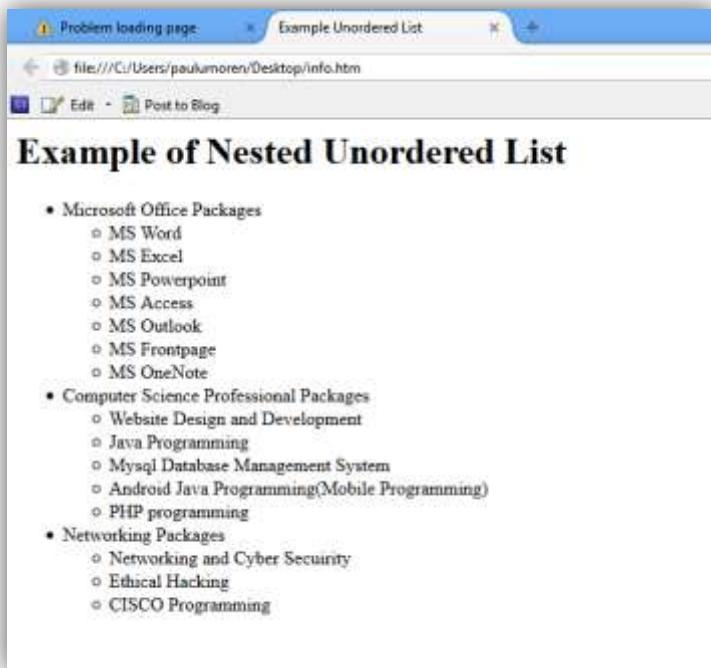


Fig. 2.3

```
<!DOCTYPE html>
<html>
<head>
<title>Example Unordered List</title>
</head>
<body>
<h1>Example of Nested Unordered List</h1>
<ul>
<li>Microsoft Office Packages</li>
<ul>
<li>MS Word</li>
<li>MS Excel</li>
<li>MS Powerpoint</li>
<li>MS Access</li>
<li>MS Outlook</li>
<li>MS Frontpage</li>
<li>MS OneNote</li>
</ul>
</li>
<li>Computer Science Professional Packages
<ul>
<li>Website Design and Development</li>
<li>Java Programming</li>
<li>Mysql Database Management System</li>
<li>Android Java Programming(Mobile
Programming)</li>
<li>PHP programming</li>
</ul>
</li>
</li>
<li>Networking Packages
<ul>
<li>Networking and Cyber Security</li>
<li>Ethical Hacking</li>
<li>CISCO Programming </li>
</ul>
</li>
</ul>
</body>
</body>
</html>
```

List items have many options of applying listing effect to them; this is achieved effectively by using CSS.

2.2.3 Definition Lists

Definition list is a list of terms and corresponding definitions typically formatted with the term on the left, with the definition following on the right or on the next line.

The `<dl>` tag is used in conjunction with `<dt>` (defines terms/names) and `<dd>` (describes each term/name).

Example:

The code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Defination List Example</title>
</head>
<body>
    <dl>
        <dt> Computer Programming</dt>
        <dd>Computer programming is the process of writing code to facilitate specific actions in a computer, application or software program, and instructs them on how to perform.
        </dd>
        <dt>Database</dt>
        <dd>A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS)
        </dd>
        <dt>Digital Marketing</dt>
        <dd>Digital marketing, also known as online marketing, refers to advertising delivered through digital channels to promote brands and connect potential customers using the internet and other forms of digital communication such as: Search engines. Websites
        </dd>
    </dl>

</body>
</html>
```

Output

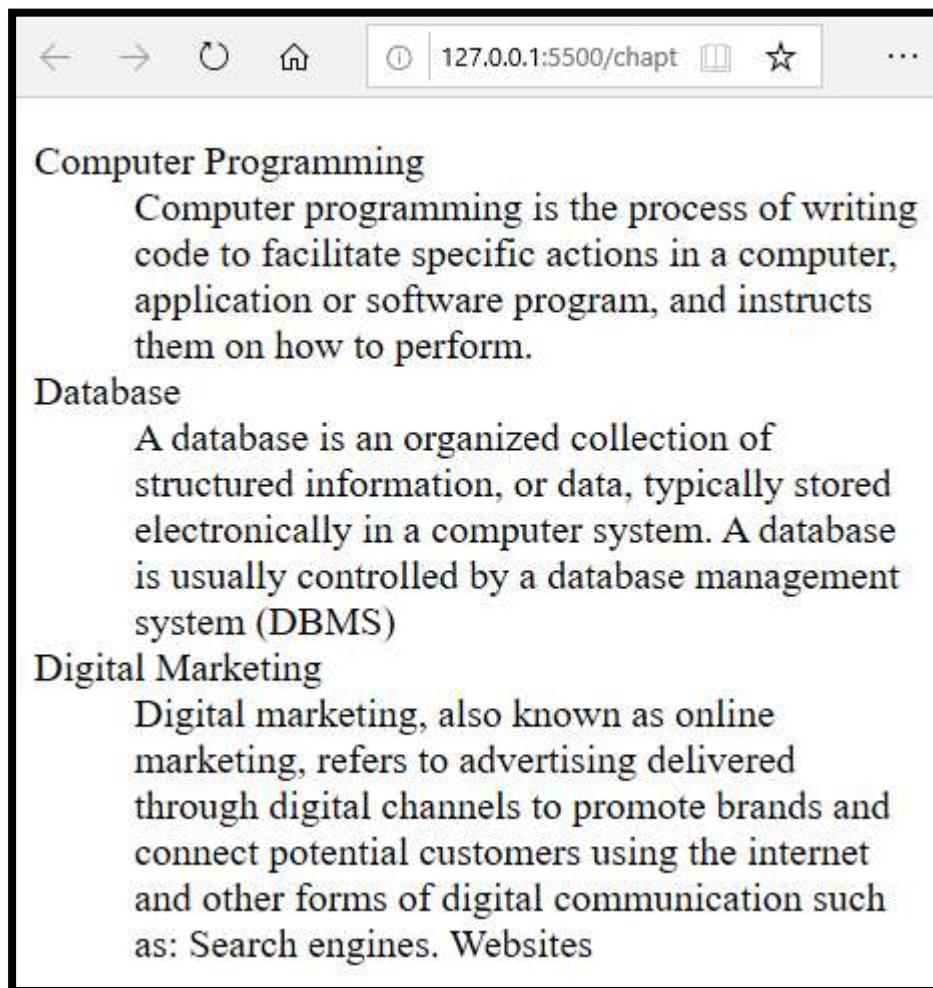


Fig. 2.4

2.3 Division (<div>) tag

The tag marks out a block of content, such as the main content block of your document, the header, or the footer. Division (<div>) can contain further elements, including more divs if required, but not apply within an inline element. For example, a simple website may have a header, a main column of content, a secondary column of content, and footer. The HTML for this could look like the following:

```
<div id="header">  
...  
</div>  
<div id="mainContent">  
...  
</div>  
<div id="secondaryContent">  
...  
</div>  
<div id="footer">
```

```
...  
</div>
```

The above content blocks can be positioned and displayed as required using CSS.

Span tag

This tag marks out sections within a block element and sometimes inside another inline element. It is an inline element, just the same as , , or <a>, except without any semantic meaning; it is simply a generic container. It can itself contain further inline elements, including more spans. For example, you wish to colour the first five words of a paragraph with a red colour using CSS class, keeping the rest of the paragraph black, can be used for this purpose as shown below;

```
<p><span class="leadingWords">  
This is for the first </span> five words of this paragraph can now be styled differently.  
</p>
```

A span cannot contain a block element—that is, you cannot place a <div> within a

Chapter Three

Working with Rules, Images and Multimedia

Though your document seems to be makeup of text, but an appropriate use of horizontal rules, images and multimedia make your document interesting and inviting. In this section, you are going to learn how to insert rules, images and multimedia in your document, as well as guiding principles on how not to overload your web pages with multimedia items and images.

3.1 Horizontal Rules

Horizontal rules separate sections of a document visually. This gives readers a clean consistent visual indication that a particular section has ended and another section has begun. Horizontal rules effectively set off small sections of text, delimit document headers and footers, and provide extra visual punch to headings within your document.

The `<hr>` tag tells the browser to insert a horizontal rule across the display window. With HTML, it has no end tag. For XHTML, include the end-tag slash (/) symbol as the last character in the tag itself after any attributes (`<hr .../>`), or include an end tag immediately following (`<hr></hr>`).

Example:

```
</head>
<body>
<h1> Example of Nested Unordered List</h1>
<hr></hr>
<ol>
<li>Microsoft Office Packages</li>
<ol list-style-type="lower-roman">
<li>MS Word</li>
<li>MS Excel</li>
<li>MS Powerpoint</li>
<li>MS Access</li>
<li>MS Outlook</li>
<li>MS Frontpage</li>
<li>MS OneNote</li>
</ol>
</li>
<hr></hr>
<li>Computer Science Professional Packages
<ol>
<li>Website Design and Development</li>
<li>Java Programming</li>
<li>MySQL Database Management System</li>
<li>Android Java Programming(Mobile Programming)</li>
<li> PHP programming</li>
</ol>
</li>
<hr></hr>
```

```
</li>
<li>Networking Packages
<ol>
<li>Networking and Cyber Security</li>
<li>Ethical Hacking</li>
<li>CISCO Programming </li>
</ol>
</li>
</ol>
<hr></hr>
</body>
</html>
```

The face bold tags are the horizontal rules tags applied to the page as shown in the diagram below.

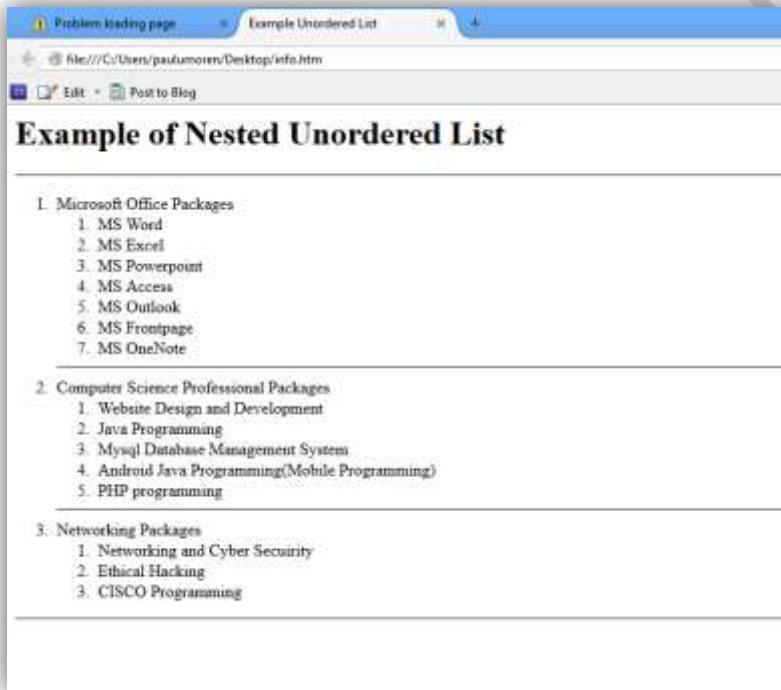


Fig. 3.1

3.1.1 The noshade attribute

This attribute produces 3D rule line effect.

By replacing `<hr size="10" noshade="noshade"></hr>` for `<hr></hr>`, in the above codes, it produces the diagram below:

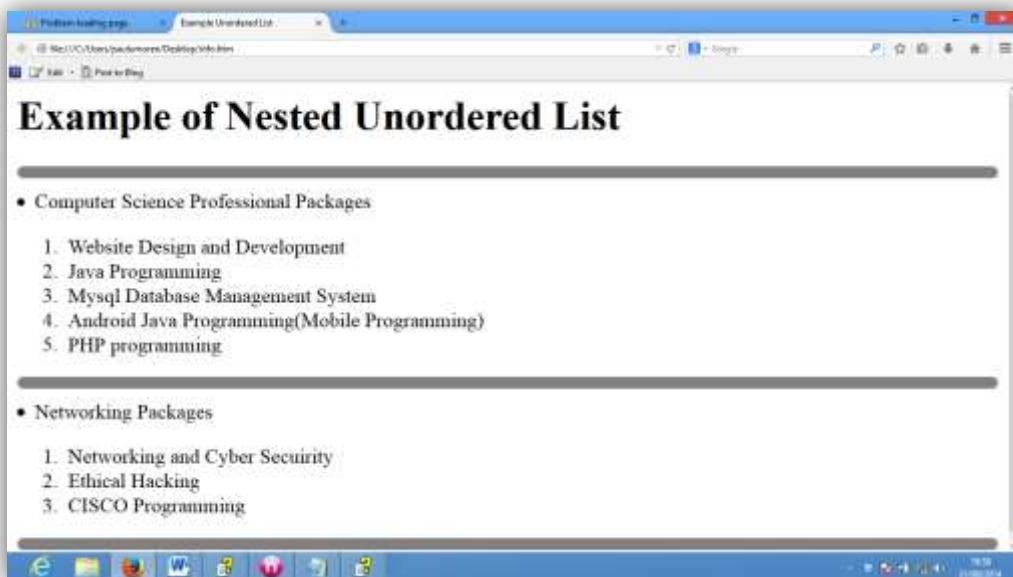


Fig. 3.2

3.2 Images

Images are very important in web designing; they give professional look to your websites. The three major formats of image commonly used are JPEG (Joint Photographers Experts Group), GIF(Graphical Interchange Format) and PNG(Portable Network Graphics). The majority of images on the Web employ GIF format, particularly buttons and banners, because of their smaller file size. The JPEG format is used to duplicate exact colours of an original image.

Inserting Images on a Page using `` tag and `src` attribute:

The `` tag lets you reference and insert an image into html document. There is no implied line or paragraph break before or after the `` tag, so images can be truly "in line" with text and other content.

To insert an image on a page:

1. Place the cursor where you want the image to appear.
2. Type `=>`

Offering Alternate Text

While images are great on a big screen with a fast connection, they can be less useful and downright problematic on handhelds, phones, slow connections, or for the blind. You can add descriptive text that will appear if the image, for whatever reason, does not display on the browser.

To offer alternate text when images do not appear:

1. Within the `img` tag, after the `src` attribute and value, type `alt="".`
2. Type the text that should appear if, for some reason, the image itself does not
3. Type `". =>`

If you would like tool tips on images in all browsers, use the title tag in addition to alt (which will still appear when the image does not). If you do not want tool tips at all, set title="".

Hints: A Tool tip set a description of an image when hovers a mouse tip on the image.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Image Example </title>
</head>
<body>
<h1>Example of Nested Unordered List</h1>

</body>
</html>
```



Fig. 3.3

Note: Sometimes, inserting pictures on your document could be a bit tricky, as you can see in this code extract; **img src='..../Pictures/honey.jpg'**, the.../ simply means that, I am jumping two steps backward before I get into the Pictures folder and load the picture onto the web server. This is because my file, which is info.htm saved to the Desktop and the picture to be loaded saved to the Pictures folder/directory in the computer.

Hints: For easy loading of images, always save the images to the location your .html files are saved.

3.2.1 How to scale an image

1. Type ``

As illustrates in the code extract below:

```

```

3.3 Background Colours and Background Images

The bgcolor attribute:

Setting the background colour is easy. To get a pure red background using RGB encoding, try:

```
<body bgcolor="#FF0000">
```

For a subtler background, try:

```
<body bgcolor="peach">
```

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Image Example </title>
</head>
<body bgcolor="gray">
<h1>Example of image <img> tag and background Colour <bgcolor> tag </h1>

</body>
</html>
```



Fig. 3.4

3.3.1 The background attribute

If a splash of colour is not enough, you may also place an image into the background of a document with the background attribute in the <body> tag.

The required value of the background attribute for an image is the URL of the image. When image is used as a background, the browser automatically repeats (tiles) the image both horizontally and vertically to fill the entire window.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Image Example </title>
</head>
<body background="../Pictures/who.png">
```

```
<h1> Example of Background Images Tag</h1>
<!---->
</body>
</html>
```



Fig. 3.5

3.3.2 Problems with background images

Here are some of the things that can go wrong with background images:

- The time to load the document increases by the amount of time needed to load the image. No further document rendering is possible while downloading background image.
- The background image takes up room in the browser's local cache, displacing other images that might actually contain useful information. This makes other documents, which might not even have backgrounds, take longer time to load.
- The colours in the image may not be available on the user's display, forcing the browser to dither the image. This replaces large areas of a single colour with repeating patterns of several other closer, but not-cleaner colours and can make the text more difficult to read.
- Because the browser must actually display an image in the background, as opposed to filling an area with a single colour, scrolling through the document can take much longer.
- Even if it is cleared onscreen, text printed on top of an image invariably is more difficult, if not impossible, to read.
- Fonts vary widely among machines; the ones you use with your browser that work fine with a background pattern often end up jagged and difficult to read on another machine.

3.4 Making Images Float (Alignment attributes)

You can use the align attribute (with the left and right values only) to make images float along one side of your page, with text and other elements wrapping around the other.

To make images float:

1. Type ``.
4. Create the elements that should flow next to the image.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Image Example </title>
</head>
<body bgcolor="gray">
<h1>Example of image <img> tag and background Colour <bgcolor> tag and alignment attributes</h1>

<h2>The Benefits of honey</h2>
<p>Honey stands as a natural sugar, as well as having medicinal values to cure some ailments</p>
</body>
</html>
```



Fig. 3.6

Exercise:

Try producing this with other alignment and observe the effect. You can try it with more than one image and more than one alignment attribute.

3.4.1 Adding Spaces around an Image

Look carefully at the above image, if you do not want your text butting right up to the image, you can use vspace and hspace attributes specifies the whitespace on top and bottom of an image. The vspace and hspace attributes are not supported in HTML5. Use CSS instead as shown below;

 for vspace and for hspace attributes.

To add space around an image:

1. Type .

By adding

`,`

it produces the diagram below;



Fig. 3.7

3.5 Working with colour

Applying colour to your web page using html tags are of two ways;

1. Apply simple colour with their names such as blue, black, white, green, grey, pink, yellow, etc., and using
2. The Hexadecimal values such as #00ff93, #fff000, etc.

There are millions of colours to be used, check Appendix 1 for hexadecimal values and names of some colours.

Hints: Colours are applied to the `<body>` text, `` tag, background `<bgcolor>` tag, and borders.

3.6 Background Audio

One other form of inline multimedia is generally available to web surfers is audio. Most browsers treat audio multimedia as separate documents, downloaded and displayed by special helper applications, applets, or plug-ins. Internet Explorer and Opera, on the other hand, contain built-in sound decoders and support a special tag (`<bgsound>`) that integrates web document with an audio file that plays in the background as a soundtrack when a web page is loaded.

Table 1.0

| Common multimedia formats and respective filename extensions | | | |
|--|----------|----------------------------|--------------------|
| Format | Type | Extension | Platform of origin |
| Graphics Interchange Format | Image | <i>Gif</i> | Any |
| Joint Photographic Experts Group | Image | <i>jpg, jpeg, jpe</i> | Any |
| X Bit Map | Image | <i>Xbm</i> | Unix |
| Tagged Image File Format | Image | <i>tif, tiff</i> | Any |
| PICT | Image | <i>pic, pict</i> | Apple |
| Rasterfile | Image | <i>Ras</i> | Sun |
| Portable Network Graphics | Image | <i>Png</i> | Any |
| Moving Pictures Expert Group | Movie | <i>mpg, mpeg</i> | Any |
| Audio Video Interleave | Movie | <i>Avi</i> | Microsoft |
| QuickTime | Movie | <i>qt, mov</i> | Apple |
| Windows Media Video | Movie | <i>Wmv</i> | Microsoft |
| Shockwave | Movie | <i>Dvr</i> | Macromedia |
| Real Video | Movie | <i>ra, rm, ram</i> | Real Networks |
| DivX | Movie | <i>div, divx, tix, mp4</i> | DivX |
| AU | Audio | <i>au, snd</i> | Sun |
| Waveform Audio | Audio | <i>Wav</i> | Microsoft |
| Audio Interchange File Format | Audio | <i>aif, aiff</i> | Apple |
| Musical Instrument Digital Interface | Audio | <i>midi, mid</i> | Any |
| PostScript | Document | <i>ps, eps, ai</i> | Any |
| Acrobat | Document | <i>Pdf</i> | Any |

Example:

<bgsound src="audio/tadum.wav" loop=10>

repeats the ta-dum soundtrack 10 times, whereas:

<bgsound src="audio/noise.wav" loop=infinite>; continuously play the noise soundtrack.

3.7 Animated Text in html

The animation is simple text scrolling horizontally across the browser. Like the <blink> tag, animated text can easily become intrusive and abusive for the reader. The <marquee> tag produces animated text effect in html. You can make the text/image scroll horizontally across or vertically down the web page.

However, <marquee> tag is obsolete but some browsers still support it.

Table 2

| <marquee> | |
|------------|---|
| Function | Creates a scrolling text marquee |
| Attributes | align, behavior, bgcolor, class, controls, direction, height, hspace, loop, scrollamount, scrolldelay, style, vspace, width |
| End tag | </marquee>; never omitted |
| Contains | plain_text |
| Used in | body_content |

The text between the <marquee>tag and its required </marquee> end tag scrolls horizontally across the display. The various tag attributes control the size of the display area, its appearance, its alignment with the surrounding text, and the scrolling speed.

Some browsers ignore the <marquee> tag and attributes, but its contents are not. They are displayed as static text.

3.7.1: The <marquee> align attribute

The popular browsers place <marquee> text into the surrounding body content just as if it were an embedded image. As a result, you can align the marquee within the surrounding text.

The align attribute accepts a value of top, middle, or bottom, meaning that the specified point of the marquee will be aligned with the corresponding point in the surrounding text.

Hints: *Thus :< marquee align=top>aligns the top of the marquee area with the top of the surrounding text. Also see the height, width, hspace, and vspace attributes (later in this chapter), which control the dimensions of the marquee.*

3.8 The behaviour, direction, and loop attributes of <marquee> attribute

These three attributes control the style, direction, and duration of the scrolling in marquee.

The behaviour attribute accepts three values as follows:

- **Scroll (default):** this value causes the marquee to act like the grand marquee in Times Square: the marquee area is initially empty; the text then scrolls in from one side

(controlled by the direction attribute), continues across until it reaches the other side of the marquee, and then scrolls off until the marquee is once again empty.

- **Slide:** this value causes the marquee to start empty text, then scrolls in from one side (controlled by the direction attribute), stops when it reaches the other side, and remains onscreen.
- **Alternate:** This value causes the marquee to start with the text fully visible at one end of the marquee area. The text then scrolls until it reaches the other end, whereupon it reverses direction and scrolls back to its starting point.

Note: *If you do not specify marquee behaviour, the default behaviour is scroll.*

The direction attribute sets the direction for marquee text scrolling. Acceptable values are either left (the default) or right. Note that the starting end for the scrolling is opposite to the direction: left means that the text starts at the right of the marquee and scrolls to the left. Remember also that rightward-scrolling text is counter-intuitive to anyone who reads left to right.

The loop attribute determines how many times the marquee text scrolls. The scrolling action repeats based on the integer value provided. If the value is infinite, the scrolling repeats until the user moves on to another document/page within the browser.

Example:

```
<!DOCTYPE html >
<html>
<head><title></title>
</head>
<body>
<bgsound src="Acapela.mp3" loop=infinite>
<marquee align=center loop=infinite>
This is an example of marquee test which could be used
as an advert
..... This looks good, Yes!
</marquee>
</body>
</html>
```

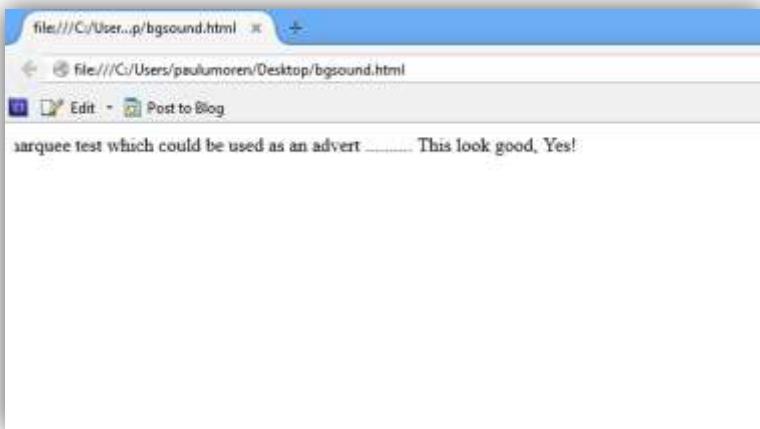


Fig. 3.8

The example message starts at the right side of the display window (default), scrolls leftward all the way across and off the display, and then starts over again until the user moves on to another page. Notice the intervening periods and spaces for the "trailer"; you cannot append one marquee to another.

3.8.1 Using bgcolor attribute in marquee

The bgcolor attribute lets you change the background colour of the marquee area. It accepts either an RGB colour value or one of the standard colour names.

To create a marquee area whose colour is yellow use:

```
<marquee bgcolor=yellow>
```

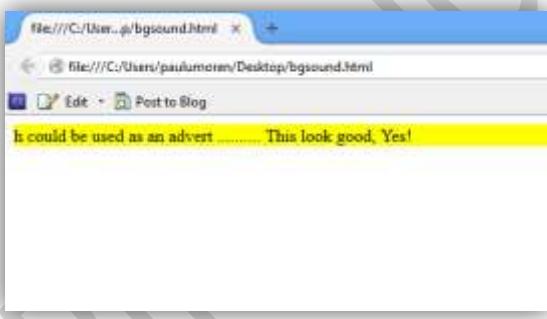


Fig. 3.9

3.8.2 The height and width attributes of <marquee>

The height and width attributes determine the size of the marquee area, if not defined, the marquee area extends all the way across the display area, which is high enough to enclose the marquee text.

Both attributes accept either a numeric value, indicating an absolute size in pixels, or a percentage, indicating the size as a percentage of the browser window height and width. For example, to create a marquee that is 50 pixels tall and occupies one-third of the display window width, use:

```
<marquee height=50 width="33%">
```

3.8.3 The hspace and vspace attributes

The hspace and vspace attributes let you create some space between the marquee and the surrounding text. This usually makes the marquee stand out from the text around it.

Both attributes require an integer value specifying the space needed in pixels. The hspace attribute creates space to the left and right of the marquee; the vspace attribute creates space above and below the marquee. To create 10 pixels of space all the way around your marquee, for example, use:

```
<marquee vspace=10 hspace=10>
```

3.8.4 The scrollamount and scrolldelay attributes

These attributes control the speed and smoothness of the scrolling marquee. The scrollamount attribute value is the number of pixels needed to move text each successive movement during the scrolling process. Lower values mean smoother but slower scrolling; higher numbers create faster, jerkier text motion.

The scrolldelay attribute lets you set the number of milliseconds to wait between successive movements during the scrolling process; the smaller this value, the faster the scrolling effect of marquee text.

You can use a low scrolldelay attribute to mitigate the slowness of a small, smooth scrollamount.

Example:

```
<marquee scrollamount=1 scrolldelay=1>
```

scrolls the text one pixel for each movement but does so as fast as possible. In this case, the capability of the user's computer limits the scrolling speed.

Try producing these attributes as an exercise.

Example:

```
<marquee align=center loop=infinite bgcolor=yellow height=50 width="33%">  
vspace=10 hspace=10 scrollamount=1 scrolldelay=1>
```

3.9 Other Multimedia Content

The Web is completely open-minded about the types of content that can be exchanged by servers and browsers. In this section, we look at a different way to reference images, along with audio, video, and other document formats.

3.9.1 Embedded Versus Referenced Content

Images currently enjoy a special status among the various media that can be included within an HTML document and displayed in-line with other content. Sometimes, however, as we discussed earlier, you may also reference images externally particularly large ones in which details are important but not immediately necessary to the document content. Other multimedia elements, including digital audio and video can be referenced as separate documents external to the current one.

You normally use the anchor tag () to link external multimedia elements to the current document. Just like other link elements selected by the user, the browser downloads the multimedia object and presents it to the user, possibly with the assistance of an external application or plug-in. Referenced content is always a two-step process: present the document that links to the desired multimedia object, then present the object if the user selects the link. In the case of images, you can choose how to present images to the user: inline and immediately available via the tag, or reference and subsequently available via the tag. If your images are small and critical to the current document, you should provide them inline. If they are large or are only a secondary element of the current document, make them available as referenced content via the tag. If you choose to provide images via the tag, it is sometimes a courtesy to your readers to indicate the size of the referenced image in the referencing document and perhaps provide a thumbnail sketch. Users can then determine whether it worth their time and expense to retrieve it.

3.9.2 Referencing Audio, Video, and Images

You reference any external document, regardless of type or format, via a conventional anchor () link:

The [National Anthem](sounds/anthem.mp3), this is National Anthem for all Nigerians all over the world, just clicking on the link loads and plays the anthem.

Just like any referenced document, the server delivers the desired multimedia object to the browser when the user clicks the link. If the browser finds that the document is not HTML but rather some other format, it automatically invokes an appropriate rendering tool (application) to display or otherwise convey the contents of the object to the user.

You can configure your browser with special helper applications that handle different document formats in different ways. Audio files, for example, might be passed to an audio-processing tool, and video files are given to a video-playing tool. If a browser has not been configured to handle a particular document format, the browser will inform you and offer to simply save the document to disk. You can later use an appropriate viewing tool to examine and use the document.

3.9.3 Embedding Other Document Types

The Web can deliver nearly any type of electronic document, not just graphics, sound, and video files. To display them, however, the client browser needs a helper application installed and referenced. Recent browsers also support plug-in accessory software.

For example, consider a company whose extensive product documentation was prepared and stored in some popular layout application such as Adobe Acrobat, FrameMaker, QuarkXPress, Microsoft Excel or CorelDraw. The Web offers an excellent way for distributing that documentation over a worldwide network, but converting to HTML would be too costly at this time.

The solution is to prepare a few HTML documents that catalogue and link the alternative files and invoke the appropriate display application; or make sure that the users' browsers have the plug-in software configured to invoke the appropriate helper application installed in the user's system. Adobe Acrobat Reader is a very popular plug-in, for example, if the document is in Acrobat (.pdf) format and a link to an Acrobat document is chosen, the tool is started and accordingly displays the document, often right in the browser's window.

Chapter Four

Working Tables

The `<table>` tag defines tables. A table is divided into rows (with the `<tr>` tag), and each row is divided into data cells (with the `<td>` tag). The letters `td` stands for "table data," which is the content of a data cell. A data cell can contain text, images, lists, paragraphs, forms, horizontal rules, tables, etc. Tables are useful for the general display of tabular data; they also serve an important role in managing document layout.

Before creating tables on your document, make sure you understand the layout of your table. You can properly sketch out the draft of your table on a paper to guide you.

4.1 Basic Table Tags

You can create a wide variety of tables with only five tags. The `<table>` tag, which encapsulates a table and its elements in the document's body content. The `<tr>` tag, which defines a table row. The `<th>` and `<td>` tags, which define the table's headers and data cells; and the `<caption>` tag, which defines a title or caption for the table. Beyond these core tags, you may also define and control whole sections of tables, including adding running headers and footers, with the `<colgroup>`, `<col>`, `<tbody>`, `<thead>`, and `<tfoot>` tags. Each tag has one or more required and optional attributes, some of which affect not only the tag itself but also related tags.

4.1.1 The `<table>` Tag

The `<table>` tag and its corresponding `</table>` end tag define and encapsulate a table within the body of your document. Unless you place them within the browser window by stylesheet, paragraph, division-level, or other alignment options, the browser stops the current text flow, breaks the line, inserts the table beginning on a new line, and then restarts the text flow on a new line below the table.

4.1.2 The `bgcolor` and `background` image attributes

You can make the background of a table with a different colour than the document's background using `bgcolor` attribute for the `<table>` tag, as well as the background image.

These mark-ups control the display of background colour and image of a table:

```
<table background="../images/logo.jpg"> for background image  
<table bgcolor="gray"> for background colour
```

4.1.3 The `border` attribute

The optional `border` attribute for the `<table>` tag-`<table border="10">` tells the browser to draw lines around the table, the rows and cells within it. The default is no borders at all. You may specify a value for border, but you do not have to with HTML. The `border` attribute carries integer value equals to 3D chiselled-edge lines that surround the outside of the table and make it appear as embossed onto the page.

4.1.4 The `cellspacing` attribute

The `cellspacing` attribute controls the amount of space placed between adjacent cells in a table and along the outer edges of cells through the edges of a table.

By including the cellspacing attribute, you can widen or reduce the interior cell borders. For instance, to make the thinnest possible interior cell borders, include the border and cellspacing=0 attributes in the table's tag.

4.1.5 The cellpadding attribute of <table> tag

The cellpadding attribute controls the amount of space between the edge of a cell and its contents, which by default is 1 pixel. You may make all the cell contents in a table touch their respective cell borders by including cellpadding=0 in the table tag. You may also increase the cellpadding space by making its value greater than 1.

4.1.6 The width and height attributes of <table> tag

Browsers automatically make a table only as wide as needed to display all of the cell contents. If necessary, you can make a table wider with the width attribute.

The value of the width attribute is either an integer number of pixels or a relative percentage of the screen width, including values greater than 100 %. For example:

```
<table width=400> or <table width="50 %">.
```

4.2 The <caption> Tag

A table commonly needs a caption to explain its contents, so the popular browsers provide a table-caption tag. The <caption> tag and its contents are typically immediately after the <table> tag. Example

```
<table>
<caption>My Caption</caption>
<tr>
<i>Creating a simple table without border</i>
<html>
<body>
<h4>This table has no borders:</h4>
<table>
<tr>
<td>100</td>
<td>200</td>
<td>300</td>
</tr>
<tr>
<td>400</td>
<td>500</td>
<td>600</td>
</tr>
</table>
<h4>And this table has no borders:</h4>
<table border='0'>
<tr>
<td>100</td>
<td>200</td>
```

```
<td>300</td>
</tr>
<tr>
<td>400</td>
<td>500</td>
<td>600</td>
</tr>
</table>
</body>
</html>
```



Fig. 4.1

The above diagram is example of 2-row and 3-column table without a border.

4.3 Creating table with border

Example:

```
<html>
<body>
<table border="10">
<tr>
<td>Name</td>
<td>Place of residence</td>
</tr>
<tr>
<td>Paul Umoren</td>
<td>Jos, Plateau</td>
</tr>
<tr>
<td>Florence Okon</td>
<td>Calabar, Cross River</td>
</tr>
<tr>
<td>Justice Okpor</td>
```

```
<td>London</td>
</tr>
<tr>
<td>Beatrice Udu</td>
<td>Ogoja, Cross River</td>
</tr>
</table>
</body>
</html>
```



Fig. 4.2

Example of table with heading `<th>` and `<caption>`tags.

```
<html>
<body>
<table border='10'>
<caption> Residence of some friends and Character Traits</caption>
<th>Name</th>
<th>Place of residence</th>
<th>Behavioural Status</th>
<tr>
<td>Paul Umoren</td>
<td>Jos, Plateau</td>
<td>Complex, Unpredictable and Carefree</td>
</tr>
<tr>
<td>Florence Okon</td>
<td>Calabar, Cross River</td>
<td>Careful and Focusing</td>
</tr>
<tr>
<td>Justice Okpor</td>
<td>London</td>
<td>Funny and Occupying</td>
</tr>
```

```
<tr>
<td>Beatrice Udu</td>
<td>Ogoja, Cross River</td>
<td> Passionate and Selective</td>
</tr>
<tr>
<td>Ugochi Ahunanya</td>
<td>Victoria Island, Lagos</td>
<td> Just good</td>
</tr>
<tr>
<td>Farida Adamu</td>

<td>Rayfield, Plateau</td>
<td> Creative and carefree</td>
</tr>
<tr>
<td>Joseph Agbor</td>
<td>Calabar, Cross River</td>
<td> Political and conscious</td>
</tr>
<tr>
<td>Cynthia Effa</td>
<td>Parliamentary, Cross River</td>
<td> Calm and Stress-free</td>
</tr>
<tr>
<td>Mary Kemi</td>
<td>Tudu Wada, Plateau</td>
<td> My world and my friends</td>
</tr>
<tr>
<td> Andrew Naughty Friend</td>
<td>Tudu Wada, Plateau</td>
<td> Discipline and assistive</td>
</tr>
</table>
</body>
</html>
```

The screenshot shows a web browser window with the title bar "HTML Tab Tryit Editor v1.4 Tryit Editor v1.4 file/_...html". The main content area displays a table with the following data:

| Name | Place of residence | Behavioural Status |
|-----------------------|----------------------------|-------------------------------------|
| Paul Umoren | Jos, Plateau | Complex, Unpredictable and Carefree |
| Florence Okon | Calabar, Cross River | Careful and Focusing |
| Justice Okpor | London | Funny and Occupying |
| Beatrice Udu | Ogoja, Cross River | Passionate and Selective |
| Ugochi Ahunanya | Victoria Island, Lagos | Just good |
| Farida Adamu | Rayfield, Plateau | Creative and carefree |
| Joseph Agbor | Calabar, Cross River | Political and conscious |
| Cynthia Effia | Parliamentary, Cross River | Calm and Stressfree |
| Mary Kemi | Tudu Wada, Plateau | My world and my friends |
| Andrew Naughty Friend | Tudu Wada, Plateau | Discipline and assistive |

Fig. 4.3

Note: The CSS part of this book handles Table styling and others. The Intention of not treating them together is for you to master ***Markup*** language before venturing into CSS.

Chapter Five Form Fundamentals

Forms enable you to build Web pages that let users actually enter information and send it back to the server. The forms can range from a single text box for entering a search string—common to all the search engines on the Web—to a complex multipart worksheet, that offers powerful submission capabilities. A form has three important parts: the form tag, which includes the URL of the script that will process the form; the form elements, like fields and menus; and the submit button which sends the data to the script on the server through the communication path between the browser and server called the *common gateway interface* (CGI).

Forms also composed of one or more text-input boxes, clickable buttons, multiple-choice checkboxes, and even pull-down menus and image maps, all placed inside the `<form></form>`. The text in the form are useful for providing form element labels, prompts, and instructions to the users on how to fill out data in the form . Within the various form elements, you can use JavaScript event handlers for a variety of effects, such as testing and verifying form contents and calculating a running sum.

When a button on the form is click to submit the form, the browser packages up the user-supplied data and sends them to a server or to an email address. The server passes the information along to a supporting program or application that processes the information and creates a reply, usually in HTML. The reply simply may be a “*Thank you*”, or it might prompt the user on how to fill out the form correctly or to supply missing fields. The server sends the reply to the browser client, which then presents it to the user.

5.1 The `<form>` Tag

The basic form tag is `<form></form>`. But there are many form attributes that can be used with the `<form>` tag such as action, method, enctype, accept, accept-charset, and name. These attributes are very important when creating forms to interact with the server side that is the database. Even though `<form>` is a block-level element, bear in mind that if you are writing XHTML, individual form controls must be contained within a further block level element—generally a `<div>` or a `<p>`-in order to validate. Form controls are all inline-level elements, so you can place them adjacent to each other within a single block container, if you so desire.

5.2 Form Attributes

The following are form attributes:-

- ⇒ **The `action` attribute:** action attribute is required for the `<form>` tag to gives the URL of the application that is to receive and process the form's data. This URL is usually the server side program that processes the form contents. Example could be `form_process.php`, `form_process.jsp`, `form_process.aspx`, etc.
- ⇒ **The `method` attribute:** The method attribute informs the user agent how the form data should be passed to the form handler. The two possible values are **GET** and **POST**. When the **GET** value is used, form sends data to the form handler in the form of a query string. For instance, the browser appends the data to the form's action URL, separated by the question mark character. When submitted, the form URL in the browser's address bar looks something like this:
`formhandler.php?name=paul+age=27+gorgeous=yes+single=yes`. This represent data collected from the input fields; and it is not advisable to use GET method to submit data with security contents such as password or security codes.

5.2.1 When and how to use POST or GET Method

Which one should you use if your forms-processing server supports both the POST and GET methods? Here are some hints:

- For best form-transmission performance, send small forms with a few short fields using the GET method without sensitive data.
 - Because some server operating systems limit the number and length of command-line arguments passed to an application at once, use the POST method to send forms that have many fields or that have long text fields.
 - If you are inexperienced in writing server-side forms-processing applications, choose GET. The extra steps involved in reading and decoding POST-style transmitted parameters, while not too difficult, may be more than you are willing to tackle.
 - If security is an issue, choose POST. GET places the form parameters directly in the application URL, where network sniffers can capture or extract from a server log file. If the parameters contain sensitive information like credit card numbers, you may be compromising your users without their knowledge. While POST applications are not without their security holes, but they can at least take advantage of encryption when transmitting the parameters as a separate transaction with the server that is using the hash or salt encrypted method.
 - If you want to invoke the server-side application outside the realm of a form, including passing it parameters, use GET, because it lets you include form-like parameters as part of a URL. POST-style applications, on the other hand, expect an extra transmission from the browser after the URL, something you cannot do as part of a conventional <a> tag.
 - If the form submission is passive, such as a search engine query, then use **GET** to submit the form. An advantage of having the queries visible in the URL is that you can bookmark a specific search query for repeated use.
- ⇒ **The enctype Attribute:** to be able to upload files such as images to the server, we use file selection control. Enctype attribute controls file selection depending on the type of file uploads to the server. The enctype attribute is used to specify which MIME type should be used to encode the form data. If this attribute is left out of the markup, then the form will default to a MIME type of application/x-www-form-urlencoded, which should be suitable for most forms, unless the form is using a file input element (described further shortly), in which case this attribute should contain a value of multipart/form-data, allowing it to cope with processing binary code. Example;

```
<form enctype="multipart/form-data" method="post" action="cgi bin/save_file">
```

The accept attribute: This attribute accepts a comma-separated list of MIME types pertaining to acceptable file types to upload to the server. For instance, if you include within your form the ability to upload JPG and GIF images to the server upon form submission, but you want to prevent users from uploading any other form of file, you would use an **accept** attribute like this:<form accept="image/gif,image/jpeg,image/jpg">

- ⇒ **The *name* Attribute:** This attribute assigns send data collected from the form to the server-side. When the form is submitted, the data/information entered in the forms are stored in the name attribute and posted to the server programming script. Example;

```
<input name='fullname' type='text' value=''/>.
```

The full name that a user entered in the form is store in the name attribute “*fullname*” as it is shown in the example above and posted to the scripting file that connect the form to the database.

5.2.2 Form Input Element

The `<input>` element is a self-closing inline element, like an image or a line break, so if you are writing XHTML, remember to include the closing forward-slash, and remember to enclose the element within a block element such as a paragraph:

```
<p><input /></p>
```

How the `<input>` element behaves and displays is dictated by the type attribute, which can take values of text, password, file, checkbox, radio, hidden, reset, submit, hidden and button. If no type is specified, current web browsers assume the default type as text input. Example of input element with type attribute ;

```
<input type='text' />
```

5.2.3 Text Input Element

A text input is use for typing text into the field. This is a single-line control and normally appears in the form of a rectangular box with an inset border. The allowed length of the input string is specified with the addition of a `maxlength` attribute, which takes a numerical value equating to the maximum number of allowed characters. There is no feedback mechanism provided in the event that the user tries to insert more than the allowed number of characters—the form control just ceases to accept extra characters and will truncate an overlong string if such a string is pasted into the control. If you wish to alert users that they have run out of room, you will need to use JavaScript.

You can also include a `value` attribute that presents instructional content of the text control to the users by using `value` or `placeholder` attributes as shown below:

```
<input type='text' value='Insert text here' /> for value  
<input type='text' placeholder='Insert text here' /> for placeholder
```

When you use `value` attribute, it displays the content for the user to erase before entering a new content, but when `placeholder` is used, as soon as the user click the text box to start typing, the displayed instructional contents disappear.

To create a text box:

1. If desired, type the label that will identify the text box to your visitor (for example, Name:).
2. Type `<input type='text'>`.
3. Type `name='name'`, where name is the text that will identify the input data to the server (your

script).

4. If desired, type value="default", where default is the data that will initially be shown in the field and that will be sent to the server if the visitor does not type something else.
5. If desired, define the size of the box on your form by typing size="n", replacing n with the desired width of the box, measured in characters.
6. If desired, type maxlength="n", where n is the maximum number of characters that can be entered in the box.
7. Finish the text box by typing a final />.

Example:

```
<!DOCTYPE html>
<html>
<head><title> Text Input(Textbox) Example</title>
</head>
<body>
<form action="registration.php" method="post" >
<b>First Name: </b><input type="text" name="firstname" size="30" maxlength="15" value="Enter first name here" /><br /><br />
<b>Surname Name: </b><input type="text" name="surname" size="30" maxlength="15" value="Enter Surname here" /><br /><br />
<b>Address: </b><input type="text" name="address" size="30" maxlength="15" value="Enter Address here" />
</body>
</html>
```

Output

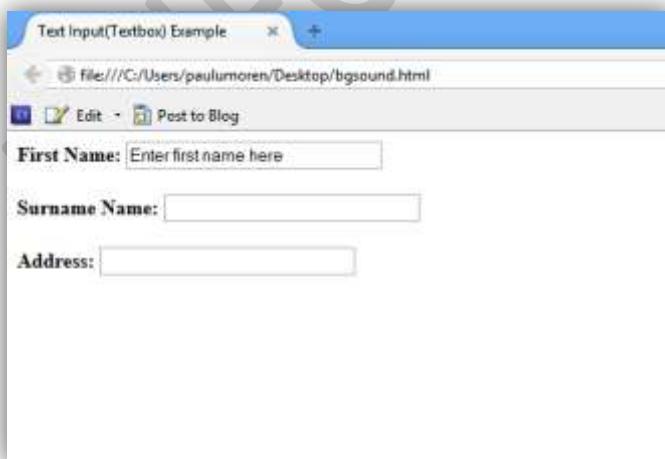


Fig. 5.1

5.2.4 Creating Password Boxes in a form

The only difference between a password input box and a text input box is that whenever password is entered in the field, it is hidden by bullets or asterisks, but the information is not encrypted when send to the server. <input type="password" />



Example

```
<!DOCTYPE html>
<html>
<head><title> Creating Password Field</title>
</head>
<body>
<center>
<form action="registration.php" method="post" >
<b>User Name:<br /></b><input type="text" name="firstname" value="Enter username here" /><br />
<b>Password:</b><br />
<input type="password" name="pw" size="30" maxlength="10" value="enter password here"/>
</center>
</body>
</html>
```

Output

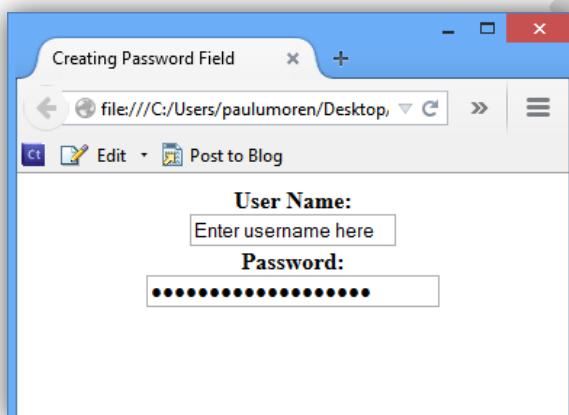


Fig. 5.2

5.2.5 File Input Element

A file input usually takes the form of a text input box followed by a Browse button. The file input control allows you to browse for a file on your local network. Once you have selected the file, the file path is then inserted automatically into the contents of the text input field.

To allow visitors to upload files:

1. Type `<form method="post" enctype="multipart/form-data">`. The `enctype` attribute ensures that the file is uploaded in the proper format.
2. Next, type `<input type="file" name="file_name" />`, where `file_name` is the name of the input field.
3. Type the caption for the file upload area so your visitors know what to do. Something like

“*What file would you like to upload?*” would work well.

4. Type `<input type="file"` to create a file upload box and a Browse button.
5. Type `name="title"`, where title contain the path of the file which is sent to the server the files being uploaded.
6. If desired, type `size="n"`, where n is the width, in characters, of the field in which the visitor will enter the path and file name.
7. Type the final `>`.
8. Complete the form as usual, including the submit button and final `</form>` tag.

Example:

```
<!DOCTYPE html>
<html>
<head><title> Creating Password Field</title>
</head>
<body>
<form enctype="multipart/form-data" method="post"
action="registration.php"
accept="image/gif,image/jpeg,image/jpg">
<b>First Name: </b><input type="text" name="firstname"
size="30" maxlength="15" value="Enter first name here" /><br /><br />
<b>Surname Name: </b><input type="text" name="surname"
size="30" maxlength="15" value="Enter Surname here" /><br /><br />
<b>Address: </b><input type="text" required= "required"
name="address" size="30"
maxlength="15" value="Enter Address here" /><br />
<b>Select Passport </b><input type="file" name="passport"
value="select passport" />
</form>
</body>
</html>
```

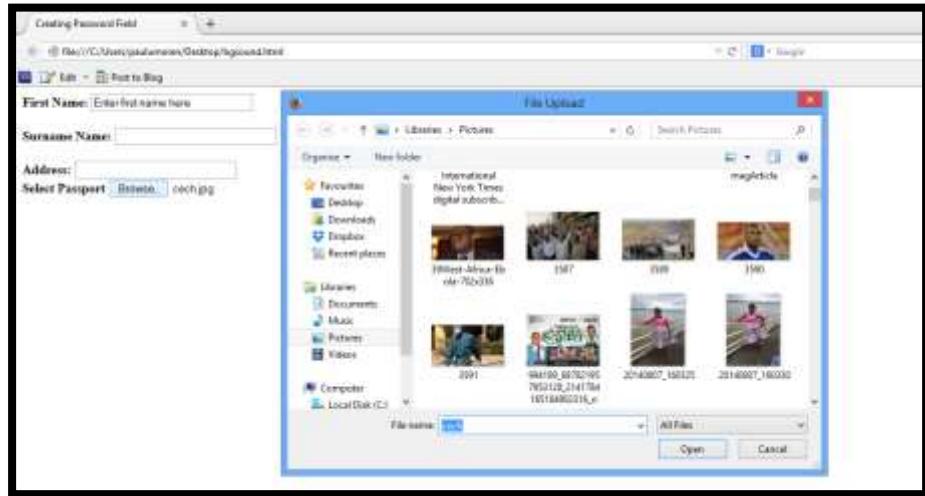


Fig. 5.3

Clicking on the *Browse...*button as shown in the diagram above will pop up a picture library folder for you to select your picture. You can as well choose from another location within your system.

5.2.6 Hidden Input Element

When you start developing robust system, the may be a situation you want to send hidden data to the server, the hidden input element handles this.

```
<input type="hidden" name="hiddenValue" value="php echo $regNumber ; ?&gt;" /&gt;</pre
```

Now the variable, \$regNumber is hidden from the user and not displayed on the browser with other information.

5.2.7 Multiple Text Area (<textarea> tag)

The <textarea> element is similar in some ways to the text input element, but it allows multiple lines of input rather than just one. It uses a pair of attributes, cols and rows to control its size, and instead of using a value attribute to present any textual instructional content, it uses the content of the element itself. It is a container element, rather than a self-closing element.

- The rows and cols attributes

A multiline text-input area stands alone onscreen: body content flows above and below, but not around it. You can control its dimensions, however, by defining the columns and rows attributes for the visible rectangular area set aside by the browser for multiline input. I suggest you set these attributes. The common browsers have a habit of setting aside the smallest, least readable region possible for <textarea> input, and the user cannot resize it.

Both rows and cols attributes require integer values for the respective dimension's size in characters. The browser automatically scrolls text that exceeds either dimension.

Example:

```
<textarea name="" rows="50" cols="20">
```

</textarea>

5.2.8 Disabled and Read-Only Elements

The disabled attribute enables you to display form elements that cannot be changed by the user and are intended to be displayed in a *greyed out* or in some other fashion that makes the disabled status obvious. The readonly attribute is very similar but it is not to be visually different from the other fields, just unchangeable.

Note: *Disable attribute does not send data to the server side via name attribute, but readonly does. So when you use scripting language to display data from the database with the intention to send it back for updating, use readonly attribute of the input element.*

Examples

```
<form method="post" action="#">
<table border="0" cellpadding="3">
<tr>
<td align="right">Name:</td>
<td><input type="text" name="yourname" /></td>
</tr><tr>
<td align="right">Login:</td>
<td><input type="text" name="login" /></td>
</tr><tr>
<td align="right">Host:</td>
<td><input type="text" name="host" value="hostname.com"
readonly="readonly" /></td>
</tr><tr>
<td align="right">Date:</td>
<td><input type="text" value="8 October, 2016"
disabled="disabled" /></td>
</tr>
</table>
</form>
```

Output

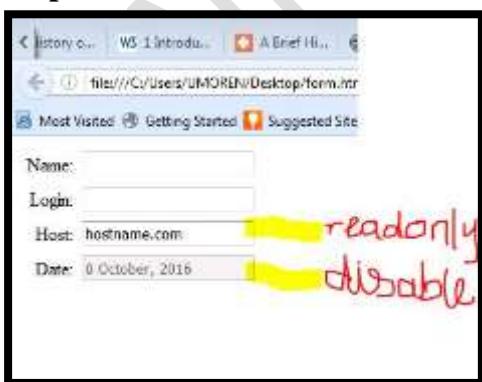


Fig. 5.4

5.3 Check boxes and Radio buttons

5.3.1 Checkbox: A checkbox input takes the form of a square box, with a check mark or an “x” character appearing when selected. As with all the input types so far, you can present the state of the check box, but rather than using the value attribute, you instead use the checked attribute. Like the randomly attribute, checked is a Boolean attribute, so if you are writing HTML or HTML5 you do not need to quote the value, but in XHTML you do. In checkbox, you can select more than one item; checkboxes are linked by the value of the name attribute.

To create checkboxes just type the following:-

```
<input type="checkbox" name="morning" checked="checked"/>
<input type="checkbox" name="afternoon" />
<input type="checkbox" name="evening" />
```

The code above will produce three checkboxes, but note the first line of code with **checked="checked"**, this will make the first checkbox to be checked at default.

5.3.2 Radio button

Unlike checkboxes, radio buttons indicate that only one choice out of several radio groups can be selected. Just like checkboxes, the value of the name attributes link radio buttons.

To create a radio button:

1. If desired, type the introductory text for your radio buttons. You might use something like Select one of the following.
2. Type <input type="radio".
3. Type name="radioset", where radioset both identifies the data sent to the script and also links the radio buttons together, ensuring that only one per set can be checked.
4. Type value="data", where data is the text that will be sent to the server if the radio button is checked, either by you (in step 5) or by the visitor. If nothing is in the value attribute, whatever that is selected from the radio button at default will be sent to the server script.
5. If desired, type checked="checked" to make the radio button active by default when the page is opened. You can only do this to one radio button in the set. (The ="checked" is optional in HTML/HTML5.)
6. Type the final />.
7. Type the text that identifies the radio button to the visitor.
8. Repeat steps 2 to 7 for each radio button in the set.

Example:

Code:

```
<!DOCTYPE html>
<html>
<head><title>Example of Radio buttons and Checkboxes</title>
</head>
<body>
<form enctype="multipart/form-data" method="post" action="registration.php"
accept="image/gif,image/jpeg,image/jpg">
<b>First Name:</b><input type="text" name="firstname"
size="30" maxlength="15" value="Enter first name here" /><br /><br />
<b>Surname Name:</b><input type="text" name="surname"
size="30" maxlength="15" value="Enter Surname here" /><br /><br />
<b>Address:</b><input type="text" name="address" size="30"
maxlength="15" value="Enter Address here" /><br />
<b>Select Passport </b><input type="file" name="passport" value="select
passport" /><br />
<h4> Example of Checkboxes</h4>
<b>Ms Word:</b><input type="checkbox" name="word"
checked="checked" /><br />
<b>Ms Excel:</b><input type="checkbox" name="excel"/><br />
<b>Ms Access:</b><input type="checkbox" name="access"/><br />
<h4>Example of Radio Buttons</h4>
<input type="radio" name="session" value="morningclass"
checked="checked" /><b>Morning Session:</b><br />
<input type="radio" name="session"
value="afternoonclass" /><b>Afternoon Session:</b><br />
<input type="radio" name="session" value="eveningclass" /><b>Evening
Session:</b><br />
<input type="radio" name="session" value="Cheddar" /> Cheddar
<input type="radio" name="session" value="Stilton" /> Stilton
<input type="radio" name="session" value="Brie" /> Brie
</form>
</body>
</html>
```

Output result



Fig. 5.5

5.4 The wrap attribute

Normally, the browser sends the text that you type into the text area to the server exactly as typed, with lines broken only where the user pressed the Enter key. Because this is often not the action the user desired, you could enable word wrapping within the text area. When the user types a line that is longer than the width of the text area, the browser automatically moves the extra text down to the next line, breaking the line at the nearest point between words in the line. With the wrap attribute set to virtual, the text wraps within the text area for presentation to the user, but transmits to the server as though no wrapping had occurred except where the user pressed the Enter key.

With the wrap attribute set to physical, the texts wrap within the text area and transmit to the server as though the user had actually typed it that way. This is the most useful way to use word wrap because the texts are transmitted exactly as the user sees them in the text area.

To obtain the default action, set the wrap attribute to off.

As an example, consider the following 60 characters of text that are being typed into a 40-character-wide text area, with **wrap=off**, the text area contains one line and the user must scroll to the right to see all of the text. But with **wrap=virtual**, the text area contains two lines of text, broken after the word makes. Only one line of text is transmitted to the server: the entire line with no embedded newline characters.

With **wrap=physical**, the text area contains two lines of text, broken after the word makes. Two lines of text are sent to the server, separated by a newline character after the word makes.

Example:

Code:

```
other Info:<br />
<textarea wrap="physical" col="20" row="5"></textarea>
</form>
.
```

Output Result:

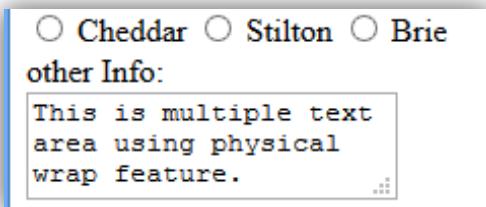


Fig. 5.6

5.5 Submit Button

Creating the Submit Button

Submit button makes it possible for users to send all the data entered in the form to the server. There are different ways of creating submit button, but to avoid any issue with some browsers, we will only stick to one, though others will be introduced.

To create a submit button:

1. Type `<input type="submit">`.
2. If desired, type `value="submit message"` where submit message is the text that will appear in the button.
3. Type the final `</>`.

Example:

Code Snippet

```
other Info:<br />
<textarea wrap="physical" col="20" row="5"></textarea><br />
<input type="submit" name="submit" value="Submit" />
</form>
</body>
.
```

Output Result

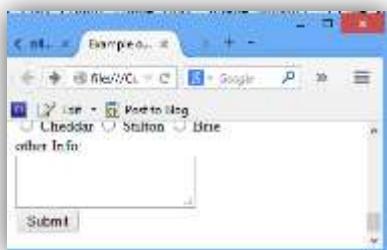


Fig. 5.7

To create a submit button with an image:

1. Type `<button type="submit">`.
2. Type the text, if any that should appear on the left side of the image in the button.
3. Type `` to complete the image.
7. Type the text, if any that should appear on the right side of the image in the button.
8. Type `</button>`

Example:

```
other Info:<br />
<textarea wrap="physical" col="20" row="5"></textarea><br />
<input type="submit" name="submit" value="Submit" /><br />
<button type="submit">
/></button>
</form>
</body>
</html>
```



Fig. 5.8

5.6 Creating Pull-down Menus

Menus are perfect for offering web visitors options of choice to choose from, the `<select>` tag is for menus, which give two compact alternatives: pull-down menus and scrolling lists. This element has only three specific attributes: name, size, and multiple. The name attribute is used here to identify the data collected from the menu when it is sent to the server, and it is not optional when interacting with the database— every `<select>` must have a name attribute. The size and multiple attributes in form element are used to select multiple options from the drop-down list. The multiple attribute is set (`multiple="multiple"` if you are writing XHTML); otherwise, just multiple will do the size attribute, which accepts a numerical value, determines how many rows of options are displayed.

Example:

Code

```
<input type="radio" name="session" value="Brie" /> Brie<br />
<select name="qualification">
<option>qualification</option>
<option>SSCE</option>
<option>Bachelor Degree</option>
<option>Master Degree</option>
<option>Doctorate Degree</option>
</select><br />
other Info:<br />
<textarea wrap="physical" col="20" row="5"></textarea><br />
<input type="submit" name="submit" value="Submit" /><br />
<button type="submit"></button>
</form>
</body>
</html>
```

Output Result:

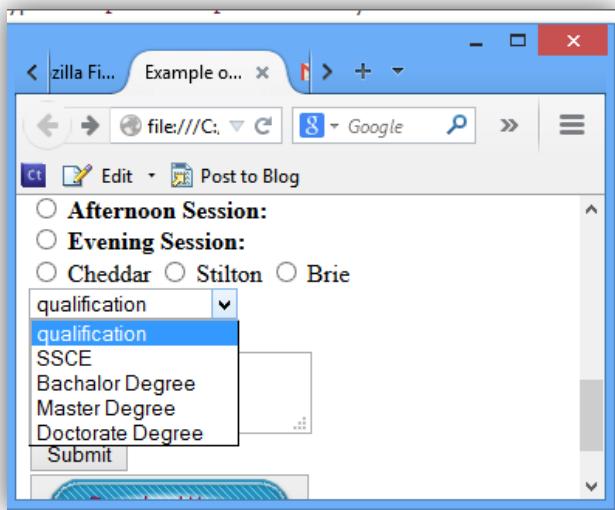


Fig. 5.9

5.6.1 The <option> tag

The **<option> tag** has three specific attributes: selected, value, and label. The selected attribute is used to indicate to the user that a particular <option> should be selected initially; without it, the browser may display either nothing at all (just a blank select box) or the first <option> it encounters.

```
<select name="cheesemenu">
<option>Cheddar</option>
<option selected='selected'>Stilton</option>
<option>Brie</option>
</select>
```

Here, stilton is preselected because of this line `<option selected='selected'>Stilton</option>` of markup.

Multiple <option> allows visitors to the website to select more than one menu option. If desired to use it, type `multiple="multiple"` for xhtml, but for html, "multiple" = "multiple" is optional, just multiple is okay.). With Ctrl or shift key press, you can do the multiple selections of the options.

Example:

Code:

```
:
.
<select name="degrees" multiple="multiple">
<option>SSCE</option>
<option selected="selected">Bachelor
degree</option>
<option>Master Degree</option>
<option>Doctorate Degree</option>
</select>
:
.
```

Output Result:

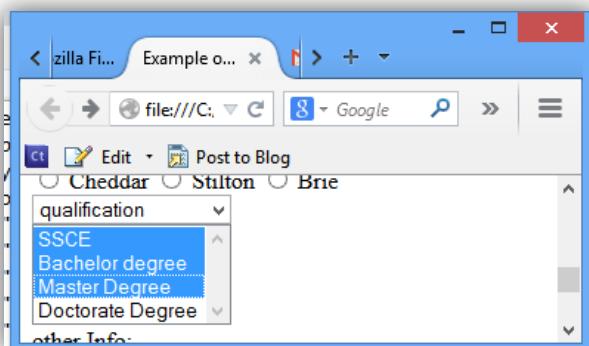


Fig. 5.10

5.6.2 The <optgroup> tag

To help provide structure to your menus, you can use the `<optgroup>` element to group Similar `<option>` elements. So, instead of the following markup:

```
<select name="foodmenu">
<option>- - -Break fast- - -</option>
<option value="tea">tea and bread</option>
<option value="rice">Rice and Stew</option>
<option> - - - Lunch- - -</option>
<option value="garri">Garri and soup</option>
<option value="amala">Amala </option>
<option value="yam">Yam and Stew </option>
<option> - - -Supper- - -</option>
<option value="vegetable">Vegetable Sauce </option>
<option value="plantain">Plantain and Stew</option>
<option value="beans">Beans and Rice </option>
</select>
```

you would use this:

```
<select name="foodmenu">
<optgroup label= "Break fast>
<option value="tea">tea and bread</option>
<option value="rice">Rice and Stew</option>
</optgroup>
<optgroup label= "Lunch ">
<option value="garri">Garri and soup</option>
<option value="amala">Amala </option>
<option value="yam">Yam and Stew </option>
</optgroup>
<optgroup label = "Supper ">
<option value="vegetable">Vegetable Sauce </option>
<option value="plantain">Plantain and Stew</option>
<option value="beans">Beans and Rice </option>
</optgroup>
</select>
```

Output Result

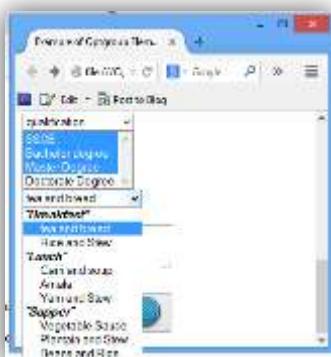


Fig. 5.11

In the above markup, you will notice that value attribute is used. This allows the submission of a value that differs from the content of the `<option>`, and if it is not present, the content instead is used as value. Just take note of this, this will be useful when you start developing a robust web applications.

5.7 Fieldset Element

The `<fieldset>` element allows web authors to divide form controls into thematically linked sections, making it easier for users to work through the form while also enhancing accessibility for assistive devices for people with disabilities. To identify each `<fieldset>`, you must use the `<legend>` attribute:

Example:

Markup

```
:  
  
<fieldset width="30">  
<legend> Example of Checkboxes</legend>  
<b>Ms Word:</b><input type="checkbox" name="word"  
checked="checked" /><br />  
<b>Ms Excel:</b><input type="checkbox" name="excel"/><br />  
<b>Ms Access:</b><input type="checkbox" name="access"/><br />  
</fieldset>  
:  
.
```

Output Result

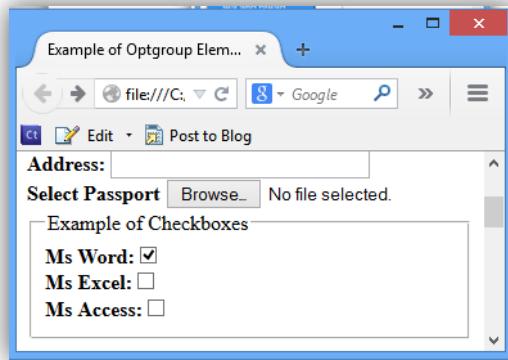


Fig. 5.12

5.8 The Anchor Tag and the href Attribute

HTML uses the **<a>** (anchor) tag to create a link to another document.

An anchor tag can point to any resource on the Web: an HTML page, an image, a sound file, a movie, etc.

The syntax of creating an anchor:

```
<a href="url">Text to be displayed</a>
```

The **<a>** tag creates an anchor to link from, the **href** attribute addresses the document to link to, and the words between the open and close of the anchor tag is displayed as a hyperlink.

5.8.1 The Target Attribute

With the target attribute, you can define **where** to open the linked document such as new tab on the browser or new window on the same browser.

The line below will open the document in a new tab of a browser window:

```
<a href="http://www.google.com.ng/"  
target="_blank">Visit Google Search Engine!</a>
```

Creating an image link with Anchor tag

```
<html>
<body>
<a href='download.htm'>
<img border='0' src='images/buttonnext.gif' width='65' height='38'>
</a>
</body>
</html>
```

The image is displayed on the page when you start your browser, clicking the image load **download.htm** page.

Example- creating a link to another file within the same location:

```
<html>
<head>
<title></title>
</head>
<body>
This is to demonstrate how to link to another
page using anchor <a> tag and href attribute.
clicking on the like <a href='table.html'>
Residence of my friends and their character</a>will
take you to the page
</body>
```

Output Result



Fig. 5.13

When the link is clicked, it displays the link page below:

| Name | Place of residence | Behavioural Status |
|-----------------------|-------------------------|-------------------------------------|
| Paul Umoren | Jos, Plateau | Complex, Unpredictable and Carefree |
| Florence Okon | Calabar, Cross River | Careful and Focusing |
| Justice Okpor | London | Funny and Occupying |
| Beatrice Udu | Ogoja, Cross River | Passionate and Selective |
| Ugochi Ahiamanya | Victoria Island, Lagos | Just good |
| Fanta Adamsu | Rayfield, Plateau | Creative and carefree |
| Joseph Agbor | Calabar, Cross River | Political and conscious |
| Cynthia Erte | Palmettary, Cross River | Calm and Stressfree |
| Mary Kemi | Tudu Wada, Plateau | My world and my friends |
| Andrew Naughty Friend | Tudu Wada, Plateau | Discipline and assistance |

Fig. 5.14

5.8.2 Linking to a Specific Anchor with a page

Once you have created an anchor, you can define a link so that a user's click brings them directly to the section of the document that contains the anchor, not just the top of that document.

To create a link to an anchor:

Type ``, where anchor name is the value of the name attribute in the destination's `a` tag or the value of the destination's id attribute

Example:

```
<html>
<head>
<title></title>
</head>
<body>
<h3>Table of Content</h3>
<div>
<a href="#intro">Introduction</a><br />
<a href="#description"> Description of Characters</a><br />
<a href="#setting"> Setting of the Sun</a/><br />
<a href="#climax"> Climax</a><br />
</div>

<h2><a name="intro">Introduction</a></h2>
<p>
This is the description of the play, "Me and you". It illustrates the
concept of individuality in conjunction with eternity of human
existence. It is a well-designed play with African setting that portrays
the heritage of African race.
Feel at home going through all the scenes and revamp your
understanding of African Culture.
</p>
```

```
<h2><a name="description">Description of Characters</a></h2>
<p>There are four key characters in the play, Bola, Aishia, Okon
and Emaka. Bola is the wife of Okon who is from the Southern region
of Nigeria, West Africa. Emaka is the husband of
Aisha from the Northern region of Nigeria while Emeka is from the
Eastern Part of Nigeria,
West Africa. The key player of the scene is Aishia </p>
</body>
</html>
```



Fig. 5.15

Clicking on the Description of Characters link will display the link section within the page as shown below.



Fig. 5.16

5.8.3 The autocomplete attribute

This attribute enables previous user inputs to be recalled and automatically entered into field as suggestions.

This attribute can be enabled for the entire form fields as well as disable for some fields as shown in the example below;

```
<form action='myform.php' method='post' autocomplete='on'>
  <input type='text' name='username'>
  <input type='text' name='address'>
  <input type='text' name='phone_number'>
  <input type='password' name='password' autocomplete='off'>
</form>
```

5.8.4 The required Attribute

The required attribute is used to ensure that a field has been completed before a form is submitted by user. This implies that a user cannot submit an empty field that is required.

```
<input type='text' name='surname' required>
```

When the browser detects attempted form submission where there's an uncompleted required input, a message is displayed, prompting the user to complete the field.

The list Attribute

This enable users to easily select from a predefined list attached list to the input;

```
Select destination: <input type='url' name='site' list='links'>
<datalist id='links'>
  <option label='Google' value='http://google.com'>
  <option label='Yahoo!' value='http://yahoo.com'>
  <option label='Bing' value='http://bing.com'>
  <option label='Ask' value='http://ask.com'>
</datalist>
```

5.8.5 Date and Time Pickers

When you choose an input type of date, month, week, time, datetime, or datetimelocal, a picker will pop up on supported browsers from which the user can make a selection based on the chosen input type as shown below;

```
<input type='date' name='time' value='dd-mm-yyyy'>
```

Exercise:

Follow the example above to complete the remaining link within the page for better understanding.

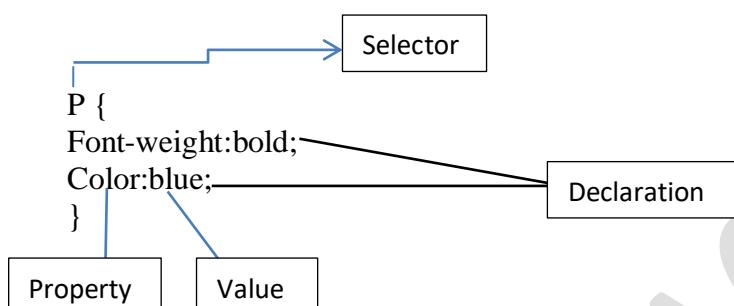
PART TWO
Cascading Style Sheet (CSS)

PINFO ICT ACADEMY

Chapter Six Cascading Style Sheet (CSS)

CSS means Cascading Style Sheet; it is the presentation part of web page layout. The main purpose of CSS is to separate structure (HTML) from presentation (CSS). CSS gives the beauty and flexibility in Web design such as the colour, background colour, background image, drop shadows and many more. CSS consists of style **rules**; each of these rules consists of a **selector** and **declaration** of property-value pairs. The rules tell web browser that understand them how to display specific types of content structures when it encounters these structures while delivering a web page to users. In this section, we will discuss implementation of CSS 2.0 and CSS 3.0

6.1 Part of Style Rule



1. A **selector** in CSS defines and references the HTML element(s) to which the rule applies; it tells the browser what to style. Examples of selectors are **p**, **body**, **table**, **caption tags**, etc. When more than one selector is in one rule, they are called **grouped**. You can use multiple selectors together in a single rule by providing a comma after each selector. This implies that rule applies to more than one selector at a time.

Example:

grouped selectors

```

th, td {
    padding: 0 12px 0;
    text-align: left;
    font-weight: bold;
}

```

declaration

2. A **declaration** block is a container that consists of everything between (and including) the curly brackets. It is composed of a property name, a colon, and one or more values depending on the property.
3. The **value** is the exact style you want to set for the property. Values can include length, percentage, color, url, keyword, and shape.
4. The **property** is an aspect of the element that you are choosing to style. There can be only one property within each declaration unless a shorthand property is used.

Examples

```

h1 {
}

```

The Selector indicates that this applies to all heading in the page

```
color: red;  
}
```

```
p1{  
font-size: small;  
color: green;  
font-family: New York, Times, serif;  
}
```

The selector `p` indicates that the style rules should be applied to all paragraphs in the document. The font-family values tell the browser to use New York as the font if the user's machine has it installed. If not, it directs the browser to use Times. And if neither of these fonts is available on the user's system, the browser should use the default font serif.

6.1.2 Length and measurement in CSS

The two kinds of lengths used in CSS are relative and absolute lengths. Absolute measurements define real world units such as inches, centimetres, points, etc., and mostly used in print work. Relative measurement is suited for onscreen layout; the four types of relative measurement that CSS allows are emphasis (em), x-height (ex), pixels (px) and percentage (%).

Ways of including CSS in html document

CSS declaration can be included in a document as follows;

1. **Inline CSS Declaration:** This is applying CSS declaration directly inside an HTML element within the HTML document.

Example :

```
<h2 style= "color: red; text-transform: uppercase;"> An Unusual Topic</h2>  
<body style= "font: 14px Times">;
```

2. **Internal (Embedded) CSS declaration;** this is included between `<style>` and `</style>` tags in the head element of the web page. It may be used if a single page has a unique style.

Example;

```
<!DOCTYPE html>  
<html lang="en" dir="ltr">  
<head>  
  <meta charset="utf-8">  
  <title>Internal CSS Declaration</title>  
  <style type= "text/css">  
    h1 ,h2{  
      color: green;  
    }  
    h3{  
      color: blue;  
    }  
  </style>  
</head>
```

3. **External CSS (in an external CSS file):** here, CSS is defined in a file that is completely separated from the web page. You then link to this file by including a link element in the

head of any web page on which you want to implement the CSS styles. This is a preferred inclusion method in html documents. This implies that a single style sheet can be applied to multiple web pages on a website, and is cached by the user's browser. Caching free the browser from continuously downloading CSS files leading to faster page load and less bandwidth used per page request. So with external CSS you can apply style rules to entire website in a single file.

Example:

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>External CSS Declaration</title>
    <link rel="stylesheet" type="text/css" href="copstyle.css" /> </head>
```



In the above CSS declaration;

- **href** : This specifies the path to the CSS file (which is often placed in a folder called **css** or **styles**).
- **type** : This attribute specifies the type of document being linked to. The value should be **text/css**. Though it is optional in HMTL5 standard.
- **rel** : This specifies the relationship between the HTML page and the file it is linked to. The value should be **stylesheet** when linking to a CSS file.

The file: copstyle.css

```
h1 ,h2{
  color: green;
}
h3{
  color: blue;
}
```

4. CSS declaration can be imported into an html document using **@import** rule.

Example;

```
<style type= "text/css">
@import url(css/copstyle.css);
</style>
```

6.2 Color and CSS

There are different options for specifying colours in CSS. These options are as follows;

- ⇒ Color keywords: here, color is specified by its name. There are 147 predefined colour names recognize by browsers.

Examples, white, orange and red color will be as shown below:-

CSS code

```
h1 ,h2{
  color: green;
}
h3{
```

```
        color: blue;
    }

body {
    color: white;
    background-color: #FFA500;
    border: thin solid red;
}
```

In the above example, I have specified that each body element in the html document should have white text, an orange background, and a thin solid red border.

Html code

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
    <head>
        <meta charset="utf-8">
        <title>External CSS Delaration</title>
        <link rel="stylesheet" href="copstyle.css" />
    </head>
    <body>

        <h1>Heading </h1>
        <div>
            Lorem ipsum dolor sit amet,
            consectetur adipiscing elit. Maecenas
            porttitor congue massa. Fusce posuere,
            magna sed pulvinar ultricies, purus lectus malesuada libero,
            sit amet commodo magna eros quis urna. Nunc viverra imperdiet enim.
            Fusce est. Vivamus a tellus. Pellentesque habitant morbi tristique senectus et
            netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris
            et orci. Aenean nec lorem.
        </div>
    </body>
</html>
```

Result

Heading

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna. Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci. Aenean nec lorem.

- ⇒ Hexadecimal: this allows you to specify a colour by special hexadecimal number that uses 6 digits code representation of red, green and blue amount in colour and preceded by a hash (#) sign.

Example

```
div{  
    color: #808080;  
    background-color: #FFA500;  
    border: thin solid #FF0000;  
}
```

The `#808080` is the hexadecimal representation of gray, `#FFA500` is for orange and `#FF0000` is for red.

- ⇒ Shorthand hexadecimal: this is a shorten representation of hexadecimal numbers.

Example;

```
div{  
    background-color: #FAB;  
}
```

`#FAB` is a shorthand hexadecimal of `#FFAABB` which is for pink colour.

- ⇒ RGB values: this allows you to specify a color through three primary colours; Red, Green and Blue. Using these colours in various combinations makes it possible to create every colour of the rainbow. CSS RGB colour uses special three numbers syntax.

Example;

```
div{  
    background-color: rgb(128, 128, 128);  
}
```

The above example is CSS keyword colour **gray**. White is 255, 255, 255 and black is 0, 0, 0.

- ⇒ RGB percentage: this is the same as RGB but uses percentage in specifying color.

- ⇒ RGBA: the A stands for alpha channel, which enables you to specify transparency (Opacity). CSS3 introduces the this property which allows you to specify the opacity of an element and any of its child elements. The value is a number between 0.0 and 1.0 (so a value of 0.5 is 50% opacity and 0.15 is 15% opacity). The CSS3 rgba property as allows of colour with an RGB value, but adds a fourth value to indicate opacity know an

alpha value and is a number between 0.0 and 1.0 (so a value of 0.5 is 50% opacity and 0.15 is 15% opacity). The rgba value will only affect the element on which it is applied (not child elements).

- ⇒ **HSL notation:** A color value can be set with the hsl() function (which stands for hue, saturation, and lightness). Hue is a degree on a color circle from 0 to 360, where 0 and 360 are red, 120 is green, and 240 is blue. Saturation is a percentage value, with 0% giving a shade of gray and 100% giving the full color. Lightness is also specified as a percentage, from 0% (black) to 100% (bright).

```
p { color: hsl(0, 100%, 100%); }
```

Although HSL colors are more intuitive than RGB colors, and the color values are easier to tweak, HSL should not be used until IE8 usage declines to a point when it is no longer necessary for web sites to support it. HSL is a CSS 3 value and is supported in Chrome, Firefox, IE9+, Safari, and Opera 10+.

HSLA notation Similar to RGB, the HSL notation can be extended with an alpha value for specifying the transparency.

```
/* Red with 50% transparency */ p { color: hsla(0, 100%, 100%, 0.5); }
```

HSLA is supported in Chrome, Firefox 3+, IE9+, Safari, and Opera 10+, which is the same as the RGBA function.

Example:

Html code

```
<div class="paragraph1">
    Lorem ipsum dolor sit amet,
    consectetur adipiscing elit. Maecenas
    porttitor congue massa. Fusce posuere,
    magna sed pulvinar ultricies, purus lectus malesuada libero,
    sit amet commodo magna eros quis urna. Nunc viverra imperdiet enim.
    Fusce est. Vivamus a tellus. Pellentesque habitant morbi tristique senectus et
    netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et
    orci. Aenean nec lorem.
</div>

<div class="paragraph2">
    Lorem ipsum dolor sit amet,
    consectetur adipiscing elit. Maecenas
    porttitor congue massa. Fusce posuere,
    magna sed pulvinar ultricies, purus lectus malesuada libero,
    sit amet commodo magna eros quis urna. Nunc viverra imperdiet enim.
    Fusce est. Vivamus a tellus. Pellentesque habitant morbi tristique senectus et
    netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et
    orci. Aenean nec lorem.
</div>
```

CSS Code

```
.paragraph1{  
    width:500px;  
    margin-left:auto;  
    margin-right:auto;  
    margin-bottom:12px;  
    background-color:rgb(102,51,0);  
}  
.paragraph2{  
    width:500px;  
    margin-left:auto;  
    margin-right:auto;  
    margin-bottom:12px;  
    background-color:rgba(102,51,0,0.5);  
}
```

Result

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna. Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci. Aenean nec lorem.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna. Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci. Aenean nec lorem.

In the above example, paragraph2 CSS class is set to 0.5 opacity, which is 50% transparency.

Note: In order not to get confuse as a beginner to CSS, stick to one, properly hexadeciml colour combination. However, it is good to know other colour combination in case you encounter them while editing someone else's work.

The main elements you will set colour for are:

- Text
- Border
- Headings
- Page background
- Background colours of text and headings

6.3 Selectors

Selectors are CSS rule set used to select HTML elements on our web pages so we can apply style to them.

Types of CSS selectors

1. Type selector
2. Class selector
3. ID selector
4. Descendant selector

6.3.1 Class and ID Selectors

Class and id selectors are the most widely used selectors apart from the type selectors that reference html element and Descendant selector, which matches all elements that are descendants of specific element.

Class Selector

Class selector references class attributes used on html element. Class name selector begins with a dot, followed by name class itself. The class name should start with a letter, no space. Name class can be combination of alphabets and numbers.

Example;

 Pay Attention in Class and Do All The Assignment

The class selector for this is;

CSS code

```
.important{  
    width: 300px;  
    margin-left:auto;  
    margin-right:auto;  
    font-weight: bold;  
    background-color: yellow;  
    color: red;  
}
```

HTML Code

```
<div class="important">  
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed  
pulvinar ultricies, purus lectus malesuada libero,sit amet commodo magna eros quis urna. Nunc viverra imperdiet enim.  
Fusce est. Vivamus a tellus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis  
egestas. Proin pharetra nonummy pede. Mauris et orci. Aenean nec lorem.  
</div>
```

You can combine two types of selectors together in the example below; type selector (div) combine with a class selector;

```
div .important{  
    font-weight: bold;  
    background-color: yellow;  
    color: red;  
}
```

In the above example, the rule only applied to div element and no other element in the document.

ID Selectors

These are unique identifiers. An ID is defined once per HTML document. ID name is preceded with a hash mark (#) and should start with letter, no space and can be combination of letters, numbers, underscores(_) and hyphen(-). The uniqueness rule of declaring the ID only applies to naming the elements, not references to them.

Example:

CSS Code

```
#main {  
    width:500px;  
    height:350px;  
    background-color:#66CC00;  
    margin-right:auto;  
    margin-left:auto;  
  
}  
h1 ,h2{  
    color: green;  
}  
h3{  
    color: blue;  
}  
  
body {  
    color: white;  
    background-color: #FFA500;  
    border: thin solid red;  
}  
.important{  
    width: 300px;  
    margin-left:auto;  
    margin-right:auto;  
    font-weight: bold;  
    background-color: yellow;  
    color: red;  
}
```

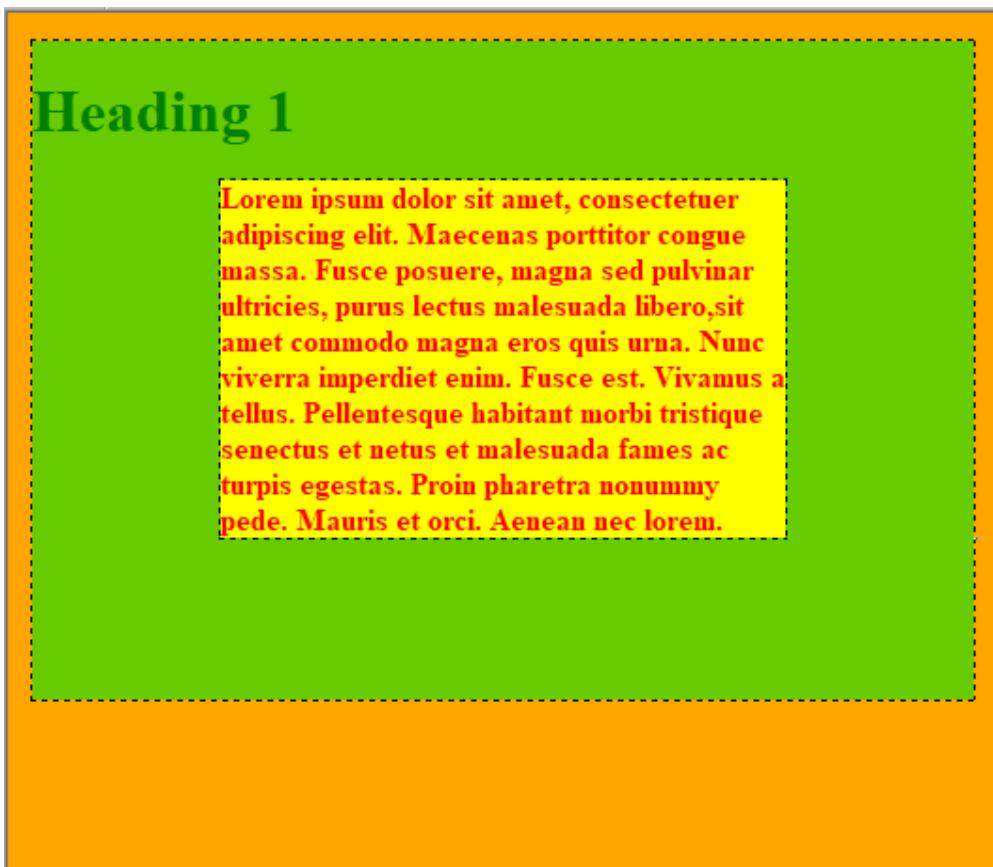
HTML Code

```
<!DOCTYPE HTML>  
<html>  
<head>  
<meta charset="utf-8">  
<title>ID Selector Example</title>
```

```
<link rel="stylesheet" href="copstyle.css" >
</head>

<body>
<div id="main">
<h1>Heading 1</h1>
<div class="important">
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna. Nunc viverra imperdiet enim.
    Fusce est. Vivamus a tellus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci. Aenean nec lorem.
    </div>
</div>
</body>
</html>
```

Result



6.3.2 The Type Selector

Type selector target html elements type of the page because html's tags are used as the selectors. Type selectors are generally used to make a broader changes to the style of a site.

Example

```
<style type="text/css">

p { /* "p" is the type selector which affect all the <p> tag on the page*/
    margin: 0 0 1em 0;
}

article, aside, details, figcaption, figure, footer, header,
main, nav, section, summary, canvas {
    display: block;
}
</style>
```

The Type Selectors above are setting the display property of those tags to block in html 5, so that any time those used throughout the site, they will display as block.

CSS Code

$h1, h2\}$

```
        color: green;
    }
    h3{
        color: blue;
    }
    p{
        text-align:center;
    }

body {
    color: white;
    background-color: #FFA500;
    border: thin solid red;
}
div{
    width:250px;
    height:250px;
    background-color:blue;
    margin-left:auto;
    margin-right:auto;
}
HTML Code
<!DOCTYPE HTML>
<html>
<head>
<meta charset='utf-8'>
<title>ID Selector Example</title>
<link rel="stylesheet" href="copstyle.css" >
</head>

<body>
<div id="main">

    <h1>Heading 1</h1>

    <div class="important">
        <blockquote>
            <p>
                    Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna. Nunc viverra imperdiet enim.
            </p>
        </blockquote>
    </div>
</div>
```

```
</body>  
</html>
```

Result



6.3.3 Descendant selector

Use this operator when you want to be more specific with selectors on html page. For instance, rather than targeting all the anchor tags on the web page, you only target anchor tags within an ordered list, this is specifically when the descendant selector comes in the picture.

Example

HTML Code

```
<!DOCTYPE HTML>  
<html>  
<head>  
<meta charset='utf-8'>  
<title>Exampel of Descendant Selector</title>  
<link rel='stylesheet' href='copstyle.css' >  
</head>
```

```
<body>
<div class='important'>
<h1>Role Players</h1>
<ol>
<li><a href="#">Paul Umoren</a></li>
<li><a href="#">Anthony Udo</a></li>
<li><a href="#">Mfon Smart</a></li>
<li><a href="#">Daniel Akpan</a></li>
</ol>

</div>
</body>
</html>
```

CSS Code

```
.important{
    width: 300px;
    margin-left:auto;
    margin-right:auto;
    font-weight: bold;
    background-color: yellow;
    color: red;
}
ol li a{
    font-weight:bold;
    font-style:italic;
    color:red;
}
```

Result



6.3.4 The Universal Selector

Universal selector is an asterisk, when used alone, it tells the CSS interpreter to apply the CSS rule to all elements in the HTML document. However, it is not commonly used in a document.

Example

```
*{  
    color: red;  
}  
  
/* All elements in the page are given a  
text color of red. */
```

6.4 Pseudo-Classes/Hyperlink Styling

Pseudo-Classes allow the author the freedom to dictate how the element should appear under different conditions. It presents dynamic events and a change in state present in html document such as hyperlinks; not easily achieved through other means. Links are active navigational elements that visitors to your site use to change location to another page either within the website or to another website entirely. When dealing with hyperlink styling, consider using anchor pseudo-classes. Pseudo-classes have single colon (:) before its properties, which is different from normal classes.

6.4.1 Dynamic Pseudo-classes in CSS

The following are dynamic pseudo-classes in CSS:

- ⇒ `:link` → this shows unvisited hyperlink
- ⇒ `:visited` → this indicates visited hyperlink
- ⇒ `:hover` → this shows an element that currently has the user's mouse pointer hovering over it.
- ⇒ `:focus` → this shows the link that is currently focus by a user using a keyboard to navigate the menus. It is also known as a keyboard focus.
- ⇒ `:active` → this indicates which the user is currently clicking on the web page and holding down the mouse button. The specified style remains in place while the user holds down the mouse button, and the element does not return to its original state until the user releases the mouse button.

Other types of pseudo classes are `:first-letter`, `:first-line`, `:first-child`, `:last-child`, `:nth-child(even)`, `:nth-child(odd)`, `:after`, `:before`.

Example;

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
<title>Pseudo-classes Examples</title>  
<style type="text/css">  
a:link{  
    text-decoration: none;  
    color: mediumblue;  
}  
a:visited {
```

```
color: magenta;
}
a:focus,
a:hover{
text-decoration: underline;
}
a:active{
color: red;
}
</style>
</head>
<body>
<h1>Links to useful sites</h1>
<ul>
<li><a href="http://www.pinfotechnologies.net/">Pinfo Technologies</a></li>
<li><a href="http://www.google.com/">Google</a></li>
<li><a href="http://www.yahoo.com/">Yahoo</a></li>
<li><a href="http://www.amazon.com/">Amazon</a></li>
</ul>
</body>
</html>
```

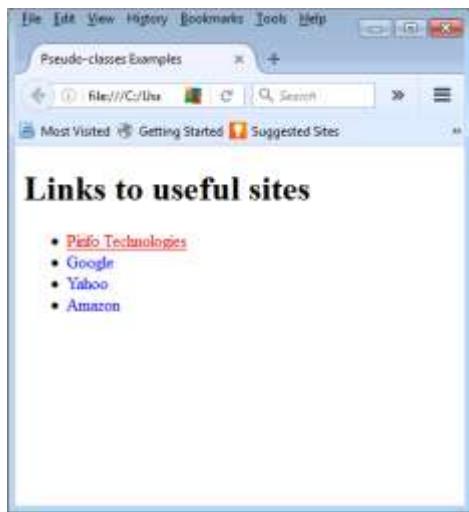


Fig. 6.1

The first dynamic pseudo-class that you used styles unvisited links. Unvisited links receive the color **mediumblue** and have the underline removed.

```
a:link {
text-decoration: none;
color: mediumblue;
}
```

The visited links receive the color magenta.

```
a:visited {  
    color: magenta;  
}
```

When a user's mouse cursor comes over a link or has the link has keyboard focus, the link is underlined.

```
a:focus,  
a:hover {  
    text-decoration: underline;  
}
```

When the user clicks and holds down the mouse button, the :active dynamic pseudo-class applies the style rule to the link as red.

```
a:active {  
    color: red;  
}
```

The above pseudo-code should be used in order. Love hate="lvha"

Grouping selectors

You can group selectors to apply the same style rules to them by separating them with commas

Two pseudo-code classes can be used in one rule as shown below.

Example:

```
a:visited,a:hover,a:focus,a:active{  
    color:blue;  
    background-color: transparent;  
}
```

It is a good idea to apply pseudo classes to an element in the following order: ***a:link, a:visited, a:hover, a:focus, a:active*** without this order, the following pseudo classes would not work as expected. An easy way to remember this order is the phrase "***Lord Vader Hates Furry Animals,*** where L stands for link, V stands for visited, H for hover, F for focus, and A for active.

6.4.2 Using Pseudo-class to apply a drop cap to a document

First-letter: This means applying the effect to first letter in the sentence.

First-line: applies styles to the first line of a block-level element. Note that the length of the first line depends on many factors, including the width of the element, the width of the document, and the font size of the text.

Example of first-letter Pseudo-class selector

CSS Code

```
p:first-letter{  
    font-family: serif;  
    font-size:500%;  
    float:left;  
    color: gray;
```

}

HTML Code

<p>

This image illustrates windows re-installation process. When you stopped working due to the fact that this has been crashed.

</p>

Result



Fig. 6.2

Example of first-line Pseudo-class selector

A screenshot of a Microsoft Notepad window titled "first_line - Notepad". The file path is "file:///C:/Users/paulumoren/Desktop/first_line.htm". The content of the file is an HTML document. A red circle highlights the CSS rule "p::first-line { background-color: yellow; font-weight:bold; }". A red arrow points from the word "Example" to this highlighted rule.

```
html>
<head>
<style>
p::first-line {
    background-color: yellow;
    font-weight:bold;
}
</style>
</head>
<body>
<h1>SYMPTOMS OF ARTHRITIS</h1>
<p>Some of the symptoms of arthritis include; Stiffness of the joint, Swelling of the joint, Pains in the joint, Weight loss, Difficulty moving the joint, Fever,Muscle ache</p>
</body>
</html>
```

Fig. 6.3

Result

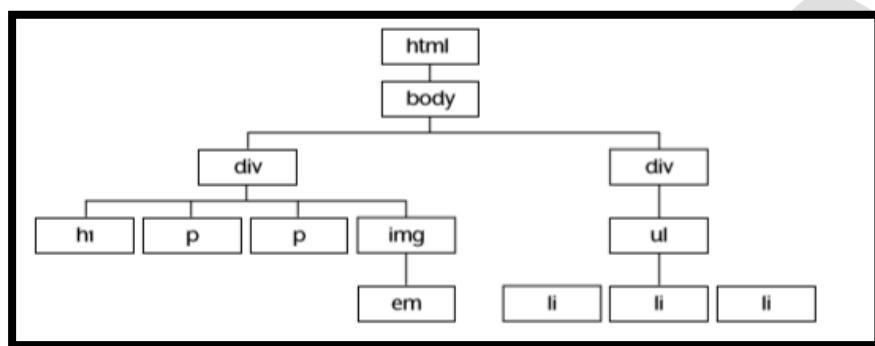


Fig. 6.4

Structural Pseudo-classes

CSS3 introduces the concept of “structural pseudo-classes” to target elements in the document tree based on unique characteristics of the elements, such as relative placement. All of the structural pseudo-classes are based on the document tree, also called the document object model (DOM). The document tree is the hierarchical structure of the HTML page, made up of elements, attributes, and text, each called a node. It contains multiple levels because elements are nested inside each other.

Elements nested directly inside other elements are called children of those outer elements; they’re also descendants, along with elements that are nested further down. The outer elements are called parents (if one level up) or ancestors (if two or more levels up). Elements that are nested at the same level with each other—in other words, they have the same parent—are called, predictably, siblings.



Below are structural pseudo-classes for CSS 2.1 and CSS 3.0

| PSEUDO-CLASS | DESCRIPTION |
|---------------------|--|
| :root | Selects the element that is the root of the document. In HTML, this is always the <code>html</code> element. |
| :nth-child() | Selects based on position within the list of its parent's children. |
| :nth-last-child() | Same as :nth-child(), but the counting for the position number is done from the last child upward instead of the first child downward. |
| :nth-of-type() | Selects based on position within the list of its parent's children, but only counting children of a certain type (such as <code>p</code> , <code>img</code> , etc.). |
| :nth-last-of-type() | Same as :nth-of-type(), but counting from the last child of the specified type instead of the first. |
| :first-child | Selects the first child of a parent element. (In Figure 5.1, the <code>h1</code> element is a first child.) |
| :last-child | Selects the last child of a parent element. (In Figure 5.1, the <code>img</code> element is a last child.) |
| :first-of-type | Selects the first sibling of its own type in a parent element. (In Figure 5.1, the first <code>p</code> element would be selected by <code>p:first-of-type</code> .) |
| :last-of-type | Selects the last sibling of its own type in a parent element. |
| :only-child | Selects an element that is the only child of its parent. (In Figure 5.1, the <code>ul</code> element is an only child.) |
| :only-of-type | Selects the only element of its own type in the parent element. |
| :empty | Selects elements that have no children elements or text inside them. |

:nth-child(pseudo-class): this is one of the most powerful and useful structural pseudo-classes. it selects an element based on its position within the list of its parent's children; in other words, it selects an element based on how many siblings it has before it.

In using the **:nth-child(pseudo-class):** , keyword such as ‘even’ and ‘odd’ as well as numbers can be used inside the parenthesis (the selector’s argument).

Example

```

<style>
  table tr:nth-child(even) {
    background: #ccc;
  }
</style>

<table>
  <tr>
    <td> Row1, cell1</td>
    <td> Row1, cell2</td>

```

```
<td> Row1, cell3</td>
</tr>
<tr >
<td> Row2, cell1</td>
<td> Row2, cell2</td>

<td> Row2, cell3</td>
</tr>
..
.
```

Here is the result

```
Row1, cell1 Row1, cell2 Row1, cell3
Row2, cell1 Row2, cell2 Row2, cell3
Row3, cell1 Row3, cell2 Row3, cell3
Row4, cell1 Row4, cell2 Row4, cell3
Row5, cell1 Row5, cell2 Row5, cell3
Row6, cell1 Row6, cell2 Row6, cell3
Row7, cell1 Row7 cell2 Row7, cell3
```

:first-child (pseudo-class): The :first-child CSS pseudo-class **represents the first element among a group of sibling elements.** /* Selects any <p> that is the first element among its siblings */ p:first-child { color: lime; } Note: As originally defined, the selected element had to have a parent.

Example

Code

```
ul li:first-child{
    color:#00C;
    font-weight:bold;
    font-size:1em;
    font-style:italic;
    text-decoration:underline;
    background-color:#FFFFCC;
}
```

HTML Code

```
<ul>
<li>Home</li>
<li>Services</li>
<li>Blog</li>
<li>Portfolio</li>
<li>About Us</li>
</ul>
```

Result

- [Home](#)
- Services
- Blog
- Portfolio
- About Us

:last-child (pseudo-class): The :last-child selector allows you to target the last element directly inside its containing element. :last-child matches the very last child of a parent element.

Example

CSS code

```
p:last-child{  
    font-size:0.75em;  
    background-color:#CC0000;  
    color:#FFFFFF;  
}
```

HTML Code

```
<div>  
    <p>Lorem ipsum...</p>  
    <p>Dolor sit amet...</p>  
    <p>Consectetur adipisicing...</p>  
    <p>Last paragraph...</p>  
</div>
```

Result

 Lorem ipsum...

 Dolor sit amet...

 Consectetur adipisicing...

 Last paragraph...

6.5 Other Selectors

The following are other CSS selectors; direct child selectors, next sibling selectors, attribute selectors and pseudo-elements.

➔ **Direct Child Selectors:** Direct child selectors operate much like descendant selectors, which rely on an ancestral relationship to decide where to apply style. Direct child selectors apply only to immediate children of the element. This is achieved by introducing a new syntax for the selector:

```
body > .intro {  
    font-weight: bold;  
}
```

Since child selector selects all the elements that are children of a specified element. The example above, **.intro** class is the child of **body** element.

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>  
<title>Advance Selectors</title>  
<style type="text/css">  
.intro {  
    font-style: italic;  
}  
body > .intro {  
    font-weight: bold;  
}  
</style>  
</head>  
<body>  
<h1>Recipes for EDIKANG Ikong Soup</h1>  
<p class="intro">EDIKANG Ikong Soup is a remarkably versatile food, prepare by the  
calabar people in Southern Nigeria, available hundreds of tribes in Nigeria or  
meal.</p>  
<div class="content">  
<h2>Submit a recipe</h2>  
<p class="intro">We would love to hear from you about your delicious recipes  
for EDIKANG Ikong Soup. Please complete our form (all fields required) or email us at  
<a href="mailto:recipes@example.com">recipes@example.com</a>.</p>  
<form method="post" action="">  
<div>  
<label for="submit-name">Name</label>  
<input type="text" name="name" id="submit-name">  
</div>  
<div>  
<label for="submit-email">Email</label>  
<input type="text" name="email" id="submit-email">  
</div>  
<div>  
<label for="submit-recipe-name">Recipe Name</label>  
<input type="text" name="recipe-name" id="submit-recipe-name">  
</div>  
<div>  
<label for="submit-ingredients">Ingredients</label>  
<textarea name="ingredients" id="submit-ingredients" rows="5" cols="50">
```

```
</textarea>
</div>
<div>
<label for="submit-recipe">Recipe</label>
<textarea name="recipe" id="submit-recipe" rows="5" cols="50">
</textarea>
</div>
<div>
<input type="submit" value="Send recipe">
</div>
</form>
</div>
</body>
</html>
```

Output



Fig. 6.5

In the above example, each element with the class name `intro` is made italic with "`font-style: italic;`". A descendant selector `body > .intro` to make only the elements with the class name of `intro`, which are also a direct child of the `body` element bold with "`font-weight: bold;`". Both paragraphs are italic in the output above, only the first is bold, because the second paragraph is a child of the element with the class name `content` and not the `body`.

6.5.1 Adjacent Sibling Selector (+)

The adjacent sibling selector selects an element that is directly after another specific element. Sibling elements must have the same parent element, and "adjacent" means "immediately following" that is the first element immediately after specified element.

Example

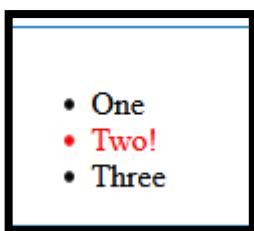
CSS code

```
li:first-of-type + li {  
    color: red;  
}
```

HTML Code

```
<ul>  
    <li>One</li>  
    <li>Two!</li>  
    <li>Three</li>  
</ul>
```

Result



In the above example, the second list item is styled since it immediately follows the first li item.

6.5.2 General Sibling Selector (~)

The CSS general sibling selector selects all elements that follow the first element such that both are children of the same parent. That is next siblings of a specified element.

Example

CSS Code

```
p ~ span {  
    color: blue;  
}
```

HTML Code

```
<span>This is not blue.</span>  
<p>This is a paragraph.</p>  
<code>Here is some code.</code>  
<span>And the blue span is here!</span>  
<span>And this is a blue span!</span>  
<code>More code...</code>  
<div> How are you? </div>  
<p> Whatever it may be, keep smiling. </p>  
<h1> Dream big </h1>  
<h2> that's all. </h2>  
<span>And yet again this is a red span!</span>
```

Result

This is not blue.

This is a paragraph.

Here is some code. And the blue span is here! And this is a blue span! More code...
How are you?

Whatever it may be, keep smiling.

Dream big

that's all.

And yet again this is a red span!

6.5.3 Attribute selector

Attribute selectors allow you to target specific elements without needing IDs or classes in the HTML. Instead, attribute selectors target an element based on the existence or value of a specific attribute on that element.

The syntax for attribute selectors is as follows;

```
input[name] {  
    border: 2px dashed #000;  
}
```

By adding the following CSS code

```
.....  
input {  
    background:#CC0;  
}  
input[name]{  
    border: 2px dashed #996600;  
}  
.....
```

To the style section of the page, it produces the following result;



Fig. 6.6

In the above, a simple type selector is used to give all input elements a yellow background. Then using a combined type and attribute selector, `input[name]`, a dashed border is only added to those inputs that have a name attribute -the final input, `<input type="submit" value="Send recipe">`, does not have a name attribute and therefore does not have a dotted border around it. This type of attribute selector is known as **Select by Presence of an Attribute**.

Attribute selector can be applied base on **select by value** as it is shown in the syntax below;

```
input[name="email"] {  
    border: 2px dashed #996600;  
}
```

Here, both an attribute (name) and a value for that attribute (email), to select on only input elements with name attribute values of email will only match this selector.

Submit a Recipe

We would love to hear from you about your
our form (all fields required) or email us at [.](#)

Name

Email

Recipe Name

Ingredients

Fig. 6.7

Now using a combined type, attribute, and value selector, input [name="email"], a dashed border is only applied to those input elements that have a name attribute with value equal to email as it is shown above.

Attribute Selectors and their functions

| ATTRIBUTE SELECTOR | FUNCTION |
|--------------------|--|
| [attr] | Matches an element with an attr attribute present, regardless of its value. |
| [attr=val] | Matches an element with an attr attribute whose value is exactly val. |
| [attr~=val] | Matches an element with an attr attribute whose value is a space-separated list of words, one of which is exactly val. |
| [attr =val] | Matches an element with an attr attribute whose value is either exactly val or begins with val immediately followed by a hyphen. |
| [attr^=val] | Matches an element with an attr attribute whose value starts with val. |
| [attr\$=val] | Matches an element with an attr attribute whose value ends with val. |
| [attr*=val] | Matches an element with an attr attribute whose value contains val somewhere within it. |

There is also **attribute substring selectors** that choose elements based on whether a particular string appears at the beginning of an attribute's value, at the end of an attribute's value, or anywhere inside an attribute's value. This is also known as regular expression in CSS3.

Examples

```
a[href^="mailto:"] {  
    padding-left: 23px;  
    background: transparent url(envelop-icon.png) no-repeat center left;  
}
```

A combined type, attribute, and a value selector with the ^ character indicating that you want to match the start of the value with your string, "mailto:", that is, you add an envelope icon as a background image.

```
input[id$="name"] {  
    border: 2px dashed #000;  
}
```

This substring attribute selector chooses elements with attributes whose value ends with a string. The selector of the preceding rule uses the dollar sign to signify that the selector matches the end of the attribute value. This changes all input elements with an id attribute value that ends in the string name as it is shown in the code snippet below;

```
.....  
<form method="post" action="">  
<div>  
    <label for="submit-name">Name</label>  
    <input type="text" name="name" id="submit-name">  
</div>  
<div>  
    <label for="submit-email">Email</label>  
    <input type="text" name="email" id="submit-email">  
</div>  
<div>  
    <label for="submit-recipe-name">Recipe Name</label>  
    <input type="text" name="recipe-name" id="submit-recipe-name">  
</div>  
<div>  
    <label for="submit-ingredients">Ingredients</label>  
    <textarea name="ingredients" id="submit-ingredients" rows="5" cols="50">  
</textarea>  
</div>  
<div>  
    .....
```

Looking at the code above, the style will affect all the id values that end with "name" such as *id="submit-recipe-name"* and *id="submit-name"*.

```
[name*="recipe"] {  
    border: 3px dashed #CC5;  
}
```

A wildcard attribute substring selector selects an element that contains an attribute whose value contains a string(*recipe*) anywhere in the value: at the beginning, the end, or anywhere in the middle. This attribute substring selector uses an asterisk in the syntax to indicate what the selector is looking for anywhere inside the value.

More examples

Using attribute selector to indicate file type such as pdf, docx, doc, mov, and dynamically add icon to it; First, prepare the a elements inside the file-download lists to have background images added to them:

```
ul a { display: block; min-height: 15px;  
padding-left: 20px;  
background-repeat: no-repeat;  
background-position: 0 3px;  
}  
a[href$=".pdf"] { background-image: url(images/icon_pdf.png);  
}  
a[href$=".doc"] { background-image: url(images/icon_doc.png);  
}  
a[href$=".mov"] {  
background-image: url(images/icon_film.png);  
}  
a[href$=".jpg"] { background-image: url(images/icon_photo.png);  
}
```

The href\$= part of each attribute selector tells the browser “find every href attribute that ends with,” and then the values in quotation marks, such as .pdf, give the ending attribute value to match against. When the browser finds a match, it applies the background image indicated, adding appropriate icons to all the links.

DOWNLOAD:  [Itinerary \(PDF\)](#)  [Itinerary \(Word\)](#)  [Map of trip locations \(PDF\)](#)

Alternative Icon Ideas

Saying it, instead of showing it.

Aside icons being more obvious and explicit, you could use generated content to write out the file-type extension at the end of each link instead of or in addition to the icons. You’d first need to make sure that this information wasn’t already manually written in each link. Then, you could add the following rule, for example, to write out “(PDF)” and “DOC” after each link to a PDF and DOC files:

```
a[href$=".pdf"]:after {  
    content: " (PDF)";  
}  
a[href$=".docx"]:after {  
    content: " (Word)";  
}
```

COMBINING MULTIPLE ATTRIBUTE SELECTOR

What if you want to show the photo icon for links to PNG images, but a graph icon for links to PNG images also happened to be a graph? Combining selectors as shown below would work:

```
a[href$=".png"]/[href*="graph"] {  
    background-image: url(images/icon_graph.png);  
}
```

This selector tells the browser “find all links that have ‘.png’ at the end of their href attributes and have ‘graph somewhere in the href attribute.’” So all of the following links would get matched:

```
<a href="images/graph_locations.png">  
<a href="images/bargraph.png">  
<a href="graph/travel.png">
```

The following are other uses of attribute selectors:

- Styling different form-field input types uniquely (using `input[type=submit]`)
- Removing bullets from lists within navigation divs (using `div[id^=nav] to match <div id="nav-primary"> and <div id="nav-secondary">`, for example)
- Styling links that go to external sites (using `a[href^=http] or a[rel=external]`), that are secure (using `a[href^=https]`), that go to a specific URL (such as `a[href*="paypal.com"]`), that open in new window (using `a[target=_blank]`), or that go to your own home page (using `a[href="http://myurl.com"] or a[href="/index.html"]`)
- Displaying the access key of a link (using `a:after { content: '[' attr(accesskey) ']' }`)
- Displaying the citation source of a blockquote (using `blockquote[cite]:after { content: ' - ' attr(cite)' }`)

6.5.4 CSS Web Developer Tools

The following tools will make your life easy when working with CSS on the web page;

- ⇒ **Google Chrome Inspect:** this makes quick CSS changes stylesheet by showing the CSS relevant to the page you are looking at. Google Chrome Inspect element helps to

diagnose the issues with the construction of your web pages. To activate this, combine Ctrl + Shift + I keys on the keyboard.

- ⇒ **Chris Pederick web developer tool:** The Chris Pederick web developer tool is an extension that adds to your browser. This is similar to Google Chrome where you can make quick CSS changes; the extension is available for Chrome and Firefox at <http://chrispederick.com/work/web-developer/>
- ⇒ **ColorZilla ColorZilla:** This is an Eyedropper that runs within Mozilla Firefox; it allows you to sample colors in your web page. ColorZilla is an extension available for Chrome and Firefox at <http://www.colorzilla.com>

PINFO ICT ACADEMY

Chapter Seven Font and CSS

Font and CSS

CSS includes variety of properties that change the font typeface (font-family), font-size, and font-style of a website. Font names are case insensitive (That is, they can be uppercase, lowercase, sentence case, or whatever mixture of cases), and strings which contain spaces must be enclosed with quotation marks. For example, “**Times New Roman**”, and a comma is always used to separate each font name.

Example:

```
<style type="text/css">
body {
    font-family: arial, helvetica, sans-serif;
}
h1 {
    font-family: "Times New Roman", Georgia, Serif;
}
</style>
```

7.1 Font-family property

Font-family specifies the font typeface used to display text on a screen. When specifying typeface, you should generally stick to the set of web safe fonts. The following table outlines the generic font family names defined in CSS.

Table 4

| GENERIC FONT | RESEMBLES |
|--------------|------------------------|
| serif | Times, Times New Roman |
| sans-serif | Helvetica, Arial |
| cursive | Zapf-Chancery |
| fantasy | Western |
| monospace | Courier, Courier New |

Examples

CSS Code

```
body {
    font-family: arial, helvetica, sans-serif;
}
h1 {
    font-family: "Times New Roman", Georgia, Serif;
}
ol {
    font-family: monospace;
```

}

HTML Code

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Font-Family</title>
</head>
<body>
<h1>Recipes for BANGA Soup</h1>
<p>BANGA Soup is a remarkably delicious meal prepared in different variety by tribes and people of Nigeria.</p>
<h2>Ingredients:</h2>
<p>500g assorted meats, 250g dried and smoked fish, 255g bushmeat, 1 palm soup base, 225g ground dried prawns, 600ml beef stock or water, 225 okra, sliced into thin rings 60g tomato puree, 2 hot chillies (eg. Scotch bonnet), pounded to a paste 100g igbo (ground-pepper) or use watercress, 1 onion, sliced 1/2 tsp atakieko (alligator pepper), 1/2 tsp ashanti pepper, ground salt to taste</p>
<h2>METHOD</h2>
<ol>
<li>Place the washed meat in a large pot, add a drop of water or stock season with salt and ground pepper and boil for 30 minutes or until tender</li>
<li>Add the smoked fish and stockfish, cook for another 10 minutes.</li>
<li>Prepare the oil-palm nut to extract the oil by boiling the washed nuts for 20 minutes until soft.</li>
<li>Remove from water and pound to remove the oil</li>
<li>Pass through a sieve to separate the kernels from the chaff.</li>
<li>Pour the strained pulp into the meat together with the sliced peppers onions' tomatoes puree and Okra</li>
<li>Sprinkle in the crayfish and cook for 15 minutes until the soup is fairly reduced and thickened to coat the back of spoon</li>
<li>Check seasoning and serve with pounded yam (Iyan) or Usin/Egun Obobo (starch or plantain pudding)</li>
</ol>
</body>
</html>
```



Fig. 7.1

This sets default fonts for all text on the page.

body {

```
    font-family: arial, helvetica, sans-serif;  
}
```

“Times New Roman”, Georgia, Serif are specified for all h1 elements.

```
h1 {  
    font-family: “Times New Roman”, Georgia, Serif;  
}
```

Finally, specify the generic system monospace font for all ol (ordered list) elements.

```
ol {  
    font-family: monospace;  
}
```

Font-style

Font-style property is used to switch between styles provided by a particular font. The values for possible font-style property are italic, normal, inherit, and oblique.

Font-weight

Font-weight property provides the functionality on how bold a font should be. The values for font-weight property are normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, and 900. However, in real-world web design, the only two values that matter are the normal and bold values. These are the values you will be using in your web design. Occasionally, you may use bolder value.

Example

CSS Code

```
body {  
    font-family: arial, helvetica, sans-serif;  
}  
h1{  
    font-family: “Times New Roman”, Georgia, Serif;  
    font-weight: bolder;  
}  
.intro {  
    font-weight: bold;  
}  
.recipe{  
    font-weight: normal;  
    font-style:italic;  
}
```

HTML Code

```
<h1>Heading One Intro</h1>  
<p class='intro'>  
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.
```

```
</p>
<p class="recipe">
Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus.
Pellentesque habitant morbi tristique senectus et netus et malesuada
fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci.
Aenean nec lorem. In porttitor. Donec laoreet nonummy augue.
Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis,
nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy.
</p>
```

Result

Heading One Intro

Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Maecenas porttitor congue massa. Fusce posuere, magna sed
pulvinar ultricies, purus lectus malesuada libero, sit amet
commodo magna eros quis urna.

Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci. Aenean nec lorem. In porttitor. Donec laoreet nonummy augue. Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy.

CSS3 Web fonts

The @font-face rule allows custom fonts to be loaded on a webpage. Once added to a stylesheet, the rule instructs the browser to download the font from where it is hosted, then display it as specified in the CSS. The font can be loaded from either a remote server or a locally-installed font on the user's own computer. Your "own" fonts are defined within the CSS3 @font-face rule. In using the CSS3 fonts it is a good idea to specify different font format. Here are the list of different font format;

Browser support for font formats

| Font format | | | | | |
|-------------|---------------|---------------|---------------|---------------|---------------|
| TTF/OTF | 9.0* | 4.0 | 3.5 | 3.1 | 10.0 |
| WOFF | 9.0 | 5.0 | 3.6 | 5.1 | 11.1 |
| WOFF2 | 14.0 | 36.0 | 39.0 | 10.0 | 26.0 |
| SVG | Not supported | Not supported | Not supported | 3.2 | Not supported |
| EOT | 6.0 | Not supported | Not supported | Not supported | Not supported |

When using @font-face rule, it should be added to the stylesheet before any styles.

The following are links to open source font

- www.fontsquirrel.com
- www.fontex.org
- www.openfontlibrary.org

The following are links to free commercial fonts

- www.typekit.com
- www.kernest.com
- www.fontspring.com

Example

```
@font-face{  
    font-family: 'master_of_break';  
    src:url('master_of_break.ttf') format('truetype'),  
    url('master_of_break.eot') format('eot'),  
    url('master_of_break.woff') format('woff'),  
    url('master_of_break.svg#webfontFHzvtks0') format('svg');  
    font-weight: bold;  
}  
  
p{  
    font-family:'master_of_break';  
}
```

*Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Maecenas porttitor congue
massa. Fusce posuere, magna sed
pulvinar ultricies, purus lectus malesuada
libero, sit amet commodo magna eros quis
urna.*

Then use it to style elements like this:

Alternative Techniques

Google Font

Google fonts Google provides open source fonts which work thorough linking your webpage to the font files on their servers or using @import as shown below;

Linking a stylesheet to google server

```
<link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
```

```
h1{  
    font-family: 'Pacifico', cursive;  
    font-weight: bold;  
}
```

```
<link rel="preconnect" href="https://fonts.googleapis.com">  
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>  
<link href="https://fonts.googleapis.com/css2?family=Open+Sans&display=swap" rel="stylesheet">
```

Then, we can style our elements like the other methods:

```
body {  
    font-family: 'Open Sans', sans-serif;  
}
```

The following is an example of using @import to load the Open Sans font from Google Fonts:

```
@import url(//fonts.googleapis.com/css?family=Open+Sans);
```

Then we can use it to style elements:

```
body {  
    font-family: 'Open Sans', sans-serif;  
}
```

Font-size Property

The "font-size" property specifies the size of a font. It can be specified as a fixed size in various units, a percentage, or as a predefined keyword as follows:

```
font-size: 1.5em;  
font-size: 12px;  
font-size: 100%;  
font-size: larger;
```

Example;

```
<style type="text/css">  
body {  
    font-family: arial, helvetica, sans-serif;  
    font-size: 12px;  
}  
h1 {
```

```
font-family: "Times New Roman", Georgia, Serif;  
font-size: 30px;  
}  
</style>
```

7.1.2 Using Short Hand Properties

This set of properties let you use a single property declaration to assign values to a number of related properties.

Example:

```
h1{  
font-weight: bold;  
font-size: 90%;  
line-height:1.8em;  
font-family: Helvetica, Arial, Sans-serif;  
}
```

This is normal for most beginners

The shorthand property of the above code

```
h1{ font: bold 90% 1.8em Helvetica, Arial, Sans-serif; }
```

line-height, letter-spacing and word-spacing Properties

The line-height property is the height of the line on which each line of text appears. It commonly set the distance between lines of text. It adds space above and under the text. You can use line-height with different values as follows:

```
line-height: normal;  
line-height: 2;  
line-height: 1em;  
line-height: 1rem;  
line-height: 200%;  
line-height: 20px;
```

The values of line-height are number, percentage and length as shown above. When a value is applied without a unit, it takes the font size and multiplies it with the font-height value. It then splits the result in two with half added to the top and half to the bottom.

Example:

```
body {  
font-size: 16px;  
line-height: 1.5;  
}
```

When we do the following calculation: $16*1.5=24\text{px}$; this implies that, our text will have a minimum height of 24px. So it will add 4 pixels under the text and above it.

Letter-spacing property controls the amount of space between letters and word-spacing property controls the space between words. Positive values of letter-spacing causes characters to spread farther apart, while negative values of letter-spacing bring characters closer together.

Example

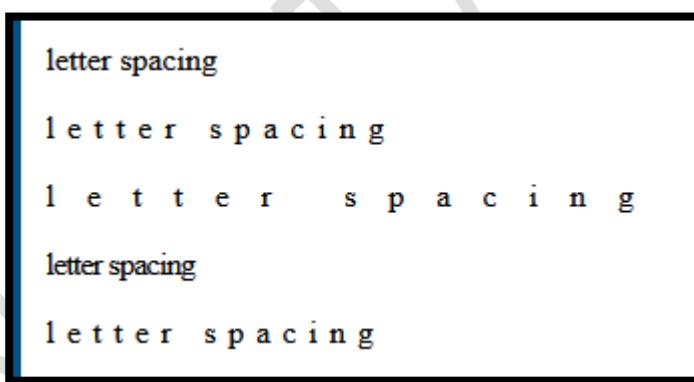
CSS Code

```
.normal { letter-spacing: normal; }
.em-wide { letter-spacing: 0.4em; }
.em-wider { letter-spacing: 1em; }
.em-tight { letter-spacing: -0.05em; }
.px-wide { letter-spacing: 6px; }
```

HTML Code

```
<p class="normal">letter spacing</p>
<p class="em-wide">letter spacing</p>
<p class="em-wider">letter spacing</p>
<p class="em-tight">letter spacing</p>
<p class="px-wide">letter spacing</p>
```

Result



Word-spacing property: This is used to provide space between words, larger gap between words increases readability. The value for the word space property can be in em, px, %. **word-space: 5px;**

Example

```
<style type="text/css">
    .important
    {
        font-style: italic;
    }
```

```
        line-height: 3;  
        letter-space: 0.5em;  
        word-space:5px;  
    }  
</style>
```

7.2 Changing text case

In changing text case CSS uses `text-transform` to deal with this.

The table below shows `text-transform` values and their meaning:

Table 5

| Value | Meaning |
|------------|--|
| Capitalize | Displays the first letter of each word as caps and all others as lowercase |
| Uppercase | Displays all letters as uppercase |
| Lowercase | Displays all letters as lowercase |
| None | Displays characters as specified |

To have a paragraph of text transformed into all uppercase, for example, use `text-transform: uppercase`; and to instantly ensure that all header level ones are in proper case, use:

```
h1 {text-transform: capitalize ;}
```

7.3 Text Indentation Properties

Indenting text in CSS is done using `text-indent` property, which accepts the values length and percentage.

Example;

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">  
<title>Text Indentation</title>  
<style type="text/css">  
    .intro{  
        text-indent:0.9em;  
    }  
</style>  
</head>  
<body>  
    <p class="intro"> Getting the layout of the page section is just a starting point of our CSS  
    class, if you are able to master this aspect you will not have problem in manipulating CSS.  
    Though some of the features that have been used in this section you might not really understand  
    them. But as the class go on you will get use to everything.</p>
```

```
</body>
</html>
```

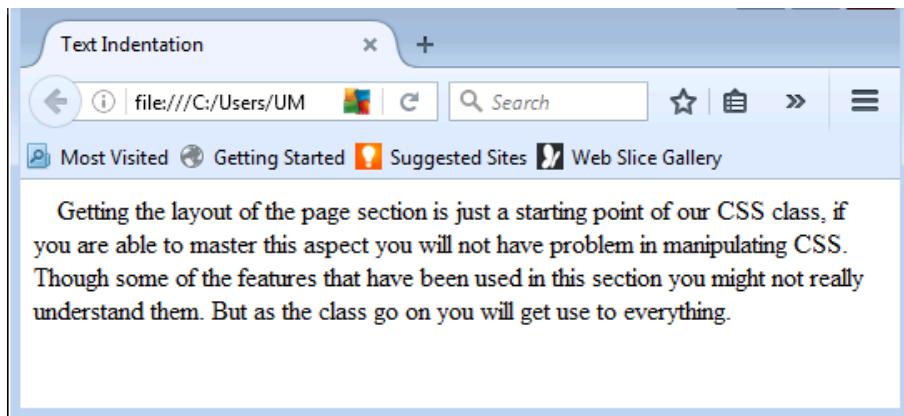


Fig. 7.2

7.4 Text Alignment Properties

Text-align property has keywords that align text to either left, right, center or justify.

Example:

```
h1{
    text-align:center;
}
h2{
    text-align: right;
}
h3{
    text-align: left;
}
p{
    text-align: justify;
}
```

7.5 Text Decorations (Underline, Overline and Strike-through)

text-decoration property add s

- ⇒ Underline
- ⇒ Overline
- ⇒ Strike-through

to text.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Text Indentation</title>
<style type="text/css">
.intro{
    text-indent:0.9em;
}
```

```
        }
h1{
    text-align:center;
    text-decoration: underline;
}
h2{
    text-align: justify;
    text-decoration: overline;
}
p{
    text-decoration: line-through;
}
</style>
</head>
<body>
<h1>Page Layout With CSS</h1>
<p class="intro"> Getting the layout of the page section is just a starting point of our CSS class, if you are able to master this aspect you will not have problem in manipulating CSS. Though some of the features that have been used in this section you might not really understand them. But as the class go on you will get use to everything.</p>
<h2> Importance of Using CSS</h2>
</body>
</html>
```

Result



Fig. 7.3

Chapter Eight

Working with Images in CSS

Images are placed on a web page using the HTML tag, but the image tag should be enclosed inside div element and position arbitrary on the page. With this, we can affect the attributes of the image such as the border, the alignment and its appearance. You can make text around the image as well.

8.1 Using image tag in CSS

You can use html img tag to reference an image in a document page as it is shown below;

Making texts round an image

```
img{  
    float: left;  
}
```



Fig. 8.1

```
img{  
    float: right;  
}
```



Fig. 8.2

Examples

This is another way of inserting image in a page using CSS classes.

```
<html>
```

```
<head>
<title></title>
<style type="text/css">
.image-replacement {
    display: block;
    -moz-box-sizing: border-box;
    box-sizing: border-box;

background:url(http://akamaicovers.oreilly.com/images/9780596517748/cat.gif) no-repeat;
    width: 180px;
    height: 236px;
    padding-left: 180px;
}

.text-replacement {
    display: block;
    -moz-box-sizing: border-box;
    box-sizing: border-box;
    height: 250px;
    width: 250px;
    padding-top: 250px;
    overflow: hidden;
    background: #333 url(http://cdn.css-tricks.com/wp-content/themes/CSS-Tricks-10/images/footer-logo-shoptalk.png) no-repeat;
}
</style>
</head>
<body>
<h2>Image replaced with Image</h2>

<h2>Text Replaced with Image</h2>
<div class="text-replacement">Some text to replace here</div>

</body>
</html>
```



Fig. 8.3

8.1.1 Working with background images in CSS

The background-image property enables you to provide an image for the background of a web document, which is, placing an image behind HTML element. This could be entire page or part of the page. When you specify a background image by default the background-image is placed at the top corner and repeat both vertically and horizontally in tiling boxes across the page.

CSS Syntax

background-image: url/none/initial/inherit;

Property Values

| Value | Description |
|-------------------|---|
| url('URL') | The URL to the image. To specify more than one image, separate the URLs with a comma |
| None | No background image will be displayed. This is default |
| linear-gradient() | <p>Sets a linear gradient as the background image. Define at least two colors (top to bottom). Eg.</p> <pre>#grad1 { height: 200px; background-image: linear-gradient(red, yellow, green); }</pre> <p>A linear gradient that starts from the left. It starts green, transitioning to blue:</p> <pre>#grad2 { height: 200px; background-image: linear-gradient(to right, green , blue);</pre> <pre>#grad3 { height: 100px; background-image: linear-gradient(180deg, pink, blue);</pre> <pre>#grad4 { height: 55px; background-image: linear-gradient(to right, red, orange, yellow, green, blue, indigo, violet);</pre> |
| radial-gradient() | <p>Sets a radial gradient as the background image. Define at least two colors (center to edges)</p> <p>Example</p> <pre>#grad-rad1 { height: 150px; width: 200px;</pre> |

```
background-image: radial-gradient(red, green, blue);  
}  
#grad-rad2{  
height: 150px;  
width: 200px;  
background-image: radial-gradient(red, yellow, green);  
}  
  
#grad-rad2 {  
height: 150px;  
width: 200px;  
background-image: radial-gradient(circle, red, yellow, green);  
}  
##grad-rad3 {  
height: 150px;  
width: 150px;  
background-image: radial-gradient(closest-side at 60%  
55%,blue,green,yellow,black);  
}  
  
#grad-rad4 {  
height: 150px;  
width: 150px;  
background-image: radial-gradient(farthest-side at 60%  
55%,blue,green,yellow,black);  
}  
  
#grad-rad5 {  
height: 150px;  
width: 150px;  
background-image: radial-gradient(closest-corner at 60%  
55%,blue,green,yellow,black);  
}  
  
#grad-rad6 {  
height: 150px;  
width: 150px;  
background-image: radial-gradient(farthest-corner at 60%  
55%,blue,green,yellow,black);  
}
```

repeating-linear-gradient()

Repeats a linear gradient

Example

```
#grad1 {  
height: 200px;  
background-image: repeating-linear-gradient(red, yellow 10%, green
```

| | |
|------------------------------------|---|
| | <code>20%); }</code> |
| <u>repeating-radial-gradient()</u> | Repeats a radial gradient Example <code>#grad1 { height: 200px; background-image: repeating-radial-gradient(violet, yellow 5%, red 10%); }</code> |
| Initial | Sets this property to its default value. |
| Inherit | Inherits this property from its parent element. |

Example:

```
body{  
background-color: white;  
color: black;  
background-image:url(name.extension); note: the extension could be .jpg,  
.png and .gif  
}
```

Tips: Always specify a background color with a background image if the image is not available.

The above code will make the graphic to be tilling or repeat itself to fill up the entire browser view point. However, CSS can be used to change both the tilling and scrolling characteristics of images.

html

```
<body>

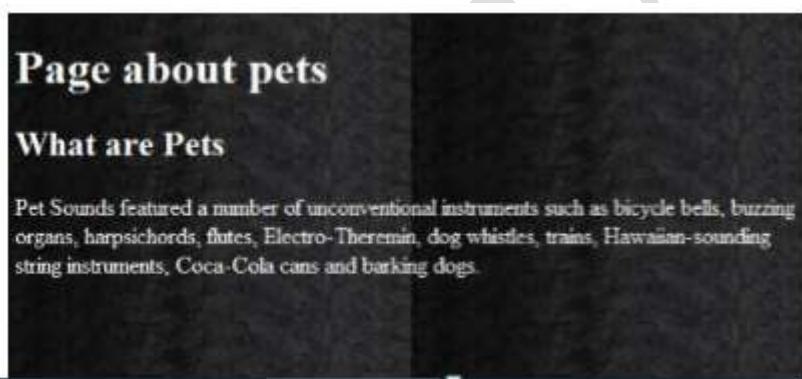
    <h1>Page about pets</h1>

    <h2>What are Pets</h2>
    <p>Pet Sounds featured a number of unconventional
    instruments such as bicycle bells, buzzing
    organs, harpsichords, flutes, Electro-Theremin,
    dog whistles, trains, Hawaiian-sounding string
    instruments, Coca-Cola cans and barking dogs.</p>
</body>
</html>
```

css

```
body{
    background-image:url("texture.png");
    color:white;
}
```

Result



The background-repeat Property

The background-repeat stops the background image from tiling. The background-repeat property has four values:

Repeat: The background image repeats both horizontally and vertically (the default way shown if the background-repeat property is not used).

Repeat-x: The image repeats horizontally

Repeat-y: The image repeats vertically only

No-repeat: The image only shown once.

The background position property

When an image is not being repeated, you can use the background-position property to specify where in the browser window the background image should be placed. This property usually has a pair of values. The first represents the horizontal position and the second represents the vertical. left top, left center, left bottom, center top, center center, center bottom, right top, right center, right bottom, 50% 50% 20px 30px

Examples

```
/* Keyword values */
background-position: top;
background-position: bottom;
background-position: left;
background-position: right;
background-position: center;

/* <percentage> values */
background-position: 25% 75%;

/* <length> values */
background-position: 0 0;
background-position: 1cm 2cm;
background-position: 10ch 8em;

/* Multiple images */
background-position: 0 0, center;

/* Edge offsets values */
background-position: bottom 10px right 20px;
background-position: right 3em bottom 10px;
background-position: bottom 10px right;
background-position: top right 10px;

/* Global values */
background-position: inherit;
background-position: initial;
background-position: revert;
background-position: unset;
```

A position defines an x/y coordinate, to place an item relative to the edges of an element's box. It can be defined using one to four values. If two non-keyword values are used as in the example above, the first value represents the horizontal position and the second represents the vertical position. If only one value is specified, the second value is assumed to be center. If three or four values are used, the length-percentage values are offsets for the preceding keyword value(s).

The background-attachment property

Background attachment is used to specify that background image is fixed or scroll the rest of the page. It has two values:

- ⇒ **fixed** The background image stays in the same position on the page.
- ⇒ **scroll** The background image moves up and down as the user scrolls up and down the page.

Example:

```
body{  
    background-color: #30293f;  
    color: white;  
    background-image:url(name.jpg);  
    background-repeat: no-repeat;  
    background-attachment: fixed;  
    background-position: bottom right;  
}
```

The Background Shorthand CSS Property: The background shorthand CSS property sets all background style properties at once, such as color, image, origin and size, or repeat method.

Background shorthand property can be used for the following CSS properties:- background-attachment, background-clip, background-color, background-image, background-origin, background-position, background-repeat, background-size

Example

```
/* Using a <background-color> */  
background: green;  
  
/* Using a <bg-image> and <repeat-style> */  
background: url("test.jpg") repeat-y;  
  
/* Using a <box> and <background-color> */  
background: border-box red;  
  
/* A single image, centered and scaled */  
background: no-repeat center/80% url('../img/image.png');
```

CSS Code

```
body {  
    background: #fffff url("img_tree.png") no-repeat right top;  
    margin-right: 200px;  
}
```

HTML code

```
<!DOCTYPE html>
```

```
<html>
<head>
<style>
body {
    background: #ffffff url('img_tree.png') no-repeat right top;
    margin-right: 200px;
}
</style>
</head>
<body>
<h1>The background Property</h1>
<p>The background property is a shorthand property for specifying all the background properties in one declaration.</p>
<p>Here, the background image is only shown once, and it is also positioned in the top-right corner.</p>
<p>We have also added a right margin, so that the text will not write over the background image.</p>
</body>
</html>
```

Result

The background Property

The background property is a shorthand property for specifying all the background properties in one declaration.

Here, the background image is only shown once, and it is also positioned in the top-right corner.

We have also added a right margin, so that the text will not write over the background image.



CSS background-size Property

The background-size property specifies the size of the background images.

There are four different syntaxes you can use with this property: the keyword syntax ("auto", "cover" and "contain"), the one-value syntax; sets the width of the image (height becomes "auto"), the two-value syntax (first value: width of the image, second value: height), and the multiple background syntax (separated with comma).

Property Values

| Value | Description | Examples |
|-------------------|---|---|
| Auto | Default value. The background image is displayed in its original size | <pre>#example1 { border: 2px solid black; padding: 25px; background: url(mountain.jpg); background-repeat: no-repeat; background-size: auto; }</pre> |
| <i>length</i> | Sets the width and height of the background image. The first value sets the width, the second value sets the height. If only one value is given, the second is set to "auto". | <pre>#example1 { border: 2px solid black; padding: 25px; background: url(mountain.jpg); background-repeat: no-repeat; background-size: 300px 100px; }</pre> |
| <i>percentage</i> | Sets the width and height of the background image in percent of the parent element. The first value sets the width, the second value sets the height. If only one value is given, the second is set to "auto" | <pre>#example1 { border: 2px solid black; padding: 25px; background: url(mountain.jpg); background-repeat: no-repeat; background-size: 100% 100%; } #example2 { border: 2px solid black; padding: 25px; background: url(mountain.jpg); background-repeat: no-repeat; background-size: 75% 50%; }</pre> |
| cover | Resize the background image to cover the entire container, even if it has to stretch the image or cut a little bit off one of the edges | <pre>#example1 { border: 2px solid black; padding: 25px; background: url(mountain.jpg); background-repeat: no-repeat; background-size: cover; }</pre> |
| contain | Resize the background image to make sure the image is fully visible | <pre>#example1 { border: 2px solid black; padding: 25px; background: url(mountain.jpg); }</pre> |

| | | |
|---------|---|--|
| | | <code>background-repeat: no-repeat; background-size: contain; }</code> |
| initial | Sets this property to its default value | |
| inherit | Inherits this property from its parent element. | |

Using two background-size properties

Example

```
#example-background-size {  
    border: 2px solid black;  
    padding: 25px;  
    background: url(img_tree.gif), url(mountain.jpg);  
    background-repeat: no-repeat;  
    background-size: contain, cover;  
}
```

In the above example, we have two background images. We specify the size of the first background image with "contain", and the second background-image with "cover".

Full Code

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
#example1 {  
    border: 2px solid black;  
    padding: 25px;  
    background: url(img_tree.gif), url(mountain.jpg);  
    background-repeat: no-repeat;  
    background-size: contain, cover;  
}  
</style>  
</head>  
<body>  
  
<h2>background-size: contain, cover:</h2>  
<div id="example1">  
    <h2>Hello World</h2>  
    <p>Here we have two background images. We specify the size of the first background image with "contain", and the second background-image with "cover".</p>  
</div>  
  
</body>  
</html>
```

Result

`background-size: contain, cover;`



8.2 How to create thumbnails for Photo gallery

Now let us create a photo gallery using CSS and HTML.
The gallery is created using the following markups;

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>Chapter 8: Gallery</title>
<link rel="stylesheet" href="gallery.css" />
</head>
<body>
<div class="wrapper">
<ul class="gallery">
<li>
<span>Widget the cat</span></li>
<li>
<span>This is my tongue</span></li>
<li>
<span>Widget the cat is looking very serious today</span></li>
<li>
<span>Widget and his very favorite person</span></li>
<li>
<span>Widget explores</span></li>
<li>
<span>Widget is sleeping</span></li>
</ul>
</div>
</body>
</html>
```

Here is CSS code for the gallery page- *gallery.css*

```
body {
background-color: #fff;
color: #000;
margin: 0;
```

```
padding: 0;
font: 0.75em/1.5 "Lucida Grande", "Lucida Sans Unicode",
"Lucida Sans", Verdana, Tahoma, sans-serif;
}
.wrapper {
width: 80%;
margin: 20px auto 40px auto;
}
.ul.gallery {
margin: 0;
padding: 0;
list-style: none;
}
.ul.gallery li {
display: inline-block;
width: 240px;
margin: 0 20px 20px 0;
border: 1px solid #000;
position: relative;
-webkit-box-shadow: 1px 2px 5px 2px rgba(0, 0, 0, 0.3);
-moz-box-shadow: 1px 2px 5px 2px rgba(0, 0, 0, 0.3);
box-shadow: 1px 2px 5px 2px rgba(0, 0, 0, 0.3);
}
.ul.gallery img {
display: block;
}
.ul.gallery span {
position: absolute;
bottom: 0;
left: 0;
background-color: rgba(0,0,0,0.7);
color: rgb(255,255,255);
width: 220px;
padding: 10px;
}
.ul.gallery span {
margin-left: -9999px;
}
.ul.gallery li:hover span {
margin-left: 0;
}
```

Discussion

Next we need to deal with the list items and make them display next to each other. I am doing this by setting their display property to inline-block by using the code below;

```
.ul.gallery li {
display: inline-block;
```

```
width: 240px;  
margin: 0 20px 20px 0;  
border: 1px solid #000;  
}
```



Fig. 8.4

Chapter Nine Controlling Margins, Borders, Padding, Width, and Height in CSS

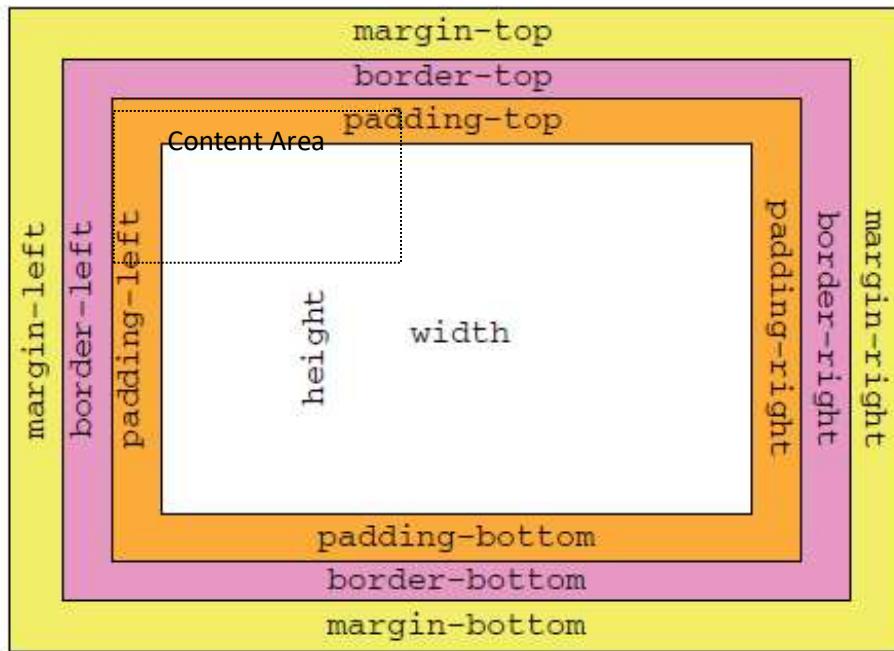


Fig. 9.1

The CSS box model is a collection of properties that define the amount of space around an element, its dimensions, its margins, its borders, and padding between the content of the element and the borders as it is shown in the diagram above. The Padding is applied around the content area. If a background is added to an element, it will be applied to the area formed by the content and padding. As such, padding creates a gutter around content so that it does not appear flush to the side of the background.

Adding a border applies a line to the outside of the padded area. These borderlines come in various styles such as solid, dashed, and dotted. Outside the border is a margin, it clears an area outside the border and it is transparent.

9.1 The Margin Property

The margin property is a shorthand property for the four individual margin properties, margin-top, margin-right, margin-bottom, and margin-left. You can set either of the four, two or one value to margin property. By setting the four values, you specify the top, bottom, right and left values differently; by setting the two values you specify the first value for the top and bottom, and the second value for the left and right; by setting one value you simultaneously set all four sides of an element's margin to the same value.

Examples

```
<style type="text/css">
.introduction {
margin-top: 70px;
margin-left: 100px;
```

By setting four values, you specify the top, bottom, right and left values differently

```
margin-bottom: 50px;  
margin-right: 100px;  
}  
</style>
```

```
<style type="text/css">  
.introduction {  
margin: 50px 100px;  
}  
</style>
```

By setting two values you specify the first value for the top and bottom, and the second value for the left and right;

```
<style type="text/css">  
.introduction {  
margin: 45px;  
}  
</style>
```

By setting one value you simultaneously set all four sides of an element's margin to the same value

Examples

Content without Margin

Content without margin example. The text is contained within a red box.

Content without margin example. The text is contained within a red box.

Content without margin example. The text is contained within a red box.

Content without margin example. The text is contained within a red box.

Content with Margin

Content with margin example. The text is contained within a red box.

Content with margin example. The text is contained within a red box.

Content with margin example. The text is contained within a red box.

9.2 Borders

Borders are used to put lines around boxes. The individual **border-top-width**, **border-right-width**, **border-bottom-width**, and **border-left-width** properties exist for setting the width of the individual sides of a box. Each of these properties can be combined into a single border-width shorthand property.

Examples

```
<!DOCTYPE html>  
<html lang="en">
```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Borders Example</title>
<style type="text/css">
.border-caption{
border-top-width: 1px;
border-right-width: 3px;
border-bottom-width: 5px;
border-left-width: 7px;
border-style: solid;
}
</style>
</head>
<body>
<h1>Example of Border</h1>
<p class="border-caption">This example shows how borders are applied in a
document.</p>
</body>
</html>
```



Fig. 9.2

border-style Property

The border-style property specifies the style of border to be used. The property is similar to the border-width; it uses an identical syntax to specify the style of border to be used for each side of the box.

The following are border-style properties and their allowed values;

Properties: *border-top-style*, *border-right-style*, *border-bottom-style*, and *border-left-style*.

Values: *dotted*, *dashed*, *solid*, *double*, *groove*, *ridge*, *inset*, *outset*, *none*, and *hidden*.

Example

```
.....
<style type="text/css">
body {
border-width: 3px;
border-top-style: ridge;
border-right-style: dashed;
```

```
border-bottom-style: dotted;  
border-left-style: double;  
}  
</style>
```



Fig. 9.3

Border Color Property

Like border-style, margin, and border-width, the border-color property can accept up to four values. This property accepts a <color> value, meaning that it can accept a color keyword(name) like red, blue,pink, a hexadecimal value like #ff0000,#000, an RGB value like rgb(0,0,255), and any color value accepted by the color property is also acceptable to the border-color properties.

Example

```
<style type="text/css">  
body {  
border-width: 3px;  
border-style: dashed;  
border-top-color: red;  
border-right-color: blue;  
border-bottom-color: green;  
border-left-color: purple;  
}  
</style>
```

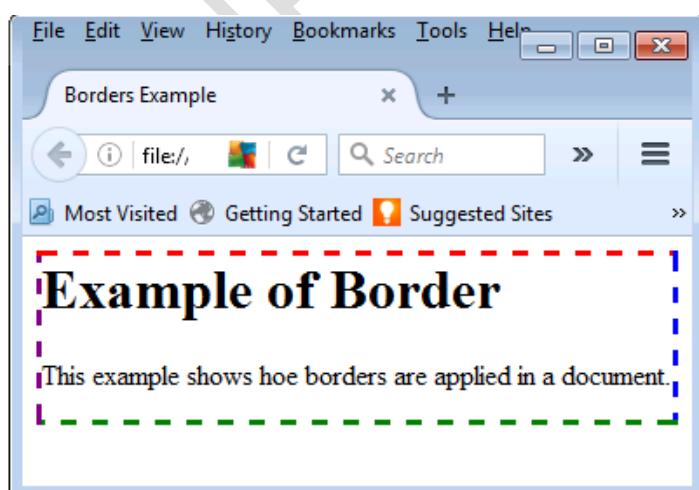


Fig.9.4

Border width property

The border-width property specifies the thickness of the border with the following values:-

- ⇒ <length>: A numeric value measured in px, em, rem, vh and vw units.
- ⇒ thin: The equivalent of 1px
- ⇒ medium: The equivalent of 3px
- ⇒ thick: The equivalent of 5px

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        .one{
            border-style: solid;
            border-width: 5px;
        }
        .two{
            border-style: solid;
            border-width: medium;
        }
        .three{
            border-style: solid;
            border-width: 2px 10px 4px 20px;
        }
    </style>
</head>
<body>
    <h2> Example of border-width properties</h2>
    <p class="one">This is 5px border device-width</p>
    <p class="two">This is medium border device-width</p>
    <p class="three">This is 2px 10px 4px 20px border device-width</p>
</body>
</html>
```

Result

Example of border-width properties

This is 5px border device-width

This is medium border device-width

This is 2px 10px 4px 20px border device-width

Example for all border properties

HTML and CSS Code

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Border Style Properties</title>
<style type="text/css">
body {
    display: inline-block;
    padding: 2em;
}
.box {
    background: #eaeaea;
    display: inline-block;
    height: 200px;
    margin-bottom: 1em;
}
.box-1 {
    border: none;
}
.box-2 {
    border: 5px hidden red;
}
.box-3 {
    border: 5px solid orange;
}
.box-4 {
    border: 5px dashed orange;
}
.box-5 {
    border: 5px dotted orange;
}
```

```
.box-6 {  
    border: 5px double orange;  
}  
.box-7 {  
    border: 5px groove orange;  
}  
.box-8 {  
    border: 5px ridge orange;  
}  
.box-9 {  
    border: 5px inset orange;  
}  
.box-10 {  
    border: 5px outset orange;  
}  
</style>  
</head>  
<body>  
<div class="box box-1">  
    none  
</div>  
<div class="box box-2">  
    hidden  
</div>  
<div class="box box-3">  
    solid  
</div>  
<div class="box box-4">  
    dashed  
</div>  
<div class="box box-5">  
    dotted  
</div>  
<div class="box box-6">  
    double  
</div>  
<div class="box box-7">  
    groove  
</div>  
<div class="box box-8">  
    ridge  
</div>  
<div class="box box-9">  
    inset  
</div>  
<div class="box box-10">  
    outset
```

```
</div>
</body>
</html>
```

Result



9.3 Padding

Padding is the space between the content of an element and its borders. Like margin, border-width, border-style, and border-color, the padding property is a shorthand property, meaning that it is a simplified representation of the other padding properties, padding-top, padding-right, padding-bottom, and padding-left.

Example

```
.....
body{
border-width: 3px;
border-style: dashed;
border-top-color: red;
border-right-color: blue;
border-bottom-color: green;
border-left-color: purple;
padding:13px;
}
.....
```



Fig.9.5

Note: In the above diagram, you will notice that spaces have been added to content of the page. This is inside padding.

Creating Triangles out of Borders

When two borders of a box meet at a corner, the browser draws their meeting point at an angle. If you reduce the box's width and height to zero, and give every border a thick width and a different color, you will end up with the appearance of four triangles pushed together, each pointing in a different direction as it is shown below;

```
.triangles {  
    border-color: red green blue orange;  
    border-style: solid;  
    border-width: 20px;  
    width: 0;  
    height: 0;  
}
```

Output

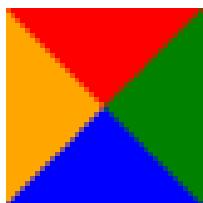


Fig. 9.6

Chapter Ten

Floating content, Positioning and Clear Property in CSS

Float property is used to layout content side-by-side; it changes how texts and images are displayed within an element. The following are possible values of float properties; left, right and none. Float property is one of the ways to use and write a layout of a page. Though flex and grid modules of CSS are currently used now for web page layout. Float is used as a main position property.

Examples

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Float CSS</title>
<style type="text/css">
.recipe h2 {
float: left;
}
.recipe .intro,
.recipe ol {
float:right;
width: 500px;
}
</style>
</head>
<body>
<h1>Recipes for Cheese</h1>
<p class="recipe">Cheese is a remarkably versatile food, available in literally
hundreds of varieties with different flavors and textures.</p>
<div class="recipe">
<h2>Welsh Rarebit</h2>
<p class="intro">Welsh Rarebit is a savory dish made from melted cheese, often
Cheddar, on toasted bread, and a variety of other ingredients such as mustard,
egg, or bacon. Here is one take on this classic.</p>
<ol>
<li>Lightly toast the bread</li>
<li>Place on a baking tray, and spread with butter.</li>
<li>Add the grated Cheddar cheese and 2 tablespoons of beer to a saucepan.
Place the saucepan over a medium heat, and stir the cheese continuously
until it has melted. Add a teaspoon of wholegrain mustard and grind in
a little pepper. Keep stirring.</li>
<li>When thick and smooth, pour over each piece of toast spreading it to
the edges to stop the toast from burning.</li>
<li>Place under the grill for a couple of minutes or until golden
brown.</li>
```

```
</ol>
</div>
</body>
```

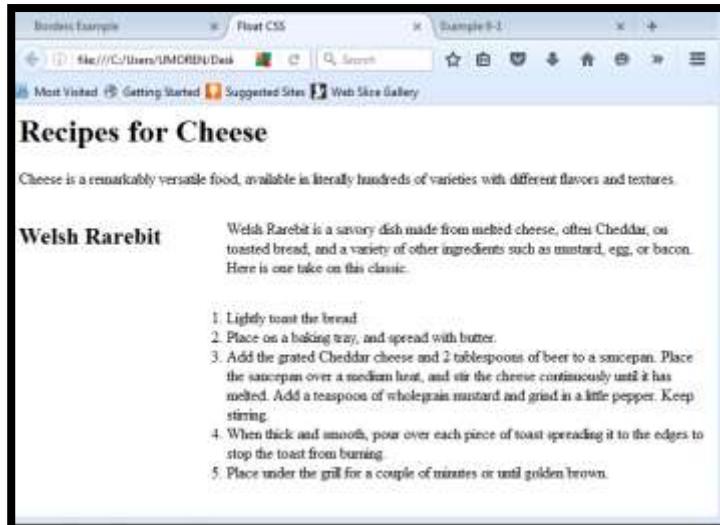


Fig. 10.1

10.1 Creating Navigation Example with Float

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Navigation Example with Float</title>
<style type="text/css">
body {
width: 600px;
margin: 1em auto;
}
h1 {
float: left;
margin-top: 0;
}
.navigation {
float: right;
margin: 0;
list-style: none;
}
.navigation li {
float: left;
}
.navigation a {
display: block;
```

```
margin-left: 0.5em;
padding: 0.5em;
border: 1px solid #CCC;
color: #233;
text-decoration: none;
}
.navigation a:focus,
.navigation a:hover {
background: #233;
color: #FFF;
}
</style>
</head>
<body>
<h1>Recipes for Cheese</h1>
<ul class="navigation">
<li><a href="#">Home</a></li>
<li><a href="#">Recipes</a></li>
<li><a href="#">Suggestions</a></li>
</ul>
</body>
</html>
```



Fig. 10.2

In the above example, the h1 element is floated left and the element with the class of navigation right. This positioned the navigation list alongside the heading, aligned to the right of the body.

10.2 Working with clear property in CSS

To control float content, clear property is usually used which specifies what should happen with the element that is next to a floating element. The possible values for clear-property are left, right, both and none. When you use the clear property, you can cancel a float on a particular element.

The clear property can have one of the following values:

- none - The element is not pushed below left or right floated elements. This is default
- left - The element is pushed below left floated elements
- right - The element is pushed below right floated elements
- both - The element is pushed below both left and right floated elements

- inherit - The element inherits the clear value from its parent

When clearing floats, you should match the clear to the float: If an element is floated to the left, then you should clear to the left. Your floated element will continue to float, but the cleared element will appear below it on the web page.

Floating text around an image

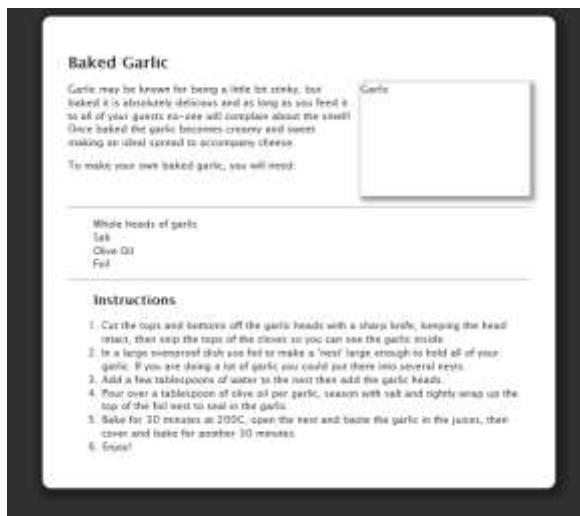


Fig. 10.3

The sample code in HTML5

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>Chapter 8: Float</title>
<link rel="stylesheet" href="float.css" />
<style type="text/css">
body {
background-color: #333;
background-image: url(brushed_alu_dark.png);
color: #fff;
margin: 0;
padding: 0;
font: 0.75em/1.3 "Lucida Grande", "Lucida Sans Unicode",
"Lucida Sans", Verdana, Tahoma, sans-serif;
}
```

```
.wrapper {  
width: 40%;  
margin: 20px auto 40px auto;  
background-color: #fff;  
color: #333;  
background-image: url(brushed_alu.png);  
-webkit-box-shadow: 3px 3px 10px 8px rgba(0, 0, 0, 0.4);  
-moz-box-shadow: 3px 3px 10px 8px rgba(0, 0, 0, 0.4);  
box-shadow: 3px 3px 10px 8px rgba(0, 0, 0, 0.4);  
-webkit-border-radius: 10px;  
-moz-border-radius: 10px;  
border-radius: 10px;  
padding: 30px 30px 30px 30px;  
  
}  
recipe {  
padding: 1em;  
}  
h2.instructions {  
background-image: url(instructions.png);  
background-repeat: no-repeat;  
background-position: left center;  
padding-left: 30px;  
}  
ul.ingredients {  
clear: both;  
border-top: 1px solid #999;  
border-bottom: 1px solid #999;  
list-style: none;  
margin: 1em 0 1em 0;  
padding: 1em 0 1em 30px;  
background-image: url(ingredients.png);  
background-repeat: no-repeat;  
background-position: 0 1em;  
}  
  
h2 {  
font-size: 125%;  
}  
  
h1 {  
font-size: 150%;  
}  
.recipe img {  
float: right;  
width: 200px;  
margin: 0 0 1em 1em;
```

```
-webkit-box-shadow: 3px 3px 5px 3px rgba(0, 0, 0, 0.4);
-moz-box-shadow: 3px 3px 5px 3px rgba(0, 0, 0, 0.4);
box-shadow: 3px 3px 5px 3px rgba(0, 0, 0, 0.4);
}

ul.ingredients {
  clear: both;
  border-top: 1px solid #999;
  border-bottom: 1px solid #999;
  list-style: none;
  margin: 1em 0 1em 0;
  padding: 1em 0 1em 30px;
  background-image: url(ingredients.png);
  background-repeat: no-repeat;
  background-position: 0 1em;
}

</style>
</head>
<body>
<div class="wrapper">
<div class="recipe">
<h1>Baked Garlic</h1>

<p>Garlic may be known for being a little bit stinky, but  
baked it is absolutely delicious and as long as you feed it  
to all of your guests no-one will complain about the smell!  
Once baked the garlic becomes creamy and sweet making an  
ideal spread to accompany cheese.</p>
<p>To make your own baked garlic, you will need:</p>
<ul class="ingredients">
<li>Whole heads of garlic</li>
<li>Salt</li>
<li>Olive Oil</li>
<li>Foil</li>
</ul>
<h2 class="instructions">Instructions</h2>
<ol>
<li>Cut the tops and bottoms off the garlic heads with a  
sharp knife, keeping the head intact, then snip the tops  
of the cloves so you can see the garlic inside.</li>
<li>In a large ovenproof dish use foil to make a 'nest'  
large enough to hold all of your garlic. If you are doing  
a lot of garlic you could put them into several nests.  
</li>
<li>Add a few tablespoons of water to the nest then add the  
garlic heads.</li>
<li>Pour over a tablespoon of olive oil per garlic, season  
with salt and tightly wrap up the top of the foil nest to
```

```
seal in the garlic.</li>
<li>Bake for 30 minutes at 200C, open the nest and baste the
garlic in the juices, then cover and bake for another
30 minutes.</li>
<li>Enjoy!</li>
</ol>
</div>
</div>
</body>
</html>
```

The Position Property

The CSS position property defines the position of an element in a document, it works with the left, right, top, bottom and z-index properties to determine the final position of an element on a page. Position property takes the following values:

- static: every element has a static position by default, so the element will stick to the normal page flow. So if there is a left/right/top/bottom/z-index set then there will be no effect on that element.

Example:

```
div.static {
    position: static;
    border: 3px solid #73AD21;
}
```

HTML Code

```
<!DOCTYPE html>
<html>
<head>
<style>
div.static {
    position: static;
    border: 3px solid #73AD21;
}
</style>
</head>
<body>
<h2>position: static;</h2>
<p>An element with position: static; is not positioned in any special way; it is
always positioned according to the normal flow of the page:</p>
<div class='static'>
    This div element has position: static;
</div>
</body>
</html>
```

Result

position: static;

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

This div element has position: static;

- relative: The "relative" value will still put the element in the normal flow, but then offset it according to top/left/right/bottom properties

Example:

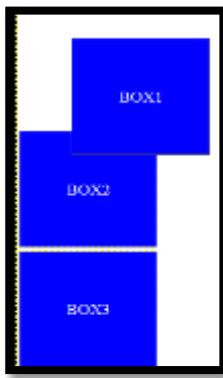
CSS Code

```
.container{  
background-color:white;  
width:400px;  
margin-top:10px;  
margin-left:auto;  
margin-right:auto;  
}  
  
.box1, .box2, .box3{  
width:130px;  
height: auto;  
background-color:blue;  
text-align: center;  
line-height: 8em;  
margin-top:5px;  
margin-left: 3px;  
}  
  
.box1{  
position:relative;  
top:30px;  
left:50px;  
}
```

HTML Code

```
<div class="container">  
 <div class="box1">Box1</div>  
 <div class="box2">Box2</div>  
 <div class="box3">Box3</div>  
</div>
```

Result

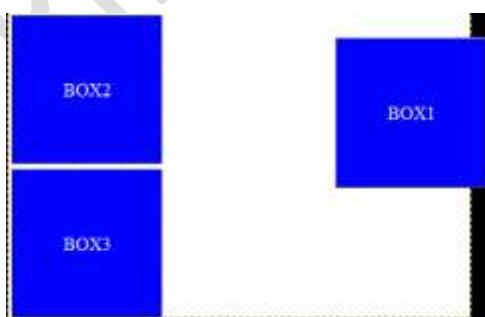


- absolute: This value takes the element out of the normal flow and position it in relation to the window (or the closest non-static element). If the closest parent element is not positioned, it is positioned relative to the next parent element that is positioned. If there's no positioned ancestor element, it is positioned relative to the <html> element.

Example:

```
.box1, .box2, .box3{  
    width:130px;  
    height: auto;  
    background-color:blue;  
    text-align: center;  
    line-height: 8em;  
    margin-top:5px;  
    margin-left: 3px;  
}  
  
.box1{  
    position:absolute;  
    top:30px;  
    right:50px;  
}
```

Result:



- fixed: Fixed positioning is a type of absolute positioning that requires the position property to have a value of fixed. It positions the element in relation to the browser window, so that when a user scrolls down the page, it stays in the exact same place.

Example

CSS Code

```
#nav, #main{  
    margin:auto;  
    width:760px;  
}  
ul li{  
    display:inline-block;  
    padding-left:5px;  
}  
ul{  
    margin-right:10px;  
    float:right;  
}  
h1{  
    color:#3E543C;  
    margin: 0px;  
    text-align: center;  
}  
#nav{  
    background-color: #C4BEBE;  
    height: 45px;  
    position:fixed;  
    top:0px;  
    left:260px;  
}  
#main{  
    margin-top: 85px;  
}
```

HTML Code

```
<div id="nav">  
    <ul>  
        <li>Home</li>  
        <li>About Us</li>  
        <li>Blog</li>  
        <li>Contact Us</li>  
    </ul>  
</div>
```

Result

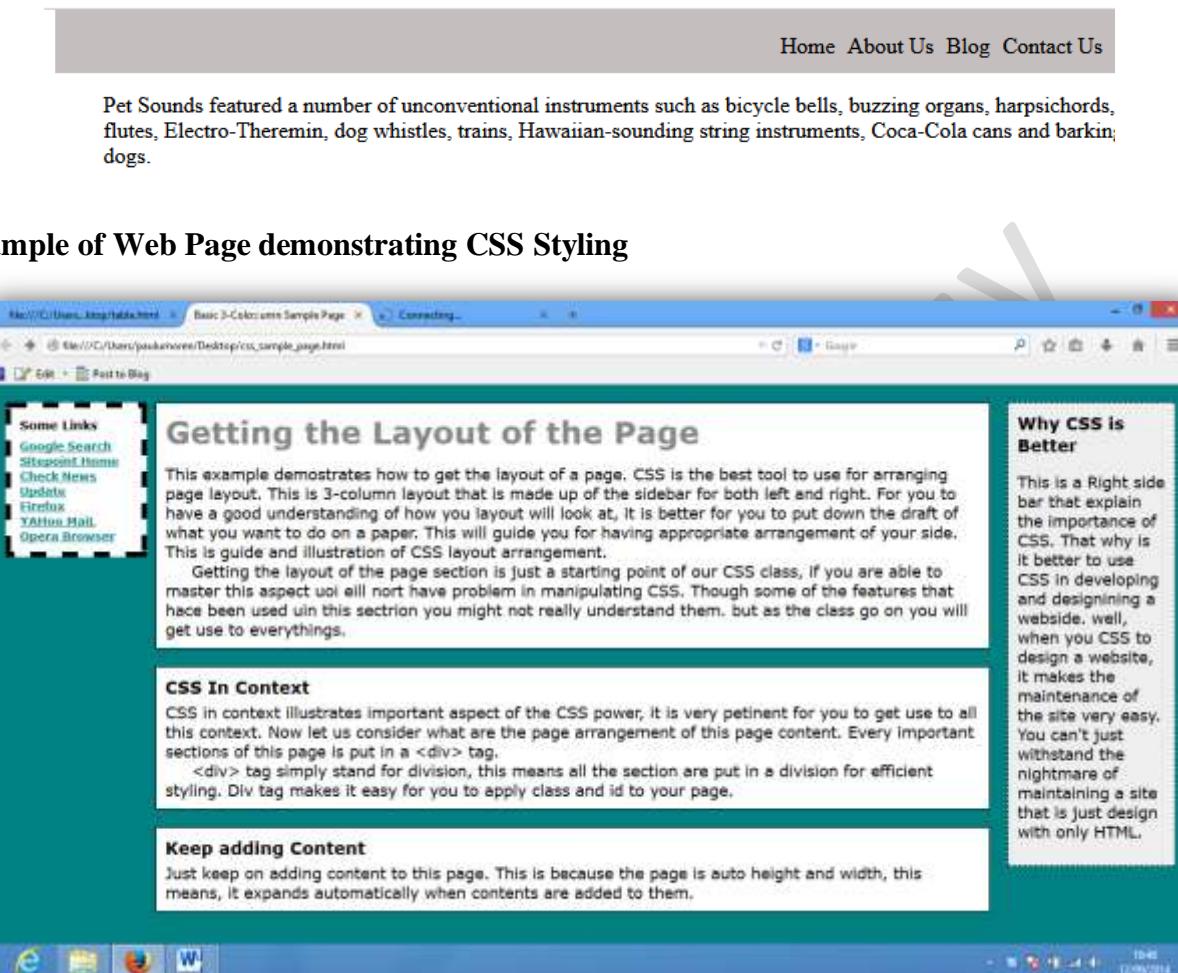


Fig. 10.4

The code that produces the page above is below:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Basic 3-Column Sample Page</title>
<meta http-equiv="Content-type" content="text/html; charset=iso-8859-1" />
<style type="text/css">
body{
background-color:teal;
margin:20px;
padding:0px;
font-size:1.1em;
font-family:verdana, Arial, Helvetica, sans-serif;
```

```
}

h1{
font-family:verdana, Arial, Helvetica, sans-serif;
margin:0 0 15px 0;
padding:0;
color:#888;
}

h2{
font-family:verdana, Arial, Helvetica, sans-serif;
margin:0 0 5px 0;
padding:0;
font-size:1.1em;
}

p{
font-family:verdana, Arial, Helvetica, sans-serif;
line-height:1.1em
margin:0 0 16px 0;
padding:0;
}

.content >p{
margin:0;
}

.content > p+p {
text-indent:30px;
}

a{
color:
teal;
font-family:verdana, Arial, Helvetica, sans-serif;
font-weight:600;
}

a:link{
color:teal;
}

a:visited{
color:teal;
}

a:hover{
background-color:#666;
}

/* All the content boxes belong to the content
class
*/
.content{
width: auto;
margin: 0 210px 20px 170px;
border: 1px solid black;
```

```
background-color: white;
padding: 10px;
z-index: 3;
}
#navleft{
position: absolute;
width: 128px;
top: 20px;
left: 20px;
font-size: small;
border: 6px dashed black;
background-color: white;
padding: 10px;
z-index: 2;
}
#navleft ul{
list-style: none;
margin: 0;
padding: 0;
}
#navright{
position: absolute;
width: 168px;
top: 20px;
border: 1px dashed black;
background-color: #eeeeee;
right: 20px;
padding: 10px;
z-index: 1;
}

```

}

</style>

</head>

<body>

<div class="content">

<h1> Getting the Layout of the Page </h1>

<p> This example demonstrates how to get the layout of a page. CSS is the best tool to use for arranging page layout. This is 3-column layout that is made up of the sidebar for both left and right. For you to have a good understanding of how your layout will look at, it is better for you to put down the draft of what you want to do on a paper. This will guide you for having appropriate arrangement of your side.

This is guide and illustration of CSS layout arrangement.</p>

<p> Getting the layout of the page section is just a starting point of our CSS class, if you are able to master this aspect you will not have problem in manipulating CSS.

Though some of the features that have been used in this section you might not really understand them. But as the class go on you will get use to everything.

</div>

<div class="content">

<h2>CSS In Context</h2>

<p>

CSS in context illustrates important aspect of the CSS power, it is very pertinent for you to get use to all this context. Now let us consider what are the page arrangement of this page content. Every important sections of this page is put in a <div> tag.

</p>

<p><div> tag simply stand for division, this means all the section are put in a division for efficient styling. Div tag makes it easy for you to apply class and id to your page.

</p>

</div>

<div class="content">

<h2>Keep adding Content</h2>

<p> Just keep on adding content to this page. This is because the page is auto height and width, this means, it expands automatically

when contents are added to them. </p>

</div>

<div id="navleft" >

<h2>Some Links</h2>

 Google Search

 Sitepoint Home

 Check News Update

 Firefox

 YAHOO Mail

 Opera Browser

</div>

<div id="navright">

<h2>Why CSS is Better</h2>

<p> This is a Right side bar that explain the importance of CSS. That why is it better to use CSS in developing and designining a webside.

well, when you CSS to design a website, it makes the maintenance of the site very easy. You can't just withstand the nightmare of maintaining a site that is just design with only HTML.

</p>

</div>

</body>

</html>

Chapter Eleven Styling Lists and Tables with CSS

Lists are versatile group of elements in HTML used for all sort of things such as site and page navigation, tab controls, and simple lists of items such as **for-tasks** or **shopping basket contents**. There are two types of list; unordered (ul) and ordered (ol) lists. The same principles apply to both in styling.

Lists consist of two parts. The first is the list container element, which is either `` for an unordered list and `` for an ordered list. The second part of a list is one or more `` elements, which contains each item. There must be at least one `` in every list.

The most common used of list in websites is site navigation. Navigation is a list of links on the website.

There are three list styling properties in CSS as shown below:-

Values list-style-type properties (ul) are

- circle → open circle
- disc → filled circle (bullet)
- square → filled square

1) List-style-type property

Example (unordered list)

```
ul{  
    list-style-type:square;  
}  
ul ul{  
    list-style-type: disc;  
}  
ul ul ul{  
    list-style-type: circle;  
}
```



Code

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">  
<title>Figure 10-3</title>  
<style type="text/css">  
.demo-disc {  
    list-style-type:disc;  
}  
.demo-circle {  
    list-style-type: circle;  
}  
.demo-square {  
    list-style-type: square;
```

```
        }
.demo-none {
list-style-type: none;
}
</style>
</head>
<body>
<h1>Unordered list bullet styles</h1>
<h2>Disc</h2>
<ul class="demo-disc">
<li>Home</li>
<li>About Us</li>
<li>Products</li>
<li>Services</li>
<li>Contact Us</li>
</ul>
<h2>Circle</h2>
<ul class="demo-circle">
<li>Home</li>
<li>About Us</li>
<li>Products</li>
<li>Services</li>
<li>Contact Us</li>
</ul>
<h2>Square</h2>
<ul class="demo-square">
<li>Home</li>
<li>About Us</li>
<li>Products</li>
<li>Services</li>
<li>Contact Us</li>
</ul>
<h2>None</h2>
<ul class="demo-none">
<li>Home</li>
<li>About Us</li>
<li>Products</li>
<li>Services</li>
<li>Contact Us</li>
</ul>
</body>
</html>
```



Fig. 11.1

Example (ordered list)

The number that precedes list items in an ordered list can be formatted with the following keywords: decimal, decimal-leading-zero, lower-roman, upper-roman, lower-greek, lower-latin, upper-latin, Armenian, Georgian, and none as shown below

1. decimal → 1,2,3,4,5,6,-----
2. decimal-leading-zero → 01,02,03,04,05,-----
3. lower-alpha → a,b,c,d,e,f,g,-----
4. lower-roman → i,ii,iii,iv,v,vi,-----
5. upper-roman → I,II,III,IV,-----

Note that the default value is decimal.

Styling the ordered list

```
ol{  
    list-style-type: upper-roman;  
}  
ol ol{  
    list-style-type: upper-alpha;  
}  
ol ol ol{  
    list-style-type: decimal;  
}
```

Code

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">  
<title>Ordered List Examples</title>
```

```
<style type="text/css">
.demo-decimal {
    list-style-type: decimal;
}
.demo-lower-roman {
    list-style-type: lower-roman;
}
.demo-upper-roman {
    list-style-type: upper-roman;
}
.demo-none {
    list-style-type: none;
}
</style>
</head>
<body>
<h1>Ordered list number styles</h1>
<h2>Decimal</h2>
<ol class="demo-decimal">
<li>Lightly toast the bread. Place on a baking tray, and spread with butter.</li>
<li>Add the grated Cheddar cheese and 2 tablespoons of beer to a saucepan. Place the saucepan over a medium heat, and stir the cheese continuously until it has melted. Add a teaspoon of wholegrain mustard and grind in a little pepper. Keep stirring.</li>
<li>When thick and smooth, pour over each piece of toast spreading it to the edges to stop the toast from burning.</li>
<li>Place under the grill for a couple of minutes or until golden brown.</li>
</ol>
<h2>Lower Roman</h2>
<ol class="demo-lower-roman">
<li>Lightly toast the bread. Place on a baking tray, and spread with butter.</li>
<li>Add the grated Cheddar cheese and 2 tablespoons of beer to a saucepan. Place the saucepan over a medium heat, and stir the cheese continuously until it has melted. Add a teaspoon of wholegrain mustard and grind in a little pepper. Keep stirring.</li>
<li>When thick and smooth, pour over each piece of toast spreading it to the edges to stop the toast from burning.</li>
<li>Place under the grill for a couple of minutes or until golden brown.</li>
</ol>
<h2>Upper Roman</h2>
<ol class="demo-upper-roman">
<li>Lightly toast the bread. Place on a baking tray, and spread with butter.</li>
<li>Add the grated Cheddar cheese and 2 tablespoons of beer to a saucepan. Place the saucepan over a medium heat, and stir the cheese continuously until it
```

has melted. Add a teaspoon of wholegrain mustard and grind in a little pepper. Keep stirring.

When thick and smooth, pour over each piece of toast spreading it to the edges to stop the toast from burning.

Place under the grill for a couple of minutes or until golden brown.

<h2>None</h2>

<ol class="demo-none">

Lightly toast the bread. Place on a baking tray, and spread with butter.

Add the grated Cheddar cheese and 2 tablespoons of beer to a saucepan. Place the saucepan over a medium heat, and stir the cheese continuously until it has melted. Add a teaspoon of wholegrain mustard and grind in a little pepper. Keep stirring.

has melted. Add a teaspoon of wholegrain mustard and grind in a little pepper. Keep stirring.

When thick and smooth, pour over each piece of toast spreading it to the edges to stop the toast from burning.

Place under the grill for a couple of minutes or until golden brown.

</body>

</html>

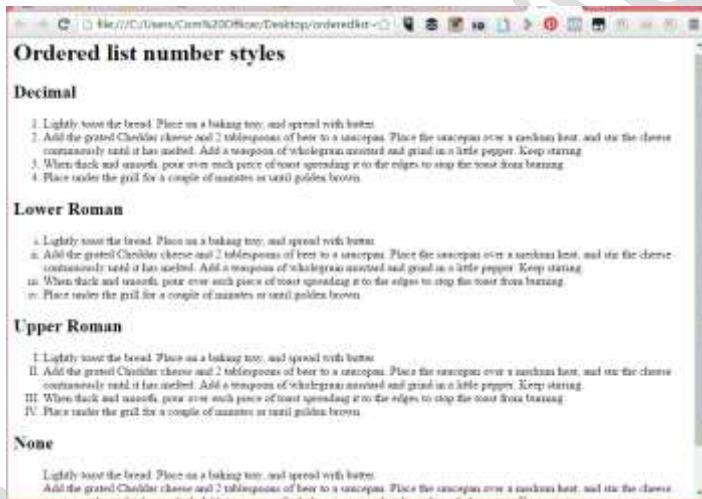


Fig. 11.2

11.1 List style image and position

The list-style-image property is used to insert images as list item markers.

Examples

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Ordered List Examples</title>
<style type="text/css">
```

```
li {  
list-style-image:url(../../../../xampp/htdocs/wp/wp-  
content/themes/travel_booking/html_v_1_6/assets/images/offer-list-  
link.png);  
}  
.alternate {  
list-style-image:url(../../../../xampp/htdocs/wp/wp-  
content/themes/travel_booking/html_v_1_6/assets/images/bottom-slider-  
next.png);  
}  
</style>  
</head>  
<body>  
<h1>List Items with Image</h1>  
<ol>  
<li>Lightly toast the bread. Place on a baking tray, and spread with  
butter.</li>  
<li>Add the grated Cheddar cheese and 2 tablespoons of beer to a  
saucepan.  
Place the saucepan over a medium heat, and stir the cheese continuously  
until it  
has melted. Add a teaspoon of wholegrain mustard and grind in a little  
pepper. Keep  
stirring.</li>  
<li>When thick and smooth, pour over each piece of toast spreading it to  
the  
edges to stop the toast from burning.</li>  
<li>Place under the grill for a couple of minutes or until golden  
brown.</li>  
</ol>  
</body>  
</html>
```

Output

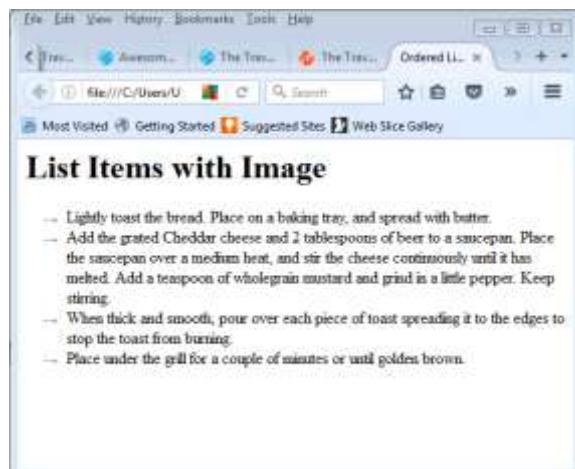


Fig. 11.3

11.2 List-style-position

The list-style-position property defines where to position the list marker, and it accepts either of these two values; inside or outside.

Example;

CSS Code

```
ul.a{  
    list-style-position: inside;  
}  
ul.b{  
    list-style-position: outside;  
}
```

HTML Code

```
<h2>The list-style-position Property Example</h2>  
<ul class='a'>  
    <li>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Repellendus eius  
        ullam, magni aut excepturi et harum recusandae veritatis culpa ipsam alias, neque  
        corporis fugit voluptatibus voluptates. Ad quia quaerat reprehenderit.</li>  
    <li>Lorem ipsum dolor sit amet consectetur adipisicing elit. Nam blanditiis quae  
        facere aliquam commodi cumque, dicta voluptatibus doloribus atque in debitis,  
        perspiciatis nam eius dignissimos consequuntur nisi dolorum. Accusamus,  
        repellendus.</li>  
    <li>Lorem ipsum dolor sit amet consectetur adipisicing elit. Eaque, modi voluptas  
        impedit libero similique distinctio veritatis iusto magnam sint natus ullam dicta,  
        eveniet corrupti dolores esse atque illum commodi alias.</li>  
</ul>  
<ul class='b'>  
    <li>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Repellendus eius  
        ullam, magni aut excepturi et harum recusandae veritatis culpa ipsam alias, neque  
        corporis fugit voluptatibus voluptates. Ad quia quaerat reprehenderit.</li>  
    <li>Lorem ipsum dolor sit amet consectetur adipisicing elit. Nam blanditiis quae  
        facere aliquam commodi cumque, dicta voluptatibus doloribus atque in debitis,
```

perspiciatis nam eius dignissimos consequuntur nisi dolorum. Accusamus, repellendus.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Eaque, modi voluptas impedit libero similique distinctio veritatis iusto magnam sint natus ullam dicta, eveniet corrupti dolores esse atque illum commodi alias.

**

Result

The list-style-position Property Example

- Lorem ipsum dolor sit amet, consectetur adipisicing elit. Repellendus eius ullam, magni aut excepturi et harum recusandae veritatis culpa ipsam alias, neque corporis fugit voluptatibus voluptates. Ad quia quaerat reprehenderit.
- Lorem ipsum dolor sit amet consectetur adipisicing elit. Nam blanditiis quae facere aliquam commodi cumque, dicta voluptatibus doloribus atque in debitis, perspiciatis nam eius dignissimos consequuntur nisi dolorum. Accusamus, repellendus.
- Lorem ipsum dolor sit amet consectetur adipisicing elit. Eaque, modi voluptas impedit libero similique distinctio veritatis iusto magnam sint natus ullam dicta, eveniet corrupti dolores esse atque illum commodi alias.
- Lorem ipsum dolor sit amet, consectetur adipisicing elit. Repellendus eius ullam, magni aut excepturi et harum recusandae veritatis culpa ipsam alias, neque corporis fugit voluptatibus voluptates. Ad quia quaerat reprehenderit.
- Lorem ipsum dolor sit amet consectetur adipisicing elit. Nam blanditiis quae facere aliquam commodi cumque, dicta voluptatibus doloribus atque in debitis, perspiciatis nam eius dignissimos consequuntur nisi dolorum. Accusamus, repellendus.
- Lorem ipsum dolor sit amet consectetur adipisicing elit. Eaque, modi voluptas impedit libero similique distinctio veritatis iusto magnam sint natus ullam dicta, eveniet corrupti dolores esse atque illum commodi alias.

Fig.11.4

11.2 Styling tables in CSS

The look of an HTML table can be greatly improved with CSS; in the HTML tag section, we have discussed table elaborately. However, let us recap some of the optional features we discussed under table before looking at how to style table.

In the preceding markup, you can see that HTML tables support many additional (optional) elements.

- The `<caption>` element is used to provide the table with a caption or the name of the table.
- The `<colgroup>` element is used to enclose each of the table `<col>` elements. `<colgroup>` elements are not displayed.
- `<col>` elements are used to control certain properties of each table column, the most common being the column width. `<col>` elements are not displayed and contain no content.
- The `<thead>` element encloses information about column headers. If you print a table that spans more than one page, the information in the `<thead>` element is repeated at the top of each page.

- The `<tbody>` element contains the main table data.
- The `<tfoot>` element is similar to the `<thead>` element, and is sometimes used to repeat column headers in long tables but may contain summary or footnote content. When you print a table that spans more than one page, the information in the `<tfoot>` element is repeated at the bottom of each page.

11.2.1 Controlling Border Spacing in Table

The border-spacing property allows more control over cell spacing. It allows the length to be specified.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Table Border Space</title>
<style type="text/css">
table{
border-spacing: 15px;
}
table,
th,
td{
border: 1px solid #000;
}
</style>
</head>
<body>
<table>
<caption>Ingredients</caption>
<colgroup>
<col class="ingredient">
<col class="quantity">
</colgroup>
<thead>
<tr>
<th>Ingredient</th>
<th>Quantity of Ingredient</th>
</tr>
<tbody>
<tr>
<th>Ingredient</th>
<th>Quantity of Ingredient</th>
</tr>
</tbody>
<tr>
```

```
<td>Bread</td>
<td>2 medium thickness slices</td>
</tr>
<tr>
<td>Butter</td>
<td>Enough for 2 slices of bread</td>
</tr>
<tr>
<td>Grated Cheddar</td>
<td>1.5 handfuls</td>
</tr>
<tr>
<td>Beer</td>
<td>One splash</td>
</tr>
<tr>
<td>Wholegrain mustard</td>
<td>One dollop</td>
</tr>
<tr>
<td>Pepper</td>
<td>To taste</td>
</tr>
</tbody>
</table>
</body>
</html>
```

A screenshot of a Microsoft Internet Explorer browser window. The title bar says 'Table Border Space'. The address bar shows 'file:///C:/Users/UMOREN/Desktop/HTML/Tables.html'. The page content is titled 'Ingredients' and contains a table with border spacing. The table has two columns: 'Ingredient' and 'Quantity of Ingredient'. The rows contain the following data:

| Ingredient | Quantity of Ingredient |
|--------------------|------------------------------|
| Bread | 2 medium thickness slices |
| Butter | Enough for 2 slices of bread |
| Grated Cheddar | 1.5 handfuls |
| Beer | One splash |
| Wholegrain mustard | One dollop |
| Pepper | To taste |
| Ingredient | Quantity of Ingredient |

Fig. 11.5

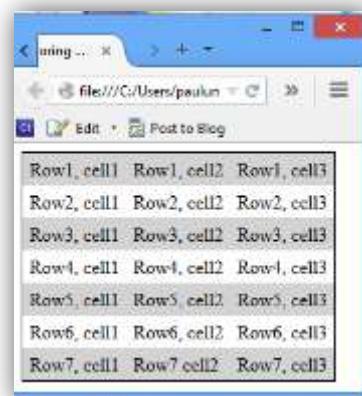
In the above example, we applied a border-spacing: 15px 5px; declaration to the <table> element, which added 15px of horizontal spacing and 5px vertical spacing between each cell.

11.2.2 Coloring Alternate Rows and Adding cell Borders in Data Tables

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
<title> Coloring Rows in a Table</title>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
<style type="text/css" >
.odd{
    Background-color:lightgrey;
}
.even{
    Background-color: white;
}
table{
    Border: 1px solid black;
    Border-spacing:0;
}
td{
```

```
Padding:4px 6px;
}
</style>
</head>
<body>
<table>
<tr class = "odd">
<td> Row1, cell1</td>
<td> Row1, cell2</td>
<td> Row1, cell3</td>
</tr>
<tr class= "even">
<td> Row2, cell1</td>
<td> Row2, cell2</td>
<td> Row2, cell3</td>
</tr>
<tr class="odd">
<td> Row3, cell1</td>
<td> Row3, cell2</td>
<td> Row3, cell3</td>
</tr>
<tr class="even">
<td> Row4, cell1</td>
<td> Row4, cell2</td>
<td> Row4, cell3</td>
</tr>
<tr class= "odd">
<td> Row5, cell1</td>
<td> Row5, cell2</td>
<td> Row5, cell3</td>
</tr>
<tr class="even">
<td> Row6, cell1</td>
<td> Row6, cell2</td>
<td> Row6, cell3</td>
</tr>
<tr class= "odd">
<td> Row7, cell1</td>
<td> Row7 cell2</td>
<td> Row7, cell3</td>
</tr>
</table>
</body>
</html>
```



PINFOICT ACADEMY

Chapter Twelve

Custom Mouse Cursor

12.1 Custom Mouse Cursor in CSS

CSS provides the cursor property to control the type of cursor displayed for a particular element. This is very useful when developing advanced web application.

The code below demonstrate how to use keywords that change how mouse cursor is displayed depending on html element used on a web page.

Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml">  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">  
<title>Example 16-2</title>  
<style type="text/css">  
.crosshair {  
cursor: crosshair;  
}  
.pointer {  
cursor: pointer;  
}  
.move {  
cursor: move;  
}  
.e-resize {  
cursor: e-resize;  
}  
.w-resize {  
cursor: w-resize;  
}  
.ne-resize {  
cursor: ne-resize;  
}  
.sw-resize {  
cursor: sw-resize;  
}  
.n-resize {  
cursor: n-resize;  
}  
.s-resize {  
cursor: s-resize;  
}  
.nw-resize {  
cursor: nw-resize;  
}
```

```
.se-resize {  
    cursor: se-resize;  
}  
.text {  
    cursor: text;  
}  
.wait {  
    cursor: wait;  
}  
.help {  
    cursor: help;  
}  
.progress {  
    cursor: progress;  
}  
.hand {  
    cursor: hand;  
}  
.all-scroll {  
    cursor: all-scroll;  
}  
.col-resize {  
    cursor: col-resize;  
}  
.row-resize {  
    cursor: row-resize;  
}  
.no-drop {  
    cursor: no-drop;  
}  
.not-allowed {  
    cursor: not-allowed;  
}  
.vertical-text {  
    cursor: vertical-text  
}  
/style>  
  
</head>  
<body>  
<h1>Cursor Types</h1>  
<ul>  
<li class="crosshair">Crosshair</li>  
<li Pointer>Pointer</li>  
<li class="move">Move</li>  
<li class="e-resize">E-Resize</li>  
<li class="w-resize">W-Resize</li>
```

```
<li class="ne-resize">NE-Resize</li>
<li class="sw-resize">SW-Resize</li>
<li class="n-resize">N-Resize</li>
<li class="s-resize">S-Resize</li>
<li class="nw-resize">NW-Resize</li>
<li class="se-resize">SE-Resize</li>
<li class="text">Text</li>
<li class="wait">Wait</li>
<li class="help">Help</li>
<li class="progress">Progress</li>
<li class="hand">Hand</li>
<li class="all-scroll">All-Scroll</li>
<li class="col-resize">Col-Resize</li>
<li class="row-resize">Row-Resize</li>
<li class="no-drop">No-Drop</li>
<li class="not-allowed">Not-Allowed</li>
<li class="vertical-text">Vertical-Text</li>
</ul>
</body>
</html>
```

Output

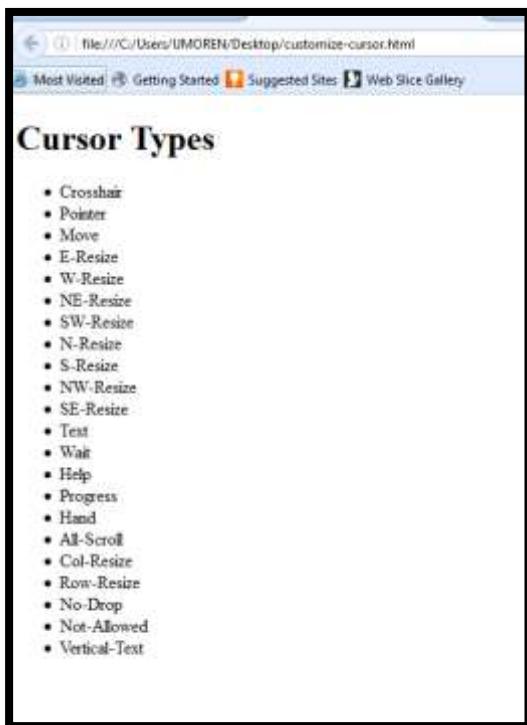


Fig. 12.14

Hovering over each of the list items triggers a different cursor as specified for the associated class.

PINFOICT ACADEMY

Chapter Thirteen

Working with CSS 3.0 (Advanced CSS)

CSS 3

CSS3 is an extension of CSS 2.1 that adds powerful new functionality such as image-free visual effects, box transformation, Transitions and animations, rounded corners using radius property, and multiple columns to web development.

13.1 Image-free visual effects.

CSS3 contains a lot of new properties that allow you to create visual effects that previously could be accomplished only with images and scripting language (Javascript) such as rounded corners, drop shadows, semitransparent backgrounds, gradients, and images for borders.

⇒ Creating rounded corners in CSS3

Border radius property is used to create rounded corners in CSS 3, the syntax is below;

Syntax

```
-webkit-border-radius: size;  
-moz-border-radius: size;  
border-radius: size;
```

The first is for Webkit-based browsers (Safari and Chrome), the second is for Mozilla-based browsers (Firefox) and the last is the un-prefixed version for browsers that support it such as Internet Explorer 9 and above, Opera, and Safari 5.

Example

```
<!DOCTYPE HTML>  
<html>  
<head>  
<meta charset="utf-8">  
<title>Untitled Document</title>  
<style type="text/css">  
blockquote {  
margin: 0 0 0 12px;  
padding: 10px 15px 5px 15px;  
-moz-border-radius: 20px;  
-webkit-border-radius: 20px;  
border-radius: 20px;  
border-top: 1px solid #fff;  
background-color: #A6DADC;  
word-wrap: break-word;  
}  
</style>  
</head>  
<body>  
<blockquote>
```

This is how rounded corners look like. It increases speed of loading page from server compare to producing the same effect using images or javascripts.

```
</blockquote>  
</body>  
</html>
```

Output

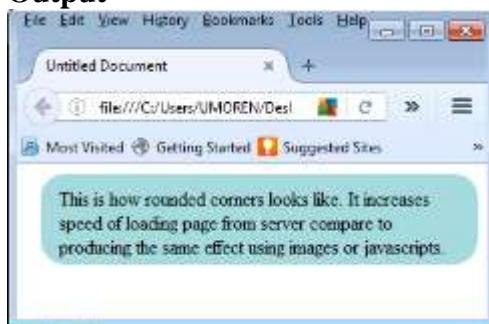


Fig.13.1

In the above example, the border-radius is 20px which is a shorthand property for the border-top-left-radius, border-top-right-radius, border-bottom-right-radius and border-bottom-left-radius properties.

Specify each corner with CSS border-radius property

This property can have up to four values such as;

border-radius:10px 4px 35px 3px;

the first value applies to the top-left-corner, the second value applies to top-right-corner, the third value applies to the bottom-right corner, the fourth value applies to bottom-left corner as shown in the examples below;

CSS Code

```
.rcorner{  
    border-radius: 10px 30px 5px 12px;  
    background-color: #83Ad21;  
    padding: 10px;  
    width: auto;  
    height: auto;  
}  
.rcorner2{  
    border-radius: 20px 30px ;  
    background-color: #95aa22;  
    padding: 10px;  
    width: auto;  
    height: auto;  
}
```

HTML Code

```
<ul class="a rcorner">
    <li>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Repellendus eius
        ullam, magni aut excepturi et harum recusandae veritatis culpa ipsam alias, neque
        corporis fugit voluptatibus voluptates. Ad quia quaerat reprehenderit.</li>
    <li>Lorem ipsum dolor sit amet consectetur adipisicing elit. Nam blanditiis quae
        facere aliquam commodi cumque, dicta voluptatibus doloribus atque in debitis,
        perspiciatis nam eius dignissimos consequuntur nisi dolorum. Accusamus,
        repellendus.</li>
    <li>Lorem ipsum dolor sit amet consectetur adipisicing elit. Eaque, modi voluptas
        impedit libero similique distinctio veritatis iusto magnam sint natus ullam dicta,
        eveniet corrupti dolores esse atque illum commodi alias.</li>
</ul>
<ul class="b rcorner2">
    <li>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Repellendus eius
        ullam, magni aut excepturi et harum recusandae veritatis culpa ipsam alias, neque
        corporis fugit voluptatibus voluptates. Ad quia quaerat reprehenderit.</li>
    <li>Lorem ipsum dolor sit amet consectetur adipisicing elit. Nam blanditiis quae
        facere aliquam commodi cumque, dicta voluptatibus doloribus atque in debitis,
        perspiciatis nam eius dignissimos consequuntur nisi dolorum. Accusamus,
        repellendus.</li>
    <li>Lorem ipsum dolor sit amet consectetur adipisicing elit. Eaque, modi voluptas
        impedit libero similique distinctio veritatis iusto magnam sint natus ullam dicta,
        eveniet corrupti dolores esse atque illum commodi alias.</li>
</ul>
```

Result

- Lorem ipsum dolor sit amet, consectetur adipisicing elit. Repellendus eius
ullam, magni aut excepturi et harum recusandae veritatis culpa ipsam alias,
neque corporis fugit voluptatibus voluptates. Ad quia quaerat reprehenderit.
 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Nam blanditiis quae
facere aliquam commodi cumque, dicta voluptatibus doloribus atque in debitis,
perspiciatis nam eius dignissimos consequuntur nisi dolorum. Accusamus,
repellendus.
 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Eaque, modi voluptas
impedit libero similique distinctio veritatis iusto magnam sint natus ullam dicta,
eveniet corrupti dolores esse atque illum commodi alias.
-
- Lorem ipsum dolor sit amet, consectetur adipisicing elit. Repellendus eius ullam,
magni aut excepturi et harum recusandae veritatis culpa ipsam alias, neque
corporis fugit voluptatibus voluptates. Ad quia quaerat reprehenderit.
 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Nam blanditiis quae
facere aliquam commodi cumque, dicta voluptatibus doloribus atque in debitis,
perspiciatis nam eius dignissimos consequuntur nisi dolorum. Accusamus,
repellendus.
 - Lorem ipsum dolor sit amet consectetur adipisicing elit. Eaque, modi voluptas
impedit libero similique distinctio veritatis iusto magnam sint natus ullam dicta,
eveniet corrupti dolores esse atque illum commodi alias.

⇒ Box shadow in CSS3

The box-shadow property is used to create drop shadows in CSS3. In this property, you set the shadow's horizontal and vertical offsets from the box, its colour, and optionally set blur radius and spread radius.

Syntax

```
-webkit-box-shadow: offset_x offset_y blur_radius color;  
-moz-box-shadow: offset_x offset_y blur_radius color;  
box-shadow: offset_x offset_y blur_radius color;
```

The first property, *offset_x* is the horizontal offset from the box, and it tells the browser to move the shadow a certain pixel to the right of the box's edge. The second property, *offset_y*, is the vertical offset, moving the shadow a certain pixel down. You can use negative values to move the shadow to the left and up instead. The third property, *blur_radius*, is the blur radius, which specifies over how many pixels the shadow should stretch. A larger value makes the shadow blurrier and softer; a value of zero would make it completely sharp-edged. The fourth property, *color*, is for the shadow colour.

Example

```
.box-shadow  
{  
    -webkit-border-radius: 10px;  
    -moz-border-radius: 10px;  
    border-radius: 10px;  
    background-color:#CCCCCC;  
    -webkit-box-shadow: 5px 5px 10px #555;  
    -moz-box-shadow: 5px 5px 10px #555;  
    box-shadow: 5px 5px 10px #555;  
    width: 160px;  
    height: 160px;  
}
```

Output



Fig.13.2

Rendering the shadow inside a box

We use inset keyword to render the shadow in a box

Example

```
.box-shadow-inside{  
    -webkit-border-radius: 10px;
```

```
-moz-border-radius: 10px;  
border-radius: 10px;  
background-color:#CCCCCC;  
-webkit-box-shadow: 0 0 20px #fff inset;  
-moz-box-shadow: 0 0 20px #fff inset;  
box-shadow: 0 0 20px #fff inset;  
width: 160px;  
height: 160px;  
}
```

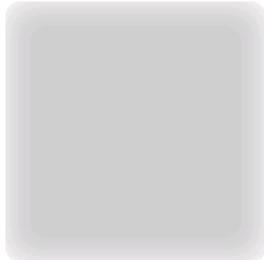


Fig. 13.3

Creating multiple shadows on a box

All we need to do is to create multiple shadows and separate them with commas;

Example

```
.shadow-inside-multiple{  
width: 160px;  
height: 160px;  
margin: 0 0 0 80px;  
-webkit-box-shadow: 0 0 20px #333 inset,  
                    20px 15px 30px yellow,  
                    -20px 15px 30px lime,  
                    -20px -15px 30px blue,  
                    20px -15px 30px red;  
  
-moz-box-shadow: 0 0 20px #333 inset,  
                20px 15px 30px yellow,  
                -20px 15px 30px lime,  
                -20px -15px 30px blue,  
                20px -15px 30px red;  
  
box-shadow: 0 0 20px #333 inset,  
            20px 15px 30px yellow,  
            -20px 15px 30px lime,  
            -20px -15px 30px blue,  
            20px -15px 30px red;  
}
```

Output



Fig.13.4

⇒ Text Shadows

The property, `text-shadow`, is used to apply shadow to text in CSS3. It uses the same properties as box shadow. The only difference is the property is applied on texts.

Syntax

```
text-shadow: offset_x offset_y blur_radius color;
```

Example

```
text-shadow {  
    text-shadow: 0 0 5px #333;  
    color:#FFF;  
}
```

This is how rounded corners looks like. It increases speed of loading page from server compared to producing the same effect using images or javascripts

Fig.13.5

⇒ Creating multiple shadow on a text

Just like we create multiple box shadows, this is possible on text by separating properties with commas.

Example

```
.multi-text-shadow{  
    text-shadow: 0 0 4px #ccc,  
                0 0 4px #ccc;  
}
```

```
0 -5px 4px #ff3,  
2px -10px 6px #fd3,  
-2px -15px 6px #f80,  
2px -18px 5px #f20;  
color:#FF0000;  
font-weight:bold;  
}
```

Output



Wao, this text is on FIRE

Fig.13.6

⇒ CSS 2D Transforms

With 2D transform properties, you are able to move, rotate, scale and skew elements. The following are 2D transformation methods:

- **translate()**: This method moves an element from its current position according to the parameters given for the X-axis and Y-axis example:

```
div{  
    transform:translate(40px,80px);  
}
```

- **rotate()**: This method rotates an element clockwise or counter-clockwise according to a given degree.

Example

```
div{  
    transform:rotate(60deg);  
}
```

- **scaleX()**: This method increases or decreases the width of an element. It uses numerical values.

Example

```
div{  
    transform:scaleX(2); => increases the element to be double of its original  
    width  
}  
  
div{  
    transform:scaleX(0.5); decreases the element to be half of its original  
    width  
}  
}
```

- `scaleY()`: This method increases or decreases the height of an element according to the given value.

Example

```
div{  
    transform:scaleY(4); increases the element four times of its original height  
}
```

- `scale()`: This method increases or decreases the size of an element according to the given parameters for width and height.

Example

```
div{  
    transform:scale(4,2); increases the element four times of its original width  
and two times of its original height.  
}
```

- `skewX()`: This method skews an element along the x-axis by the given angle.

Example

```
header{  
    transform:skewX(30deg); skews the header element 30 degree along the  
X-axis  
}
```

- `skewY()`: This method skews an element along the y-axis by the given angle.

Example

```
header{  
    transform:skewY(30deg); skews the header element 30 degree along the  
Y-axis  
}
```

- `skew()`: This method skews an element along the X-axis and Y-axis by the given angles.

Example

```
div{  
    transform:skew(30deg,20deg); skews the div element 30 degree along the X-axis  
and the Y-axis  
}
```

- `matrix()`: This method combines all the 2D transform methods into one by taking six parameters containing mathematical functions which allows elements to be rotated, scaled, moved(translated), and skewed.

Example

```
div{  
    transform:matrix(1,-0.5,0,1,0,0);  
}
```

⇒ 3D Transform

```
.btn {  
    width: 100px;
```

```
height: 100px;  
border: none;  
background: slateblue;  
color: white;  
font-size: 20px;  
font-weight: 50;  
line-height: 1;  
}  
.btn:hover {  
transform: translateY(-10px);  
}
```

HTML Code

```
<button class="btn">  
Hello World  
</button>
```

Result



Hello
World

⇒ CSS Transition

In CSS, transition is an animation that moves a property between two states. It is an implicit animation, which means it is triggered only when a new value is set for a CSS property. For a transition to occur, four conditions must be in place: an initial value, an end value, the transition itself, and a trigger.

The shorthand CSS syntax is written as follows:

```
div {  
  
transition: <property> <duration> <timing-function> <delay>;  
  
}
```

CSS Transition Properties

The following table lists all the CSS transition properties:

| Property | Description |
|----------|-------------|
|----------|-------------|

| | |
|----------------------------|--|
| Transition | A shorthand property for setting the four transition properties into a single property <div style="margin-left: 20px;"><pre>div { width: 50px; transition: width 2s; }</pre></div> |
| transition-delay | Specifies a delay (in seconds) for the transition effect <div style="margin-left: 20px;"><pre>div { transition-delay: 5s; }</pre></div> |
| transition-duration | Specifies how many seconds or milliseconds a transition effect takes to complete <div style="margin-left: 20px;"><pre>div { transition-duration: 3s; }</pre></div> |
| transition-property | Specifies the name of the CSS property the transition effect is for <div style="margin-left: 20px;"><pre>div { transition-property: height; }</pre></div> |
| transition-timing-function | <p>Specifies the speed curve of the transition effect.</p> <p>The transition-timing-function property can have the following values:</p> <ul style="list-style-type: none"> • ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default) • linear - specifies a transition effect with the same speed from start to end • ease-in - specifies a transition effect with a slow start • ease-out - specifies a transition effect with a slow end • ease-in-out - specifies a transition effect with a slow start and end • cubic-bezier(n,n,n,n) - lets you define your own values in a cubic-bezier function <pre>(.div1 {transition-timing-function: linear;} .div2 {transition-timing-function: ease;} .div3 {transition-timing-function: ease-in;} .div4 {transition-timing-function: ease-out;} .div5 {transition-timing-function: ease-in-out;})</pre> |

Example

CSS Code

```
div{  
    width:200px;  
    height: 200px;  
    background-color: #95aa22;  
    transition: width 3s;  
}  
div:hover{
```

```
width: 400px;  
}
```

HTML

```
<div>  
    <p>  
        this is the effect of transition  
    </p>  
</div>
```

Result



In the above example. When cursor moves over the object it changes from its original width to 400px; when the cursor moves out of the element, it gradually change back to its original size which is 200px.

Changing several property values

In the example below, a transition effect is added for both the width and height property, with a duration of 3 seconds for the width and 5 seconds for the height and the transition width and height 400px and 500px respectively:

CSS Code

```
div{  
    width:200px;  
    height: 200px;  
    background-color: #95aa22;  
    transition: width 3s height 5s;  
}  
div:hover{  
    width: 400px;  
    height: 500px;  
}
```

HTML

```
<h2>Hover the mouse cursor to see the transition effect</h2>  
<div>  
    <p>  
        this is the effect of transition  
    </p>  
</div>
```

Result



The example below performs a four-second font size transition with a two-second delay between the time the user mouse hovers over the element and the beginning of the animation effect:

CSS Code

```
.delfont {  
    font-size: 14px;  
    transition-property: font-size;  
    transition-duration: 4s;  
    transition-delay: 2s;  
}  
.delfont:hover {  
    font-size: 36px;  
}
```

HTML Code

```
<p class="delfont">This is the effect</p>
```

Multiple animated properties example

CSS Code

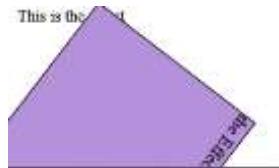
```
.shape {  
    border-style: solid;  
    border-width: 1px;  
    display: block;  
    width: 100px;  
    height: 100px;  
    background-color: #00F;  
    transition: width 2s, height 2s, background-color 2s, transform 2s;  
}
```

```
.shape:hover {  
    background-color: #FCC;  
    width: 250px;  
    height: 250px;  
    transform: rotate(180deg);  
}
```

HTML

```
<div class='shape'>  
    the Effect  
</div>
```

Result



Using transitions when highlighting menus

A common use of CSS is to highlight items in a menu as the user hovers the mouse cursor over them. It's easy to use transitions to make the effect even more attractive.

First, we set up the menu using HTML:

CSS Code

```
a {  
    flex:1;  
    background-color: #333;  
    color: #fff;  
    border: 1px solid;  
    padding: 0.5rem;  
    text-align: center;  
    text-decoration: none;  
    transition: all 0.5s ease-out;  
}  
a:hover,  
a:focus {  
    background-color: #fff;  
    color: #333;  
}
```

HTML Code

```
<nav>
  <a href="#">Home</a>
  <a href="#">About Us</a>
  <a href="#">Portfolio</a>
  <a href="#">Contact Us</a>
</nav>
```

Result



⇒ CSS Animation

Animations are powerful tools for engaging and delighting visitors on a site which make loading experience more entertaining, direct the attention to an important element on the page, as well as improve usability. Animations consist of two components, a style describing the CSS animation and a set of keyframes that indicate the start and end states of the animation's style, with possible intermediate waypoints. CSS animations are great for websites because they add dynamic, engaging content without placing much more weight on the page, since they do not require extra scripts. CSS animations are unlikely to slow down your pages.

The following are CSS Animation properties:

- ⇒ `@keyframes`: To use CSS animation, you must first specify some keyframes for the animation. Keyframes hold what styles the element will have at certain times. When you specify CSS styles inside the `@keyframes` rule, the animation will gradually change from the current style to the new style at certain times. The following example binds the "sample" animation to the `<div>` element.

The animation will last for 4 seconds, and it will gradually change the background-color of the `<div>` element from "red" to "yellow":

CSS

```
/* The animation code */
@keyframes sample {
  from {background-color: red;}
  to {background-color: yellow;}
}

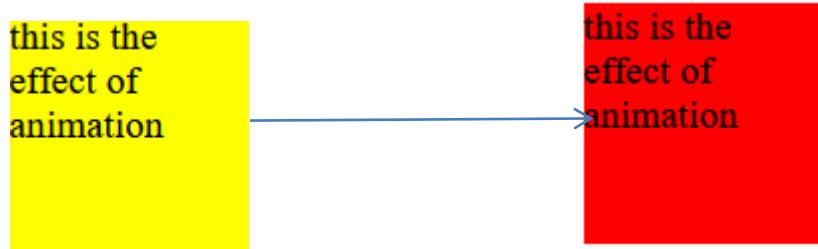
/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
```

```
background-color: red;  
animation-name: example;  
animation-duration: 4s;  
}
```

HTML Code

```
<div>  
  <p>  
    this is the effect of animation  
  </p>  
</div>
```

Result



- ⇒ `animation-name`: Specifies the name of the `@keyframes` at-rule describing the animation's keyframes.
- ⇒ `animation-duration`: The `animation-duration` property defines how long an animation should take to complete. If the `animation-duration` property is not specified, no animation will occur, because the default value is 0s (0 seconds)

Example

```
/* The animation code */  
@keyframes sample {  
  0% {background-color: red;}  
  25% {background-color: yellow;}  
  50% {background-color: blue;}  
  100% {background-color: green;}  
}  
  
/* The element to apply the animation to */  
div {  
  width: 100px;  
  height: 100px;  
  background-color: red;  
  animation-name: sample;  
  animation-duration: 4s;  
}
```

The above example will change the background-color of the <div> element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete with 4s duration.

- ⇒ **animation-delay:** The animation-delay property specifies a delay for the start of an animation. The following example has a 4 seconds delay before starting the animation:

```
div {  
    width: 100px;  
    height: 100px;  
    position: relative;  
    background-color: red;  
    animation-name: example;  
    animation-duration: 4s;  
    animation-delay: 4s;  
}
```

- ⇒ **animation-iteration-count:** The animation-iteration-count property specifies the number of times an animation should run. The following example will run the animation 5 times before it stops:

```
div {  
    width: 100px;  
    height: 100px;  
    position: relative;  
    background-color: red;  
    animation-name: example;  
    animation-duration: 4s;  
    animation-iteration-count: 5;  
}
```

The following example uses the value "infinite" to make the animation continue for ever:

```
div {  
    width: 100px;  
    height: 100px;  
    position: relative;  
    background-color: red;  
    animation-name: example;  
    animation-duration: 4s;  
    animation-iteration-count: infinite;  
}
```

- ⇒ **animation-direction:** This property specifies whether an animation should be played forwards, backwards or in alternate cycles. The animation-direction property have the following values:

- normal - The animation is played as normal (forwards). This is default
- reverse - The animation is played in reverse direction (backwards)
- alternate - The animation is played forwards first, then backwards
- alternate-reverse - The animation is played backwards first, then forwards

example

```
div {  
    width: 100px;  
    height: 100px;  
    position: relative;  
    background-color: red;  
    animation-name: example;  
    animation-duration: 4s;  
    animation-iteration-count: 2;  
    animation-direction: alternate;  
}
```

⇒ animation-timing-function: This property specifies the speed curve of the animation. The animation-timing-function property can have the following values:

- ease - Specifies an animation with a slow start, then fast, then end slowly (this is default)
- linear - Specifies an animation with the same speed from start to end
- ease-in - Specifies an animation with a slow start
- ease-out - Specifies an animation with a slow end
- ease-in-out - Specifies an animation with a slow start and end
- cubic-bezier(n,n,n,n) - Lets you define your own values in a cubic-bezier function

The following example shows some of the different speed curves that can be used:

```
.div1 {animation-timing-function: linear;}  
.div2 {animation-timing-function: ease;}  
.div3 {animation-timing-function: ease-in;}  
.div4 {animation-timing-function: ease-out;}  
.div5 {animation-timing-function: ease-in-out;}
```

⇒ animation-fill-mode: The animation-fill-mode property specifies a style for the target element when the animation is not playing (before it starts, after it ends, or both).

The animation-fill-mode property can have the following values:

- none - Default value. Animation will not apply any styles to the element before or after it is executing
- forwards - The element will retain the style values that is set by the last keyframe (depends on animation-direction and animation-iteration-count)

- backwards - The element will get the style values that is set by the first keyframe (depends on animation-direction), and retain this during the animation-delay period
- both - The animation will follow the rules for both forwards and backwards, extending the animation properties in both directions

example

```
div {  
    width: 100px;  
    height: 100px;  
    background: red;  
    position: relative;  
    animation-name: example;  
    animation-duration: 3s;  
    animation-fill-mode: forwards;  
}
```

The following example lets the <div> element get the style values set by the first keyframe before the animation starts (during the animation-delay period):

```
div {  
    width: 100px;  
    height: 100px;  
    background: red;  
    position: relative;  
    animation-name: example;  
    animation-duration: 3s;  
    animation-delay: 2s;  
    animation-fill-mode: backwards;  
}
```

⇒ animation: The animation shorthand uses six properties for example;

```
div {  
    animation-name: example;  
    animation-duration: 5s;  
    animation-timing-function: linear;  
    animation-delay: 2s;  
    animation-iteration-count: infinite;  
    animation-direction: alternate;  
}
```

The same animation effect as above can be achieved by using the shorthand animation property:

```
div {  
    animation: example 5s linear 2s infinite alternate;  
}
```

Making text slide across the browser windows

The example below styles `<p>` so that the text slide in from off the right edge of the browser window.

CSS code

```
p {  
    animation-duration: 4s;  
    animation-name: slideintext;  
}
```

```
@keyframes slideintext {  
    from {  
        margin-left: 100%;  
        width: 250%;  
    }  
  
    to {  
        margin-left: 0%;  
        width: 100%;  
    }  
}
```

In the above example, the style for `<p>` element specifies that the animation should take 4 seconds to execute from start to finish, using the `animation-duration` property, and that the name of the `@keyframes` at-rule defining the keyframes for the animation sequence is named "slideintext".

The keyframes are defined using the `@keyframes` at-rule. In this case, we have just two keyframes. The first occurs at 0% (using the alias `from`). Here, we configure the left margin of the element to be at 100% (that is, at the far right edge of the containing element), and the width of the element to be 250% (or 2.5 times the width of the containing element). This causes the first frame of the animation to have the header drawn off the right edge of the browser window.

The second (and final) keyframe occurs at 100% (using the alias `to`). The left margin is set to 0% and the width of the element is set to 100%. This causes the header to finish its animation flush against the left edge of the content area.

Adding another frame

If we want the header's font size to increase as it moves from right to left for a while, then to decrease back to its original size, it is as simple as adding another keyframe as shown below:

```
75% {  
    font-size: 300%;  
    margin-left: 25%;  
    width: 150%;  
}
```

At the end of the day, we have the full code as shown below;

```
p {  
    animation-duration: 3s;  
    animation-name: slidein;  
}  
  
@keyframes slidein {  
from {  
    margin-left: 100%;  
    width: 300%;  
}  
  
75% {  
    font-size: 300%;  
    margin-left: 25%;  
    width: 150%;  
}  
  
to {  
    margin-left: 0%;  
    width: 100%;  
}
```

This tells the browser that 75% of the way through the animation sequence, the header should have its left margin at 25% and the width should be 150%.

Making CSS Animation repeats

To make the animation repeat itself, use the animation-iteration-count property to indicate how many times to repeat the animation. In this case, let's use infinite to have the animation repeat indefinitely:

```
p {  
    animation-duration: 3s;  
    animation-name: slidein;  
    animation-iteration-count: infinite;  
}
```

Making CSS Animation move back and forth

The code below move the animation back and forth across the screen

```
p {  
    animation-duration: 3s;  
    animation-name: slidein;  
    animation-iteration-count: infinite;  
    animation-direction: alternate;  
}
```

Setting multiple animation property values

The CSS animation longhand values can accept multiple values, separated by commas. This feature can be used when applying multiple animations in a single rule, and set separate durations, iteration counts, etc. for the different animations as shown in the example below;

```
p{  
    animation-name: fadeInOut, moveLeft300px, bounce;  
    animation-duration: 2.5s, 5s, 1s;  
    animation-iteration-count: 2, 1, 5;  
}
```

In the example above, we have three values set on all three properties. In this case each animation is run with the corresponding values in the same position on each property, so for example fadeInOut has a duration of 2.5s and an iteration count of 2, etc.

CSS Buttons

Button adds clickable visibility to a website. In this section, you will learn how to style buttons to enhance it visibility in the examples below;

Examples

CSS Code

```
.button {  
    background-color: #4CAF50; /* Green */  
    border: none;  
    color: white;  
    padding: 16px 32px;  
    text-align: center;  
    text-decoration: none;  
    display: inline-block;  
    font-size: 16px;  
    margin: 4px 2px;  
    transition-duration: 0.4s;  
    cursor: pointer;  
}
```

```
.button1 {  
background-color: white;  
color: black;  
border: 2px solid #4CAF50;  
}  
  
.button1:hover {  
background-color: #4CAF50;  
color: white;  
}  
  
.button2 {  
background-color: white;  
color: black;  
border: 2px solid #008CBA;  
}  
  
.button2:hover {  
background-color: #008CBA;  
color: white;  
}  
  
.button3 {  
background-color: white;  
color: black;  
border: 2px solid #f44336;  
}  
  
.button3:hover {  
background-color: #f44336;  
color: white;  
}  
  
.button4 {  
background-color: white;  
color: black;  
border: 2px solid #e7e7e7;  
}  
  
.button4:hover {background-color: #e7e7e7;}  
  
.button5 {  
background-color: white;  
color: black;  
border: 2px solid #555555;  
}
```

```
.button5:hover {  
    background-color: #555555;  
    color: white;  
}
```

HTML Code

```
<h2>Hoverable Buttons</h2>  
<p>Use the :hover selector to change the style of the button when you move the mouse over it.</p>  
<p><strong>Tip:</strong> Use the transition-duration property to determine the speed of the "hover" effect:</p>  
<button class="button button1">Green</button>  
<button class="button button2">Blue</button>  
<button class="button button3">Red</button>  
<button class="button button4">Gray</button>  
<button class="button button5">Black</button>
```

Result



More Examples on buttons

CSS Code

```
button {  
    font-family: inherit;  
    font-size: 20px;  
    background: royalblue;  
    color: white;  
    padding: 0.7em 1em;  
    padding-left: 0.9em;  
    display: flex;  
    align-items: center;  
    border: none;  
    border-radius: 16px;  
    overflow: hidden;  
    transition: all 0.2s;  
}  
  
button span {  
    display: block;  
    margin-left: 0.3em;  
    transition: all 0.3s ease-in-out;  
}
```

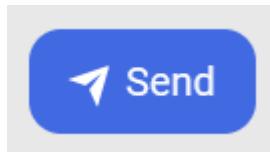
```
button svg {  
    display: block;  
    transform-origin: center center;  
    transition: transform 0.3s ease-in-out;  
}  
  
button:hover .svg-wrapper {  
    animation: fly-1 0.6s ease-in-out infinite alternate;  
}  
  
button:hover svg {  
    transform: translateX(1.2em) rotate(45deg) scale(1.1);  
}  
  
button:hover span {  
    transform: translateX(5em);  
}  
  
button:active {  
    transform: scale(0.95);  
}  
  
@keyframes fly-1 {  
from {  
    transform: translateY(0.1em);  
}  
to {  
    transform: translateY(-0.1em);  
}  
}
```

HTML Code

```
<button>  
    <div class="svg-wrapper-1">  
        <div class="svg-wrapper">  
            <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24"  
width="24" height="24">  
                <path fill="none" d="M0 0h24v24H0z"></path>  
                <path fill="currentColor" d="M1.946 9.315c-.522-.174-.527-.455.01-.634l19.087-6.362c.529-.176.832.12.684.638l-5.454 19.086c-.15.529-.455.547-.679.045L12 14l6-8-8 6-8.054-2.685z"></path>  
            </svg>  
        </div>  
    </div>  
    <span>Send</span>
```

```
</button>
```

Results



CSS Pagination Examples

If you have a website with lots of pages, you may wish to add some sort of pagination to each page; this is process of dividing the document into pages and providing them with numbers. The example below illustrates simple pagination with pages number align on the center.

CSS Code

```
.center {  
    text-align: center;  
}  
  
.pagination {  
    display: inline-block;  
}  
  
.pagination a {  
    color: black;  
    float: left;  
    padding: 8px 16px;  
    text-decoration: none;  
    transition: background-color .3s;  
    border: 1px solid #ddd;  
    margin: 0 4px;  
}  
  
.pagination a.active {  
    background-color: #4CAF50;  
    color: white;  
    border: 1px solid #4CAF50;  
}  
.pagination a:hover:not(.active) {  
    background-color: #ddd;  
}
```

HTML Code

<h2>Centered Pagination</h2>

```
<div class="center">
  <div class="pagination">
    <a href="#">&laquo;</a>
    <a href="#">1</a>
    <a href="#" class="active">2</a>
    <a href="#">3</a>
    <a href="#">4</a>
    <a href="#">5</a>
    <a href="#">6</a>
    <a href="#">&raquo;</a>
  </div>
</div>
```

Multiple-column layouts

CSS3 introduces a few new modules that make multi-column layouts easier to create. The Multicolumn Layout module deals with flowing text of a single block into multiple columns, similar to newspaper layout.

CSS Multi-column Properties

column-count: this property specifies the number of columns an element should be divided into. The following example will divide the text in the <div> element into 3 columns:

```
div {
  column-count: 3;
}
```

column-gap: This property specifies the gap between the columns. The following example specifies a 10 pixels gap between the columns:

```
div {
  column-gap: 10px;
}
```

column-rule-style: this property specifies the style of the rule between columns:

```
div {
  column-rule-style: solid;
}
```

column-rule-width: this property specifies the width of the rule between columns:

```
div {
  column-rule-width: 2px;
}
```

column-rule-color: this property specifies the color of the rule between columns as it is shown below:

```
div {  
  column-rule-color: lightblue;  
}
```

column-rule: The column-rule property is a shorthand property for setting all the column-rule properties as it is shown above. The example below sets the width, style, and color of the rule between columns:

```
div {  
  column-rule: 2px dash gray;  
}
```

column-span: this property specifies how many columns an element should span across. The following example specifies that the <h1> element should span across all columns:

```
h1 {  
  column-span: all;  
}
```

column-width: this property specifies a suggested, optimal width for the columns. The example below specifies that the suggested, optimal width for the columns should be 50px:

```
div {  
  column-width: 50px;  
}
```

Example:

CSS Code

```
.news {  
  column-count: 3;  
  column-gap: 10px;  
  column-rule-style: solid;  
  column-rule-color:#333;  
  column-width: 20px;
```

```
}
```

```
h1 {
```

```
  column-span: all;
```

```
}
```

HTML Code

```
<p class="news">
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.....

</p>

Result

Add a Rule Between the Columns

 Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit

lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit

praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum.

CSS Variables - The var() Function

The var() function is used to insert the value of a CSS variable. A good way to use CSS variables is when it comes to the colors of your design. Instead of copy and paste the same colors over and over again, you can place them in variables.

Syntax of the var() Function

The var() function is used to insert the value of a CSS variable. The syntax of the var() function is as follows:

var(--name, value)

- ⇒ **name** The variable name (must start with two dashes) is required
- ⇒ **value** The fallback value (used if the variable is not found) is optional

Note: The variable name must begin with two dashes (--) and it is case sensitive!

CSS variable scopes

The two types of CSS variable scopes are the global and the local variable. The global variable can be used in the entire page when it is defined while the local variable can only be used inside the selected where it is declared.

Advantages of using var() are:

- makes the code easier to read (more understandable)
- makes it much easier to change the color values

To create a variable with global scope, declare it inside the :root selector. The `:root` selector matches the document's root element as shown in the example below;

```
:root {  
  --blue: #1e90ff;  
  --white: #ffffff;  
}  
  
body {  
  background-color: var(--blue);  
}  
  
h2 {  
  border-bottom: 2px solid var(--blue);  
}  
  
.container {  
  color: var(--blue);  
  background-color: var(--white);  
  padding: 15px;  
}  
  
button {  
  background-color: var(--white);  
  color: var(--blue);  
  border: 1px solid var(--blue);  
  padding: 5px;  
}
```

⇒ Grid layout

Grid CSS module defines a two-dimensional grid based layout system, optimised for user interface design, which offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.. In the grid layout model, the children of a grid container can be positioned into arbitrary slots in a flexible or fixed predefined layout grid.

Display/Define Grid Property

An HTML element becomes a grid container when its display property is set to grid or inline-grid. To define a Grid use `display:grid` or `display:inline-grid` on the parent element. You can then create a grid using the `grid-template-columns` and `grid-template-rows` properties.

Examples

CSS Code

```
.main{  
    display: grid;  
    grid-template-columns: 25% 25% 25%;  
    grid-gap: 10px;  
    background-color: #fff;  
    color: #444;  
}  
  
.box{  
    background-color: rgb(7, 32, 24);  
    color: #fff;  
    border-radius: 5px;  
    padding: 20px;  
    font-size: 100%;  
    text-align: center;  
}
```

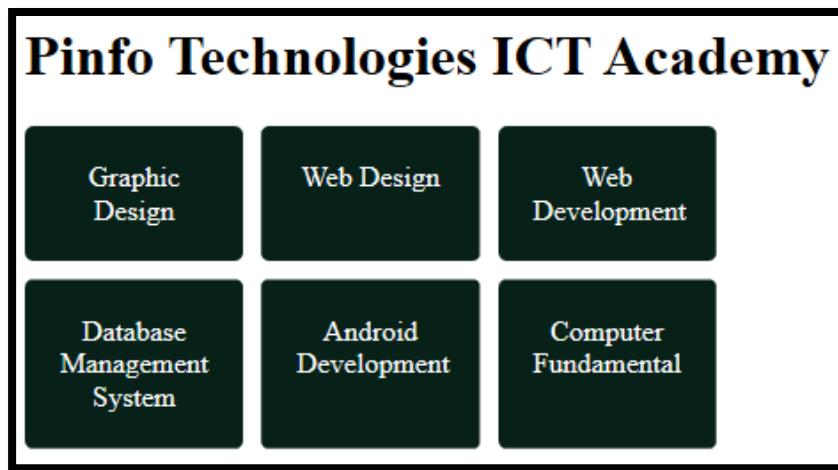
}

HTML Code

```
<div class="main">  
    <div class="box ">Graphic Design</div>  
    <div class="box ">Web Design</div>  
    <div class="box ">Web Development</div>  
    <div class="box ">Database Management System</div>
```

```
<div class="box ">Android Development</div>  
<div class="box ">Computer Fundamental</div>  
</div>
```

Result



Grid Elements

A grid layout consists of a parent element, with one or more child elements.

EXAMPLE

CSS Code

```
.grid-container {  
    display: grid;  
    grid-template-columns: auto auto auto;  
    background-color: #2196F3;  
    padding: 10px;  
}  
  
.grid-item {
```

```
background-color: rgba(255, 255, 255, 0.8);  
border: 1px solid rgba(0, 0, 0, 0.8);  
padding: 20px;  
font-size: 30px;  
text-align: center;  
}  
  
</style>
```

HTML Code

```
<h1>Grid Elements</h1>  
  
<p>A Grid Layout must have a parent element with the <em>display</em> property set to <em>grid</em> or <em>inline-grid</em>. </p>  
  
<p>Direct child element(s) of the grid container automatically becomes grid items.</p>  
  
<div class="grid-container">  
  <div class="grid-item">1</div>  
  <div class="grid-item">2</div>  
  <div class="grid-item">3</div>  
  <div class="grid-item">4</div>  
  <div class="grid-item">5</div>  
  <div class="grid-item">6</div>  
  <div class="grid-item">7</div>  
  <div class="grid-item">8</div>  
  <div class="grid-item">9</div>  
</div>
```

Grid technology

Grid container: This establishes a new grid formatting context for its contents. It is great for dividing up the major regions of a page into smaller sections or setting the relationship between elements in terms of size and position

Grid Columns: The vertical lines of grid items are called columns.

Grid Rows: The horizontal lines of grid items are called rows.

Grid lines: These are horizontal or vertical lines between grid cells. They can be named or referred by numbers.

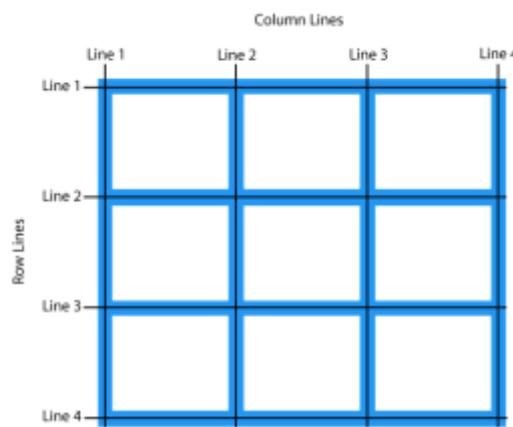
Grid cell: It is the space between two adjacent row and two adjacent column grid lines. It is the smallest unit of the grid that can be referenced when positioning grid items.

Grid items: These are the direct children of the grid container. They are arranged vertically into columns and horizontally into rows.

The space between a row and a column is called a **gap**. You can adjust the gap using these properties:

- **grid-column-gap**, which defines the gap between columns;
- **grid-row-gap**, which defines the gap between rows; or
- **grid-gap**, which is a shorthand property for grid-column-gap and grid-row-gap.

In between columns and rows, there are lines referred to as column lines and row lines, respectively as shown below;



When positioning a grid item inside a container, you reference the line numbers.

The **grid-template-columns** Property

The **grid-template-columns** property defines the number of columns in your grid layout, as well as the width of each column. The value is a space-separated-list, where each value defines the width of the respective column.

```
.grid-container {  
    display: grid;  
    grid-template-columns: auto auto auto auto;  
}
```

If you want your grid layout to contain 4 columns, specify the width of the 4 columns, or "auto" if all columns should have the same width as shown above.

The **grid-template-rows** Property

The **grid-template-rows** property defines the height of each row. The value is a space-separated-list, where each value defines the height of the respective row as shown below;

```
.grid-container {  
    display: grid;
```

```
    grid-template-rows: 40px 100px;  
}
```

The align-content Property

The align-content property is used to *vertically* align the whole grid inside the container. The grid's total height has to be less than the container's height for the align-content property to have any effect. Example;

```
.grid-container {  
  display: grid;  
  height: 200px;  
  align-content: center;  
}
```

Naming Grid Items

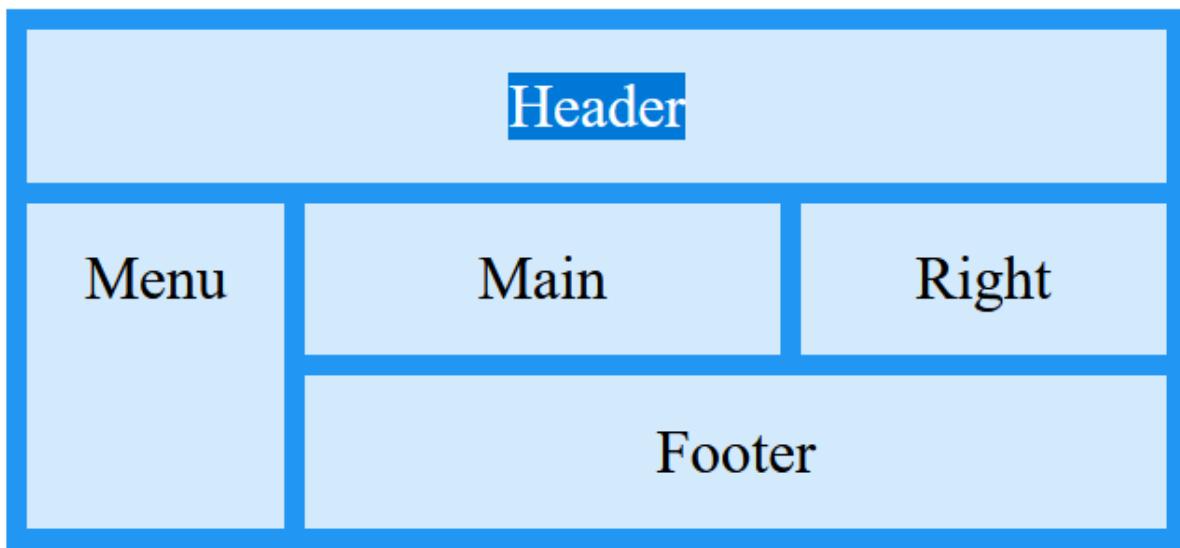
The grid-area property can also be used to assign names to grid items. In naming grip-area, each row is defined by apostrophes (' ') while the columns in each row is defined inside the apostrophes, separated by a space as it is shown in the example below;

```
.item1 { grid-area: header; }  
.item2 { grid-area: menu; }  
.item3 { grid-area: main; }  
.item4 { grid-area: right; }  
.item5 { grid-area: footer; }  
  
.grid-container {  
  display: grid;  
  grid-template-areas:  
    'header header header header header header'  
    'menu main main main right right'  
    'menu footer footer footer footer footer';  
  gap: 10px;  
  background-color: #2196F3;  
  padding: 10px;  
}  
  
.grid-container > div {  
  background-color: rgba(255, 255, 255, 0.8);  
  text-align: center;  
  padding: 20px 0;  
  font-size: 30px;  
}
```

HTMLCode

```
<h1>The grid-area Property</h1>
<p>You can use the <em>grid-area</em> property to name grid items.</p>
<p>You can refer to the name when you set up the grid layout, by using the
<em>grid-template-areas</em> property on the grid container.</p>
<p>This grid layout contains six columns and three rows:</p>
<div class="grid-container">
  <div class="item1">Header</div>
  <div class="item2">Menu</div>
  <div class="item3">Main</div>
  <div class="item4">Right</div>
  <div class="item5">Footer</div>
</div>
```

Result:



How to Center a Grid in CSS

To start centering your grid items, use the **align-items** property. Some of the values that you can assign to this property are **start**, **end**, and **stretch**.

When the value of the **align-items** property is set to **start**, this places the items to the vertical start of their corresponding grid area while setting it to **end** places them at the end. When the value is **stretch**, the items are spread out across the grid area.

The **justify-items** property works similarly to the **align-items** property and accepts the same values on the inline axis.

To target just one item so you can center it, use the **align-self** and **justify-self** properties, respectively, at the item level.

Example

How to Make a CSS Grid Responsive

To build a responsive Grid, first set the **box-sizing** property of all HTML elements to **border-box**. This includes the padding and border to the width and height of the elements.

Add the code below to your CSS code for grid responsive design

```
* {  
    box-sizing: border-box;  
}
```

In grid layout, each row is wrapped in a <div> and the number of columns inside a row always adds up to 12;

Using responsive grid-view with 12 columns

First we must calculate the percentage for one column: $100\% / 12 \text{ columns} = 8.33\%$. Then we make one class for each of the 12 columns, class="col-" and a number defining how many columns the section should span as it is shown below;

```
.col-1 {width: 8.33%;}  
.col-2 {width: 16.66%;}  
.col-3 {width: 25%;}  
.col-4 {width: 33.33%;}  
.col-5 {width: 41.66%;}  
.col-6 {width: 50%;}  
.col-7 {width: 58.33%;}  
.col-8 {width: 66.66%;}  
.col-9 {width: 75%;}  
.col-10 {width: 83.33%;}  
.col-11 {width: 91.66%;}  
.col-12 {width: 100%;}
```

Example

CSS Code

```
* {
  box-sizing: border-box;
}

.row::after {
  content: "";
  clear: both;
  display: table;
}

[class*="col-"] {
  float: left;
  padding: 15px;
}

.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}

html {
  font-family: "Lucida Sans", sans-serif;
}

.header {
  background-color: #9933cc;
  color: #ffffff;
  padding: 15px;
}

.menu ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}

.menu li {
  padding: 8px;
  margin-bottom: 7px;
```

```
background-color: #33b5e5;
color: #ffffff;
box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}
.menu li:hover {
background-color: #0099cc;
}
```

Html Code

```
<div class="header">
<h1>Chania</h1>
</div>

<div class="row">
<div class="col-3 menu">
<ul>
<li>The Flight</li>
<li>The City</li>
<li>The Island</li>
<li>The Food</li>
</ul>
</div>

<div class="col-9">
<h1>The City</h1>
<p>Chania is the capital of the Chania region on the island of Crete. The city can be divided in two parts, the old town and the modern city.</p>
<p>Resize the browser window to see how the content respond to the resizing.</p>
</div>
</div>
```

In the example above, all these columns should be floating to the left, and have a padding of 15px:

```
[class*="col-"] {
float: left;
padding: 15px;
border: 1px solid red;
}
```

Each row should be wrapped in a <div>. The number of columns inside a row should always add up to 12:

```
<div class="row">
<div class="col-3">...</div> <!-- 25% -->
```

```
<div class="col-9">...</div> <!-- 75% -->  
</div>
```

The columns inside a row are all floating to the left, and are therefore taken out of the flow of the page, and other elements will be placed as if the columns do not exist. To prevent this, we will add a style that clears the flow:

```
.row::after {  
    content: "";  
    clear: both;  
    display: table;  
}
```

We also want to add some styles and colors to make it look better:

```
html {  
    font-family: "Lucida Sans", sans-serif;  
}  
.header {  
    background-color: #9933cc;  
    color: #ffffff;  
    padding: 15px;  
}  
.menu ul {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
}  
.menu li {  
    padding: 8px;  
    margin-bottom: 7px;  
    background-color :#33b5e5;  
    color: #ffffff;  
    box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);  
}  
.menu li:hover {  
    background-color: #0099cc;  
}
```

⇒ Flexbox Layout

The Flexible Box Layout Module makes it easier to design flexible responsive layout structure without using float or positioning in CSS. Any child elements that reside within the flex container are called flex items, arranged in a one-dimensional layout in rows or columns which

expands to fill additional space or shrink to fit smaller spaces. The flex-items(contents) are distributed along the Main Axis and Cross Axis.

The diagram below shows flex box chart

| Flex Box Chart | |
|-----------------------|---|
| Property | Value(s) |
| #1 display | flex |
| #2 flex-direction | row column column-reverse row-reverse |
| #3 justify-content | flex-start flex-end center space-between space-around space-evenly |
| #4 align-items | flex-start flex-end center stretch baseline |
| #5 align-content | flex-start flex-end center stretch space-between space-around |
| #6 align-self | auto flex-start flex-end center baseline stretch |
| #7 Order | /* Any positive Value */ |
| #8 flex-grow | /* Any positive Value */ |
| #9 flex-shrink | /* Any positive Value */ |
| #10 flex-wrap | nowrap wrap wrap-reverse |

Terminologies

The flex container: this is an area of a document laid out using flexbox. To start using flex, you first of all define the flex container; to start using flex, you first of all define the flex container display property to **flex** or **inline-flex**. As soon as this property is set, all the direct children of that container become **flex items**.

The following are initial values defined when creating a flex container;

- Items display in a row (the flex-direction property's default is row).
- The items start from the start edge of the main axis.
- The items do not stretch on the main dimension, but can shrink.
- The items will stretch to fill the size of the cross axis.
- The flex-basis property is set to auto.
- The flex-wrap property is set to nowrap.

```
.flex-container {  
    display: flex;  
    background-color: orange;  
}
```

Example

```
.flex-container {  
    display: flex;  
    background-color: orange;  
}
```

```
.flex-container > div {  
    background-color: #f1f;  
    margin: 15px;  
    padding: 25px;  
    font-size: 40px;  
}
```

Html

```
<div class='flex-container'>  
    <div>A</div>  
    <div>B</div>  
    <div>C</div>  
</div>
```

Result;

⇒ **flex Property**

The flex CSS property specifies how a flex item will grow or shrink so as to fit within the space available in its flex container. This is a shorthand property that declares the following properties in order on a single line:

- flex-grow
- flex-shrink
- flex-basis

```
/* Three properties declared on three lines: */
```

```
.first-flex-item {  
    flex-grow: 2;  
    flex-shrink: 1;  
    flex-basis: 150px;  
}
```

```
/* Same three properties declared on one line: */
```

```
.first-flex-item {  
    flex: 2 1 150px;  
}
```

⇒ **flex-shrink Property**

The CSS flex-shrink property determines how an element should shrink as the parent container decreases in size horizontally. This property accepts a numerical value which specifies the ratios for the shrinkage of a flex item compared to its other sibling elements within its parent container.

The default value for this property is 1.

```
.container {  
  display: flex;  
}  
  
.item-a {  
  flex-shrink: 1;  
  /* The value 1 indicates that the item should shrink. */  
}  
  
.item-b {  
  flex-shrink: 2;  
  /* The value 2 indicates that the item should shrink twice than the element item-a.  
 */  
}
```

Css flex-basis property

The flex-basis CSS property sets the initial base size for a flex item before any other space is distributed according to other flex properties.

```
// Default Syntax  
flex-basis: auto;
```

The flex-direction CSS property

This specifies how flex items are placed in the flex container - either vertically or horizontally. This property also determines whether those flex items appear in order or in reverse order as shown below;

```
.flex-container {  
  display: flex;  
  flex-direction: column;  
}
```

CSS Code

```
flex-container {  
  display: flex;  
  flex-direction: column;  
  background-color: DodgerBlue;
```

}

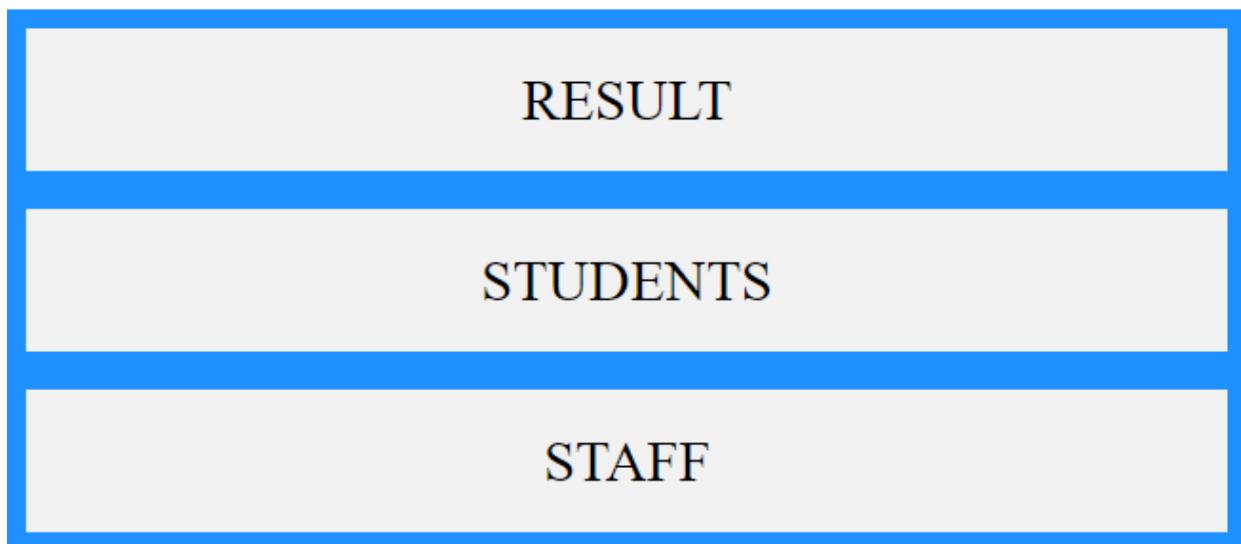
```
.flex-container > div {  
background-color: #f1f1f1;  
width: auto;  
margin: 10px;  
text-align: center;  
line-height: 75px;  
font-size: 30px;  
}
```

HTML Code

```
<div class="flex-container">  
<div>RESULT</div>  
<div>STUDENTS</div>  
<div>STAFF</div>  
</div>
```

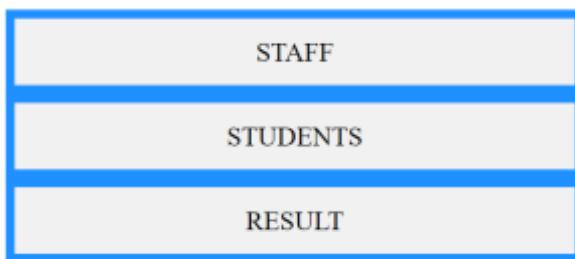
Result

The "flex-direction: column;" stacks the flex items vertically (from top to bottom):



The *column-reverse* value stacks the flex items vertically (but from bottom to top) as shown below;

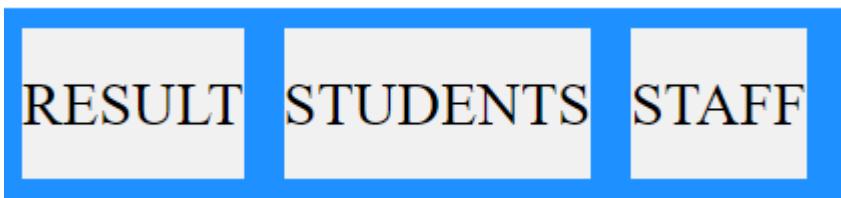
```
.flex-container {  
display: flex;  
flex-direction: column-reverse;  
background-color: DodgerBlue;  
}
```



The *row* value stacks the flex items horizontally (from left to right):

Example

```
.flex-container {  
    display: flex;  
    flex-direction: row;  
    background-color: DodgerBlue;  
}
```



The *row-reverse* value stacks the flex items horizontally (but from right to left):

```
.flex-container {  
    display: flex;  
    flex-direction: row-reverse;  
    background-color: orange;  
}
```



The wrap property: though flexbox is a one dimensional model, it is possible to cause flex items to wrap onto multiple lines by adding the flex-wrap property as shown below;

```
.flex-container {  
    display: flex;  
    flex-wrap: wrap;  
}
```

Example

CSS Code

```
.flex-container {  
    display: flex;
```

```
flex-wrap: wrap;  
background-color: DodgerBlue;  
}
```

```
.flex-container > div {  
background-color: #f1f1f1;  
width: 100px;  
margin: 10px;  
text-align: center;  
line-height: 75px;  
font-size: 30px;  
}
```

HTML

```
<h1>The flex-wrap Property</h1>
```

<p>The "flex-wrap: wrap;" specifies that the flex items will wrap if necessary:</p>

```
<div class="flex-container">  
<div>1</div>  
<div>2</div>  
<div>3</div>  
<div>4</div>  
<div>5</div>  
<div>6</div>  
<div>7</div>  
<div>8</div>  
<div>9</div>  
<div>10</div>  
<div>11</div>  
<div>12</div>  
<div>13</div>  
<div>14</div>  
<div>15</div>  
  
</div>
```

<p>resizing the browser window adjust to the device screen resolution. </p>

Result

The flex-wrap Property

The "flex-wrap: wrap;" specifies that the flex items will wrap if necessary:

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

resizing the browser window adjust to the device screen resolution.

The *nowrap* value specifies that the flex items will not wrap (this is default):

```
.flex-container {  
    display: flex;  
    flex-wrap: nowrap;  
}
```

The *wrap-reverse* value specifies that the flexible items will wrap if necessary, in reverse order:

```
.flex-container {  
    display: flex;  
    flex-wrap: wrap-reverse;  
}
```

flex-grow Property

The CSS flex-grow property allows flex items to grow as the parent container increases in size horizontally. This property accepts numerical values and specifies how an element should grow relative to its sibling elements based on this value.

The default value for this property is 0.

```
.boxA {  
    width: 100px;  
    flex-grow: 1;  
}  
  
/* This boxB element will stretch twice wider than the boxA element */  
.boxB {  
    width: 100px;  
    flex-grow: 2;  
}
```

The flex-flow Property

You can combine the two properties flex-direction and flex-wrap into the flex-flow shorthand. The first value specified is flex-direction and the second value is flex-wrap as shown in the examples below;

```
.flex-container {  
    display: flex;  
    flex-flow: row wrap;  
}
```

The justify-content property

The CSS justify-content flexbox property defines how the browser distributes space between and around content items along the main-axis of their container. This is when the content items do not use all available space on the major-axis (horizontally). justify-content can have the values of: flex-start, flex-end, center, space-between and space-around.

- ⇒ The *center* value aligns the flex items at the center of the container:

```
.flex-container {  
    display: flex;  
    justify-content: center;  
}
```

- ⇒ The *flex-start* value aligns the flex items at the beginning of the container (this is default):

```
.flex-container {  
    display: flex;  
    justify-content: flex-start;  
}
```

- ⇒ The *flex-end* value aligns the flex items at the end of the container:

```
.flex-container {  
    display: flex;  
    justify-content: flex-end;  
}
```

- ⇒ The *space-around* value displays the flex items with space before, between, and after the lines:

```
.flex-container {  
    display: flex;  
    justify-content: space-around;  
}
```

- ⇒ The *space-between* value displays the flex items with space between the lines:

```
.flex-container {  
    display: flex;  
    justify-content: space-between;  
}
```

The align-items Property

The `align-items` property is used to align the flex items vertically.

- ⇒ The *center* value aligns the flex items in the middle of the container:

```
.flex-container {  
    display: flex;  
    height: 200px;  
    align-items: center;  
}
```

- ⇒ The *flex-start* value aligns the flex items at the top of the container:

```
.flex-container {  
    display: flex;  
    height: 200px;  
    align-items: flex-start;  
}
```

- ⇒ The *flex-end* value aligns the flex items at the bottom of the container:

```
.flex-container {  
    display: flex;  
    height: 200px;  
    align-items: flex-end;  
}
```

- ⇒ The *stretch* value stretches the flex items to fill the container (this is default):

```
.flex-container {  
    display: flex;  
    height: 200px;  
    align-items: stretch;  
}
```

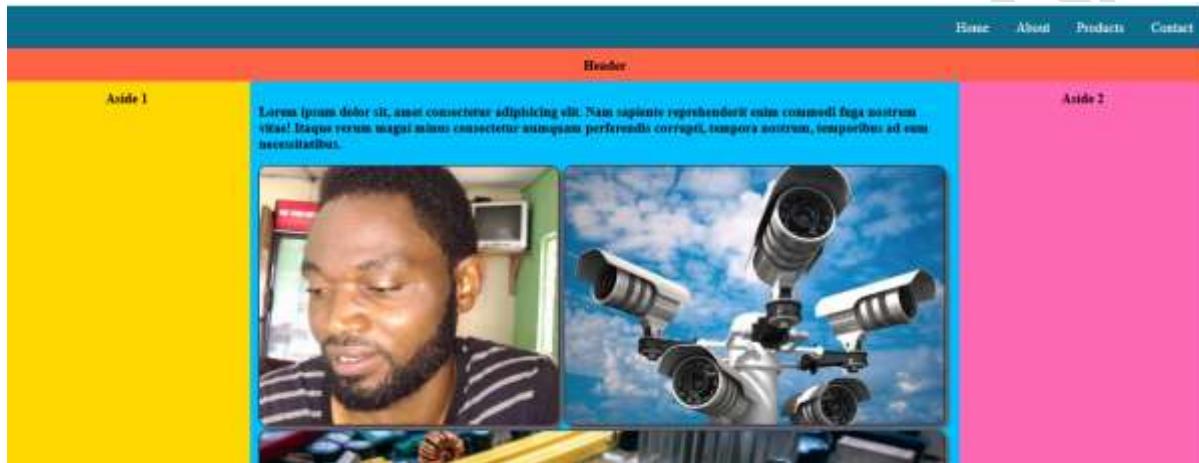
- ⇒ The *baseline* value aligns the flex items such as their baselines aligns:

```
.flex-container {  
    display: flex;
```

```
height: 200px;  
align-items: baseline;  
}
```

More Examples on creating CSS Layout with flex box property

We want to design the CSS layout using Flex box module as shown below'



CSS Code

```
<style type="text/css">  
.flex-container {  
/* We first create a flex layout context */  
display: flex;  
  
/* Then we define the flow direction  
and if we allow the items to wrap  
* Remember this is the same as:  
* flex-direction: row;  
* flex-wrap: wrap;  
*/  
flex-flow: row wrap;  
/* Then we define how is distributed the remaining space */  
justify-content: space-around;  
padding: 0;  
margin: 0;  
list-style: none;  
}  
.flex-item img{  
background: tomato;
```

```
padding: 5px;
width: 200px;
height: 150px;
margin-top: 10px;
line-height: 150px;
color: orange;
font-weight: bold;
font-size: 3em;
text-align: center;
}
.nav{

display: flex;
flex-flow: row wrap;
justify-content: flex-end;
list-style: none;
margin: 0;
background: rgb(12, 109, 141);
}
.nav a{

text-decoration: none;
display: block;
padding: 1em;
color: white;
}
.nav a:hover{
background:#dd7d10;
}

@media all and (max-width: 800px){

.nav {
justify-content: space-around;
}
}

@media all and (max-width: 600px){

.nav{
flex-flow:column wrap;
padding: 0;
}
.nav a{
text-align: center;
padding:10px;
border-top: 1px solid rgba(255, 255, 255, 0.3);
border-bottom: 1px solid rgba(0, 0, 0, 0.1);
}
}
```



```
.nav li:last-of-type a{  
    border-bottom:none;  
}  
}  
.wrapper {  
    display:flex;  
    flex-flow: row wrap;  
    font-weight: bold;  
    text-align: center;  
}  
  
.wrapper > * {  
    padding: 10px;  
    flex: 1 100%;  
}  
  
.header {  
    background: tomato;  
}  
  
.footer {  
    background: lightgreen;  
}  
  
.main {  
    text-align: left;  
    background: deepskyblue;  
}  
  
.aside-1 {  
    background: gold;  
}  
}  
.aside-2 {  
    background: hotpink;  
}  
}  
  
@media all and (min-width: 600px) {  
    .aside {flex: 1 0 0; }  
}  
}  
  
@media all and (min-width: 800px) {  
    .main {flex: 3 0px; }  
    .aside-1 {order: 1; }  
    .main {order: 2; }  
}
```

```
.aside-2 { order: 3; }
.footer { order: 4; }
}

body {
    padding: 2em;
}

ul#contactsheet {
    display: flex;
    flex-wrap: wrap;
    padding: 0;
}
ul#contactsheet:after {
    content: "";
    display: block;
    flex-grow: 10; /* Prevent justify/stretch of last photo. 10 seems best */
}
#contactsheet li {
    height: 40vh;
    flex-grow: 1;
    margin: 0 0.5rem 0.5rem 0;
    list-style: none; /* override common/Normalize */
}
#contactsheet img {
    max-height: 100%;
    min-width: 100%;
    object-fit: cover;
    vertical-align: bottom;
    border: 2px solid #600;
    border-radius: 0.6rem;
    box-shadow: 0.3rem 0.3rem 0.3rem #655;
}
#contactsheet img:hover {
    border-color: #c60;
    box-shadow: 0.3rem 0.3rem 0.3rem #974;
}

/* Portrait: */
@media (max-aspect-ratio: 1 / 1) {
    #contactsheet li {
        height: 30vh;
    }
}
```

```
/* Short screens: */
@media (max-height: 480px) {
  #contactsheet li {
    height: 80vh;
  }
}

/* Smaller screens in portrait: */
@media (max-aspect-ratio: 1 / 1) and (max-width: 480px) {
  ul#contactsheet {
    flex-direction: row;
  }

  #contactsheet li {
    height: auto;
    width: 100%;
  }

  #contactsheet img {
    width: 100%;
    max-height: 75vh;
    min-width: 0;
  }
}
</style>
```

HTML Code

```
<body>

<ul class="nav">
  <li><a href="#">Home</a></li>
  <li><a href="#">About</a></li>
  <li><a href="#">Products</a></li>
  <li><a href="#">Contact</a></li>
</ul>
<div class="wrapper">
  <header class="header">Header</header>
  <article class="main">
    <p>
        Lorem ipsum dolor sit, amet consectetur adipisicing elit. Nam sapiente reprehenderit enim commodi fuga nostrum vitae! Itaque rerum magni minus consectetur numquam perferendis corrupti, tempora nostrum, temporibus ad eum necessitatibus.
    </p>
    <ul class="flex-container">
```

```
<ul id="contactsheet">
  <li>
    <a href="https://areabeyond.com"></a>
  </li>
  <li>
    <a href="https://areabeyond.com"></a>
  </li>
  <li>
    
  </li>
  <li>
    
  </li>
  <li>
    
  </li>
  <li>
    
  </li>
  <li>
    
  </li>
  <li>
    
  </li>

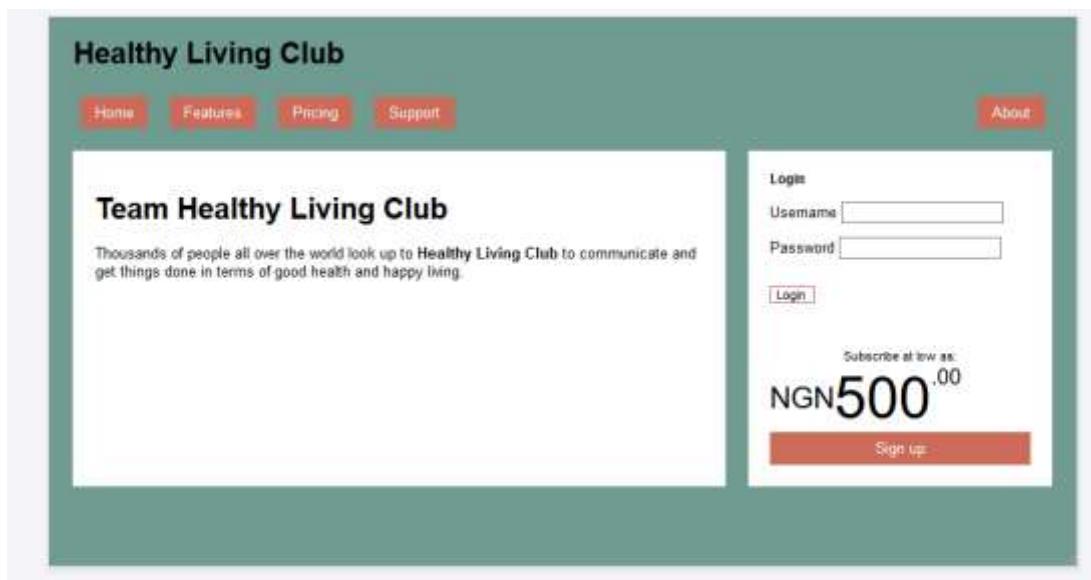
</ul>

</article>
<aside class="aside aside-1">Aside 1</aside>
<aside class="aside aside-2">Aside 2</aside>
<footer class="footer">Footer</footer>
</div>

</body>
```

The example below create 2-columns layout using CSS Flexbox Modules

The Result



```
<!doctype html>
<head>
<title>Flexbox Page</title>
<link href="style.css" rel="stylesheet"
type="text/css" />
</head>
<body>
<div class="container">
<header>
<h1>Healthy Living Club</h1>
</header>
<nav>
<ul class="site-nav">
<li><a href="/">Home</a></li>
<li><a href="#">Features</a></li>
<li><a href="#">Pricing</a></li>
<li><a href="#">Support</a></li>
<li class="nav-right">
<a href="#">About</a>
</li>
</ul>
</nav>
<main class="flex">
<div class="column-main tile">
<h1>Team Healthy Living Club</h1>
<p>Thousands of people all over the world look up to <b>Healthy Living Club</b> to communicate and get things done in terms of good health and happy living.</p>
</div>
<div class="column-sidebar">
```

```
<div class="tile">
<form class="login-form">
<h3>Login</h3>
<p>
<label for="username">Username</label>
<input id="username" type="text"
name="username"/>
</p>
<p>
<label for="password">Password</label>
<input id="password" type="password"
name="password"/>
</p>
<button type="submit">Login</button>
</form>
</div>
<div class="tile centered">
<small>Subscribe at low as:</small>
<div class="cost">
<span class="cost-currency">NGN</span>
<span class="cost-dollars">500</span>
<span class="cost-cents">.00</span>
</div>
<a class="cta-button" href="/pricing">
Sign up 15
</a>
</div> 2082375351
</div>
</main>
</div>
</body>
</html>
```

Css file

```
/* setting a global box-sizing fix */
:root{
box-sizing:border-box;
}
*, 
::before,
::after{
box-sizing:inherit;
}
body{
background-color:#709b90;
font-family: Helvetica,Arial, sans-serif; /* green background color and sans-serif font
for the page
```

```
}
```

```
body * + * {
```

```
margin-top:1.5em; // setting global margins
```

```
}
```

```
.container{ /* container class to center page content */
```

```
max-width: 1080px;
```

```
margin: 0 auto;
```

```
}
```

First output page



Next it to make the menu aligned horizontally by building a basic flex menu

Achieving this, the flex container should be the unordered list () and the list items which are its children

As shown below;

```
<nav>
```

```
  <ul class="site-nav">
```

```
    <li><a href="/">Home</a></li>
```

```
    <li><a href="#">Features</a></li>
```

```
    <li><a href="#">Pricing</a></li>
```

```
    <li><a href="#">Support</a></li>
```

```
    <li class="nav-right">
```

```
      <a href="#">About</a>
```

```
    </li>
```

```
  </ul>
```

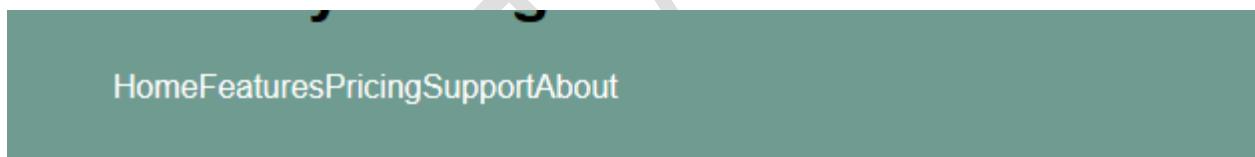
```
</nav>
```

Next we apply `display: flex` to the list,

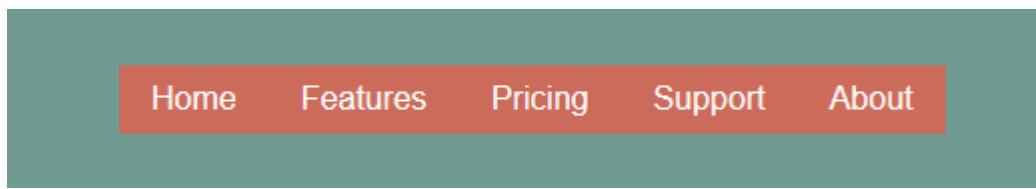
We then add the following styles

```
.site-nav{  
display:flex; /* this makes flex container and its children flex items */  
padding-left: 0; /* remove the default padding */  
list-style-type: none; /* remove the default list bullets in the user agent styles */  
background-color: #5f4b44;  
}  
  
.site-nav > li{  
margin-top:0; /* this overide the lobotomized owl top margin */  
}  
.site-nav > li > a{  
background-color: #cc6b5a;  
color: #fff;  
text-decoration: none; /* remove the defualt underline from the link in the user  
agent styles */  
}
```

This display the diagram below;



Adding padding and spacing to it



Adding spaces between the menu

With the style rule sets below;

```
.site-nav > li + li{ /* with the use of plus(+) operator, every list items that follow another list items are targeted */
  margin-left: 1.5em;
}

.site-nav > .nav-right{
  margin-left: auto; /* this push the About us link to the right of the menu with the use of auto margins inside a flexbox */
}
```



Flex item size

The flex property controls the size of the flex items along the main axis,

Now let us apply flexbox to the main container as follows;

```
.tile{
  padding:1.5em;
  background-color:#fff;
}

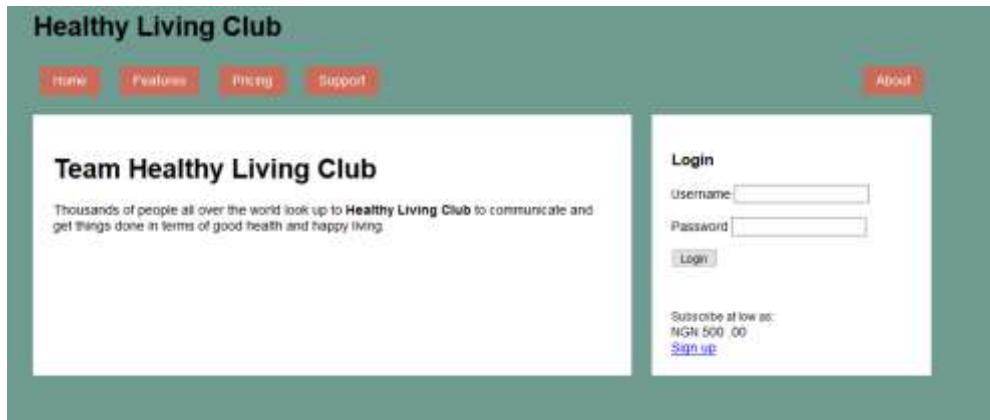
.flex{
  display: flex; /* flexbox layout is applied to the main container */
}

.flex > * + * { /* this remove the top margin and applies space between the flex items */
  margin-top: 0;
  margin-left: 1.5em;
}
```

Applying the flex property to set column widths

```
.column-main{
```

```
flex:2;  
}  
.column-sidebar{  
flex:1;  
}
```



Setting/changing the flex direction

This property is used to shift direction of flex items from left-to-right, from top to bottom. In the diagram above, two tiles are stacked on top one another, so we will use the flex direction to separate them.

Styling the login form

```
.login-form h3{  
margin:0;  
font-size: .9em;  
font-weight:bold;  
text-align:right;  
text-transform:uppercase;  
}  
.login-form button{  
margin-top: 1em;  
border: 1px solid #cc6b5a;  
background-color:white;  
cursor: pointer;  
}
```

Login

Username

Password

Using alignment property of flexbox and styling the button

```
.centered {  
    text-align: center;  
}  
  
.cost{  
    display: flex;  
    justify-content: center;  
    align-items: center;  
}  
  
.cost > span {  
    margin-top: 0;  
}  
  
.cost-currency{  
    font-size: 2rem;  
}  
  
.cost-naira{  
    font-size: 4rem;  
}  
  
.cost-kobos{  
    font-size: 1.5rem;  
    align-self: flex-start;  
}  
  
.cta-button{  
    display: block;  
    background-color: #cc6b5a;  
    color: #fff;  
    padding: .5em 1em;  
}
```

Subscribe at low as:
NGN 500.00
[Sign up](#)

```
text-decoration: none;  
}
```

PINFOICT ACADEMY

Chapter Fourteen

Working with Frames and Favicon

Frames

Frame enables you to create pages within page. It breaks the screen into sections. Individual frames are specified with the frame tag, which is wrapped in a frameset specifier that indicates the amount of space to be allocated to each pane of information.

Example

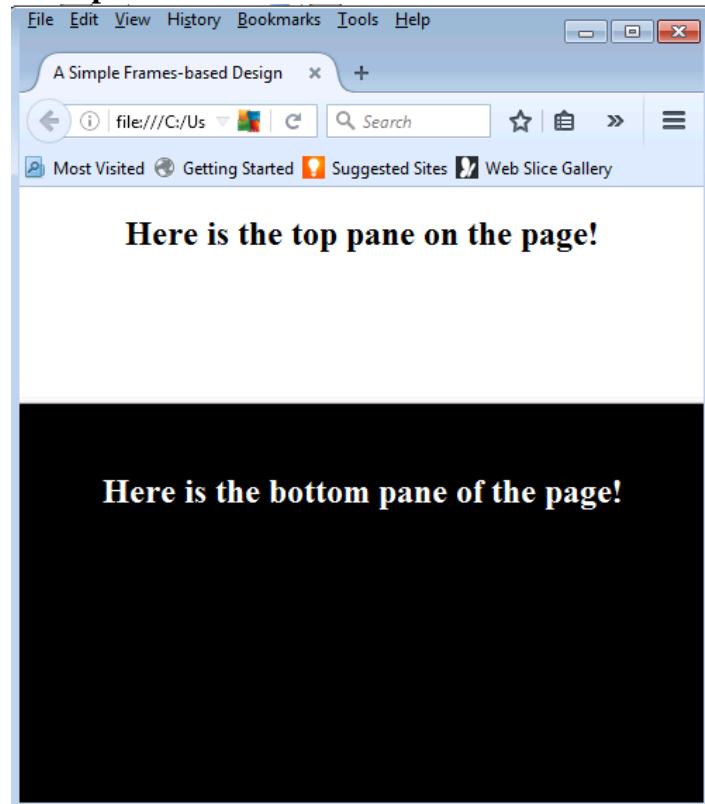


Fig. 13.1

Looking at the diagram above, two pages have been link to one page making it three pages display.

The sample code

Main page saving with frames.html

```
<html>
<title>A Simple Frames-based Design</title>
<frameset rows="75,*">
<frame src="frames/top.html" />
<frame src="frames/bottom.html" />
</frameset>
<noframes></noframes>
</html>
```

Top page saved with top.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Top page</title>
</head>
<body bgcolor="white">
<h2 style="text-align:center;">Here
is the top pane on the page!</h2>
</body>
</html>
```

13.1 Bottom page saved with bottom.html

In the above example, Instead of a single page defining all the information displayed to the visitor, the information is split into multiple pages, each given its own small piece of the window.

The most important tag to learn about is frameset

The frameset tag creates a *frameset*: a set of frames into which the Web page is split. In addition to being able to specify rows to split the Web page into horizontal panes, you can alternatively use cols to specify vertical panes. You can use three different values for these attributes:

- A simple number to specify the desired size in screen pixels
- An asterisk to specify the remaining space on the page
- A percentage of page width by using the n% notation

What does <frameset cols="30%,19,"> mean?*

The sequence cols="30%,19,*" is interpreted as the first column being allocated 30 percent of the width of the window, the next column being allocated a slim 19 pixels, and the third column getting the remainder of the space on the window.

You can create complex multipane Web pages, where each pane has autonomous behavior, by combining these attributes in creative ways as shown below:

```
<html>
<title>A Simple Frames-based Design</title>
<frameset cols='70%,*'>
<frameset rows='70%,30%,*'*>
<frame src='frames/top.html' />
<frame src='frames/bottom.html' />
</frameset>
<frameset rows='33%,33%,*'>
```

```
<frame src="frames/advert1.html" />
<frame src="frames/advert2.html" />
<frame src="frames/advert3.html" />
</frameset>
</frameset>
</html>
```

In this case, two columns of information are specified. One column is 70 percent of the width of the screen; the latter gets the remaining width. That is specified with the following line:

```
<frameset cols="70%, *">
```

The first pane here is the second frameset: two rows, the first (top.html) 30 percent of the available height, and the second (bottom.html) the remaining 70 percent:

```
<frameset rows="30%, 70%">
<frame src="frames/top.html" />
<frame src="frames/bottom.html" />
</frameset>
```

The second column of information (the * width in the first frameset specification) contains three advertisements evenly spaced, each 33 percent of the vertical space:

```
<frameset rows="33%, 33%, *">
<frame src="frames/advert1.html" />
<frame src="frames/advert2.html" />
<frame src="frames/advert3.html" />
</frameset>
```

Output

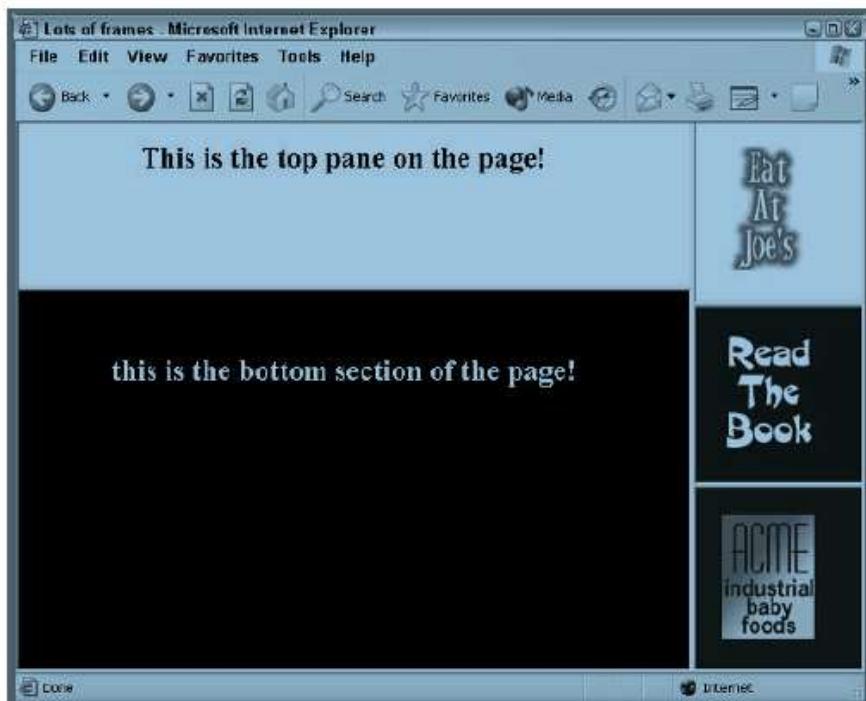


Fig.13.2

A couple of different attributes can be specified for frames, the most important of which is the name= attribute. Each specific frame can be given a unique name (similar to) that can then be used as a way to control which window is affected by specific actions. For instance, your site includes a table of contents in a small pane that is always present. Any user who clicks one of the links on the table of contents actually causes the information in the main pane to change—not the information in the table of contents pane.

That is the idea behind the name= attribute. A partner attribute also appears in the anchor tag for any hypertext reference (a href).

Example

```
<html>
<frameset cols="20%, *">
<frame src="frames/toc.html" />
<frame src="frames/default.html" name="main" />
</frameset>
</html>
```

Notice in the example that the second frame tag now has a name associated with it: main. Here are the contents of the default.html page:

```
<html>
<body style="text-align:center;">

</body>
</html>
```

Here is the all-important toc.html page with the target="main" attribute, where "main" is the name of the specific target pane as specified in the frame tag itself:

```
<html>
<body style="background-color:yellow">
<div style="text-align:center; font-size:120%; font-weight:bold;">
<h2>Pick An Animal</h2>
<div style='line-height:2.0;'>
<a href="dog.html" target="main">DOG</a>
<br />
<a href="cat.html" target="main">CAT</a>
<br />
<a href="bird.html" target="main">BIRD</a>
<br />
<a href="default.html" target="main">(HOME)</a>
</div>
</div>
</body>
</html>
```

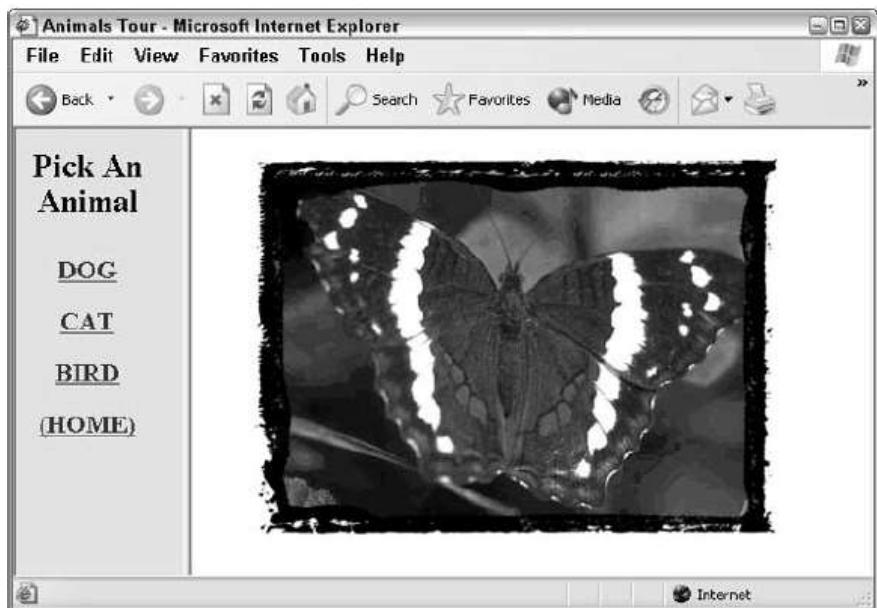


Fig.13.3

13.2 Marginheight and Marginwidth attributes of a frame

These attributes are used to set the margin height and width of a frame;

Examples

```
<html>
<title>The Gettysburg Address</title>
<frameset cols="50%, *" border="5">
<frame src="frames/gettysburg1.html" marginheight="0" marginwidth="0"
/>
<frame src="frames/gettysburg2.html" marginheight="30" marginwidth="30"
/>
</frameset>
```

Inline Frames

One of the coolest things that Microsoft introduced into the HTML language with its popular Internet Explorer browser is the concept of *inline frames*—frame windows completely enclosed by their surrounding window.

An inline frame is specified with the iframe tag in a manner quite similar to how frame tag is specified.

Example

```
<iframe src="inset-info.html" height="40%" width="50%"></iframe>
```

In the above example, inline frame window that is 40 percent of the height is specified, and 50 percent of the width of the current page and that the HTML within should be the page.

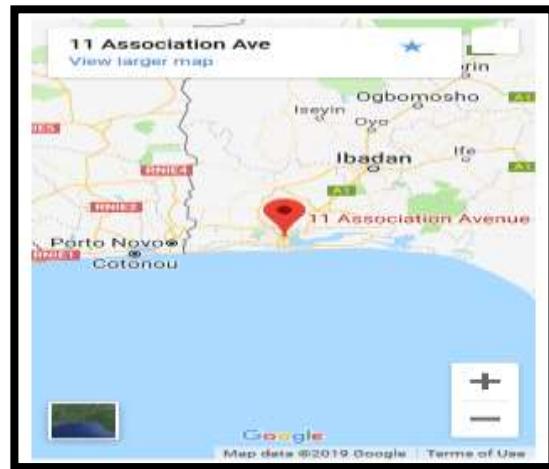
Inline Frame(iframe) and Google Map

In contact page of a website, iframe is also used to include Google map that points to the address of your page. The code snippet below shows how to include Google Map in a page using iframe.

```
<div class="shortcode-google-frame">
```

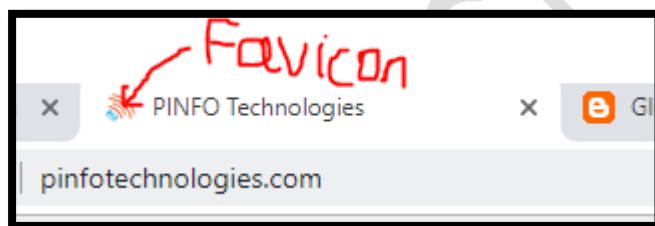
```
<iframe class="contact-iframe" height="480" frameborder="0"
scrolling="no" marginheight="0" marginwidth="0"
src="http://maps.google.com/maps?f=q&source=s_q&hl=en&geocode=&q=11,+association+Avenue+Lagos+Nigeria&aq=0&oq=lagos&sll=6.5474406,3.3638344&sspn=0.013153,0.021136&ie=UTF8&hq=&hnear=Lagos,+Nigeria&t=m&z=7&output=embed"></iframe>
```

Output



Favicon

Favicon is a small icon that appears at the beginning of your website URL bar in a browser. It helps to enhance the brand of your website. Favicon is an acronym for *Favourites Icon*. This small icon is better save with .ico extension such as favicon.ico.



How to setup Favicon

The following are steps to achieve it;

1. Create a 16 x 16 pixels image using ms paint software or an online tool such as <https://www.favicon-generator.org/>, <http://tools.dynamicdrive.com/favicon/>, etc. Mostly, your logo is always used to reduce to the required size.
2. Save the image as *favicon.ico*
3. You can now upload the image to your website root folder via any of FTP application such as Filezilla, etc
4. Then add the following code to link to the location of the saved image :- <link rel="shortcut icon" href="/favicon.ico" type="image/x-icon" /> you must always take note of the path.

PINFOICT ACADEMY

Chapter Fourteen Media Queries(Responsive Website)

Media queries let you customize styles based on the characteristics of the user's device or display, such as the viewport width, whether it's in portrait or landscape mode, or whether it shows color. This is different from the media types, such as screen and print, that you can specify for your style sheets in CSS 2.1. With media queries, you specify not only the media type to which you want to apply a set of styles, but also one or more characteristics of the user's display.

Example

```
@media screen and (max-width: 600px) {  
    body {  
        font-size: 88%;  
    }  
    #content-main {  
        float: none;  
        width: 100%;  
    }  
}
```

The above media query starts with the @media rule, and then specifies a media type (in this case, screen). Next there is the word and, followed by the characteristic we want to match against, called the media feature. This particular media feature, max-width: 600px, tells the browser that the styles for this media query, which are contained within a set of curly brackets for the media query as a whole, should apply only up to a maximum width of 600 pixels. If the viewport width exceeds 600 pixels, the browser will ignore the styles inside the media query. This media query can be dropped right into your main style sheet, keeping all your styles in one place for easy debugging and maintenance, as well as saving an HTTP request. If you want, however, you can apply media queries to separate style sheets on the link element or @import rule as seen below;

```
@import url(narrow.css) only screen and (max-width:600px);  
<link rel="stylesheet" media="only screen and (max-width:600px)" href="narrow.css">
```

Creating a responsive image gallery using @media rules

Example:

Code

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
div.img {  
    border: 1px solid #ccc;  
}  
div.img:hover {  
    border: 1px solid #777;  
}
```

```
div.img img {  
    width: 100%;  
    height: auto;  
}  
  
div.desc {  
    padding: 15px;  
    text-align: center;  
}  
* {  
    box-sizing: border-box;  
}  
.responsive {  
    padding: 0 6px;  
    float: left;  
    width: 24.99999%;  
}  
  
@media only screen and (max-width: 700px){  
    .responsive {  
        width: 49.99999%;  
        margin: 6px 0;  
    }  
}  
@media only screen and (max-width: 500px){  
    .responsive {  
        width: 100%;  
    }  
}  
.clearfix:after {  
    content: "";  
    display: table;  
    clear: both;  
}  
</style>  
</head>  
<body>  
<h2>Responsive Image Gallery</h2>  
<h4>Resize the browser window to see the effect.</h4>  
  
<div class="responsive">  
    <div class="img">  
        <a target="_blank" href="images/Chrysanthemum.jpg">  
              
        </a>  
        <div class="desc">Add a description of the image here</div>  
</div>
```

@media Rules for the page

```
</div>
</div>
<div class="responsive">
  <div class="img">
    <a target="_blank" href="images/Desert.jpg">
      
    </a>
    <div class="desc">Add a description of the image here</div>
  </div>
</div>
<div class="responsive">
  <div class="img">
    <a target="_blank" href="images/Hydrangeas.jpg">
      
    </a>
    <div class="desc">Add a description of the image here</div>
  </div>
</div>
<div class="responsive">
  <div class="img">
    <a target="_blank" href="images/Koala.jpg">
      
    </a>
    <div class="desc">Add a description of the image here</div>
  </div>
</div>
<div class="clearfix"></div>
<div style="padding:6px;">
  <p>This example use media queries to re-arrange the images on different screen sizes: for screens larger than 700px wide, it will show four images side by side, for screens smaller than 700px, it will show two images side by side. For screens smaller than 500px, the images will stack vertically (100%).</p>
</div>
</body>
</html>
```

Output

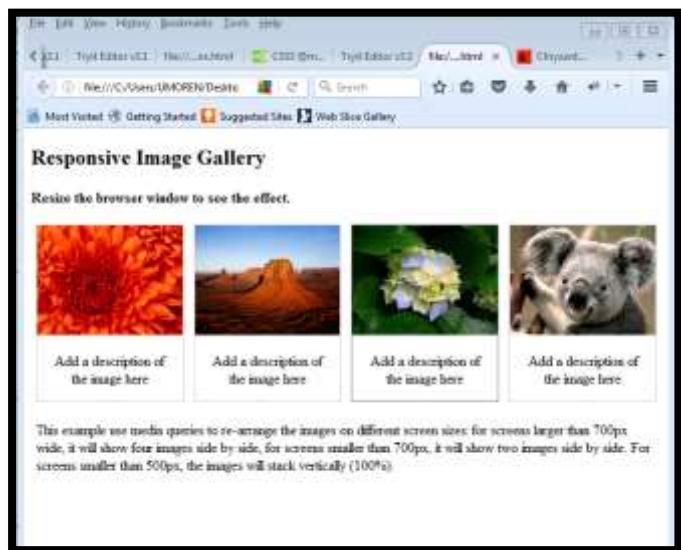


Fig. 14.1

PART Three HTML5

Chapter Sixteen Introduction to HTML5

16.1 HTML5

Many web sites contain HTML code like: <div id="nav"> <div class="header"> <div id="footer"> to indicate navigation, header, and footer.

HTML5 helps you to create awesome websites, it offers new semantic elements to define different parts of a web page:

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>

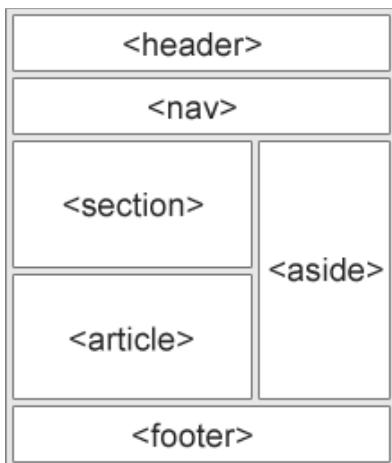


Fig.16.1

We start HTML5 with a doctype as follow;

```
<!doctype html>
```

Next you need <html> tag which is exactly type you need in XHTML and in the <html> tag, you need a language attribute call lang="" as follow;

```
<!doctype html>
```

```
<html lang="en">
```

```
</html>
```

The lang attribute needs to be set to English(en) if you are using English to create your website. You can set the attribute to any language such as fr which refers to French language, etc.

We add the <mata> tag which define the character set in the head tag

```
<!doctype html>
<html lang="en">
<head>
<mata charset="utf-8">
</head>
<body>
</body>
</html>
```

<mata charset="utf-8"> this shows how the browser interpret text; it is kind of character encode which you need to worry about it.

16.2 HTML5 Elements

Html 5 helps you to identify the different part of your body. Tags have been made for each of the pieces, and now your browser know where each tag belongs to.

i. <header> Element

The <header> element specifies a header for a document or section. The <header> element should be used as a container for introductory content such as the logo and the title of your website. You can have several <header> elements in one document.

The following example defines a header for an article:

Example

```
<article>
<header>
<h1>What Does WWF Do?</h1>
<p>WWF's mission:</p>
</header>
<p>WWF's mission is to stop the degradation of our planet's natural environment,
and build a future in which humans live in harmony with nature.</p>
</article>
```

ii. HTML5 <nav> Element

The <nav> element defines a set of navigation/menu links.

The <nav> element is intended for large blocks of navigation links. However, not all links in a document should be inside a <nav> element!

Example

```
<nav>
  <a href="/html/">HTML</a>
  <a href="/css/">CSS</a>
  <a href="/js/">JavaScript</a>
  <a href="/jquery/">jQuery</a>
</nav>
```

iii. <section> Element

The <section> element defines a section in a document. This is where all the information on the webpage goes in.

Example

```
<section>
  <h1>WWF</h1>
  <p>The World Wide Fund for Nature (WWF) is....</p>
</section>
```

iv. HTML5 <article></article>Element

This is where similar information are group together such as Forum post,Blog post
Newspaper article, and it is always in between <section> tag.

Example

```
<article>
  <h1>What Does WWF Do?</h1>
  <p>WWF's mission is to stop the degradation of our planet's natural
environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

Note that each article has its own layout such as header , footer,etc.

v. <hgroup></hgroup> Element

Whenever you have more than one heading tag <h1>,<h2>,... You have to group them in the <hgroup> tag in html5. This tells html5 that this is part of a big group, all the heading items in there are important.

Example

```
<article>
  <header>
    <hgroup>
      <h1>All about PINFO Technologies</h1>
```

```
<h2> Pinfo Tech is an ICT firm that brings your ideas to  
reality</h2>  
</hgroup>  
</header>  
</article>
```

vi. HTML5 <footer> Element

The <footer> element specifies footer for a document or section. A footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc. You can have multiple footers on the same page – if you have several blog entries in one page, every blog entry can have its own footer and the page itself can have its own footer too.

Example

```
<footer>  
<p>Posted by: Hege Refsnes</p>  
<p>Contact information: <a href="mailto:someone@example.com">  
someone@example.com</a>.</p>  
</footer>
```

If your footer is only for links, the <footer> tag should be used this way:

```
<footer>  
<ul>  
<li>About us</li>  
<li>Contact</li> ...  
</ul>  
</footer>
```

vii <aside> Element

The <aside> element defines some content aside from the content it is placed in (like a sidebar). The aside content should be related to the surrounding content.

Example

```
<p>My family and I visited The Epcot center this summer.</p>  
<aside>  
<h4>Epcot Center</h4>  
<p>The Epcot Center is a theme park in Disney World, Florida.</p>  
</aside>
```

viii <section> Element

The section element creates a structure and document outline (semantic markup). It is similar to the div from HTML4, and represents a logical section or component of a web application or document. However, the div is a semantically neutral element, whereas the section element is not. The section element may create items inside an outline of a document, or divide page content into related pieces (like an Introduction) followed by some background information on the topic.

Example from https://www.w3.org/WAI/GL/wiki/Using_HTML5_section_element

```
<article>
  <hgroup>
    <h1>The Guitar Gallery</h1>
    <h2>Lots of guitars</h2>
  </hgroup>

  <section>
    <h2>Fenders</h2>
    <p>Do you like Fenders or Gibsons? Well if a Fender is good enough for Jimi, it's good enough for me!</p>
    <p>[...]</p>
    <h3> Fender Guitars</h3>
    <p>[...]</p>
  </section>

  <section>
    <h2>Gibson Guitars</h2>
    <p>I want an SG but don't want to take out a mortgage to get it</p>
    <p>More about my feelings of deprivation due to lack of antique Gibson guitars[...]</p>
  </section>

  <section>
    <h2>Acoustic Dreams</h2>
    <p>For the softer moments we have nylon acoustic guitars</p>
    <p>Well, I really like John Fahey and Leo Kottke, [...]</p>
    <h3>What kind of guitar did Robbie Basho play??</h3>
  </section>
</article>
```

16.3 HTML5 Style Guide and Coding Conventions

⇒ HTML Coding Conventions

Web developers are often uncertain about the coding style and syntax to use in HTML. Between 2000 and 2010, many web developers converted from HTML to XHTML. With XHTML, developers were forced to write valid and "well-formed" code.

HTML5 is a bit sloppier when it comes to code validation.

With HTML5, you must create your own **Best Practice, Style Guide and Coding Conventions**. A consistent use of style makes it easier for others to understand and use your HTML.

⇒ Use Correct Document Type

Always declare the document type as the first line in your document:

```
<!DOCTYPE html>
```

If you want consistency with lower case tags, you can use:

```
<!doctype html>
```

⇒ Use Lower Case Element Names

HTML5 allows mixing uppercase and lowercase letters in element names.

It is recommended to use lowercase element names:

- Mixing uppercase and lowercase names is bad
- Developers are used to using lowercase names (as in XHTML)
- Lowercase look cleaner
- Lowercase are easier to write

Bad:

```
<SECTION>
  <p>This is a paragraph.</p>
</SECTION>
```

Very Bad:

```
<Section>
  <p>This is a paragraph.</p>
</SECTION>
```

Good:

```
<section>
  <p>This is a paragraph.</p>
</section>
```

⇒ Close All HTML Elements

In HTML5, you do not have to close all elements (for example the `<p>` element).

We recommend closing all HTML elements:

Looking bad:

```
<section>
  <p>This is a paragraph.
  <p>This is a paragraph.
</section>
```

Looking good:

```
<section>
  <p>This is a paragraph.</p>
  <p>This is a paragraph.</p>
</section>
```

⇒ Close Empty HTML Elements

In HTML5, it is optional to close empty elements.

This is allowed:

```
<meta charset="utf-8">
```

This is also allowed:

```
<meta charset="utf-8" />
```

The slash (/) is required in XHTML and XML.

⇒ Use Lower Case Attribute Names

Looking bad:

```
<div CLASS="menu">
```

Looking good:

```
<div class="menu">
```

⇒ Quote Attribute Values

HTML5 allows attribute values without quotes.

We recommend quoting attribute values:

- You have to use quotes if the value contains spaces
- Mixing styles is never good
- Quoted values are easier to read

This will not work, because the value contains spaces:

```
<table class=table striped>
```

This will work:

```
<table class="table striped">
```

⇒ Image Attributes

Always use the **alt** attribute with images. It is important when the image cannot be viewed.

```

```

Always define image size. It reduces flickering because the browser can reserve space for images before they are loaded.

```

```

⇒ Spaces and Equal Signs

A space around equal signs is legal:

```
<link rel = "stylesheet" href = "styles.css">
```

But space-less is easier to read, and groups entities better together:

```
<link rel="stylesheet" href="styles.css">
```

⇒ Avoid Long Code Lines

When using an HTML editor, it is inconvenient to scroll right and left to read the HTML code.

Try to avoid code lines longer than 80 characters.

⇒ Blank Lines and Indentation

Do not add blank lines without a reason. For readability, add blank lines to separate large or logical code blocks. For readability, add 2 spaces of indentation. Do not use TAB.

⇒ Omitting <html> and <body>

In the HTML5 standard, the <html> tag and the <body> tag can be omitted.

The following code will validate as HTML5:

Example

```
<!DOCTYPE html>
<head>
  <title>Page Title</title>
</head>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
```

However, it is not recommended to omit <html> and <body> tags. The <html> element is the document root. It is the recommended place for specifying the page language:

```
<!DOCTYPE html>
<html lang="en-US">
```

Declaring a language is important for accessibility applications (screen readers) and search engines.

Omitting <html> or <body> can crash DOM and XML software, and can produce errors in older browsers (IE9).

⇒ Omitting <head>

In the HTML5 standard, the <head> tag can also be omitted.

By default, browsers will add all elements before <body>, to a default <head> element.

You can reduce the complexity of HTML, by omitting the <head> tag:

Example

```
<!DOCTYPE html>
<html>
  <title>Page Title</title>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

⇒ Meta Data in HTML5

The <title> element is required in HTML5. Make the title as meaningful as possible:

```
<title>HTML5 Syntax and Coding Style</title>
```

To ensure proper interpretation, and correct search engine indexing, both the language and the character encoding should be defined as early as possible in a document:

```
<!DOCTYPE html>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>HTML5 Syntax and Coding Style</title>
</head>
```

⇒ Style Sheets in HTML5

Use simple syntax for linking style sheets (the type attribute is not necessary):

```
<link rel="stylesheet" href="styles.css">
```

Short rules can be written compressed, on one line, like this:

```
p.into {font-family: Verdana; font-size: 16em;}
```

Long rules should be written over multiple lines:

```
body {
  background-color: lightgrey;
  font-family: "Arial Black", Helvetica, sans-serif;
  font-size: 16em;
  color: black;
}
```

- Place the opening bracket on the same line as the selector.
- Use one space before the opening bracket.
- Use 2 spaces of indentation.
- Use colon plus one space between each property and its value.
- Use space after each comma or semicolon.
- Use semicolon after each property-value pair, including the last.
- Only use quotes around values if the value contains spaces.
- Place the closing bracket on a new line, without leading spaces.
- Avoid lines over 80 characters.

16.4 Creating a simple page layout using html5

This involves usig css and html5 codes, the layout to be created will be responsive, this imples it can adjust base on the size of a display screen.

- ➔ First let us create the html to work with

```
<!doctype html>
<html lang="en">
<head>
  <title>
```

```
</title>
<meta charset='utf-8' />
<link rel="stylesheet" href="style.css">
</head>
<body>
  <div id="big_wrapper">
    <header id="top_header">
      <h1>Welcome to the HTML5</h1>
    </header>
    <nav id="top_menu">
      <ul>
        <li>Home</li>
        <li>Tutorial</li>
        <li>Recipes</li>
      </ul>
    </nav>
    <section id="main_section">
      <article>
        <header>
          <hgroup>
            <h1> Title of Article</h1>
            <h2> Subtitle for the Article</h2>
          </hgroup>
        </header>
        <p> This is the best article so far!</p>
      </article>
    </section>
    <aside id="side_news">
      <h4>Up Coming News</h4>
      There is a new herbs formula coming up
    </aside>
    <footer id="the_footer">
      copyright Umoren 2017
    </footer>
  </div>
</body>
</html>
```

Next we need to link our CSS file for the html ids created with the file name style.css

We now move on to our css file to style our layout.

-> We set up our css file as follow;

```
*{
  margin:0px;
```

```
        padding:0px;
    }
    h1{
        font: bold 20px Tahoma;
    }
    h2{
        font: bold 14px Tahoma;
    }
    header, section, footer, aside, nav, article, hgroup{
        display:block;
    }
    body{
        width:100%;
        display:-webkit-box;
        -webkit-box-pack:center; /* centring the website */
    }
}
```

After setting up our css file as shown above, The output will look like this;

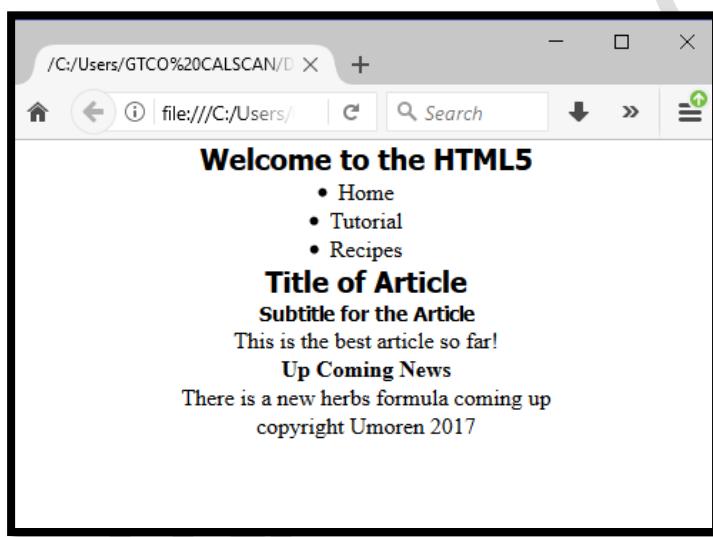


Fig. 16.2

This shows that not a serious style has been added to it.

Next we work on the header and menu bars by adding the following css codes to our existing css file

```
#big_wrapper{
    max-width:1000px;
    margin:20px 0px;
```

```
display:-webkit-box;
-webkit-box-orient:vertical;
-webkit-box-flex:1;
}
#top_header{
background: yellow;
border:3px solid black;
padding:20px;
}

}
#top_menu{
border:red;
background: blue;
color:white;
}

#top_menu li{
display:inline-block;
list-style:none;
padding:5px;
font:bold 14px Tahoma;
}

}
```

Then the output will look like this



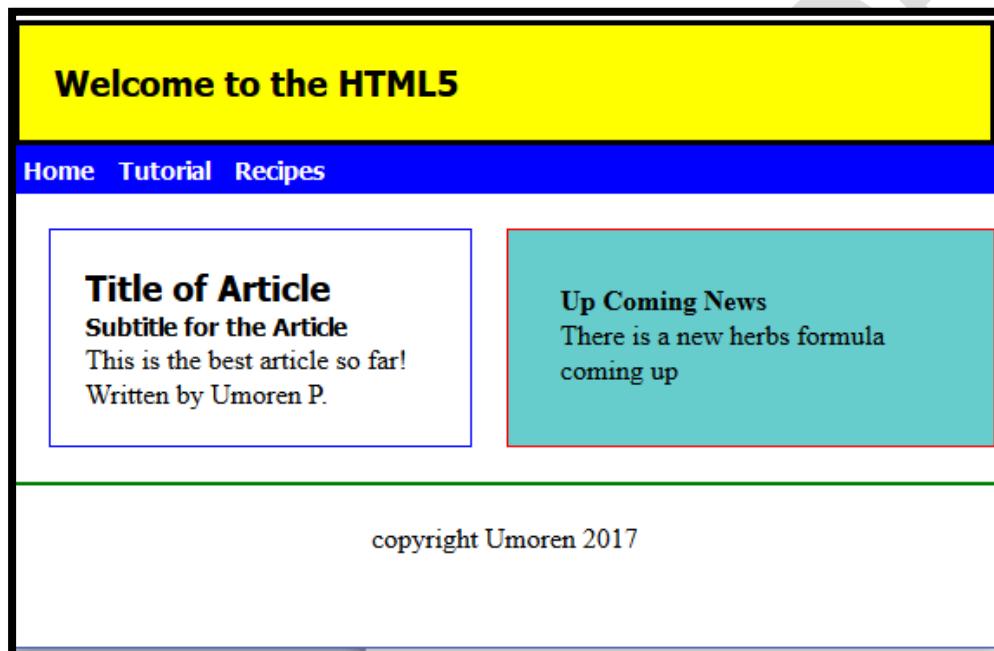
Fig. 16.3

This implies that the <header> tags and the <nav> have been style accordingly using the above css codes.

Now we style the main section as follows;

```
#main_section{
border:1px solid blue;
-webkit-box-flex:1; /* make the box flexible*/
margin:20px;
```

```
padding: 20px;  
}  
#side_news{  
    border: 1px solid red;  
    width: 220px;  
    margin: 20px 0px;  
    padding: 30px;  
    background: #66CCCC;  
}  
#the_footer{  
    text-align: center;  
    padding: 20px;  
    border-top: 2px solid green;  
}
```



16.4

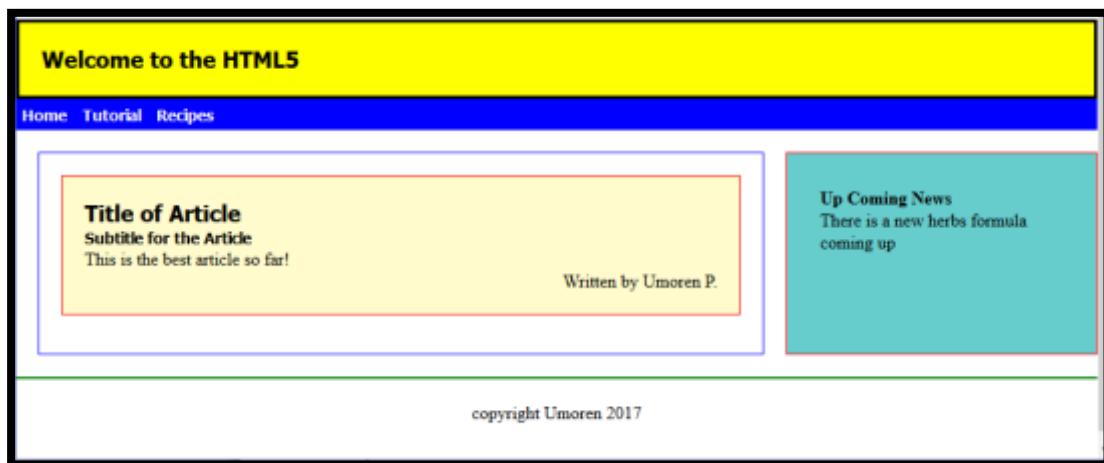
The above output implies that the main contents of the webpage have been styled such as the `<section>`, `<footer>`, `<aside>` tags.

Let us finish styling the `<article>` tag of our website by adding the following css code which affect the article and the footer inside the article;

```
article{  
    background: #FFFBCC;  
    border: 1px solid red;  
    padding: 20px;
```

```
        margin-bottom:15px;  
    }  
    article footer{  
        text-align:right;  
    }
```

Output;



16.5

This is the code css code save in the file call style.css

```
*{  
    margin:0px;  
    padding:0px;  
}  
h1{  
    font: bold 20px Tahoma;  
}  
h2{  
    font: bold 14px Tahoma;  
}  
header, section, footer, aside, nav, article, hgroup{  
    display:block;  
}  
body{  
    width:100%;  
    display:-webkit-box;  
    -webkit-box-pack:center; /* centering the website */  
}  
#big_wrapper{
```

```
max-width:1000px;
margin:20px 0px;
display:-webkit-box;
-webkit-box-orient:vertical;
-webkit-box-flex:1;
}
#top_header{
background: yellow;
border:3px solid black;
padding:20px;

}
#top_menu{

border:red;
background: blue;
color:white;
}

#top_menu li{

display:inline-block;
list-style:none;
padding:5px;
font:bold 14px Tahoma;

}
#new_div{
display:-webkit-box;
-webkit-box-orient:horizontal;
}

#main_section{
border:1px solid blue;
-webkit-box-flex:1; /* make the box flexible*/
margin:20px;
padding: 20px;

}
#side_news{
border: 1px solid red;
width:220px;
margin:20px 0px;
padding: 30px;
background:#66CCCC;
}

#the_footer{
```

```
text-align:center;
padding:20px;
border-top:2px solid green;
}

article{
background: #FFFBC;
border:1px solid red;
padding:20px;
margin-bottom:15px;
}
article footer{
text-align:right;
}
```

Video tags in HTML5

In order to add video to your website, make use of video tag as it is shown below;

```
<video> </video>
```

next is to add the **src** attribute, this link to source of your video either online or within your local storage as shown below;

```
<video src="videos/comedy.mp4"></video>
```

if you load the video above, it will not play because controls have not been added to it, to get the video playing, we add **controls** attribute to it which gives us a default video browser control panel as it is shown below;

you can change the width of the video control panel as it is shown below;

```
<video src="videos/34.mp4" width ="640" height="360" controls></video>
```

To make the video plays continuously, we add **loop** attribute to it, this makes it continue playing with it gets to the end as shown below;

```
<video src="videos/34.mp4" width ="640" height="360" loop controls></video>
```

another attribute to use is poster which is the still image that will show when your video is not playing as it is shown below;

```
<video src="videos/34.mp4" width ="640" height="360"
poster="images/mycapface.jpeg" loop controls></video>
```

PART Four
JavaScript Fundamentals

PINFOICM ACADEMY

Chapter Seventeen

Javascript Fundamentals and Introduction to Algorithm

JavaScript is a multi-paradigm language that supports event-driven, functional, object-oriented, and prototype-based programming styles. JavaScript was initially used only for the client-side but, in more recent times, it has also been used as a server-side programming language.

17.1 What you can do with javascript

- ⇒ Create an active user interface that reacts to events
- ⇒ Validate user input on forms
- ⇒ Create custom HTML pages on the fly
- ⇒ Control web browsers and detects the visitor's browser
- ⇒ To create cookies in your application

17.2 How to incorporate JavaScript into the web page.

JavaScript is inserted into the webpage in the head of the html document, or any part of the part of the document. But the best practise is to put it in the head of a document as follow;

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Untitled Document</title>
<script type="text/javascript" language="javascript">
</script>
</head>
```

Fig. 17.1

In between this tag `<script type="text/javascript" language="javascript">` javascript code goes in here `</script>`. Another way of incorporating JavaScript into the page is to link it from the external file as follow;

```
<script language="javascript" type="text/javascript" src="../javascript/zoneChecker.js">
</script>
<script language="javascript" type="text/javascript" src="../javascript/modifications/modify_contents.js">
</script>
```

Fig.17.2

17.3 Ways of displaying data in JavaScript

The following are ways Javascript displays data :

- Writing into an HTML element, using innerHTML.
- Writing into the HTML output using document.write().
- Writing into an alert box, using window.alert().
- Writing into the browser console, using console.log().

17.3.1 Using innerHTML

The document.getElementById(id) method accesses the HTML element, the id attribute defines the HTML element, the innerHTML property defines the HTML content as it is shown in the example below;

```
<body>

<h1>This is my header H1</h1>

<p>This is my paragraph p-tag</p>

<p id="docId"></p>

<script>

document.getElementById("docId").innerHTML="Displaying data with innerHTML
example";

</script>

</body>
```

Output

```
<title>Javascript Display Examples</title>
</head>
<body>
    <h1>This is my header H1</h1>
    <p>This is my paragraph p-tag</p>
    <p id="docId"></p>
    <script>
        document.getElementById("docId").
        innerHTML="Displaying data with innerHTML example";
    </script>
</body>
</html>
```

This is my header H1

This is my paragraph p-tag

Displaying data with innerHTML example

17.3.2.Using document.write()

The code snippet below illustrates how to use document.write() method to display data in JavaScript;

```
<body>
<h1>This is my header H1</h1>
<p>This is my paragraph p-tag</p>
<script>
document.write("Displaying data with document.write method in Javascript");
</script>
</body>
```

Output

```
<script>
document.write("Displaying data with document.write method in Javascript");
</script>
```

Displaying data with document.write method in Javascript

17.3.3 Using window.alert()

The example below shows how to use an alert box to display data in javascript;

```
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>
<script>
window.alert('Displaying data using alert box in Javascript');
</script>
</body>
```

Output

```
<!-- Example of displaying data in javascript with document.write() method-->
<script>
document.write("Displaying data with document.write method in Javascript");
</script>
<!-- Example of displaying data in javascript with window.alert() box-->
<script>
window.alert('Displaying data using alert box in Javascript');
</script>
```



17.3.4 Using console.log()

You can call console.log() method in the browser to display data for debugging purposes as shown in the example below;

```
<script>
console.log("displaying data in browser console");
</script>
```

Output



The screenshot shows the Microsoft Edge DevTools interface with the 'Console' tab selected. The console window displays the output of a JavaScript script. The script contains three parts of code: a comment block, a `document.write()` call, and a `window.alert()` call. The output in the console shows the text "displaying data in browser console".

```
<!-- Example of displaying data in javascript with document.write() method-->
<script>
  document.write("Displaying data with document.write method in Javascript");
</script>
<!--Example of displaying data in javascript with window.alert() box-->
<script>
  window.alert("Displaying data using alert box in Javascript");
</script>
<!--Example of displaying data in browser console-->
<script>
  console.log("displaying data in browser.console");
</script>
```

17.4 Comments

This is a line of code that browser ignores. It gives explanation to your program which makes it possible for other programmers to edit as well as you to edit in the future.

Types of comments

17.4.1 Single line comment

This only affects one line

// this is a single line comment

17.4.2 Multiple line comment

This affect more than one line

```
/*
  This is a multiple line comment
*/
```

17.5 Basic parts of JavaScript

The following are the basic parts of Javascript:

Variable: A variable is a word that represents a piece of text, a number, a boolean (true or false value) or an object which takes a space in a computer memory. Remember that variables are case sensitive, they should not start with numbers and your variable name should be meaningful.

17.5.1 Variable naming conversion

Underscore: first_name, last_name, master_card_pin, country_id.

Upper Camel Case (Pascal Case): FirstName, LastName, MasterCardPin, CountryId.

Lower Camel Case: firstName, lastName, masterCardPin, countryId.

17.5.2 Ways of declaring variables in JavaScript

The following are four ways of declaring variables in JavaScript

- Using var : If you want your code to run in older browser, you must use var

Example

```
var x = 5;  
var y = 6;  
var z = x + y;  
  
var first_name
```

- Using let: Variables defined with let cannot be Redeclared, it must be declared before use and variables defined with let have Block Scope.

```
let lastName  
let x = 5;  
let y = 6;  
let z = x + y;
```

- Using const

The variables declare with **const** cannot be changed, the general rule is to declare variables with const unless you know that the variable will change. Note that JavaScript const variables must be assigned a value when they are declared.

Example

Correct

```
const VOL = 35;
```

Incorrect

```
const VOL;  
VOL = 35;
```

In the example below, price_a, price_b, and total_price, are variables:

Example

```
const price_a = 5;  
const price_b = 6;  
let total_price = price_a + price_b;
```

The two variables price_a and price_b are declared with the const keyword.

```
const PI = 3.141592653589793;  
PI = 3.14; // This will give an error  
PI = PI + 13; // This will also give an error
```

These are constant values cannot be changed. The total_price is declared with let keyword which can be changed.

When to use JavaScript const

Use const when you are declaring the following;

⇒ A new array:

Example

```
const cars = ["Camry", "Volvo", "BENZ"];  
  
cars = ["Toyota", "Volvo", "Audi"]; // ERROR
```

this implies that you can not reassign the array:

⇒ A new Object : In the example below, you cannot reassign the object.

```
const car = {type: "Fiat", model: "500", color: "blue"};  
car = {type: "Volvo", model: "EX60", color: "white"}; // ERROR
```

but you can change the property as well as add a new property as shown below;

```
// You can change a property:  
car.color = "red";
```

```
// You can add a property:  
car.owner = "Umoren";
```

- Using nothing

In the example below, x, y, and z, are undeclared variables:

Example

```
x = 50;  
y = 60;  
z = x + y;
```

The basic variable types are:

character = 'a' or '@' or 'k' etc.

string = 'hdttgs 8 -46hs () Get the idea'

integer = 0 or 1 or 2 or 3 or 4 etc. or -1 or -2 or -3 or -4 etc.

float (or double) = 23.546 or -10.46

boolean = true or false

17.5 Operator

An operator is a command or a symbol that combine two or more operands and produces result of the combination. Operands may be variables of values as the case maybe. Operators serve to bring values and/or variables together and yield new results. Operators are important in JavaScript programming, as they are the main way to evaluate current conditions and change future conditions.

The following are common arithmetic operators (binary operators): This is because they work their magic given two values (binary meaning "two")

- + addition
- subtraction
- *multiplication
- /division
- % modulus (returns the remainder of a division)

+ Operator can also be used for concatenation. Concatenation is the process of combining multiple strings to yield one new, bigger string.

⇒ Unary operators

The unary operators are: **Unary plus (+)** which convert an operand into a number; **Unary minus (-)** which convert an operand into a number and negate the value after that, as well as **prefix / postfix increments (++)** that adds one to its operand, and **prefix / postfix decrements (--)** that minus one to its operand.

Examples

```
let a = 10;  
a = +a; // add 10 to initial variable value => 20  
a = -a; // -10 from the initial variable value => 0
```

```
let i = 10;  
i++;  
i--;
```

⇒ Assignment Operators

This is used to assign one value, either a literal or a variable to a variable.

price = 100;

price = cost;

total += price; //this is the same as total = total+price;

⇒ Comparison Operators

Comparison is a simple way to make your program makes simple decision. When you are comparing two values you use '==' signs.

Mugs==10; //is equal to

Mug!=10 ; //is not-equal to comparison operator

Two other common comparison operators are greater than (>) and less than (<). They are used in the following manner:

mugs > 10; // mugs greater than 10

mugs < 10; // mugs less than 10

mugs >= 10 // mugs is greater than or equal to 10

mugs <= 10 // mugs is less than or equal to 10

Example:

```
<script type="">
  Var apples = 34;
  Var hotdogs=53;
  //If(comparision) If a condition is true, run this code;
  If(apples==hotdogs)
    document.write("yay it worked!");
  }
</script>
```

➔ Logical Operators:

&& (the AND Operator)

The && operator looks at two comparisons, and if each is true in and of itself, the && operator returns a true but if one is false, the && operator returns false.

(mugs > 5) && (price < 10)

|| (the OR Operator)

It requires that only one of the two comparisons be true in order for it to return a true for the whole logical comparison. For example:

(mugs > 5) || (price < 10)

Just as with the && operator, both comparisons above are evaluated. If either is true, the OR operator returns a true. Of course, if both are true, it still returns a true. Only if both comparisons yield false results will the || operator returns a false result.

! (the NOT Operator)

This is a negation operator, it shows that the value is not same as the original one.

Example

```
<body>
    <script type="text/javascript">
        var posNum = 4;
        var negNum = -2;
        //using AND operator
        if(posNum > 0 && negNum < 0) {
            document.write('This is AND operator<br>');
        }

        //using OR operator
        if(posNum > 0 || negNum > 0) {
            document.write('This is OR operator<br>');
        }

        //using the NOT operator
        if(posNum > 0 && !(negNum > 0)) {
            document.write('This is NOT operator');
        }
    </script>
</body>
```

17.5.2 Operators Table

| Operator | Uses |
|----------|--|
| + | adds two numbers or appends two strings if more than one type of variable is appended, including a string appended to a number or vice-versa, the result will be a string |
| - | subtracts the second number from the first |
| / | divides the first number by the second |
| * | multiplies two numbers |
| % | divide the first number by the second and return the remainder |
| = | assigns the value on the right to the object on the left |
| += | the object on the left = the object on the left + the value on the right this also works when appending strings |
| -= | the object on the left = the object on the left - the value on the right |
| > | number on the right must be greater than the number on the left this also works with strings and values |

| | |
|----|--|
| < | number on the right must be less than the number on the left this also works with strings and values |
| >= | number on the right must be greater than or equal to the number on the left this also works with strings and values |
| <= | number on the right must be less than or equal to the number on the left this also works with strings and values |
| ++ | increment the number |
| -- | decrement the number |
| == | the numbers or objects or values must be equal |
| != | the numbers or objects or values must not be equal |
| << | bitwise leftshift |
| >> | bitwise rightshift |
| & | bitwise AND |
| | bitwise OR |
| ^ | bitwise XOR |
| ~ | bitwise NOT |
| ! | logical NOT (the statement must not be true) |
| && | logical AND (both statements must be true) |
| | logical OR (either statement must be true) |
| In | object or array on the right must have the property or cell on the left |

17.5.3 The Math object operators

In reality, these are methods of the Math object but they act like operators.

| Operator | What it does |
|-----------------|---|
| Math.abs(n) | Returns the absolute value of n |
| Math.acos(n) | Returns (in radians) \cos^{-1} of n |
| Math.asin(n) | Returns (in radians) \sin^{-1} of n |
| Math.atan(n) | Returns (in radians) \tan^{-1} of n |
| Math.atan2(n,k) | Returns the angle (rads) from cartesian coordinates 0,0 to n,k |
| Math.ceil(n) | Returns n rounded up to the nearest whole number |
| Math.cos(n) | Returns cos n (where n is in radians) |
| Math.exp(n) | Returns e^n |
| Math.floor(n) | Returns n rounded down to the nearest whole number |
| Math.log(n) | Returns $\ln(n)$ Note, to find $\log_{10}(n)$, use $\text{Math.log}(n) / \text{Math.log}(10)$ |

| | |
|---------------------------|--|
| Math.max(a,b,c,d,e,.....) | Returns the largest number |
| Math.min(a,b,c,d,e,.....) | Returns the smallest number |
| Math.pow(n,k) | Returns n^k |
| Math.random() | Returns a random number between 0 and 1 |
| Math.round(n) | Returns n rounded up or down to the nearest whole number |
| Math.sin(n) | Returns sin n (where n is in radians) |
| Math.sqrt(n) | Returns the square root of n |
| Math.tan(n) | Returns tan n (where n is in radians) |

17.6 Arrays

Array is a special variable like a box that holds similar elements in a program. If you have a list of items such as a list of courses, storing the courses in single variables could look like this:

```
let course1 = "react";
let course2 = "Java";
let course3 = "PHP";
```

but if you have more than fifty(50) courses, to loop through it and find a specific one will be a problem, the best solution to that is using an Array.

⇒ Creating an array

To create an array, use:

Syntax:

```
const array_name = [item1, item2, item3, ...];
const cars = ["Saab", "Volvo", "BMW", "Camry"];
```

const nameOfArray = new Array(); - or

const nameOfArray = new Array(5); where 5 is the desired length of the array

const nameOfArray = new Array('content_of_first_cell', 'and_the_second', '8', 'blah', 'etc'); - or

const nameOfArray = ['content_of_first_cell', 'and_the_second', '8', 'blah', 'etc'];

and then you can refer to it using its name. However, the two examples above using “new Array” and “[]” do exactly the same thing.

You can also create an array, and then provide the elements as shown in the example below;

```
const cars = [];
cars[0]= "Saab";
cars[1]= "Volvo";
cars[2]= "BMW";
cars[3]= "Camry";
```

⇒ Accessing Array Elements using Index

You access an array element by referring to its **index number** as shown below:

```
const cars = ["Saab", "Volvo", "BMW", "Camry"];
let car = cars[0];
```

Note: Array indexes start with 0, [0] refers to the first element in the array, [1] refers to the second element, [2] refers to the third element and so on.

Example

```
<body>
```

```
<script type="text/javascript">
    const rainbowColors = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet'];
    //accessing an array
    var firstColor = rainbowColors[0];
    var fifthColor = rainbowColors[4];
    document.write('The first value in the array is '+firstColor+'<br>');
    document.write('The fifth value in the array is '+fifthColor+'<br>');

</script>
</body>
```

Associative Arrays

An associative array is one in which its elements are referenced by name rather than by numeric offset. To create an associative array, define a block of elements within curly braces. For each element, place the key on the left and the contents on the right of a colon (:).

Creating and displaying an associative array

```
<script>
balls = {
    "golf": "Golf balls, 6",
    "tennis": "Tennis balls, 3",
    "soccer": "Soccer ball, 1",
    "ping": "Ping Pong balls, 1 doz"
}
```

```
for (ball in balls)
document.write(ball + " = " + balls[ball] + "<br>")
</script>
```

To verify that the array has been correctly created and populated, I have used another kind of for loop using the in keyword. This creates a new variable to use only within the array (ball, in this example) and iterates through all elements of the array to the right of the in keyword (balls, in this example). The loop acts on each element of balls, placing the key value into ball. Using this key value stored in ball, you can also get the value of the current element of balls. The result of calling up the example script in a browser is as follows:

```
golf = Golf balls, 6
tennis = Tennis balls, 3
soccer = Soccer ball, 1
ping = Ping Pong balls, 1 doz
```

To get a specific element of an associative array, you can specify a key explicitly, in the following manner (in this case, outputting the value Soccer ball, 1):

```
document.write(balls['soccer'])
```

⇒ Array Properties and Methods

The strength of JavaScript arrays are the built-in array properties and methods. The following examples show array properties and methods:

```
cars.length // Returns the number of elements
```

```
//length of an array
document.write('The length of the array is '+rainbowColors.length+'<br>');
```

Iterating through Array

You can access elements of the array using iteration as shown below;

```
<body>
<script type="text/javascript">
const rainbowColors = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet'];
for (var i = 0; i < rainbowColors.length; i++) {
    document.write(rainbowColors[i]+'<br>');
}
</script>
</body>
```

The examples below shows how to iterate through an array of images

```
<body>
<script>
    //Arrays
    const students= ['a','b','c','d','e','f','g'];

    for(var i=0; i<students.length; i++){
        document.write("<img src='images/" + students[i] + ".jpg'>" );
    }
</script>
</body>
```

You can also use the Array.forEach() function to loop through an array:

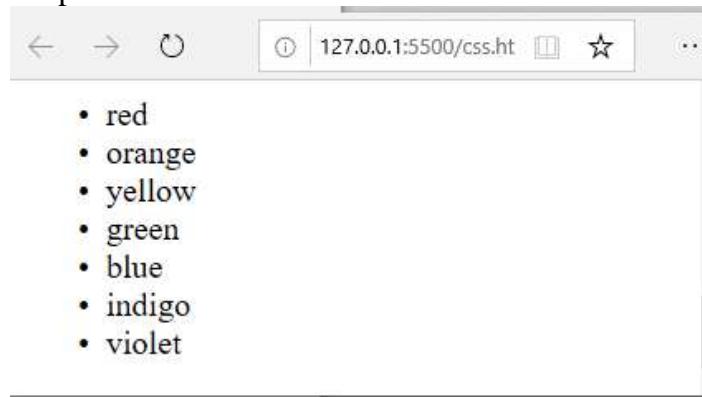
```
<p id="demoarray"></p>
<script>
const rainbowColors = ['red', 'orange', 'yellow','green', 'blue', 'indigo', 'violet'];

let text = "<ul>";
rainbowColors.forEach(myFunction);

text += "</ul>";

function myFunction(value) {
    text += "<li>" + value + "</li>";
    document.getElementById("demoarray").innerHTML=text;
}
</script>
```

Output



⇒ Adding Array Elements

The easiest way to add a new element to an array is using the push() method as shown below;

```
//adding value to the end of an array
rainbowColors.push('gold');
document.write('The length of the array is '+rainbowColors.length+'<br>');
```

- red
- orange
- yellow
- green
- blue
- indigo
- violet
- gold

The length of the array is 8

⇒ Converting Arrays to Strings

JavaScript uses method toString() to convert an array to a string of (comma separated) array values.

```
<p id="demo"></p>

<script>
const rainbowColors = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet'];
document.getElementById("demo").innerHTML = rainbowColors.toString();
</script>
```

Output

```
red,orange,yellow,green,blue,indigo,violet,gold
```

⇒ Using join() method in an array

The join() method also joins all array elements into a string by specifying the separator.
Example

```
<p id="demo"></p>

<script>
const rainbowColors = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet'];
document.getElementById("demo").innerHTML = rainbowColors.join(" * ");
```

```
</script>
```

Output

```
red * orange * yellow * green * blue * indigo * violet * gold
```

⇒ JavaScript Array pop()

When you work with Javascript, it is very easy to remove elements from an array, the pop() method handles this by removing the last element from an array.

Example

```
<script>
const rainbowColors = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet'];
rainbowColors.pop();
document.getElementById("demo2").innerHTML=rainbowColors;
</script>
```

Output

```
red,orange,yellow,green,blue,indigo
```

⇒ JavaScript Array shift()

The shift() method removes the first array element and "shifts" all other elements to a lower index. It works like the pop() method, but in this case, it affects the first element.

Example

```
<script>
const rainbowColors = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet'];
rainbowColors.shift();
document.getElementById("demo2").innerHTML=rainbowColors;
</script>
```

In the result below, the first element have been removed from the array

Output

```
orange,yellow,green,blue,indigo,violet
```

⇒ JavaScript Array unshift()

The unshift() method adds a new element to an array at the beginning, and "unshifts" older elements to the right:

Example

```
<script>
const rainbowColors = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo'];
rainbowColors.unshift("gold");
document.getElementById("demo2").innerHTML=rainbowColors;
```

```
</script>
```

Output

```
gold,red,orange,yellow,green,blue,indigo
```

⇒ Merging (Concatenating) Arrays

The concat() method creates a new array by merging existing arrays, as shown in the example below:

Example

```
<p id="demo4"></p>
```

```
<script>
```

```
const business_staff = ["Daniel", "Esther", "Nathaniel", "Samuel"];
const training_staff = ["Stella", "Edna", "Blessing", "Legend", "Simon"];

const all_staff = business_staff.concat(training_staff);

document.getElementById("demo4").innerHTML = all_staff;
```

```
</script>
```

Output

```
Daniel,Esther,Nathaniel,Samuel,Stella,Edna,Blessing,Legend,Simon
```

⇒ JavaScript Sorting Arrays

The sort() method sorts the elements of an array as strings in alphabetical and ascending order. Sorting alphabetically works well for strings, but sorting numbers can produce incorrect results. However, this can be fixed by providing a "*compare function*" as shown in the example below;

```
all_staff.sort();
document.getElementById("demo5").innerHTML = all_staff;
</script>
```

After the array is sorted with sort(), the result is shown below;

```
Blessing,Daniel,Edna,Esther,Legend,Nathaniel,Samuel,Simon,Stella
```

Sorting numbers using compare function:

```
<p id="demo"></p>
```

```
<script>
const nums = [40, 100, 1, 4, 7, 9, 90, 5, 25, 10];
nums.sort(function(a, b){return a-b});
document.getElementById("demo").innerHTML = nums;
</script>
```

Output

```
1,4,5,7,9,10,25,40,90,100
```

⇒ Reversing an Array

The reverse() method reverses the elements in an array, by combining sort() and reverse() you can sort an array in descending order as shown in the example below;

```
<p id="demo7"></p>
<script>
const rainbowColors = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo'];

// First sort the array
rainbowColors.sort();

// Then reverse it:
rainbowColors.reverse();

document.getElementById("demo7").innerHTML = rainbowColors;

</script>
```

Output

```
yellow,red,orange,indigo,green,gold,blue
```

⇒ JavaScript Array map()

The Array.map() method allows you to iterate over an array and modify its elements using a callback function. The callback function will then be executed on each of the array's elements.

For instance, you are required to multiply each of the element in an array and produce a new array, this could easily be done with array map() method.

Example

In this example, we multiply each array value by 5.

```
<p id="demo8"></p>
<script>
const nums2 = [40, 100, 1, 4, 7, 9, 90, 5, 25, 10];
const num3 = nums2.map(myArrayFunction);
```

```
document.getElementById("demo8").innerHTML=num3;  
  
function myArrayFunction(value){  
    return value*5;  
}  
  
</script>
```

Output

```
200,500,5,20,35,45,450,25,125,50
```

⇒ JavaScript Array reduce()

The Array reduce() method reduces an array elements to a single value. It works from left to right by executing the callback function.

Example

```
<p id="demo9"></p>  
<script>  
const numbers = [45, 4, 9, 16, 25, 46, 90];  
let sum = numbers.reduce(myFunction);  
  
document.getElementById("demo9").innerHTML = "The sum is " + sum;  
  
function myFunction(total, value) {  
    return total + value;  
}  
</script>
```

Output

```
The sum is 235
```

17.6 Control Structure

Control structures say what scripts should be run if a test is satisfied. Control structures come as statements such as If...else Statement, While Statement, Do .. While Statement, for loop statement, Switch Statement, etc.

17.6.1 The If...else and else if Statements

```
if (income >= 1000) {  
    lights = true;  
}  
else {  
    lights = false;  
}
```

If the expression is false, JavaScript skips ahead to the else clause and executes the statements within the set of brackets immediately below the else keyword.

In the example below, if you have more than one condition to test, ***else if*** statement is appropriate to use. Here we test for numerous conditions, if all the condition is false, the statement within the else clause is executed.

```
<body>
<script type="text/javascript">
var score = 58;

if (score >= 70) {
    document.write('You got an A');
} else if (score >= 60) {
    document.write('You got a B');
} else if (score >= 50) {
    document.write('You got a C');
} else if (score >= 45) {
    document.write('You got a D');
} else if (score >= 40) {
    document.write('You got an E');
} else{
    document.write('You got an F');
}
</script>
</body>
```

17.6.2 The while statement

The while statement functions in the following way:

```
while (condition) {
    statements
}
```

In this example, the series of statements within the brackets are executed for as long as the specified condition holds true.

```
count=1;
track="";
while ( count <= 20 ) {
    track += "repeat "; count++;
}
```

```
<body>
<script type="text/javascript">
```

```
var x = 10;
while(x <= 15){
    x = x + 1;
    document.write(x+'<br>');
}
</script>
</body>
```

17.6.3 The for statement

The **for statement** is much like the while statement. The real difference is that **for statement** is more specifically structured to handle numerical loops, while the **while statement**, although capable of the same is more flexible to function on conditions other than numerical loops. This loop takes 3 different statements.

1. Variable name
2. Ending point
3. How much time do you want to execute the code and print it out.

Example

```
<script type= "text/javascript">
For(x=0;x<10;x++){
document.write("I love Chidinma Obiano ");
}
</script>
```

The above code will print out 9 results, starting from 0

Eg.

```
track = '';
for (count = 1; count <= 20; count++) {
track += "repeat ";
}
```

17.6.4 The 'Switch' Statement

Switch statement takes one variable and test for all the possible scenarios. This statement is mostly use alongside with **break** statement which makes the program to break out of the test as soon as the program sees the answer and **default** statement which is executed if all the match is not found.

Example

```
<script type= "text/javascript">
let girl= "Mfon";
switch(girl){
case "Mfon":
document.write("You must link going to Spar");
```

```
break;  
case "Chidinma":  
document.write("You must like processing transcripts at YABATECH");  
break;  
default:  
document.write("You don't like anyone of them");  
}  
</script>
```

Switch statement save you from writing a whole bunch of ***if...else*** statement if you have multiple variables to test.

```
switch(myVar) {  
    case 1:  
        //if myVar is 1 this is executed  
    case 'sample':  
        //if myVar is 'sample' (or 1, see the next paragraph)  
        //this is executed  
    case false:  
        //if myVar is false (or 1 or 'sample', see the next paragraph)  
        //this is executed  
    default:  
        //if myVar does not satisfy any case, (or if it is  
        //1 or 'sample' or false, see the next paragraph)  
        //this is executed  
}
```

If a case is satisfied, the code beyond that ***case*** statement will also be executed unless the ***break*** statement is used. In the above example, if myVar is 1, the code for case 'sample', case false and default will all be executed as well. The solution is to use ***break*** statement as follows (The use of break statement is described below):

```
switch(myVar) {  
    case 1:  
        //if myVar is 1 this is executed  
        break;  
    case 'sample':  
        //if myVar is 'sample' this is executed  
        break;  
    case false:  
        //if myVar is false this is executed  
        break;  
    default:  
        //if myVar does not satisfy any case, this is executed  
        //break; is unnecessary here as there are no cases following this  
}
```

17.7 Functions in JavaScript

Function is a mini program or a block of code designs to perform a specific task, it is executed when it is invoked(call).

In order to make a function, first type the keyword **function**, second give a function a name then follow by two () then { //where the function codes go in} .

17.7.1 General Function Syntax

There are different ways of writing functions in JavaScript as shown below;

1. Function without parameters

```
function nameOfFunction() {
    // codes to be executed when the function is called
}
```

Eg.

```
var x = 20;
var y = 10;
function sum() {
    var z = x + y;
    document.write(z+'<br>');
}
sum();
```

2. Function with multiple parameters

```
function nameOfFunction(a,b,c) {
    // codes to be executed when the function is called
}
```

Eg.

```
//function with argument
function subtraction(a, b) {
    var c = a - b;
    document.write(c+'<br>');
}
subtraction(10, 4);
```

3. Anonymous function

The anonymous functions don't have **names**. They need to be tied to something: *variable or an event to run*.

```
nameOfFunction = function (listOfVariableNames) {
    // codes to be executed when the function is called
};
```

Eg.

```
let oddNumber = function (number){  
    If(number%2==0){  
        return "The number is even number"  
    }else{  
        return "The number is an odd number";  
    }  
}
```

⇒ Invoked function expression

Invoked function expression runs as soon as the browser encounters it. The benefit of this function is that it **runs immediately** where it is located in the code and produces a direct output.

```
let nameOfFunction = function anotherNameForTheFunction(listOfVariableNames) {  
    // codes to be executed when the function is called};
```

```
<body>  
<script type="text/javascript">  
  
let factorial = {  
    'factitCall': function fact(number){  
        if(number<=1){  
            return 1;  
        }  
        return number*fact(number-1);  
    }  
};  
document.write(factorial.factitCall(5));  
</script>  
</body>
```

17.7.2 Calling a Function

Functions are called using

```
nameOfFunction(listOfVariables);  
or  
window.nameOfFunction(listOfVariables);  
or  
object.onEventName = nameOfFunction;
```

⇒ Passing variables to functions

Variables passed to a function are known as arguments. When a function is called, the variables or values passed to it in the brackets are assigned to the variable names in the brackets of the function definition.

```
function checkval(passvar) {  
    //if I ran the function using the command "checkval('hello')"  
    //then passvar would take on the value 'hello'  
    if( passvar != "" ) {  
        document.myform.mytextinput.value = passvar;  
    }  
}
```

This function, when called, will set the value of a text input to whatever value of passvar was, as long as passvar was not blank.

⇒ Passing more than one variable to a function

```
function functionName(variable1,variable2,variable3,etc.) {  
    function code  
}  
functionName(5,6,7,etc.);
```

⇒ Passing no variables to a function

```
function functionName() {  
    //function code  
}  
functionName();
```

⇒ Using return value in a function

```
Function addNumbers(a,b){  
    var c=a+b;  
    return c;  
}  
document.write(addNumber(3,6));
```

When the function is called, it returns the value of c which is 9 this implies that $c=a+b$; in order to get the value printed out, we need to use `document.write()` or `document.getElementById()` to printout the value to the screen.

Eg:

```
<body>
<script type="text/javascript">
//function declaration
    var x = 20;
    var y = 10;
    function sum() {
        var z = x + y;
        document.write(z+'<br>');
    }
    sum();

//function with argument
function subtraction(a, b) {
    var c = a - b;
    document.write(c+'<br>');
}
subtraction(10, 4);

//return function
function square(a) {
    var result = a * a;
    return result;
}
document.write(square(78));
</script>
</body>
```

It is important to make sure that if the return statement is used to return a value, you must ensure that in all cases, the function returns a value.

17.8.4 Variables Scope

Javascript has 3 type of variable scopes such as Block scope, Function scope and Global scope.

Local variable: any variable declare within JavaScript function becomes a local variable which cannot be accessed out the declared function. This implies that local variables have function scope.

Eg.

```
function subtraction() {
    var a,b,c;
    document.write(a,b,c);
}
```

Global variable: When a variable is declared outside a function, it becomes a global variable which can be accessed within the declared page.

Eg.

```
<script type="text/javascript">
//global scope
    var x = 20;
    var y = 10;
    var z = x - y;
    function sum() {
        var z = x + y;
        document.write(z+'<br>');
    }
    sum();

    function subtraction() {
        document.write(z);
    }
    subtraction();
</script>
```

Block Scope

Block scoping means **declaring a variable, not just inside a function**, but around any curly brackets like if statements or loops. **let** and **const** keywords are used for declaring variables that are block scope.

Eg.

```
1 function loop(){
2     for (let i=0; i < 5; i++){
3         console.log(i);
4     }
5     console.log("final", i)
6 }
7 loop();
```

The line of code **console.log("final", i)** will flag error that i is not defined because i is a block scope variable within the for loop block{}.

Returning an Array in a function

The function returned only one parameter, but what if you need to return multiple parameters? You can do this by returning an array;

Returning an array of values

```
<script>
words = fixNames("the", "DALLAS", "CowBoys")
for (j = 0 ; j < words.length ; ++j)
document.write(words[j] + "<br>")
function fixNames() {
var s = new Array()
for (j = 0 ; j < fixNames.arguments.length ; ++j)
s[j] = fixNames.arguments[j].charAt(0).toUpperCase() +
fixNames.arguments[j].substr(1).toLowerCase()
return s
}
</script>
```

Here the variable words is automatically defined as an array and populated with the returned result of a call to the function fixNames. Then a for loop iterates through the array and displays each member.

After each word has been processed, it is stored as an element of this array, which is returned by the return statement. This function enables the extraction of individual parameters from its returned values, like the following (the output from which is simply The Cowboys):

```
words = fixNames("the", "DALLAS", "CowBoys")
document.write(words[0] + " " + words[2])
```

creates an array, loads it with some values, and then displays them.

```
<script>
numbers = []
numbers.push("One")
numbers.push("Two")
numbers.push("Three")
for (j = 0 ; j < numbers.length ; ++j)
document.write("Element " + j + " = " + numbers[j] + "<br>")
</script>
```

The output from this script is as follows:

Element 0 = One

Element 1 = Two

Element 2 = Three

JavaScript Events

Javascript interaction with HTML is handled via event that occur when a user or a browser manipulate a web page. Events could be loading a page, clicking a button, resizing a windows, closing a window, etc.

Common HTML Events

The table below shows a list of common HTML events:

| Event | Description |
|-------------|--|
| Onchange | An HTML element has been changed |
| Onclick | The user clicks an HTML element |
| onmouseover | The user moves the mouse over an HTML element |
| onmouseout | The user moves the mouse away from an HTML element |
| onkeydown | The user pushes a keyboard key |
| Onload | The browser has finished loading the page |

In the following example, an onclick attribute (with code), is added to a <button> element:

```
<button onclick="document.getElementById('currentDate').innerHTML = Date()">The time  
is?</button>
```

```
<p id="currentDate"></p>
```

Output

The time is?

Sun Aug 21 2022 12:32:30 GMT+0100 (GMT Summer Time)

```
<script>  
    function greetings() {  
        alert('Hello World, welcome to event in Javascript!');  
    }  
</script>  
</head>  
<body>  
    <button type="button" onclick="greetings()">Please click here! </button>
```

</body>

Output



<body>

⇒ Onkeyup Event and Syntax

This is a scenario where you press a keyboard event and it performs as per your code logic.

Example

body

Enter your First name:

```
<input type="text" id="firstname" onkeyup="onKeyUpFunction()>
<p>The first name is: <span id="testkey"></span></p>
<script>
function onKeyUpFunction() {
    let input = document.getElementById("firstname").value;
```

docu

}

```
</script>
```



⇒ Onmouseover/Onmouseout Events and Syntax

We can use this event for hovering the mouse pointer when we put the cursor and it performs as per the logic of the element which is connected to and its child's elements as well out moving the mouse cursor away from the element.

<body>

```
<h1 id="demoEvent">Test Mouse over me Event</h1>
```

```
<script>

document.getElementById("demoEvent").onmouseover = function() {mouseOverEvent()};

function mouseOverEvent() {
    document.getElementById("demoEvent").style.color = "Red";
}
document.getElementById("demoEvent").onmouseout = function() {mouseOut()};
function mouseOut() {
    document.getElementById("demoEvent").style.color = "Black";

}

</script>
</body>
```

Output

Test Mouse over me Event **Test Mouse over me Event**

Onchange Event

The *onchange event* occurs when the value of an element has been changed as shown in the example below:

```
<body>

Please Enter name: <input type="text" id="fullname">

<script>

document.getElementById("fullname").onchange = function() { myFullscreen()};

function myFullscreen() {

    var x = document.getElementById("fullname");

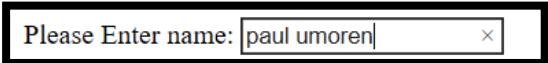
    x.value = x.value.toUpperCase();

}

</script>

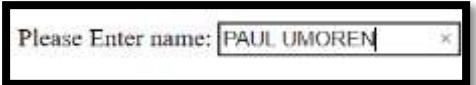
</body>
```

Output



Please Enter name: paul umoren

On changing, the output becomes



Please Enter name: PAUL UMOREN

Onfocus Event

This event performs when the given instruction receives the focus as per the change or click event. In the example below, we use an eventListener to attach a focus event to an input field.

<p>This is the best scenario to uses the addEventListener() function to attach a "focus" event to an input element box.</p>

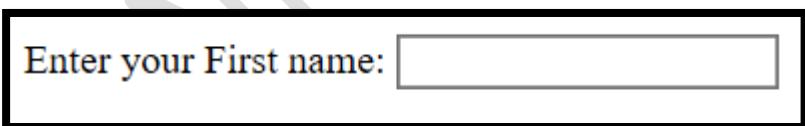
Enter your First name: <input type="text" id="Firstname">
<script>
document.getElementById("Firstname").addEventListener("focus", myFunction);

function myFunction() {

 document.getElementById("Firstname").style.backgroundColor = "gray";

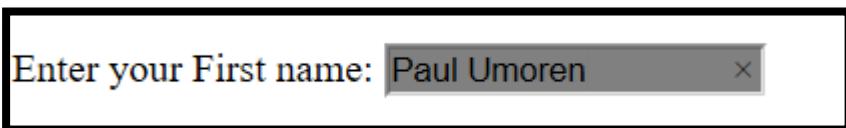
}

</script>
Output



Enter your First name:

Output after onfocus event



Enter your First name: Paul Umoren

JavaScript addEventListener()

The `addEventListener()` method is an inbuilt function of [JavaScript](#) that we can use to add multiple event handlers to a particular element without overwriting the existing event handlers.

Syntax

```
element.addEventListener(event, function, useCapture);
```

the function has three parameters, the parameters ***event*** and ***function*** are widely used. The third parameter is optional to define.

Parameter Values

The event: It is a required parameter. It can be defined as a string that specifies the event's name.

The function: It is also a required parameter. It is a JavaScript function which responds to the event occur.

The useCapture: It is an optional parameter. It is a Boolean type value that specifies whether the event is executed in the bubbling or capturing phase. Its possible values are **true** and **false**. When it is set to true, the event handler executes in the capturing phase. When it is set to false, the handler executes in the bubbling phase. Its default value is **false**.

Example

```
<!DOCTYPE html>
<html>
<body>
<p> Example of the addEventListener() method. </p>
<p> Click the following button to see the effect. </p>
<button id = "btn"> Click me </button>
<p id = "para"></p>
<script>
document.getElementById("btn").addEventListener("click", fun);
function fun() {
```

```
document.getElementById("para").innerHTML = "Hello World" + "<br>" + "Welcome to the  
pinfotechnologies.net";  
}  
</script>  
</body>  
</html>
```

Adding multiple events to the same element.

```
<!DOCTYPE html>  
<html>  
<body>  
<p> This is an example of adding multiple events to the same element. </p>  
<p> Click the following button to see the effect. </p>  
<button id = "btn">Click me </button>  
<p id = "para"></p>  
<p id = "para1"></p>  
<script>  
function fun() {  
    alert("Welcome to the pinfotechnologies.net");  
}  
  
function fun1() {  
    document.getElementById("para").innerHTML = "This is second function";  
}  
function fun2() {  
    document.getElementById("para1").innerHTML = "This is third function";  
}  
var mybtn = document.getElementById("btn");  
mybtn.addEventListener("click", fun);  
mybtn.addEventListener("click", fun1);  
mybtn.addEventListener("click", fun2);  
</script>  
</body>  
</html>
```

Event Bubbling or Event Capturing

In HTML DOM, **Bubbling** and **Capturing** are the two ways of event propagation. We can understand these ways by taking an example.

Suppose we have a div element and a paragraph element inside it, and we are applying the "click" event to both of them using the **addEventListener()** method. So, in **Bubbling**, the event of paragraph element is handled first, and then the div element's event is handled. It means that in bubbling, the inner element's event is handled first, and then the outermost element's event will be handled.

In **Capturing** the event of div element is handled first, and then the paragraph element's event is handled. It means that in capturing the outer element's event is handled first, and then the innermost element's event will be handled.

Example

```
addEventListener(event, function, useCapture);
```

We can specify the propagation using the **useCapture** parameter. When it is set to false (which is its default value), then the event uses bubbling propagation, and when it is set to true, there is the capturing propagation.

Example

In this example, there are two div elements. We can see the bubbling effect on the first div element and the capturing effect on the second div element.

When we double click the span element of the first div element, then the span element's event is handled first than the div element. It is called *bubbling*.

But when we double click the span element of the second div element, then the div element's event is handled first than the span element. It is called *capturing*.

```
<!DOCTYPE html>
<html>
<head>
<style>
div{
background-color: lightblue;
border: 2px solid red;
font-size: 25px;
text-align: center;
}
span{
border: 2px solid blue;
```

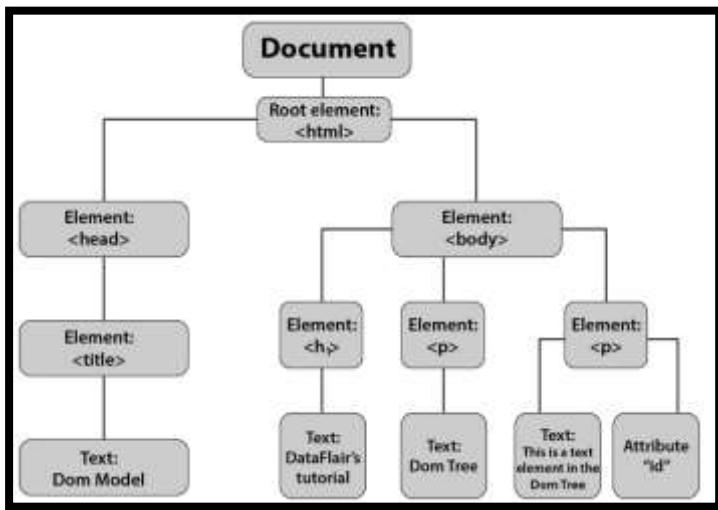
```
}

</style>
</head>
<body>
<h1> Bubbling </h1>
<div id = "d1">
This is a div element.
<br><br>
<span id = "s1"> This is a span element. </span>
</div>
<h1> Capturing </h1>
<div id = "d2"> This is a div element.
<br><br>
<span id = "s2"> This is a span element. </span>
</div>
<script>
document.getElementById('d1').addEventListener('dblclick', function() {alert('You have double clicked on div element')}, false);
document.getElementById('s1').addEventListener('dblclick', function() {alert('You have double clicked on span element')}, false);
document.getElementById('d2').addEventListener('dblclick', function() {alert('You have double clicked on div element')}, true);
document.getElementById('s2').addEventListener('dblclick', function() {alert('You have double clicked on span element')}, true);
</script>
</body>
</html>
```

JavaScript Document Object Model (DOM)

A **Document Object Model** is a programming interface for **HTML (HyperText Markup Language)** and **XML (eXtensible Markup Language)** documents that provides a data representation comprising all the objects, depicting the structure and content of the document on the web.

Javascript DOM Tree



Note: **DOM is a separate set of rules neither HTML or Javascript**

A DOM tree consists of four main types of nodes:

- ⇒ **Document node:** this is at the top of the tree and it represents the entire browser page. It is the starting point in the DOM tree. You have to navigate through the document node to access any other node in the DOM tree.
- ⇒ **Element node:** All the HTML elements like heading tags (`<h1>` to `<h6>`) and paragraph tags (`<p>`) in the page create an element node in the tree. You use these nodes to gain access to the elements' attribute and text nodes
- ⇒ **Attribute node:** When the opening tags in the HTML document contain attributes, the tree represents them as attribute nodes.
- ⇒ **Text node:** Once you have access to the element node, you can reach the text content within that element, stored inside the text nodes of the DOM tree. These nodes cannot have child nodes. Thus, a text node always creates a new branch in the DOM tree, and no further branches come out of it.

Accessing the Elements

DOM Queries are the methods that find elements in the DOM tree. They return one element or a collection of elements in a NodeList.

Example

HTML

`<body>`

```
<div id="page">

<h1 id="header">TO DO LIST</h1>
<ul>
    <h2>PINFOTECH's JavaScript Tutorial</h2>
    <li id="one" class="mandatory">Learning the concepts</li>
    <li id="two" class="mandatory">Practising the codes</li>
    <li id="three">Taking quizzes</li>
    <li id="four">Solving Interview Questions</li>
</ul>

<!-- JavaScript code -->
</div>
</body>
CSS

<!-- CSS styling -->
<style type="text/css">
#page{
    max-width: 400px;
    margin: 20px auto 20px auto;
    border: 3px solid aqua;
}
h1, h2, ul{
    padding: auto;
}
p{
    text-align: center;
    font-size: large;
    font-weight: bold;
}
</style>
```

Methods to Select an Individual Element Node

Following are the methods to select an individual element in the tree:

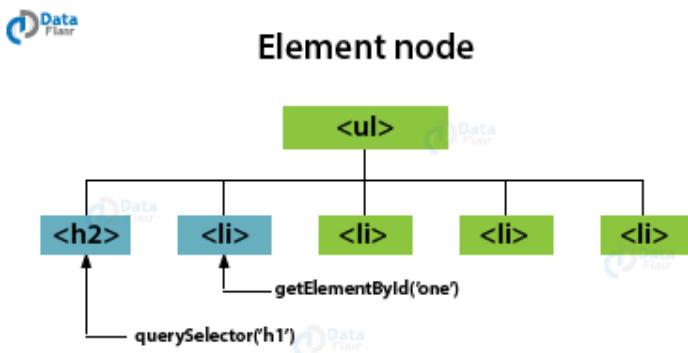
- **getElementById('id')**: Uses the unique value of the element's id attribute. The HTML must have an id attribute for the method to select it.

For example – getElementById('one')

- **querySelector('css selector')**: Uses a CSS selector, returns the first matching element.

For example – querySelector('h1')

The code below combines these methods to add styling to the webpage.



Code:

Javascript

```
<script type="text/JavaScript">

    document.getElementById('one').style.color="maroon"; //change font color

    document.querySelector("h1").style.backgroundColor = "blue"; //change background color

    document.querySelector("p").style.color="red"; // change the text color to red

</script>
```

Output

TO DO LIST

PINFOTECH's JavaScript Tutorial

- Learning the concepts
- Practising the codes
- Taking quizzes
- Solving Interview Questions

This is to test for querySelector paragraph example

To Select Multiple Elements (NodeLists)

There are three common ways to select multiple elements in the tree:

- **getElementsByClassName()**: Selects all the elements that have a specified value for the class attribute.

For example – `getElementsByClassName('mandatory')`

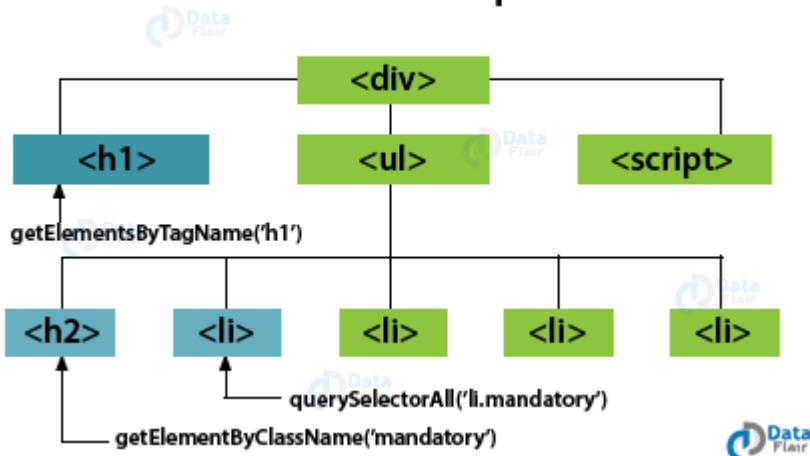
- **getElementsByTagName()**: Selects all the elements that have the specified tag names.

For example – `getElementsByTagName('h1')`

- **querySelectorAll()**: Uses a CSS selector, returns all the matching elements.

For example – `querySelectorAll('li.mandatory')`

Methods to Select Multiple Elements



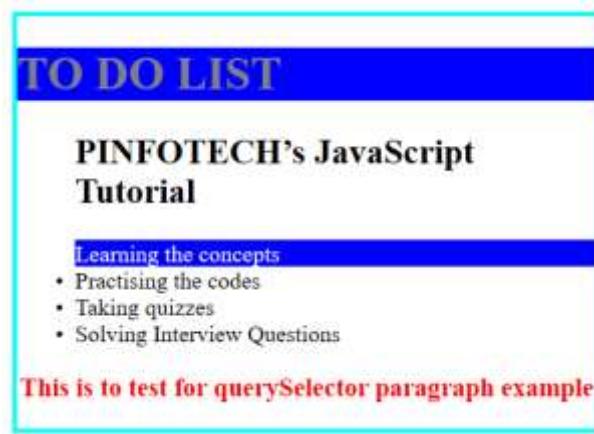
The above methods always return a NodeList, even if the list contains only a single element, so it is important to use square brackets [] if you want to access any element in the NodeList. The following code explains all the methods of returning a NodeList.

```
var list_qs = document.querySelectorAll("li.mandatory"); //NodeList
list_qs[0].style.backgroundColor = "blue"; //change background color

var list_cn = document.getElementsByClassName('mandatory'); //NodeList
list_cn[0].style.color="white"; //change font color

var list_tn = document.getElementsByTagName('h1'); //NodeList
```

```
list_tn[0].style.color="gray"; //change font color
```

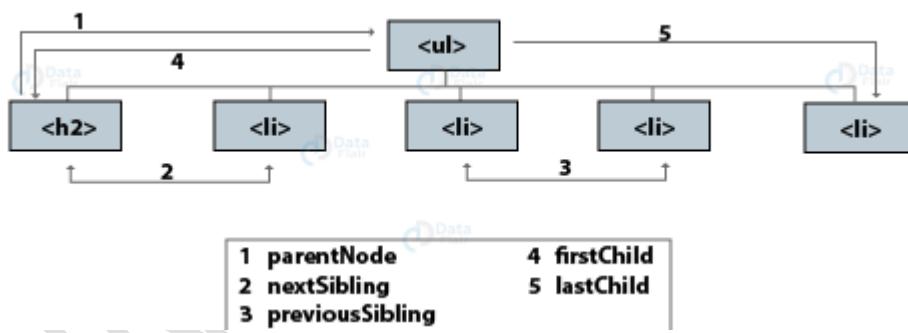


Traversing between Element Nodes

You can navigate between related nodes using the following techniques available

- **parentNode**: Selects the parent of the current element node (returns a single element).
- **previousSibling/ nextSibling**: Selects the previous or next sibling from the tree.
- **firstChild/ lastChild**: Selects the first or last child of the current element.

The above methods are illustrated in the diagram below;



DOM Manipulation

To add/remove HTML content is DOM Manipulation. The following three steps are needed to add HTML content to your webpage:

- Create a new element using **createElement()**.
- Give it content by creating a text node using **createTextNode()** and adding it to the element using the **appendChild()** method.
- Add the element to the DOM tree by finding the location and then using **appendChild()**.

To remove an element from the webpage, the steps are as follows:

- Store the element you want to remove in a variable.
- Store the parent of that element in another variable.
- Remove the element from its containing element using the **removeChild()** method.

Let's implement these methods with the help of a code. We will use the same HTML and CSS code we discussed at the beginning of this tutorial. All you need to do is change the JavaScript in your program. In this example, we will add a paragraph **<p>** tag in the page, and remove the **<h1>** tag from it.

```
<script type="text/JavaScript">

    //Adding new element
    var newEl = document.createElement('p'); //create element node
    var newText = document.createTextNode('Node added using DOM manipulation.');//create text node
    newEl.appendChild(newText); //add text node to element node
    var position = document.getElementById("page"); //find the position where you want to add the node
    position.appendChild(newEl); //add element in that position

    //Removing an element
    var removeEl = document.getElementById('header'); //store the element you want to remove
    var containerEl = document.getElementById('page'); //find the element which contains the above element
    containerEl.removeChild(removeEl); //remove the element

</script>
```

JavaScript Class

In terms of OOP, we often call classes as a ‘**blueprint**’ of an object. It determines what the properties of an object are and how it behaves. Note that javascript class is not an object; it is a template for Javascript objects.

How to define a class in JavaScript

Class is defined with a keyword **class** instead of function. Though similar to **functions**, the class syntax has two components:

- class declarations
- class expressions

```
class className{
    constructor(parameters){
        //properties
    }
}
```

```
//other class methods  
}
```

Or

```
let className = class{  
    constructor(parameters){  
        //properties  
    }  
    //other class methods  
}
```

Like a function expression, a class expression can be named or anonymous. If named, the name is only visible inside the class expression

The Constructor Method

The constructor method is a special method:

- It has to have the exact name "constructor"
- It is executed automatically when a new object is created
- It is used to initialize object properties

If you do not define a constructor method, JavaScript will add an empty constructor method.

The new keyword

The **new className()** creates a new object with all the properties and methods defined in the class. All the properties associated with the objects need to be defined inside the constructor() function.

The example below shows the basic concept of a class definition.

```
<html>  
  <body>  
  
    <script>  
      class Shape{ //defining a class  
        constructor(){ //method called automatically  
          document.write("Class <i>Shape</i> created.")  
  
          this.name = "circle";  
          this.color = "black";  
          this.radius = 5; //adding properties  
        }  
      }  
    </script>  
  </body>  
</html>
```

```
        }
    }
var shape1 = new Shape(); //creating a new instance of the class
</script>

</body>
</html>
```

In the above program, we defined a class Shape with a constructor() function. We instantiated the class i.e. created an object named **shape1**.

Class Body and Methods

All the code inside the **curly brackets { }** after the class name (class declaration) or **class** keyword (class expression) is considered to be the body of that class

Class Methods

Class methods are created with the same syntax as object methods as shown in the example below;

Steps:

Use the keyword **class** to create a class.

Always add a **constructor()** method.

Then add any number of methods.

Syntax

```
class ClassName {
    constructor() { ... }
    method_1() { ... }
    method_2() { ... }
    method_3() { ... }
}
```

Now let us create a Class with a method named "age", that returns the Car age:

Example

```
class Car {
    constructor(name, year) {
        this.name = name;
```

```
this.year = year;
}
age() {
    let date = new Date();
    return date.getFullYear() - this.year;
}
let myCar = new Car("Ford", 2014);
document.getElementById("demo").innerHTML = "My car is " + myCar.age() + " years old.";
```

Sending parameters to a class method

We can send parameters to class method as shown in the example below;

```
class Car {
    constructor(name, year) {
        this.name = name;
        this.year = year;
    }
    age(x) {
        return x - this.year;
    }
}
let date = new Date();
let year = date.getFullYear();

let myCar = new Car("Ford", 2014);
document.getElementById("demo").innerHTML=
"My car is " + myCar.age(year) + " years old.;"
```

Prototype Methods: Getters and Setters

Getters and setters define methods on a class that is then used as a class property.

Getters get the function linked to an object property when we attempt to look up the property, using the keyword **get**. Setters set the function we need to call when we access the object property by binding them together with the keyword **set**.

The syntax of a getter is as follows:

```
get getName( ){ }
```

The syntax of a setter looks something like this:

```
set setterName(getterName){ }
```

These methods help us share the data of our class between different class instances (objects).

Example

```
<html>
  <body>
    <h3>Pinfo Tech: Implementing Getters and Setters</h3>
    <script>
      class Circle{ //class declaration
        //constructor method with a single parameter
        constructor(radius){
          this.radius = radius; //setting radius value to object
        }
        get area(){ //getter
          return 3.14*Math.pow(this.radius, 2); //returning area using radius
        }
        set area(area){ //setter
          this.radius = Math.sqrt(area) / 3.14; //returning radius using area
        }
      }
      var circle1 = new Circle(5); //new class instance
      document.write("Area of circle with radius "+circle1.radius + " is "+circle1.area
      + "<br>");
      document.write("Radius of circle with area "+circle1.area + " is "+
      circle1.radius + "<br>");
    </script>
  </body>
</html>
```

Static Methods

A static method is defined in a class, but it isn't a part of the object instantiated once it's created. It doesn't require any instance and is accessed directly by the class name rather than the object name.

The static methods let us allocate memory to our program and defines various keywords and methods that JavaScript interpreter already understands. The code below shows the working of a static method clearly.

EXAMPLE

```
<html>
<body>

<script>
class tellCourse{ //defining a class
constructor(){ //method called automatically
    this.course = "JavaScript"; //adding course name
}
normalMsg(){ //class method
    return "PinfoTech says learn " + this.course + " today.<br>";
}
static staticMsg(){ //static method
    return "Which course to choose?<br>"}
}
var course1 = new tellCourse(); //creating a new instance of the class
document.write(tellCourse.staticMsg()); //calling static method
document.write(course1.normalMsg()); //calling class method
</script>

</body>
</html>
```

Inheritance

Inheritance is an important concept in Object-Oriented Programming. This is a mechanism that allows the child class to acquire all the properties and methods of a parent class (except private properties and methods). We use the keyword **extends** in a class declaration or a class expression to create a child class of another class.

Note: If there is a constructor present in the subclass, it needs to first call **super()** before using “**this**”

```
<html>
<body>
<script>
class Shape{ //defining a class
constructor(color){
    this.color = color;
}
msg(){ //parent class method
    document.write("The color of the shape is " + this.color + "<br>");
}
}
class Circle extends Shape{ //inheritance
constructor(color){
```

```
super(color); //using the constructor of parent class to set property
}
msg(){ //child class method
    document.write("The color of the circle is " + this.color + "<br>");
}
}
var c1 = new Circle("Blue"); //creating a new instance of the child class
c1.msg(); //calling child method
</script>

</body>
</html>
```

JavaScript Objects

JavaScript objects are containers for named values called properties. *Object Methods.* Objects can also have methods. Object values are written as **name : value** pairs (name and value separated by a colon).

The following example creates a new JavaScript object with four properties:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

    <p id="demo"></p>
    <script>
        const person={firstName:"Paul", lastName:"Umoren", age:30,
eyeColor:"blue"};

        document.getElementById("demo").innerHTML = "Your name is "+
person.firstName+ " "+person.lastName+" and "+
"your age is "+person.age + " with a color of eye "+person.eyeColor;
    </script>
</body>
</html>
```

JavaScript for...in Loop

The JavaScript for...in statement loops through the properties of an object.

Syntax

```
for (let variable in object) {  
    // code to be executed  
}
```

Example

```
<!DOCTYPE html>  
<html>  
<body>  
<h2>JavaScript Objects</h2>  
<p>Display object properties:</p>  
<p id="demo"></p>  
<script>  
const person = {  
    name: "paul",  
    age: 30,  
    city: "Calabar",  
    state:"Cross River"  
};  
document.getElementById("demo").innerHTML = person.name + ', ' + person.age +  
, " + person.city+ ', ' + person.state;  
</script>  
</body>  
</html>
```

Nested Objects

```
myObj = {  
    name:"John",  
    age:30,  
    cars: {  
        car1:"Ford",  
        car2:"BMW",  
        car3:"Fiat"  
    }  
}
```

You can access nested objects using the dot notation or the bracket notation:

Example

```
myObj.cars.car2;
```

or:

Example

```
myObj.cars["car2"];
```

or:

Example

```
myObj["cars"]["car2"];
```

or:

Example

```
let p1 = "cars";
let p2 = "car2";
myObj[p1][p2];
```

Nested Arrays and Objects

Values in objects can be arrays, and values in arrays can be objects as it is shown in the example below;

```
const myObj = {
  name: "John",
  age: 30,
  cars: [
    {name: "Ford", models: ["Fiesta", "Focus", "Mustang"]},
    {name: "BMW", models: ["320", "X3", "X5"]},
    {name: "Fiat", models: ["500", "Panda"]}
  ]
}
```

To access arrays inside arrays, use a for-in loop for each array:

```
for (let i in myObj.cars) {
  x += "<h1>" + myObj.cars[i].name + "</h1>";
  for (let j in myObj.cars[i].models) {
    x += myObj.cars[i].models[j];
```

```
}
```

Example

```
<!DOCTYPE html>
<html>
<body>

<h2>Nested JavaScript Objects and Arrays.</h2>
<p id="demo"></p>

<script>
let x = '';
const myObj = {
  name: "John",
  age: 30,
  cars: [
    {name:"Ford", models:["Fiesta", "Focus", "Mustang"]},
    {name:"BMW", models:["320", "X3", "X5"]},
    {name:"Fiat", models:["500", "Panda"]}
  ]
}

for (let i in myObj.cars) {
  x += "<h2>" + myObj.cars[i].name + "</h2>";
  for (let j in myObj.cars[i].models) {
    x += myObj.cars[i].models[j] + "<br>";
  }
}
document.getElementById("demo").innerHTML = x;
</script>
</body>
</html>
```

JavaScript Object Methods

Example 1

```
<!DOCTYPE html>
<html>
<body>

<h1>The JavaScript <i>this</i> Keyword</h1>
<p>In this example, <b>this</b> refers to the <b>person</b> object.</p>
<p>Because <b>fullName</b> is a method of the person object.</p>
```

```
<p id="demo"></p>

<script>
// Create an object:
const person = {
  firstName: "Paul",
  lastName: "Umoren",
  id: 5066,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};

// Display data from the object:
document.getElementById("demo").innerHTML = person.fullName();
</script>

</body>
</html>
```

Example 2

Adding a new method to an object is easy:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Objects</h2>
<p id="demo"></p>

<script>
const person = {
  firstName: "Paul",
  lastName: "Umoren",
  id: 506
};
person.name = function() {
  return this.firstName + " " + this.lastName;
};
document.getElementById("demo").innerHTML =
"My name is " + person.name();
</script>
</body>
</html>
```

Canva

The **Canvas API** provides a means for drawing graphics using [JavaScript](#) and the [HTML <canvas>](#) element. Canvas can also be used for animation, game graphics, data visualization, photo manipulation, and real-time video processing.

HTMLCanvasElement

This provides properties and methods for manipulating the layout and presentation of [<canvas>](#) elements, as well as inherits the properties and methods of the [HTMLElement](#) interface.

HTML5 Canvas - Drawing Rectangles

here are three methods that draw rectangles on the canvas –

| Sr.No. | Method and Description |
|--------|--|
| 1 | fillRect(x,y,width,height) This method draws a filled rectangle. |
| 2 | strokeRect(x,y,width,height) This method draws a rectangular outline. |
| 3 | clearRect(x,y,width,height) This method clears the specified area and makes it fully transparent |

Here *x* and *y* specify the position on the canvas (relative to the origin) of the top-left corner of the rectangle and *width* and *height* are width and height of the rectangle.

```
<!DOCTYPE HTML>

<html>
  <head>
    <style>
      #test {
        width: 100px;
        height:100px;
        margin: 0px auto;
      }
    </style>

    <script type = "text/javascript">
      function drawShape() {

        // Get the canvas element using the DOM
```

```
var canvas = document.getElementById('mycanvas');

// Make sure we don't execute when canvas isn't supported
if (canvas.getContext) {

    // use getContext to use the canvas for drawing
    var ctx = canvas.getContext('2d');

    // Draw shapes
    ctx.fillRect(25,25,100,100);
    ctx.clearRect(45,45,60,60);
    ctx.strokeRect(50,50,50,50);
} else {
    alert('You need Safari or Firefox 1.5+ to see this demo.');
}
}

</script>
</head>

<body id = "test" onload = "drawShape();">
    <canvas id = "mycanvas"></canvas>
</body>

</html>
```

HTML5 Canvas - Drawing Paths

| S.No. | Method and Description |
|-------|---|
| 1 | beginPath() This method resets the current path. |
| 2 | moveTo(x, y) This method creates a new subpath with the given point. |
| 3 | closePath() This method marks the current subpath as closed, and starts a new subpath with a point the same as the start and end of the newly closed subpath. |
| 4 | fill() This method fills the subpaths with the current fill style. |
| 5 | stroke() |

| | |
|---|--|
| | This method strokes the subpaths with the current stroke style. |
| | arc(x, y, radius, startAngle, endAngle, anticlockwise) |
| 6 | Adds points to the subpath such that the arc described by the circumference of the circle described by the arguments, starting at the given start angle and ending at the given end angle, going in the given direction, is added to the path, connected to the previous point by a straight line. |

Example

```
<!DOCTYPE HTML>

<html>
  <head>

    <style>
      #test {
        width: 100px;
        height:100px;
        margin: 0px auto;
      }
    </style>

    <script type = "text/javascript">
      function drawShape() {

        // get the canvas element using the DOM
        var canvas = document.getElementById('mycanvas');

        // Make sure we don't execute when canvas isn't supported
        if (canvas.getContext) {

          // use getContext to use the canvas for drawing
          var ctx = canvas.getContext('2d');

          // Draw shapes
          ctx.beginPath();
          ctx.arc(75,75,50,0,Math.PI*2,true); // Outer circle

          ctx.moveTo(110,75);
          ctx.arc(75,75,35,0,Math.PI,false); // Mouth

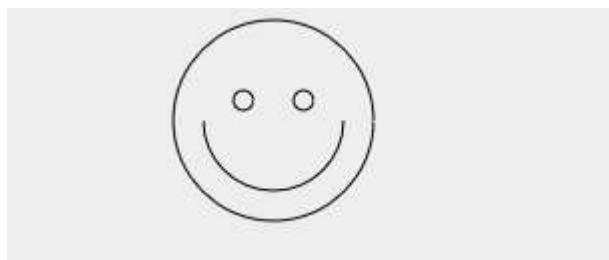
          ctx.moveTo(65,65);
          ctx.arc(60,65,5,0,Math.PI*2,true); // Left eye
        }
      }
    </script>
  </head>
  <body>
    <div id="test"></div>
  </body>
</html>
```

```
ctx.moveTo(95,65);
ctx.arc(90,65,5,0,Math.PI*2,true); // Right eye
ctx.stroke();
} else {
    alert('You need Safari or Firefox 1.5+ to see this demo.');
}
}
</script>
</head>

<body id = "test" onload = "drawShape();">
    <canvas id = "mycanvas"></canvas>
</body>

</html>
```

Result



HTML5 Canvas - Drawing Lines

| Sr.No. | Method and Description |
|--------|---|
| 1 | beginPath() This method resets the current path. |
| 2 | moveTo(x, y) This method creates a new subpath with the given point. |
| 3 | closePath() This method marks the current subpath as closed, and starts a new subpath with a point the same as the start and end of the newly closed subpath. |
| 4 | fill() This method fills the subpaths with the current fill style. |
| 5 | stroke() |

| | |
|---|---|
| | This method strokes the subpaths with the current stroke style. |
| 6 | lineTo(x, y) This method adds the given point to the current subpath, connected to the previous one by a straight line. |

```
<!DOCTYPE HTML>

<html>
  <head>

    <style>
      #test {
        width: 100px;
        height:100px;
        margin: 0px auto;
      }
    </style>

    <script type = "text/javascript">
      function drawShape() {

        // get the canvas element using the DOM
        var canvas = document.getElementById('mycanvas');

        // Make sure we don't execute when canvas isn't supported
        if (canvas.getContext) {

          // use getContext to use the canvas for drawing
          var ctx = canvas.getContext('2d');

          // Filled triangle
          ctx.beginPath();
          ctx.moveTo(25,25);
          ctx.lineTo(105,25);
          ctx.lineTo(25,105);
          ctx.fill();

          // Stroked triangle
          ctx.beginPath();
          ctx.moveTo(125,125);
          ctx.lineTo(125,45);
          ctx.lineTo(45,125);
          ctx.closePath();
          ctx.stroke();

        } else {
      
```

```
        alert('You need Safari or Firefox 1.5+ to see this demo.');
    }
}
</script>
</head>

<body id = "test" onload = "drawShape();">
    <canvas id = "mycanvas"></canvas>
</body>

</html>
```

Canvas - Images

To draw an image on a canvas, use the following method:

- `drawImage(image,x,y)`

Example

```
<!DOCTYPE html>
<html>
<body>

<p>Image to use:</p>



<p>Canvas:</p>

<canvas id="myCanvas" width="240" height="297"
style="border:1px solid #d3d3d3;">
Your browser does not support the HTML5 canvas tag.
</canvas>

<script>
window.onload = function() {
    var canvas = document.getElementById("myCanvas");
    var ctx = canvas.getContext("2d");
    var img = document.getElementById("scream");
    ctx.drawImage(img, 10, 10);
};
</script>

</body>
```

</html>

Using canvas to draw a clock

HTML code:

```
<!DOCTYPE html>
<html>
<body>

<canvas id="canvas" width="400" height="400" style="background-color:#333"></canvas>

<script>
var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");
var radius = canvas.height / 2;
ctx.translate(radius, radius);
radius = radius * 0.90
drawClock();

function drawClock() {
    ctx.arc(0, 0, radius, 0 , 2 * Math.PI);
    ctx.fillStyle = "white";
    ctx.fill();
}
</script>

</body>
</html>
```

Code Explained

Add an HTML <canvas> element to your page:

```
<canvas id="canvas" width="400" height="400" style="background-color:#333"></canvas>
```

Create a canvas object (var canvas) from the HTML canvas element:

```
var canvas = document.getElementById("canvas");
```

Create a 2d drawing object (var ctx) for the canvas object:

```
var ctx = canvas.getContext("2d");
```

Calculate the clock radius, using the height of the canvas:

```
var radius = canvas.height / 2;
```

Using the canvas height to calculate the clock radius, makes the clock work for all canvas sizes.

Remap the (0,0) position (of the drawing object) to the center of the canvas:

```
ctx.translate(radius, radius);
```

Reduce the clock radius (to 90%) to draw the clock well inside the canvas:

```
radius = radius * 0.90;
```

Create a function to draw the clock:

```
function drawClock() {  
    ctx.arc(0, 0, radius, 0, 2 * Math.PI);  
    ctx.fillStyle = "white";  
    ctx.fill();  
}
```

Part II - Draw a Clock Face

The clock needs a clock face. Create a JavaScript function to draw a clock face:

JavaScript:

```
function drawClock() {  
    drawFace(ctx, radius);  
}  
  
function drawFace(ctx, radius) {  
    var grad;  
  
    ctx.beginPath();  
    ctx.arc(0, 0, radius, 0, 2 * Math.PI);  
    ctx.fillStyle = 'white';  
    ctx.fill();  
  
    grad = ctx.createRadialGradient(0, 0, radius * 0.95, 0, 0, radius * 1.05);  
    grad.addColorStop(0, "#333");  
    grad.addColorStop(0.5, 'white');  
    grad.addColorStop(1, "#333");  
    ctx.strokeStyle = grad;  
    ctx.lineWidth = radius * 0.1;  
    ctx.stroke();
```

```
ctx.beginPath();
ctx.arc(0, 0, radius * 0.1, 0, 2 * Math.PI);
ctx.fillStyle = '#333';
ctx.fill();
}
```

Code Explained

Create a drawFace() function for drawing the clock face:

```
function drawClock() {
  drawFace(ctx, radius);
}

function drawFace(ctx, radius) {
```

Draw the white circle:

```
ctx.beginPath();
ctx.arc(0, 0, radius, 0, 2 * Math.PI);
ctx.fillStyle = 'white';
ctx.fill();
```

Create a radial gradient (95% and 105% of original clock radius):

```
grad = ctx.createRadialGradient(0, 0, radius * 0.95, 0, 0, radius * 1.05);
```

Create 3 color stops, corresponding with the inner, middle, and outer edge of the arc:

```
grad.addColorStop(0, '#333');
grad.addColorStop(0.5, 'white');
grad.addColorStop(1, '#333');
```

The color stops create a 3D effect.

Define the gradient as the stroke style of the drawing object:

```
ctx.strokeStyle = grad;
```

Define the line width of the drawing object (10% of radius):

```
ctx.lineWidth = radius * 0.1;
```

Draw the circle:

```
ctx.stroke();
```

Draw the clock center:

```
ctx.beginPath();
ctx.arc(0, 0, radius * 0.1, 0, 2 * Math.PI);
ctx.fillStyle = '#333';
ctx.fill();
```

Part III - Draw Clock Numbers

The clock needs numbers. Create a JavaScript function to draw clock numbers:

JavaScript:

```
function drawClock() {
    drawFace(ctx, radius);
    drawNumbers(ctx, radius);
}

function drawNumbers(ctx, radius) {
    var ang;
    var num;
    ctx.font = radius * 0.15 + "px arial";
    ctx.textBaseline = "middle";
    ctx.textAlign = "center";
    for(num = 1; num < 13; num++){
        ang = num * Math.PI / 6;
        ctx.rotate(ang);
        ctx.translate(0, -radius * 0.85);
        ctx.rotate(-ang);
        ctx.fillText(num.toString(), 0, 0);
        ctx.rotate(ang);
        ctx.translate(0, radius * 0.85);
        ctx.rotate(-ang);
    }
}
```

Example Explained

Set the font size (of the drawing object) to 15% of the radius:

```
ctx.font = radius * 0.15 + "px arial";
```

Set the text alignment to the middle and the center of the print position:

```
ctx.textBaseline = "middle";
ctx.textAlign = "center";
```

Calculate the print position (for 12 numbers) to 85% of the radius, rotated (PI/6) for each number:

```
for(num = 1; num < 13; num++) {
    ang = num * Math.PI / 6;
    ctx.rotate(ang);
    ctx.translate(0, -radius * 0.85);
    ctx.rotate(-ang);
    ctx.fillText(num.toString(), 0, 0);
    ctx.rotate(ang);
    ctx.translate(0, radius * 0.85);
    ctx.rotate(-ang);
}
```

Part IV - Draw Clock Hands

The clock needs hands. Create a JavaScript function to draw clock hands:

JavaScript:

```
function drawClock() {
    drawFace(ctx, radius);
    drawNumbers(ctx, radius);
    drawTime(ctx, radius);
}

function drawTime(ctx, radius){
    var now = new Date();
    var hour = now.getHours();
    var minute = now.getMinutes();
    var second = now.getSeconds();
    //hour
    hour = hour%12;
    hour = (hour*Math.PI/6)+(minute*Math.PI/(6*60))+(second*Math.PI/(360*60));
    drawHand(ctx, hour, radius*0.5, radius*0.07);
    //minute
    minute = (minute*Math.PI/30)+(second*Math.PI/(30*60));
    drawHand(ctx, minute, radius*0.8, radius*0.07);
    // second
```

```
second = (second*Math.PI/30);
drawHand(ctx, second, radius*0.9, radius*0.02);
}

function drawHand(ctx, pos, length, width) {
  ctx.beginPath();
  ctx.lineWidth = width;
  ctx.lineCap = "round";
  ctx.moveTo(0,0);
  ctx.rotate(pos);
  ctx.lineTo(0, -length);
  ctx.stroke();
  ctx.rotate(-pos);
}
```

Example Explained

Use Date to get hour, minute, second:

```
var now = new Date();
var hour = now.getHours();
var minute = now.getMinutes();
var second = now.getSeconds();
```

Calculate the angle of the hour hand, and draw it a length (50% of radius), and a width (7% of radius):

```
hour = hour%12;
hour = (hour*Math.PI/6)+(minute*Math.PI/(6*60))+(second*Math.PI/(360*60));
drawHand(ctx, hour, radius*0.5, radius*0.07);
```

Use the same technique for minutes and seconds.

The drawHand() routine does not need an explanation. It just draws a line with a given length and width.

In these chapters we build an analog clock using HTML Canvas.

Part V - Start the Clock

To start the clock, call the drawClock function at intervals:

JavaScript:

```
var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");
var radius = canvas.height / 2;
ctx.translate(radius, radius);
radius = radius * 0.90
//drawClock();
setInterval(drawClock, 1000);
```

Example Explained

The only thing you have to do (to start the clock) is to call the drawClock function at intervals.

Substitute:

```
drawClock();
```

With:

```
setInterval(drawClock, 1000);
```

The interval is in milliseconds. drawClock will be called for each 1000 milliseconds.

```
<!DOCTYPE html>
<html>
<body>

<canvas id="canvas" width="400" height="400"
style="background-color:#333">
</canvas>

<script>
var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");
var radius = canvas.height / 2;
ctx.translate(radius, radius);
radius = radius * 0.90
setInterval(drawClock, 1000);

function drawClock() {
  drawFace(ctx, radius);
  drawNumbers(ctx, radius);
```

```
drawTime(ctx, radius);
}

function drawFace(ctx, radius) {
    var grad;
    ctx.beginPath();
    ctx.arc(0, 0, radius, 0, 2*Math.PI);
    ctx.fillStyle = 'white';
    ctx.fill();
    grad = ctx.createRadialGradient(0,0,radius*0.95, 0,0,radius*1.05);
    grad.addColorStop(0, '#333');
    grad.addColorStop(0.5, 'white');
    grad.addColorStop(1, '#333');
    ctx.strokeStyle = grad;
    ctx.lineWidth = radius*0.1;
    ctx.stroke();
    ctx.beginPath();
    ctx.arc(0, 0, radius*0.1, 0, 2*Math.PI);
    ctx.fillStyle = '#333';
    ctx.fill();
}

function drawNumbers(ctx, radius) {
    var ang;
    var num;
    ctx.font = radius*0.15 + "px arial";
    ctx.textBaseline="middle";
    ctx.textAlign="center";
    for(num = 1; num < 13; num++){
        ang = num * Math.PI / 6;
        ctx.rotate(ang);
        ctx.translate(0, -radius*0.85);
        ctx.rotate(-ang);
        ctx.fillText(num.toString(), 0, 0);
        ctx.rotate(ang);
        ctx.translate(0, radius*0.85);
        ctx.rotate(-ang);
    }
}

function drawTime(ctx, radius){
    var now = new Date();
    var hour = now.getHours();
    var minute = now.getMinutes();
    var second = now.getSeconds();
    //hour
    hour=hour%12;
```

```
hour=(hour*Math.PI/6)+  
(minute*Math.PI/(6*60))+  
(second*Math.PI/(360*60));  
drawHand(ctx, hour, radius*0.5, radius*0.07);  
//minute  
minute=(minute*Math.PI/30)+(second*Math.PI/(30*60));  
drawHand(ctx, minute, radius*0.8, radius*0.07);  
// second  
second=(second*Math.PI/30);  
drawHand(ctx, second, radius*0.9, radius*0.02);  
}  
  
function drawHand(ctx, pos, length, width) {  
    ctx.beginPath();  
    ctx.lineWidth = width;  
    ctx.lineCap = "round";  
    ctx.moveTo(0,0);  
    ctx.rotate(pos);  
    ctx.lineTo(0, -length);  
    ctx.stroke();  
    ctx.rotate(-pos);  
}  
</script>  
  
</body>  
</html>
```



17.9 Introduction to Algorithm

This introduction to algorithm development is to create in your mind the knowledge of efficient steps on solving problems; this implies that Algorithm is the fundamental part of computer science/programming. As a programmer, your focus is always to devise ways on how to proffer solutions to problems. Learning and visualizing algorithm will change the way you tackle real life situation by looking for an optimized way to do things. This brief topic on algorithm is to lay foundation on processes involve in problem solution.

Every problem solution starts with a plan and the plan for solving a problem is known as algorithm. By definition, an algorithm is a list of steps that you can follow to complete a task. An algorithm includes calculations, reasoning and data processing. Algorithms can be presented by natural languages, pseudo code and flowcharts.

17.9.1 Properties of algorithm

- An algorithm is an unambiguous (definite) description that makes clear what has to be implemented.
- An algorithm expects a defined set of inputs. For example, it might require two numbers where both numbers are greater than zero, or it might require a word, or a list of zero or more numbers.
- An algorithm produces a defined set of outputs. It might output the larger of the two numbers, an all-uppercase version of a word, or a sorted version of the list of numbers.
- An algorithm is to terminate and produce a result after a finite time or operations.
- Most algorithms are guaranteed to produce the correct result.
- If an algorithm imposes a requirement on its inputs (called a *precondition*), that requirement must be met. For example, a precondition might be that an algorithm will only accept positive numbers as an input. If preconditions aren't met, then the algorithm is allowed to fail by producing the wrong answer or never terminating.

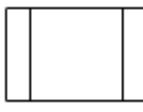
17.9.2 Different ways of presenting algorithm

Flowchart

A flowchart is the graphical or pictorial representation of an algorithm with the help of different symbols, shapes and arrows in order to demonstrate a process or a program. The main purpose of a flowchart is to analyze different processes.

The table below shows Basic Symbols you will often use for flowcharting

| S/N | Flowchart Symbols | Symbols Function |
|-----|-------------------|--|
| 1 | | This is called Terminal Symbol which indicates the start, pulse and end of a program's logic |
| 2 | | This is called Input/Output Symbol which denotes a function of input and output action of a program. |
| 3 | | This is Process/Instruction Symbol used to denote arithmetic and data movement instructions, like a step in progress of a program. |
| 4 | | This Decision Symbol indicates decision to be taken by a program at which the user chooses Yes or No, True or False. Based on the decision taken, the program logic branches to different parts of the flowchart. |

| | | |
|---|---|--|
| 5 |  | These are Flow lines Symbols with arrowheads that indicate a flow of operations/activities in a program. |
| 6 |  | This is called a Connector Symbol. This is used when a flowchart become complex and spread over more than a page, the connector symbol is used to indicate the entry and exit points of the flowchart pages. |
| 7 |  | This symbol is called Document which represents steps that result in document. |
| 8 |  | This is called Database Symbol. This is used to indicate steps that result in information being store in the database. |
| 9 |  | This symbol is called Subroutine or Predefined Process . Subroutines are portions of code that run and return the execution point to the calling function. This allows you to write one subroutine and call it as often as you like from anywhere in the code. So this symbol shows steps that result in subroutine or predefined process which is refers to a process that is defined elsewhere. |

17.12.2 Rules for drawing flowchart:

1. Use common statements that are easy to understand for words within flowchart symbols.
2. Be consistent in using names and variables in the flowchart.
3. Go from left to right and top to bottom in constructing flowcharts.
4. Keep the flowchart as simple as possible.
5. If a new flow charting page is needed, break the flowchart at an input or output point.
Use properly labelled connectors to link the flowchart on different pages.

17.12.3 Advantages of drawing a flowchart:

1. It helps a programmer to get a good visual reference of the structure of a program.
2. It not only serves as program documentation but also helps as a means to communicate with several programmers, as it is language independent.
3. It allows a programmer to test alternative solutions to a problem without even coding the program. It is much easier to find and correct logic errors prior to coding.
4. It serves as a useful reference or a programming aid due to its simple and efficient method of representing the program logic of a problem.

17.12.4 Examples of Algorithms in Programming

Problem: Given a list of positive numbers, return the smallest number on the list.

Algorithm:

1. Start
2. Declare variable min,x,L
3. Set min to 0.
4. For each number x in the list L, compare it to min.
5. If x is smaller, set min to x.
6. min is now set to the smallest number in the list
7. Stop

JavaScript Programming implementation

```
var min,x,L;// variables declaration  
L = [1,2,3,4,5];// array of numbers  
x= Math.min(...L); // finding of the minimum number  
console.log(x); // printing to the JavaScript console
```

Flowchart

Exercise

Example 2: Print 1 to 60:

Algorithm:

Step 1: Start

Step 2: Declare variable X

Step 3: Initialize X as 0,

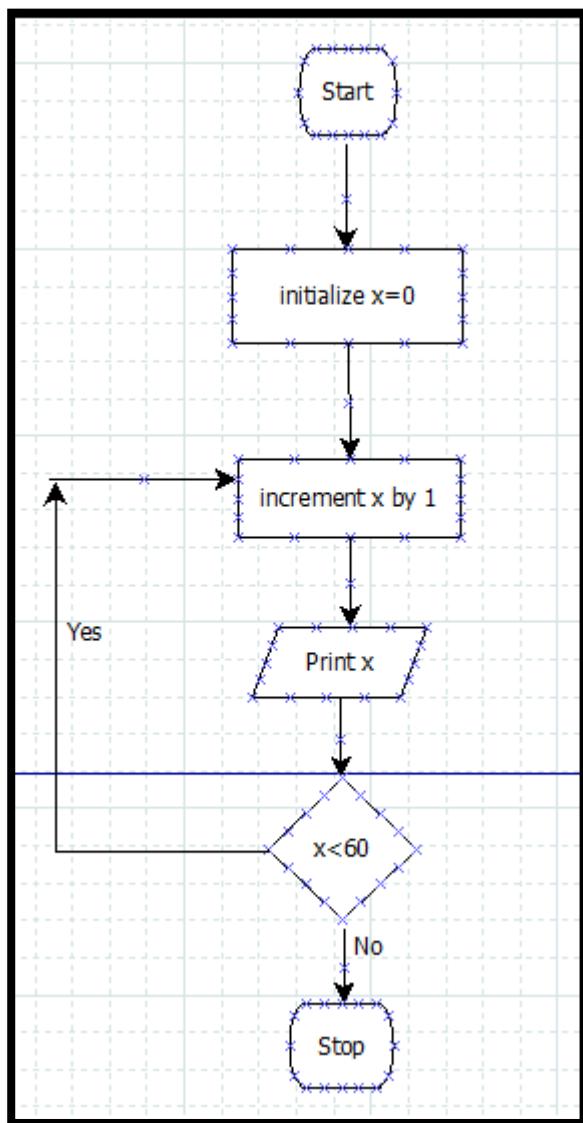
Step 4: Increment X by 1,

Step 5: Print X,

Step 6: Check if X is less than 60 then go back to step 4

Step 7: Stop

Flowchart



JavaScript Programming Implementation

```
for (x=0; x<60;x++)  
    console.log(x);  
}
```

Example 3: Convert Temperature from Fahrenheit ($^{\circ}\text{F}$) to Celsius $^{\circ}\text{C}$)

Algorithm:

Step 1: Start

Step 2: Declare variable Fahrenheit

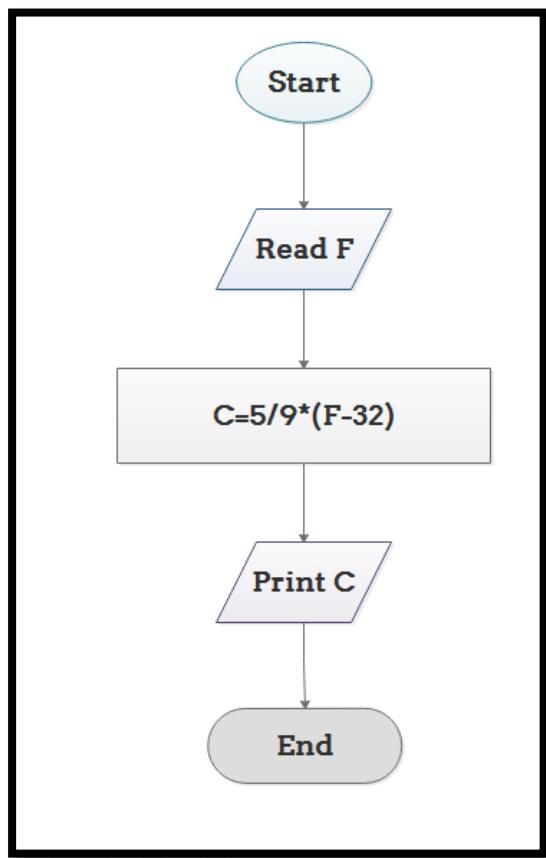
Step 3: Read temperature in Fahrenheit,

Step 4: Calculate temperature with formula $C=5/9*(F-32)$,

Step 5: Print C,

Step 6: Stop

Flowchart



JavaScript Programming Implementation

```
var F;  
F=45;  
C=5/9*(F-32);  
Console.log(C);
```

Example 4: Write an algorithm to add three numbers entered by user.

Step 1: Start

Step 2: Declare variables number1, number2, number3 and sum.

Step 3: Read values number1, number2 and number3

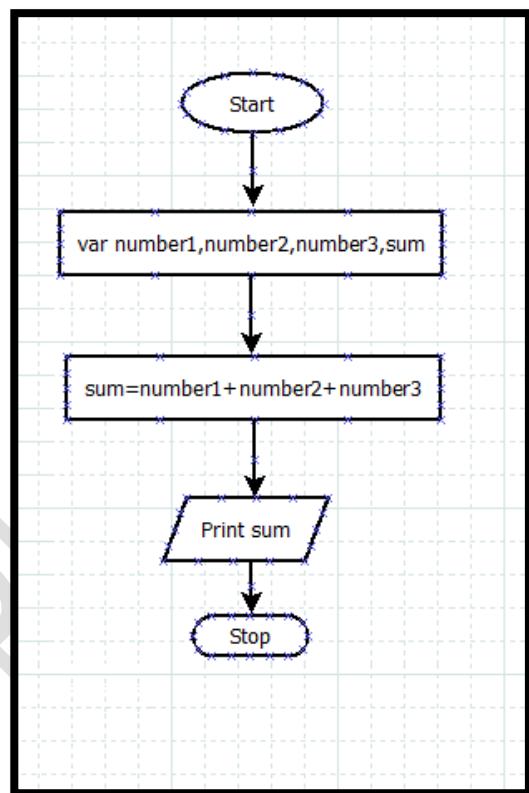
Step 4: Add number1, number2 and number3 and assign the result to sum.

sum=num1+num2

Step 5: Output sum

Step 6: Stop

Flowchart



```
var number1,number2,number3,sum;  
  
number1=23;  
  
number2=25;  
  
number3=30;  
  
sum=number1+number2+number3;  
  
console.log(sum);
```

Example 5: Write an algorithm to find the largest among three different numbers entered by user.

Step 1: Start

Step 2: Declare variables a,b and c.

Step 3: Read variables a,b and c.

Step 4: If a>b

If a>c

 Display a is the largest number.

Else

 Display c is the largest number.

Else

If b>c

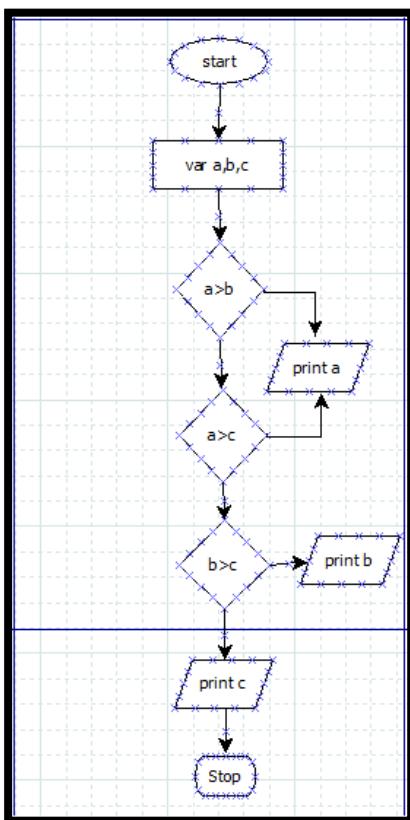
 Display b is the largest number.

Else

 Display c is the greatest number.

Step 5: Stop

Flowchart



Example 6: Write an algorithm to find all roots of a quadratic equation $ax^2+bx+c=0$.

Step 1: Start

Step 2: Declare variables a, b, c, D, x1, x2, rp and ip;

Step 3: Calculate discriminant

$$D=b^2-4ac$$

Step 4: If $D \geq 0$

$$r_1 = (-b + \sqrt{D})/2a$$

$$r_2 = (-b - \sqrt{D})/2a$$

Display r1 and r2 as roots.

Else

Calculate real part and imaginary part

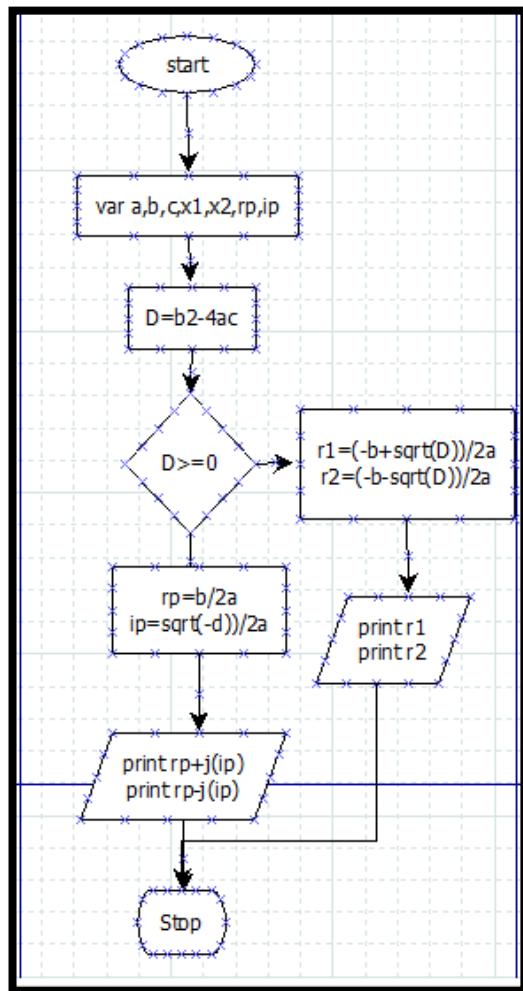
$$rp = b/2a$$

$$ip = \sqrt{(-D)/2a}$$

Display rp+j(ip) and rp-j(ip) as roots

Step 5: Stop

Flowchart



JavaScript Programming Implementation

```

var a,b,c,D,x1,x2, rp, ip;
D = b2-4ac;
If(D>=0){
R1=(-b+sqrt(D))/2a;
R2=(-b-sqrt(D))/2a;
console.log(r1);
console.log(r2);
}
Else{
rp = b/2a;
ip=sqrt(-D)/2a;
console.log(rp+j(ip));
console.log(rp-j(ip));
}
  
```

Example 7: Write an algorithm to find the factorial of a number entered by user.

Step 1: Start

Step 2: Declare variables n,factorial and i.

Step 3: Initialize variables

 factorial=1

 i=1

Step 4: Read value of n

Step 5: Repeat the steps until i=n

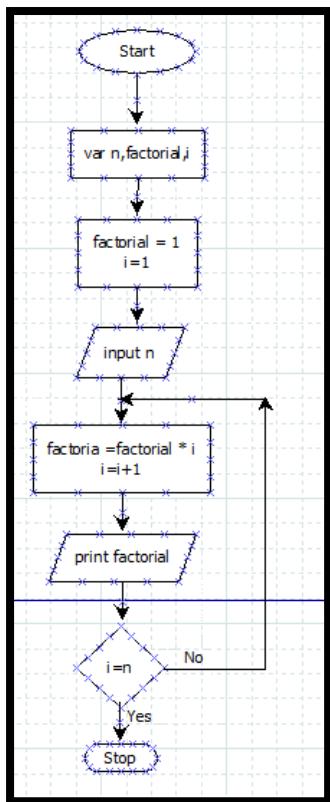
 5.1: factorial=factorial*i

 5.2: i=i+1

Step 6: Display factorial

Step 7: Stop

Flowchart



JavaScript Programming Implementation

```
var n, factorial;  
factorial = 1;  
i=1;  
n=23;  
If(i!=n){  
    factorial=factorial*I;  
    I=i+1;  
    Console.log(factorial);  
}
```

Chapter Eighteen

Introduction to Bootstrap and Version Control (Git/GitHub)

Brief Introduction to Bootstrap Framework

- Bootstrap is a free front-end framework for faster and easier web development

- Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins
- Bootstrap also gives you the ability to easily create responsive designs

Responsive web design is about creating web sites which automatically adjust themselves to look good on all devices, from small phones to large desktops.

18.1 Advantages of Bootstrap:

- **Easy to use:** Anybody with just basic knowledge of HTML and CSS can start using Bootstrap
- **Responsive features:** Bootstrap's responsive CSS adjusts to phones, tablets, and desktops
- **Mobile-first approach:** In Bootstrap 3, mobile-first styles are part of the core framework
- **Browser compatibility:** Bootstrap is compatible with all modern browsers (Chrome, Firefox, Internet Explorer, Safari, and Opera)

Ways of using Bootstrap in a webpage

There are two ways to use Bootstrap in a webpage; the first way is to use a CDN or Content delivery network. Using bootstrap CDN means that we will not download and store the bootstrap files in our server or local machine. We will just include the bootstrap CSS and JavaScript links

Use BootstrapCDN, provided for free by the folks at StackPath

Including the CSS

```
<link rel="stylesheet"  
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"  
      integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"  
      crossorigin="anonymous">
```

Including the JS

Place the following <script>s near the end of your pages, right before the closing </body> tag, to enable them. jQuery must come first, then Popper.js, and then our JavaScript plugins.

Starter template

Be sure to have your pages set up with the latest design and development standards. That means using an HTML5 doctype and including a viewport meta tag for proper responsive behaviors as shown below;

```
<!doctype html>

<html lang="en">

    <head>

        <!-- Required meta tags -->

        <meta charset="utf-8">

        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

        <!-- Bootstrap CSS -->

        <link rel="stylesheet"
            href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
            integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
            crossorigin="anonymous">

    <title>Hello, world!</title>

    </head>

    <body>

        <h1>Hello, world!</h1>

        <!-- Optional JavaScript -->

        <!-- jQuery first, then Popper.js, then Bootstrap JS -->

        <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"
            crossorigin="anonymous"></script>

        <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
            integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
            crossorigin="anonymous"></script>
```

```
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
crossorigin="anonymous"></script>

</body>

</html>
```

Output

Hello, world!

Containers

Containers are the most basic layout element in Bootstrap and are **required when using our default grid system**. Containers are used to contain, pad, and (sometimes) center the content within them. While containers *can* be nested, most layouts do not require a nested container.

Bootstrap comes with three different containers:

- `.container`, which sets a max-width at each responsive breakpoint
- `.container-fluid`, which is width: 100% at all breakpoints
- `.container-{breakpoint}`, which is width: 100% until the specified breakpoint

| | Extra small | Small | Medium | Large | Extra large |
|-------------------------------|-------------|--------|--------|--------|-------------|
| | <576px | ≥576px | ≥768px | ≥992px | ≥1200px |
| <code>.container</code> | 100% | 540px | 720px | 960px | 1140px |
| <code>.container-sm</code> | 100% | 540px | 720px | 960px | 1140px |
| <code>.container-md</code> | 100% | 100% | 720px | 960px | 1140px |
| <code>.container-lg</code> | 100% | 100% | 100% | 960px | 1140px |
| <code>.container-xl</code> | 100% | 100% | 100% | 100% | 1140px |
| <code>.container-fluid</code> | 100% | 100% | 100% | 100% | 100% |

Example

```
<div class="container">
  <div class="row">
    <div class="col-sm">
```

One of three columns

```
</div>
```

```
<div class="col-sm">
```

One of three columns

```
</div>
```

```
<div class="col-sm">
```

One of three columns

```
</div>
```

```
</div>
```

```
</div>
```

One of three columns

One of three columns

One of three columns

The above example creates three equal-width columns on small, medium, large, and extra large devices using our predefined grid classes. Those columns are centered in the page with the parent .container.

Row columns

Use the responsive .row-cols-* classes to quickly set the number of columns that best render your content and layout. Whereas normal .col-* classes apply to the individual columns (e.g., .col-md-4), the row columns classes are set on the parent .row as a shortcut.

```
<div class="container">  
<div class="row row-cols-4">  
  <div class="col">Column</div>  
  <div class="col">Column</div>  
  <div class="col">Column</div>  
  <div class="col-6">Column</div>  
</div>
```

```
</div>
```

```
output
```

```
Column  Column  Column  
Column
```

Alignment

Use flexbox alignment utilities to vertically and horizontally align columns.

```
<div class="container">  
  <div class="row align-items-start">  
    <div class="col">  
      One of three columns  
    </div>  
    <div class="col">  
      One of three columns  
    </div>  
    <div class="col">  
      One of three columns  
    </div>  
  </div>  
  <div class="row align-items-center">  
    <div class="col">
```

```
One of three columns  
  </div>  
  
  <div class="col">  
    One of three columns  
      </div>  
    <div class="col">  
      One of three columns  
       </div>  
      </div>  
    <div class="row align-items-end">  
      <div class="col">  
        One of three columns  
        </div>  
      <div class="col">  
        One of three columns  
        </div>  
      <div class="col">  
        One of three columns  
        </div>  
      </div>  
    </div>  
  </div>  
  </div>
```

| | | |
|----------------------|----------------------|----------------------|
| One of three columns | One of three columns | One of three columns |
| One of three columns | One of three columns | One of three columns |
| One of three columns | One of three columns | One of three columns |

Example 2

```
<div class="container">  
  
  <div class="row justify-content-start">  
  
    <div class="col-4">  
  
      One of two columns  
  
    </div>  
  
    <div class="col-4">  
  
      One of two columns  
  
    </div>  
  
  </div>  
  
  <div class="row justify-content-center">  
  
    <div class="col-4">  
  
      One of two columns  
  
    </div>  
  
    <div class="col-4">  
  
      One of two columns  
  
    </div>  
  
  </div>  
  
  <div class="row justify-content-end">  
  
    <div class="col-4">  
  
      One of two columns  
  
    </div>  
  
  </div>
```

```
</div>

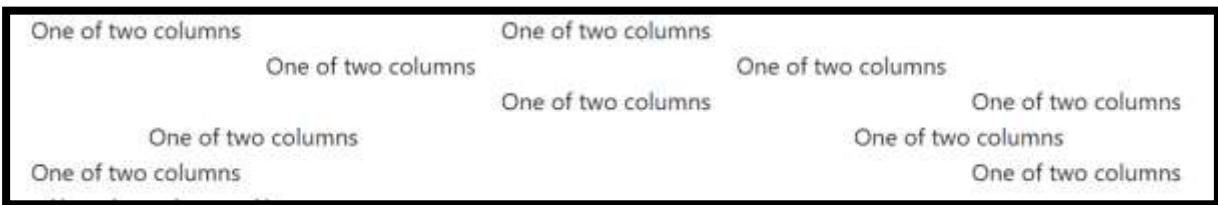
<div class="col-4">
    One of two columns
</div>

</div>

<div class="row justify-content-around">
    <div class="col-4">
        One of two columns
    </div>
    <div class="col-4">
        One of two columns
    </div>
</div>

<div class="row justify-content-between">
    <div class="col-4">
        One of two columns
    </div>
    <div class="col-4">
        One of two columns
    </div>
</div>

</div>
```



Column breaks

Breaking columns to a new line in flexbox requires a small hack: add an element with width: 100% wherever you want to wrap your columns to a new line. Normally this is accomplished with multiple .rows, but not every implementation method can account for this.

```
<div class="container">  
  <div class="row">  
    <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>  
    <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>  
  
    <!-- Force next columns to break to new line -->  
    <div class="w-100"></div>  
  
    <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>  
    <div class="col-6 col-sm-3">.col-6 .col-sm-3</div>  
  </div>  
</div>
```

Output

| | |
|--------------------------------------|--------------------------------------|
| .col-6 .col-sm-3 .col-6 .col-sm-3 | .col-6 .col-sm-3 .col-6 .col-sm-3 |
|--------------------------------------|--------------------------------------|

Lead

Make a paragraph stand out by adding .lead.

```
<p class="lead">
```

Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor. Duis mollis, est non commodo luctus.

```
</p>
```

Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor. Duis mollis, est non commodo luctus.

Inline

Remove a list's bullets and apply some light margin with a combination of two classes, .list-inline and .list-inline-item.

```
<ul class="list-inline">  
  <li class="list-inline-item">Lorem ipsum</li>  
  <li class="list-inline-item">Phasellus iaculis</li>  
  <li class="list-inline-item">Nulla volutpat</li>  
</ul>
```

 Lorem ipsum Phasellus iaculis Nulla volutpat

Description list alignment

```
<dl class="row">  
  <dt class="col-sm-3">Description lists</dt>
```

```
<dd class="col-sm-9">A description list is perfect for defining terms.</dd>
```

```
<dt class="col-sm-3">Euismod</dt>
```

```
<dd class="col-sm-9">
```

```
<p>Vestibulum id ligula porta felis euismod semper eget lacinia odio sem nec elit.</p>
```

```
<p>Donec id elit non mi porta gravida at eget metus.</p>
```

```
</dd>
```

```
<dt class="col-sm-3">Malesuada porta</dt>
```

```
<dd class="col-sm-9">Etiam porta sem malesuada magna mollis euismod.</dd>
```

```
<dt class="col-sm-3 text-truncate">Truncated term is truncated</dt>
```

```
<dd class="col-sm-9">Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.</dd>
```

```
<dt class="col-sm-3">Nesting</dt>
```

```
<dd class="col-sm-9">
```

```
<dl class="row">
```

```
<dt class="col-sm-4">Nested definition list</dt>
```

```
<dd class="col-sm-8">Aenean posuere, tortor sed cursus feugiat, nunc augue blandit nunc.</dd>
```

```
</dl>
```

```
</dd>
```

```
</dl>
```

Output

| | | |
|-----------------------------|--|---|
| Description lists | A description list is perfect for defining terms. | |
| Euismod | Vestibulum id ligula porta felis euismod semper eget lacinia odio sem nec elit. Donec id elit non mi porta gravida at eget metus. | |
| Malesuada porta | Etiam porta sem malesuada magna mollis euismod. | |
| Truncated term is truncated | Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. | |
| Nesting | Nested definition list | Aenean posuere, tortor sed cursus feugiat, nunc augue blandit nunc. |

Responsive images

Images in Bootstrap are made responsive with .img-fluid, max-width: 100%; and height: auto; are applied to the image so that it scales with the parent element.

```

```

Image thumbnails

In addition to our [border-radius utilities](#), you can use .img-thumbnail to give an image a rounded 1px border appearance.

```

```

Aligning images

Align images with the [helper float classes](#) or [text alignment classes](#). block-level images can be centered using [the .mx-auto margin utility class](#) as shown below;

```

```

```

```

```

```

```
<div class="text-center">
```

```
    
```

```
</div>
```

Tables

```
<table class="table">

<thead>

<tr>

<th scope="col">#</th>

<th scope="col">First</th>

<th scope="col">Last</th>

<th scope="col">Handle</th>

</tr>

</thead>

<tbody>

<tr>

<th scope="row">1</th>

<td>Mark</td>

<td>Otto</td>

<td>@mdo</td>

</tr>

<tr>

<th scope="row">2</th>

<td>Jacob</td>

<td>Thornton</td>

<td>@fat</td>


```

```
</tr>  
</table>
```

| # | First | Last | Handle |
|---|-------|----------|----------|
| 1 | Mark | Otto | @mdo |
| 2 | Jacob | Thornton | @fat |
| 3 | Larry | the Bird | @twitter |

Table example 2

```
<table class="table table-dark">  
  <thead>  
    <tr>  
      <th scope="col">#</th>  
      <th scope="col">First</th>  
      <th scope="col">Last</th>  
      <th scope="col">Handle</th>  
    </tr>  
  </thead>  
  <tbody>
```

```
<tr>

<th scope="row">1</th>

<td>Mark</td>

<td>Otto</td>

<td>@mdo</td>

</tr>

<tr>

<th scope="row">2</th>

<td>Jacob</td>

<td>Thornton</td>

<td>@fat</td>

</tr>

</table>
```

| # | First | Last | Handle |
|---|-------|----------|----------|
| 1 | Mark | Otto | @mdo |
| 2 | Jacob | Thornton | @fat |
| 3 | Larry | the Bird | @twitter |

Table head options

Similar to tables and dark tables, use the modifier classes `.thead-light` or `.thead-dark` to make `<thead>`s appear light or dark gray.

```
table class="table">| # | First | Last | Handle |
| --- | --- | --- | --- |
| 1 | Mark | Otto | @mdo |
| 2 | Jacob | Thornton | @fat |
|  |  |  |  |

```

```
<th scope="row">3</th>  
  
<td>Larry</td>  
  
<td>the Bird</td>  
  
<td>@twitter</td>  
  
</tr>  
  
</tbody>  
  
</table>
```

| # | First | Last | Handle |
|---|-------|----------|----------|
| 1 | Mark | Otto | @mdo |
| 2 | Jacob | Thornton | @fat |
| 3 | Larry | the Bird | @twitter |

Example 2

```
<table class="table">  
  
<thead class="thead-light">  
  
<tr>  
  
  <th scope="col">#</th>  
  
  <th scope="col">First</th>  
  
  <th scope="col">Last</th>
```

```
<th scope="col">Handle</th>

</tr>

</thead>

<tbody>

<tr>

<th scope="row">1</th>

<td>Mark</td>

<td>Otto</td>

<td>@mdo</td>

</tr>

<tr>

<th scope="row">2</th>

<td>Jacob</td>

<td>Thornton</td>

<td>@fat</td>

</tr>

<tr>

<th scope="row">3</th>

<td>Larry</td>

<td>the Bird</td>

<td>@twitter</td>

</tr>

</tbody>
```

</table>

| # | First | Last | Handle |
|---|-------|----------|----------|
| 1 | Mark | Otto | @mdo |
| 2 | Jacob | Thornton | @fat |
| 3 | Larry | the Bird | @twitter |

Striped rows

Use .table-striped to add zebra-striping to any table row within the <tbody>.

Example 1

```
<table class="table table-striped">

<thead>

  <tr>
    <th scope="col">#</th>
    <th scope="col">First</th>
    <th scope="col">Last</th>
    <th scope="col">Handle</th>
  </tr>

</thead>

<tbody>

  <tr>
```

```
<th scope="row">1</th>  
<td>Mark</td>  
<td>Otto</td>  
<td>@mdo</td>  
</tr>  
<tr>  
  <th scope="row">2</th>  
  <td>Jacob</td>  
  <td>Thornton</td>  
  <td>@fat</td>  
</tr>  
</tbody>  
</table>
```

Example 2

```
<table class="table table-striped table-dark">  
  <thead>  
    <tr>  
      <th scope="col">#</th>  
      <th scope="col">First</th>  
      <th scope="col">Last</th>  
      <th scope="col">Handle</th>  
    </tr>
```

```
</thead>

<tbody>

<tr>

<th scope="row">1</th>

<td>Mark</td>

<td>Otto</td>

<td>@mdo</td>

</tr>

<tbody>

</table>
```

The screenshot displays two examples of tables.
Example 1: A table with a light gray border. It has four columns: '#', 'First', 'Last', and 'Handle'. The data rows are: #1, First: Mark, Last: Otto, Handle: @mdo; and #2, First: Jacob, Last: Thornton, Handle: @fat.
Example 2: A table with a dark header row (#, First, Last, Handle) and white body rows. The data rows are: #1, First: Mark, Last: Otto, Handle: @mdo.

| # | First | Last | Handle |
|---|-------|----------|--------|
| 1 | Mark | Otto | @mdo |
| 2 | Jacob | Thornton | @fat |

| # | First | Last | Handle |
|---|-------|------|--------|
| 1 | Mark | Otto | @mdo |

Bordered table

Add .table-bordered for borders on all sides of the table and cells.

```
<table class="table table-bordered">

<thead>

<tr>

<th scope="col">#</th>

<th scope="col">First</th>

<th scope="col">Last</th>
```

```
<th scope="col">Handle</th>
</tr>
</thead>
<tbody>
<tr>
<th scope="row">1</th>
<td>Mark</td>
<td>Otto</td>
<td>@mdo</td>
</tr>
<tr>
<th scope="row">2</th>
<td>Jacob</td>
<td>Thornton</td>
<td>@fat</td>
</tr>
</tbody>
</table>
```

| # | First | Last | Handle |
|---|-------|----------|--------|
| 1 | Mark | Otto | @mdo |
| 2 | Jacob | Thornton | @fat |

Hoverable rows

Add .table-hover to enable a hover state on table rows within a <tbody>.

Example 1

```
<table class="table table-hover">

<thead>

<tr>

<th scope="col">#</th>

<th scope="col">First</th>

<th scope="col">Last</th>

<th scope="col">Handle</th>

</tr>

</thead>

<tbody>

<tr>

<th scope="row">1</th>

<td>Mark</td>

<td>Otto</td>

<td>@mdo</td>

</tr>

<tr>

<th scope="row">2</th>

<td>Jacob</td>

<td>Thornton</td>
```

```
<td>@fat</td>  
</tr>  
</tbody>  
</table>
```

| # | First | Last | Handle |
|---|-------|----------|--------|
| 1 | Mark | Otto | @mdo |
| 2 | Jacob | Thornton | @fat |

Small table

Add .table-sm to make tables more compact by cutting cell padding in half.

```
<table class="table table-sm">  
  
<thead>  
  
<tr>  
  <th scope="col">#</th>  
  <th scope="col">First</th>  
  <th scope="col">Last</th>  
  <th scope="col">Handle</th>  
  
</tr>  
</thead>  
  
<tbody>  
  
<tr>  
  <th scope="row">1</th>  
  <td>Mark</td>
```

```
<td>Otto</td>  
  
<td>@mdo</td>  
  
</tr>  
  
<tr>  
  
    <th scope="row">2</th>  
  
    <td>Jacob</td>  
  
    <td>Thornton</td>  
  
    <td>@fat</td>  
  
</tr>  
  
</tbody>  
  
</table>
```

Captions

A `<caption>` functions like a heading for a table. It helps users with screen readers to find a table and understand what it's about and decide if they want to read it.

```
<table class="table">  
  
    <caption>List of users</caption>  
  
    <thead>  
  
        <tr>  
            <th scope="col">#</th>  
            <th scope="col">First</th>  
            <th scope="col">Last</th>  
            <th scope="col">Handle</th>  
  
        </tr>
```

```
</thead>

<tbody>

    <tr>

        <th scope="row">1</th>

        <td>Mark</td>

        <td>Otto</td>

        <td>@mdo</td>

    </tr>

    ...

</tbody>

</table>
```

| List of users | | | |
|---------------|-------|------|--------|
| # | First | Last | Handle |
| 1 | Mark | Otto | @mdo |

Responsive tables

Responsive tables allow tables to be scrolled horizontally with ease. Make any table responsive across all viewports by wrapping a `.table` with `.table-responsive`.

```
<div class="table-responsive">  
  <table class="table">  
    ...  
  </table>
```

```
</div>
```

Or

```
<div class="table-responsive-sm">  
  <table class="table">  
    ...  
  </table>  
</div>
```

Figures

Anytime you need to display a piece of content—like an image with an optional caption, consider using a `<figure>`.

Example

```
<figure class="figure">  
    
  <figcaption class="figure-caption">A caption for the image.</figcaption>  
</figure>
```

Alerts

Provide contextual feedback messages for typical user actions with the handful of available and flexible alert messages.

Example

```
<div class="alert alert-primary" role="alert">  
  A simple primary alert—check it out!  
</div>  
  
<div class="alert alert-secondary" role="alert">
```

A simple secondary alert—check it out!

```
</div>
```

```
<div class="alert alert-success" role="alert">
```

A simple success alert—check it out!

```
</div>
```

```
<div class="alert alert-danger" role="alert">
```

A simple danger alert—check it out!

```
</div>
```

```
<div class="alert alert-warning" role="alert">
```

A simple warning alert—check it out!

```
</div>
```

```
<div class="alert alert-info" role="alert">
```

A simple info alert—check it out!

```
</div>
```

```
<div class="alert alert-light" role="alert">
```

A simple light alert—check it out!

```
</div>
```

```
<div class="alert alert-dark" role="alert">
```

A simple dark alert—check it out!

```
</div>
```

A simple primary alert—check it out!

A simple secondary alert—check it out!

A simple success alert—check it out!

A simple danger alert—check it out!

A simple warning alert—check it out!

A simple info alert—check it out!

A simple light alert—check it out!

A simple dark alert—check it out!

Whenever you need to, be sure to use margin utilities to keep things nice and tidy as shown in the example below;

```
<div class="alert alert-success" role="alert">  
  <h4 class="alert-heading">Well done!</h4>  
  <p>Aww yeah, you successfully read this important alert message. This example text is going  
  to run a bit longer so that you can see how spacing within an alert works with this kind of  
  content.</p>  
  
  <hr>  
  <p class="mb-0">Whenever you need to, be sure to use margin utilities to keep things nice and  
  tidy.</p>  
</div>
```

Well done!

Aww yeah, you successfully read this important alert message. This example text is going to run a bit longer so that you can see how spacing within an alert works with this kind of content.

Whenever you need to, be sure to use margin utilities to keep things nice and tidy.

Buttons

Bootstrap's custom button styles are used for actions in forms, dialogs, and more with support for multiple sizes, states, and more.

Bootstrap includes several predefined button styles, each serving its own semantic purpose, with a few extras thrown in for more control.

Examples

```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-danger">Danger</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-light">Light</button>
<button type="button" class="btn btn-dark">Dark</button>
<button type="button" class="btn btn-link">Link</button>
```



Button tags

The .btn classes are designed to be used with the <button> element. However, you can also use these classes on <a> or <input> elements (though some browsers may apply a slightly different rendering).

When using button classes on <a> elements that are used to trigger in-page functionality (like collapsing content), rather than linking to new pages or sections within the current page, these links should be given a role="button" to appropriately convey their purpose to assistive technologies such as screen readers.

Example

```
<a class="btn btn-primary" href="#" role="button">Link</a>

<button class="btn btn-primary" type="submit">Button</button>

<input class="btn btn-primary" type="button" value="Input">

<input class="btn btn-primary" type="submit" value="Submit">

<input class="btn btn-primary" type="reset" value="Reset">
```



Outline buttons

In need of a button, but not the hefty background colors they bring? Replace the default modifier classes with the .btn-outline-* ones to remove all background images and colors on any button as shown below;

```
<button type="button" class="btn btn-outline-primary">Primary</button>

<button type="button" class="btn btn-outline-secondary">Secondary</button>

<button type="button" class="btn btn-outline-success">Success</button>

<button type="button" class="btn btn-outline-danger">Danger</button>
```

```
<button type="button" class="btn btn-outline-warning">Warning</button>
```

```
<button type="button" class="btn btn-outline-info">Info</button>
```

```
<button type="button" class="btn btn-outline-light">Light</button>
```

```
<button type="button" class="btn btn-outline-dark">Dark</button>
```



Sizes

Fancy larger or smaller buttons? Add .btn-lg or .btn-sm for additional sizes.

Example

```
<button type="button" class="btn btn-primary btn-lg">Large button</button>
```

```
<button type="button" class="btn btn-secondary btn-lg">Large button</button>
```

```
<button type="button" class="btn btn-primary btn-sm">Small button</button>
```

```
<button type="button" class="btn btn-secondary btn-sm">Small button</button>
```



Disabled state

Make buttons look inactive by adding the disabled boolean attribute to any <button> element as shown below;

```
<button type="button" class="btn btn-lg btn-primary" disabled>Primary button</button>
```

```
<button type="button" class="btn btn-secondary btn-lg" disabled>Button</button>
```

Primary button

Button

Disabled buttons using the `<a>` element behave a bit different:

- `<a>`s don't support the `disabled` attribute, so you must add the `.disabled` class to make it visually appear disabled.
- Some future-friendly styles are included to disable all pointer-events on anchor buttons. In browsers which support that property, you won't see the disabled cursor at all.
- Disabled buttons should include the `aria-disabled="true"` attribute to indicate the state of the element to assistive technologies.

```
<a href="#" class="btn btn-primary btn-lg disabled" tabindex="-1" role="button" aria-disabled="true">Primary link</a>
```

```
<a href="#" class="btn btn-secondary btn-lg disabled" tabindex="-1" role="button" aria-disabled="true">Link</a>
```

Toggle states

Add `data-toggle="button"` to toggle a button's active state. If you're pre-toggling a button, you must manually add the `.active` class **and** `aria-pressed="true"` to the `<button>`.

```
<button type="button" class="btn btn-primary" data-toggle="button" aria-pressed="false">
```

Single toggle

```
</button>
```

Single toggle

Button group

Group a series of buttons together on a single line with the `button-group` and super-power them with JavaScript.

Basic example

Wrap a series of buttons with .btn in .btn-group. Add on optional JavaScript radio and checkbox style behavior with [our buttons plugin](#).

Example

```
<div class="btn-group" role="group" aria-label="Basic example">  
  <button type="button" class="btn btn-secondary">Left</button>  
  <button type="button" class="btn btn-secondary">Middle</button>  
  <button type="button" class="btn btn-secondary">Right</button>  
</div>
```



Button toolbar

Combine sets of button groups into button toolbars for more complex components. Use utility classes as needed to space out groups, buttons, and more.

Example

```
<div class="btn-toolbar" role="toolbar" aria-label="Toolbar with button groups">  
  <div class="btn-group mr-2" role="group" aria-label="First group">  
    <button type="button" class="btn btn-secondary">1</button>  
    <button type="button" class="btn btn-secondary">2</button>  
    <button type="button" class="btn btn-secondary">3</button>  
    <button type="button" class="btn btn-secondary">4</button>  
  </div>  
  <div class="btn-group mr-2" role="group" aria-label="Second group">
```

```
<button type="button" class="btn btn-secondary">5</button>  
<button type="button" class="btn btn-secondary">6</button>  
<button type="button" class="btn btn-secondary">7</button>  
</div>  
  
<div class="btn-group" role="group" aria-label="Third group">  
  <button type="button" class="btn btn-secondary">8</button>  
</div>  
</div>
```



Nesting

Place a .btn-group within another .btn-group when you want dropdown menus mixed with a series of buttons as shown in the example below;

```
<div class="btn-group" role="group" aria-label="Button group with nested dropdown">  
  <button type="button" class="btn btn-secondary">1</button>  
  <button type="button" class="btn btn-secondary">2</button>  
  
  <div class="btn-group" role="group">  
    <button id="btnGroupDrop1" type="button" class="btn btn-secondary dropdown-toggle" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">  
      Dropdown  
    </button>  
    <div class="dropdown-menu" aria-labelledby="btnGroupDrop1">
```

```
<a class="dropdown-item" href="#">Dropdown link</a>  
<a class="dropdown-item" href="#">Dropdown link</a>  
</div>  
</div>  
</div>
```



Cards

A **card** is a flexible and extensible content container. It includes options for headers and footers, a wide variety of content, contextual background colors, and powerful display options. If you're familiar with Bootstrap 3, cards replace our old panels, wells, and thumbnails. Similar functionality to those components is available as modifier classes for cards.

Card title

Some quick example text to build on the card title and make up the bulk of the card's content.

```
<div class="card" style="width: 18rem;">  
    
  <div class="card-body">  
    <h5 class="card-title">Card title</h5>  
    <p class="card-text">Some quick example text to build on the card title and make up the bulk  
      of the card's content.</p>  
    <a href="#" class="btn btn-primary">Go somewhere</a>  
  </div>  
</div>
```



Card title

Some quick example text to build on the card title and make up the bulk of the card's content.

[Go somewhere](#)

Content types

Cards support a wide variety of content, including images, text, list groups, links, and more. Below are examples of what's supported.

Body

The building block of a card is the .card-body. Use it whenever you need a padded section within a card.

This is some text within a card body.

```
<div class="card">  
  <div class="card-body">  
    This is some text within a card body.  
  </div>  
</div>
```

Titles, text, and links

Card titles are used by adding .card-title to a `<h*>` tag. In the same way, links are added and placed next to each other by adding .card-link to an `<a>` tag.

Subtitles are used by adding a .card-subtitle to a `<h*>` tag. If the .card-title and the .card-subtitle items are placed in a .card-body item, the card title and subtitle are aligned nicely.

Images

.card-img-top places an image to the top of the card. With .card-text, text can be added to the card. Text within .card-text can also be styled with the standard HTML tags.

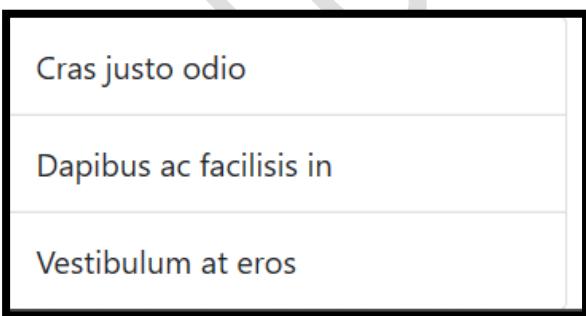
Some quick example text to build on the card title and make up the bulk of the card's content.

```
<div class="card" style="width: 18rem;">  
    
  <div class="card-body">  
    <p class="card-text">Some quick example text to build on the card title and make up the bulk  
    of the card's content.</p>  
  </div>  
</div>
```

List groups

Create lists of content in a card with a flush list group.

```
<div class="card" style="width: 18rem;">  
  <ul class="list-group list-group-flush">  
    <li class="list-group-item">Cras justo odio</li>  
    <li class="list-group-item">Dapibus ac facilisis in</li>  
    <li class="list-group-item">Vestibulum at eros</li>  
  </ul>  
</div>
```



Some quick example text to build on the card title and make up the bulk of the card's content.

```
<div class="card text-white bg-primary mb-3" style="max-width: 18rem;">  
  <div class="card-header">Header</div>
```

```
<div class="card-body">

<h5 class="card-title">Primary card title</h5>

<p class="card-text">Some quick example text to build on the card title and make up the bulk
of the card's content.</p>

</div>

</div>

<div class="card text-white bg-secondary mb-3" style="max-width: 18rem;">

<div class="card-header">Header</div>

<div class="card-body">

<h5 class="card-title">Secondary card title</h5>

<p class="card-text">Some quick example text to build on the card title and make up the bulk
of the card's content.</p>

</div>

</div>

<div class="card text-white bg-success mb-3" style="max-width: 18rem;">

<div class="card-header">Header</div>

<div class="card-body">

<h5 class="card-title">Success card title</h5>

<p class="card-text">Some quick example text to build on the card title and make up the bulk
of the card's content.</p>

</div>

</div>

<div class="card text-white bg-danger mb-3" style="max-width: 18rem;">

<div class="card-header">Header</div>

<div class="card-body">
```

```
<h5 class="card-title">Danger card title</h5>

<p class="card-text">Some quick example text to build on the card title and make up the bulk
of the card's content.</p>

</div>

</div>

<div class="card text-white bg-warning mb-3" style="max-width: 18rem;">

<div class="card-header">Header</div>

<div class="card-body">

<h5 class="card-title">Warning card title</h5>

<p class="card-text">Some quick example text to build on the card title and make up the bulk
of the card's content.</p>

</div>

</div>

<div class="card text-white bg-info mb-3" style="max-width: 18rem;">

<div class="card-header">Header</div>

<div class="card-body">

<h5 class="card-title">Info card title</h5>

<p class="card-text">Some quick example text to build on the card title and make up the bulk
of the card's content.</p>

</div>

</div>

<div class="card bg-light mb-3" style="max-width: 18rem;">

<div class="card-header">Header</div>

<div class="card-body">

<h5 class="card-title">Light card title</h5>
```

```
<p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>
```

```
</div>
```

```
</div>
```

```
<div class="card text-white bg-dark mb-3" style="max-width: 18rem;">
```

```
<div class="card-header">Header</div>
```

```
<div class="card-body">
```

```
<h5 class="card-title">Dark card title</h5>
```

```
<p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>
```

```
</div>
```

```
</div>
```



Border

Use [border utilities](#) to change just the border-color of a card. Note that you can put `.text-{color}` classes on the parent `.card` or a subset of the card's contents in the example:

Some quick example text to build on the card title and make up the bulk of the card's content

```
<div class="card border-primary mb-3" style="max-width: 18rem;">
```

```
<div class="card-header">Header</div>
```

```
<div class="card-body text-primary">
```

```
<h5 class="card-title">Primary card title</h5>
```

```
<p class="card-text">Some quick example text to build on the card title and make up the bulk  
of the card's content.</p>  
  
</div>  
  
</div>  
  
<div class="card border-secondary mb-3" style="max-width: 18rem;">  
  
  <div class="card-header">Header</div>  
  
  <div class="card-body text-secondary">  
  
    <h5 class="card-title">Secondary card title</h5>  
  
    <p class="card-text">Some quick example text to build on the card title and make up the bulk  
    of the card's content.</p>  
  
  </div>  
  
</div>  
  
<div class="card border-success mb-3" style="max-width: 18rem;">  
  
  <div class="card-header">Header</div>  
  
  <div class="card-body text-success">  
  
    <h5 class="card-title">Success card title</h5>  
  
    <p class="card-text">Some quick example text to build on the card title and make up the bulk  
    of the card's content.</p>  
  
  </div>  
  
</div>  
  
<div class="card border-danger mb-3" style="max-width: 18rem;">  
  
  <div class="card-header">Header</div>  
  
  <div class="card-body text-danger">  
  
    <h5 class="card-title">Danger card title</h5>
```

```
<p class="card-text">Some quick example text to build on the card title and make up the bulk  
of the card's content.</p>  
  
</div>  
  
</div>  
  
<div class="card border-warning mb-3" style="max-width: 18rem;">  
  
    <div class="card-header">Header</div>  
  
    <div class="card-body text-warning">  
  
        <h5 class="card-title">Warning card title</h5>  
  
        <p class="card-text">Some quick example text to build on the card title and make up the bulk  
        of the card's content.</p>  
  
    </div>  
  
</div>  
  
<div class="card border-info mb-3" style="max-width: 18rem;">  
  
    <div class="card-header">Header</div>  
  
    <div class="card-body text-info">  
  
        <h5 class="card-title">Info card title</h5>  
  
        <p class="card-text">Some quick example text to build on the card title and make up the bulk  
        of the card's content.</p>  
  
    </div>  
  
</div>  
  
<div class="card border-light mb-3" style="max-width: 18rem;">  
  
    <div class="card-header">Header</div>  
  
    <div class="card-body">  
  
        <h5 class="card-title">Light card title</h5>
```

```
<p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>
```

```
</div>
```

```
</div>
```

```
<div class="card border-dark mb-3" style="max-width: 18rem;">
```

```
<div class="card-header">Header</div>
```

```
<div class="card-body text-dark">
```

```
<h5 class="card-title">Dark card title</h5>
```

```
<p class="card-text">Some quick example text to build on the card title and make up the bulk of the card's content.</p>
```

```
</div>
```

```
</div>
```



Carousel

A slideshow component for cycling through elements—images or slides of text—like a carousel.

How it works

The carousel is a slideshow for cycling through a series of content, built with CSS 3D transforms and a bit of JavaScript. It works with a series of images, text, or custom markup. It also includes support for previous/next controls and indicators.

Slides only

Here's a carousel with slides only. Note the presence of the .d-block and .w-100 on carousel images to prevent browser default image alignment.

```
<div id="carouselExampleSlidesOnly" class="carousel slide" data-ride="carousel">  
  <div class="carousel-inner">  
    <div class="carousel-item active">  
        
    </div>  
    <div class="carousel-item">  
        
    </div>  
    <div class="carousel-item">  
        
    </div>  
  </div>  
</div>
```

With controls

Adding in the previous and next controls:

[Previous](#) [Next](#)

```
<div id="carouselExampleControls" class="carousel slide" data-ride="carousel">  
  <div class="carousel-inner">  
    <div class="carousel-item active">  
        
    </div>  
    <div class="carousel-item">  
      
```

```
</div>

<div class="carousel-item">

</div>

</div>

<a class="carousel-control-prev" href="#carouselExampleControls" role="button" data-slide="prev">

    <span class="carousel-control-prev-icon" aria-hidden="true"></span>

    <span class="sr-only">Previous</span>

</a>

<a class="carousel-control-next" href="#carouselExampleControls" role="button" data-slide="next">

    <span class="carousel-control-next-icon" aria-hidden="true"></span>

    <span class="sr-only">Next</span>

</a>

</div>
```

With indicators

You can also add the indicators to the carousel, alongside the controls, too.

- 1.
- 2.
- 3.

[Previous](#) [Next](#)

```
<div id="carouselExampleIndicators" class="carousel slide" data-ride="carousel">

    <ol class="carousel-indicators">

        <li data-target="#carouselExampleIndicators" data-slide-to="0" class="active"></li>
```

```
<li data-target="#carouselExampleIndicators" data-slide-to="1"></li>
<li data-target="#carouselExampleIndicators" data-slide-to="2"></li>
</ol>
<div class="carousel-inner">
  <div class="carousel-item active">
    
  </div>
  <div class="carousel-item">
    
  </div>
  <div class="carousel-item">
    
  </div>
</div>
<a class="carousel-control-prev" href="#carouselExampleIndicators" role="button" data-slide="prev">
  <span class="carousel-control-prev-icon" aria-hidden="true"></span>
  <span class="sr-only">Previous</span>
</a>
<a class="carousel-control-next" href="#carouselExampleIndicators" role="button" data-slide="next">
  <span class="carousel-control-next-icon" aria-hidden="true"></span>
  <span class="sr-only">Next</span>
</a>
```

</div>

With captions

Add captions to your slides easily with the .carousel-caption element within any .carousel-item. They can be easily hidden on smaller viewports, as shown below, with optional display utilities. We hide them initially with .d-none and bring them back on medium-sized devices with .d-md-block.

1.

2.

[Previous](#) [Next](#)

```
<div id="carouselExampleCaptions" class="carousel slide" data-ride="carousel">

<ol class="carousel-indicators">

  <li data-target="#carouselExampleCaptions" data-slide-to="0" class="active"></li>

  <li data-target="#carouselExampleCaptions" data-slide-to="1"></li>

  <li data-target="#carouselExampleCaptions" data-slide-to="2"></li>

</ol>

<div class="carousel-inner">

  <div class="carousel-item active">

    <div class="carousel-caption d-none d-md-block">

      <h5>First slide label</h5>

      <p>Nulla vitae elit libero, a pharetra augue mollis interdum.</p>

    </div>

  </div>

  <div class="carousel-item">

    
```

```
<div class="carousel-caption d-none d-md-block">  
    <h5>Second slide label</h5>  
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>  
</div>  
</div>  
  
<div class="carousel-item">  
    <br/>  
    <div class="carousel-caption d-none d-md-block">  
        <h5>Third slide label</h5>  
        <p>Praesent commodo cursus magna, vel scelerisque nisl consectetur.</p>  
    </div>  
</div>  
</div>  
  
<a class="carousel-control-prev" href="#carouselExampleCaptions" role="button" data-slide="prev">  
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>  
    <span class="sr-only">Previous</span>  
</a>  
  
<a class="carousel-control-next" href="#carouselExampleCaptions" role="button" data-slide="next">  
    <span class="carousel-control-next-icon" aria-hidden="true"></span>  
    <span class="sr-only">Next</span>  
</a>  
</div>
```

Crossfade

Add .carousel-fade to your carousel to animate slides with a fade transition instead of a slide.

[Previous](#) [Next](#)

```
<div id="carouselExampleFade" class="carousel slide carousel-fade" data-ride="carousel">

  <div class="carousel-inner">

    <div class="carousel-item active">
      
    </div>

    <div class="carousel-item">
      
    </div>

    <div class="carousel-item">
      
    </div>

  </div>

  <a class="carousel-control-prev" href="#carouselExampleFade" role="button" data-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>

  <a class="carousel-control-next" href="#carouselExampleFade" role="button" data-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</div>
```

```
</a>
```

```
</div>
```

Individual .carousel-item interval

Add data-interval="" to a .carousel-item to change the amount of time to delay between automatically cycling to the next item.

[Previous](#) [Next](#)

```
<div id="carouselExampleInterval" class="carousel slide" data-ride="carousel">

  <div class="carousel-inner">

    <div class="carousel-item active" data-interval="10000">
      
    </div>

    <div class="carousel-item" data-interval="2000">
      
    </div>

    <div class="carousel-item">
      
    </div>

  </div>

  <a class="carousel-control-prev" href="#carouselExampleInterval" role="button" data-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>

  <a class="carousel-control-next" href="#carouselExampleInterval" role="button" data-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</div>
```

```
<span class="carousel-control-next-icon" aria-hidden="true"></span>  
<span class="sr-only">Next</span>  
</a>  
</div>
```

Collapse

Toggle the visibility of content across your project with a few classes and our JavaScript plugins.

Example

Click the buttons below to show and hide another element via class changes:

- .collapse hides content
- .collapsing is applied during transitions
- .collapse.show shows content

You can use a link with the href attribute, or a button with the data-target attribute. In both cases, the data-toggle="collapse" is required.

```
<p>  
  <a class="btn btn-primary" data-toggle="collapse" href="#collapseExample" role="button"  
    aria-expanded="false" aria-controls="collapseExample">  
    Link with href  
  </a>  
  
  <button class="btn btn-primary" type="button" data-toggle="collapse" data-  
    target="#collapseExample" aria-expanded="false" aria-controls="collapseExample">  
    Button with data-target  
  </button>  
  
</p>  
  
<div class="collapse" id="collapseExample">  
  <div class="card card-body">
```

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident.

```
</div>
```

```
</div>
```

[Link with href](#) [Button with data-target](#)

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident.

Multiple targets

A `<button>` or `<a>` can show and hide multiple elements by referencing them with a JQuery selector in its href or data-target attribute. Multiple `<button>` or `<a>` can show and hide an element if they each reference it with their href or data-target attribute as shown in the example below;

```
<p>
```

```
<a class="btn btn-primary" data-toggle="collapse" href="#multiCollapseExample1" role="button" aria-expanded="false" aria-controls="multiCollapseExample1">Toggle first element</a>
```

```
<button class="btn btn-primary" type="button" data-toggle="collapse" data-target="#multiCollapseExample2" aria-expanded="false" aria-controls="multiCollapseExample2">Toggle second element</button>
```

```
<button class="btn btn-primary" type="button" data-toggle="collapse" data-target=".multi-collapse" aria-expanded="false" aria-controls="multiCollapseExample1 multiCollapseExample2">Toggle both elements</button>
```

```
</p>
```

```
<div class="row">
```

```
<div class="col">
```

```
<div class="collapse multi-collapse" id="multiCollapseExample1">
```

```
<div class="card card-body">  
  
    Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad  
    squid. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea  
    proident.  
  
</div>  
  
</div>  
  
</div>  
  
<div class="col">  
  
<div class="collapse multi-collapse" id="multiCollapseExample2">  
  
<div class="card card-body">  
  
    Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad  
    squid. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea  
    proident.  
  
</div>  
  
</div>  
  
</div>  
  
</div>
```

[Toggle first element](#)

[Toggle second element](#)

[Toggle both elements](#)

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident.

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident.

Accordion example

Using the [card](#) component, you can extend the default collapse behavior to create an accordion. To properly achieve the accordion style, be sure to use .accordion as a wrapper.

```
<div class="accordion" id="accordionExample">

  <div class="card">

    <div class="card-header" id="headingOne">
      <h2 class="mb-0">
        <button class="btn btn-link" type="button" data-toggle="collapse" data-target="#collapseOne" aria-expanded="true" aria-controls="collapseOne">
          Collapsible Group Item #1
        </button>
      </h2>
    </div>

    <div id="collapseOne" class="collapse show" aria-labelledby="headingOne" data-parent="#accordionExample">
      <div class="card-body">
        Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them accusamus labore sustainable VHS.
      </div>
    </div>
  </div>

<div class="card">

  <div class="card-header" id="headingTwo">
    <h2 class="mb-0">
```

```
<button class="btn btn-link collapsed" type="button" data-toggle="collapse" data-target="#collapseTwo" aria-expanded="false" aria-controls="collapseTwo">
```

```
    Collapsible Group Item #2
```

```
    </button>
```

```
    </h2>
```

```
    </div>
```

```
    <div id="collapseTwo" class="collapse" aria-labelledby="headingTwo" data-parent="#accordionExample">
```

```
        <div class="card-body">
```

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them accusamus labore sustainable VHS.

```
        </div>
```

```
    </div>
```

```
    </div>
```

```
    <div class="card">
```

```
        <div class="card-header" id="headingThree">
```

```
            <h2 class="mb-0">
```

```
                <button class="btn btn-link collapsed" type="button" data-toggle="collapse" data-target="#collapseThree" aria-expanded="false" aria-controls="collapseThree">
```

```
                    Collapsible Group Item #3
```

```
                </button>
```

```
                </h2>
```

```
            </div>
```

```
<div id="collapseThree" class="collapse" aria-labelledby="headingThree" data-parent="#accordionExample">

<div class="card-body">

    Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them accusamus labore sustainable VHS.

</div>

</div>

</div>

</div>
```

[Collapsible Group Item #1](#)

[Collapsible Group Item #2](#)

Anim pariatur cliche reprehenderit, enim eiusmod high life accusamus terry richardson ad squid. 3 wolf moon officia aute, non cupidatat skateboard dolor brunch. Food truck quinoa nesciunt laborum eiusmod. Brunch 3 wolf moon tempor, sunt aliqua put a bird on it squid single-origin coffee nulla assumenda shoreditch et. Nihil anim keffiyeh helvetica, craft beer labore wes anderson cred nesciunt sapiente ea proident. Ad vegan excepteur butcher vice lomo. Leggings occaecat craft beer farm-to-table, raw denim aesthetic synth nesciunt you probably haven't heard of them accusamus labore sustainable VHS.

[Collapsible Group Item #3](#)

Dropdowns

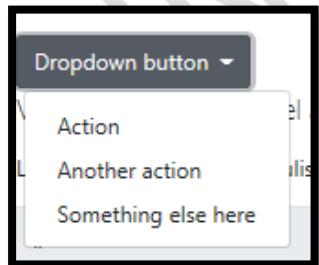
Toggle contextual overlays for displaying lists of links and more with the Bootstrap dropdown plugin.

Dropdowns are built on a third party library, [Popper.js](#), which provides dynamic positioning and viewport detection. Be sure to include [popper.min.js](#) before Bootstrap's JavaScript or use bootstrap.bundle.min.js / bootstrap.bundle.js which contains Popper.js. Popper.js isn't used to position dropdowns in navbars though as dynamic positioning isn't required.

Single button

Any single .btn can be turned into a dropdown toggle with some markup changes. Here's how you can put them to work with either <button> elements:

```
<div class="dropdown">  
  <button class="btn btn-secondary dropdown-toggle" type="button"  
    id="dropdownMenuButton" data-toggle="dropdown" aria-haspopup="true" aria-  
    expanded="false">  
    Dropdown button  
  </button>  
  
  <div class="dropdown-menu" aria-labelledby="dropdownMenuButton">  
    <a class="dropdown-item" href="#">Action</a>  
    <a class="dropdown-item" href="#">Another action</a>  
    <a class="dropdown-item" href="#">Something else here</a>  
  </div>  
</div>
```



And with <a> elements:

```
<div class="dropdown">
```

```
<a class="btn btn-secondary dropdown-toggle" href="#" role="button"
id="dropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">
```

Dropdown link

```
</a>
```

```
<div class="dropdown-menu" aria-labelledby="dropdownMenuLink">
```

```
  <a class="dropdown-item" href="#">Action</a>
```

```
  <a class="dropdown-item" href="#">Another action</a>
```

```
  <a class="dropdown-item" href="#">Something else here</a>
```

```
</div>
```

```
</div>
```

The best part is you can do this with any button variant, too:

```
<!-- Example single danger button -->
```

```
<div class="btn-group">
```

```
  <button type="button" class="btn btn-danger dropdown-toggle" data-toggle="dropdown" aria-
haspopup="true" aria-expanded="false">
```

```
    Action
```

```
  </button>
```

```
  <div class="dropdown-menu">
```

```
    <a class="dropdown-item" href="#">Action</a>
```

```
    <a class="dropdown-item" href="#">Another action</a>
```

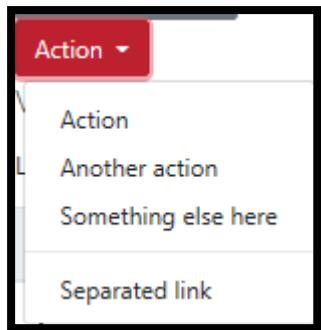
```
    <a class="dropdown-item" href="#">Something else here</a>
```

```
    <div class="dropdown-divider"></div>
```

```
    <a class="dropdown-item" href="#">Separated link</a>
```

```
</div>
```

```
</div>
```



Split button

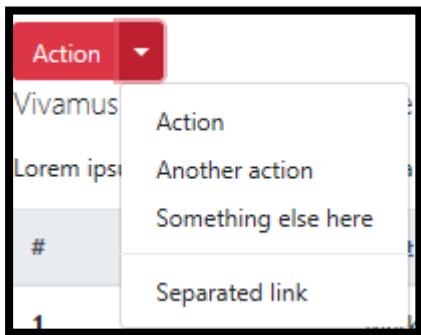
Similarly, create split button dropdowns with virtually the same markup as single button dropdowns, but with the addition of .dropdown-toggle-split for proper spacing around the dropdown caret.

We use this extra class to reduce the horizontal padding on either side of the caret by 25% and remove the margin-left that's added for regular button dropdowns. Those extra changes keep the caret centered in the split button and provide a more appropriately sized hit area next to the main button.

```
<!-- Example split danger button -->

<div class="btn-group">
  <button type="button" class="btn btn-danger">Action</button>
  <button type="button" class="btn btn-danger dropdown-toggle dropdown-toggle-split" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    <span class="sr-only">Toggle Dropdown</span>
  </button>
  <div class="dropdown-menu">
    <a class="dropdown-item" href="#">Action</a>
    <a class="dropdown-item" href="#">Another action</a>
```

```
<a class="dropdown-item" href="#">Something else here</a>  
  
<div class="dropdown-divider"></div>  
  
<a class="dropdown-item" href="#">Separated link</a>  
  
</div>  
  
</div>
```



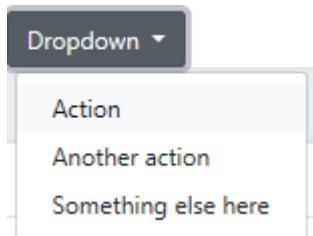
Menu items

you can now optionally use `<button>` elements in your dropdowns instead of just `<a>`s.

```
<div class="dropdown">  
  
  <button class="btn btn-secondary dropdown-toggle" type="button" id="dropdownMenu2"  
        data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">  
  
    Dropdown  
  
  </button>  
  
  <div class="dropdown-menu" aria-labelledby="dropdownMenu2">  
  
    <button class="dropdown-item" type="button">Action</button>  
  
    <button class="dropdown-item" type="button">Another action</button>  
  
    <button class="dropdown-item" type="button">Something else here</button>  
  
</div>
```

```
</div>
```

```
</div>
```

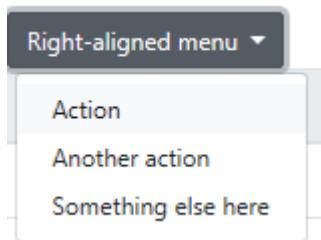


Menu alignment

By default, a dropdown menu is automatically positioned 100% from the top and along the left side of its parent. Add `.dropdown-menu-right` to a `.dropdown-menu` to right align the dropdown menu.

Heads up! Dropdowns are positioned thanks to Popper.js (except when they are contained in a navbar).

```
<div class="btn-group">  
  <button type="button" class="btn btn-secondary dropdown-toggle" data-toggle="dropdown"  
         aria-haspopup="true" aria-expanded="false">  
    Right-aligned menu  
  </button>  
  <div class="dropdown-menu dropdown-menu-right">  
    <button class="dropdown-item" type="button">Action</button>  
    <button class="dropdown-item" type="button">Another action</button>  
    <button class="dropdown-item" type="button">Something else here</button>  
  </div>  
</div>
```



Forms

Examples and usage guidelines for form control styles, layout options, and custom components for creating a wide variety of forms.

Overview

Bootstrap's form controls expand on [our Rebooted form styles](#) with classes. Use these classes to opt into their customized displays for a more consistent rendering across browsers and devices.

```
<form>

<div class="form-group">
  <label for="exampleInputEmail1">Email address</label>
  <input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp">
  <small id="emailHelp" class="form-text text-muted">We'll never share your email with anyone else.</small>
</div>

<div class="form-group">
  <label for="exampleInputPassword1">Password</label>
  <input type="password" class="form-control" id="exampleInputPassword1">
</div>

<div class="form-group form-check">
  <input type="checkbox" class="form-check-input" id="exampleCheck1">
  <label class="form-check-label" for="exampleCheck1">Check me out</label>
```

```
</div>

<button type="submit" class="btn btn-primary">Submit</button>

</form>
```

The form includes a placeholder message 'We'll never share your email with anyone else.' below the email input field.

Form controls

Textual form controls—like `<input>`s, `<select>`s, and `<textarea>`s—are styled with the `.form-control` class. Included are styles for general appearance, focus state, sizing, and more.

```
<form>

<div class="form-group">
  <label for="exampleFormControlInput1">Email address</label>
  <input type="email" class="form-control" id="exampleFormControlInput1"
  placeholder="name@example.com">
</div>

<div class="form-group">
  <label for="exampleFormControlSelect1">Example select</label>
  <select class="form-control" id="exampleFormControlSelect1">
    <option>1</option>
    <option>2</option>
    <option>3</option>
  </select>
</div>
```

```
<option>4</option>
<option>5</option>
</select>
</div>

<div class="form-group">
    <label for="exampleFormControlSelect2">Example multiple select</label>
    <select multiple class="form-control" id="exampleFormControlSelect2">
        <option>1</option>
        <option>2</option>
        <option>3</option>
        <option>4</option>
        <option>5</option>
    </select>
</div>

<div class="form-group">
    <label for="exampleFormControlTextarea1">Example textarea</label>
    <textarea class="form-control" id="exampleFormControlTextarea1" rows="3"></textarea>
</div>
</form>
```

The screenshot shows a web form with four input fields:

- Email address: A single-line text input field containing "name@example.com".
- Example select: A single-select dropdown menu with the value "1" selected.
- Example multiple select: A multiple-select dropdown menu with items 1, 2, 3, and 4. Item 4 is currently selected.
- Example textarea: A multi-line text area with a single character "I" typed into it.

For file inputs, swap the .form-control for .form-control-file.

Example file input

```
<form>  
  
<div class="form-group">  
  
  <label for="exampleFormControlFile1">Example file input</label>  
  
  <input type="file" class="form-control-file" id="exampleFormControlFile1">  
  
</div>  
  
</form>
```

Sizing

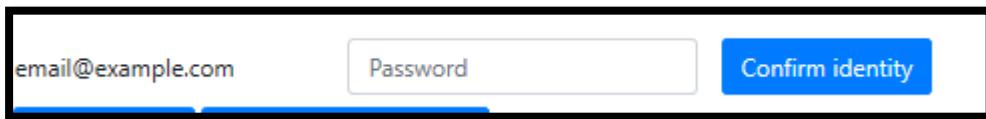
Set heights using classes like .form-control-lg and .form-control-sm.

```
<input class="form-control form-control-lg" type="text" placeholder=".form-control-lg">  
  
<input class="form-control" type="text" placeholder="Default input">  
  
<input class="form-control form-control-sm" type="text" placeholder=".form-control-sm">
```

```
<select class="form-control form-control-lg">  
  <option>Large select</option>  
</select>  
  
<select class="form-control">  
  <option>Default select</option>  
</select>  
  
<select class="form-control form-control-sm">  
  <option>Small select</option>  
</select>
```

Example

```
<form class="form-inline">  
  
  <div class="form-group mb-2">  
    <label for="staticEmail2" class="sr-only">Email</label>  
    <input type="text" readonly class="form-control-plaintext" id="staticEmail2"  
           value="email@example.com"/>  
  </div>  
  
  <div class="form-group mx-sm-3 mb-2">  
    <label for="inputPassword2" class="sr-only">Password</label>  
    <input type="password" class="form-control" id="inputPassword2"  
           placeholder="Password"/>  
  </div>  
  
  <button type="submit" class="btn btn-primary mb-2">Confirm identity</button>  
</form>
```



Range Inputs

Set horizontally scrollable range inputs using .form-control-range.

Example Range input

```
<form>

<div class="form-group">

<label for="formControlRange">Example Range input</label>

<input type="range" class="form-control-range" id="formControlRange">

</div>

</form>
```

Checkboxes and radios

Default checkboxes and radios are improved upon with the help of .form-check, **a single class for both input types that improves the layout and behavior of their HTML elements**.

Checkboxes are for selecting one or several options in a list, while radios are for selecting one option from many.

```
<div class="form-check">

<input class="form-check-input" type="checkbox" value="" id="defaultCheck1">

<label class="form-check-label" for="defaultCheck1">

  Default checkbox

</label>

</div>

<div class="form-check">

<input class="form-check-input" type="checkbox" value="" id="defaultCheck2" disabled>
```

```
<label class="form-check-label" for="defaultCheck2">
```

Disabled checkbox

```
</label>
```

```
</div>
```

Or

```
<div class="form-check">
```

```
  <input class="form-check-input" type="radio" name="exampleRadios" id="exampleRadios1" value="option1" checked>
```

```
  <label class="form-check-label" for="exampleRadios1">
```

Default radio

```
</label>
```

```
</div>
```

```
<div class="form-check">
```

```
  <input class="form-check-input" type="radio" name="exampleRadios" id="exampleRadios2" value="option2">
```

```
  <label class="form-check-label" for="exampleRadios2">
```

Second default radio

```
</label>
```

```
</div>
```

```
<div class="form-check">
```

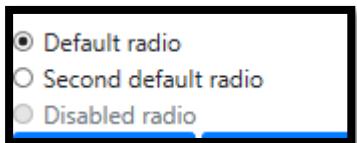
```
  <input class="form-check-input" type="radio" name="exampleRadios" id="exampleRadios3" value="option3" disabled>
```

```
  <label class="form-check-label" for="exampleRadios3">
```

Disabled radio

```
</label>
```

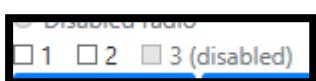
```
</div>
```



Inline

Group checkboxes or radios on the same horizontal row by adding `.form-check-inline` to any `.form-check`.

```
<div class="form-check form-check-inline">  
  <input class="form-check-input" type="checkbox" id="inlineCheckbox1" value="option1">  
  <label class="form-check-label" for="inlineCheckbox1">1</label>  
</div>  
  
<div class="form-check form-check-inline">  
  <input class="form-check-input" type="checkbox" id="inlineCheckbox2" value="option2">  
  <label class="form-check-label" for="inlineCheckbox2">2</label>  
</div>  
  
<div class="form-check form-check-inline">  
  <input class="form-check-input" type="checkbox" id="inlineCheckbox3" value="option3" disabled>  
  <label class="form-check-label" for="inlineCheckbox3">3 (disabled)</label>  
</div>
```



Form groups

The `.form-group` class is the easiest way to add some structure to forms. It provides a flexible class that encourages proper grouping of labels, controls, optional help text, and form validation messaging. By default it only applies margin-bottom, but it picks up additional styles in `.form-inline` as needed. Use it with `<fieldset>`s, `<div>`s, or nearly any other element.

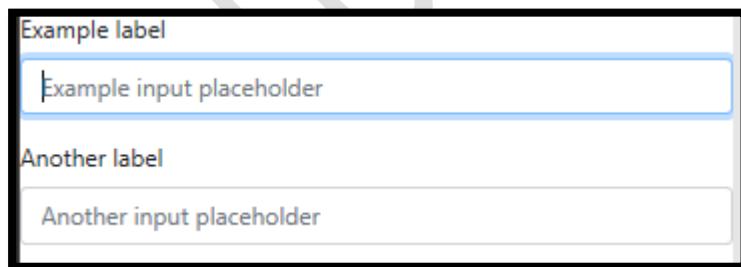
Example

```
<form>

  <div class="form-group">
    <label for="formGroupExampleInput">Example label</label>
    <input type="text" class="form-control" id="formGroupExampleInput"
      placeholder="Example input placeholder">
  </div>

  <div class="form-group">
    <label for="formGroupExampleInput2">Another label</label>
    <input type="text" class="form-control" id="formGroupExampleInput2"
      placeholder="Another input placeholder">
  </div>

</form>
```



Form grid

More complex forms can be built using our grid classes. Use these for form layouts that require multiple columns, varied widths, and additional alignment options.

Example

```
<form>

<div class="form-row">

    <div class="form-group col-md-6">
        <label for="inputEmail4">Email</label>
        <input type="email" class="form-control" id="inputEmail4">
    </div>

    <div class="form-group col-md-6">
        <label for="inputPassword4">Password</label>
        <input type="password" class="form-control" id="inputPassword4">
    </div>

</div>

<div class="form-group">
    <label for="inputAddress">Address</label>
    <input type="text" class="form-control" id="inputAddress" placeholder="1234 Main St">
</div>

<div class="form-group">
    <label for="inputAddress2">Address 2</label>
    <input type="text" class="form-control" id="inputAddress2" placeholder="Apartment, studio,
or floor">
</div>

<div class="form-row">

    <div class="form-group col-md-6">
        <label for="inputCity">City</label>
        <input type="text" class="form-control" id="inputCity">
    </div>

```

```
</div>

<div class="form-group col-md-4">
    <label for="inputState">State</label>
    <select id="inputState" class="form-control">
        <option selected>Choose...</option>
        <option>...</option>
    </select>
</div>

<div class="form-group col-md-2">
    <label for="inputZip">Zip</label>
    <input type="text" class="form-control" id="inputZip">
</div>
</div>

<div class="form-group">
    <div class="form-check">
        <input class="form-check-input" type="checkbox" id="gridCheck">
        <label class="form-check-label" for="gridCheck">
            Check me out
        </label>
    </div>
</div>

<button type="submit" class="btn btn-primary">Sign in</button>
</form>
```

The form consists of several input fields and controls:

- Email: A text input field.
- Password: A text input field.
- Address: A text input field containing "1234 Main St".
- Address 2: A text input field containing "Apartment, studio, or floor".
- City: A text input field.
- State: A dropdown menu labeled "Choose...".
- Zip: A text input field.
- Check me out: A checkbox.
- Sign In: A blue rectangular button.

Horizontal form

Create horizontal forms with the grid by adding the .row class to form groups and using the .col-*-* classes to specify the width of your labels and controls. Be sure to add .col-form-label to your <label>s as well so they're vertically centered with their associated form controls.

Example

```
<form>

<div class="form-group row">
    <label for="inputEmail3" class="col-sm-2 col-form-label">Email</label>
    <div class="col-sm-10">
        <input type="email" class="form-control" id="inputEmail3">
    </div>
</div>

<div class="form-group row">
    <label for="inputPassword3" class="col-sm-2 col-form-label">Password</label>
    <div class="col-sm-10">
        <input type="password" class="form-control" id="inputPassword3">
    </div>
</div>

<fieldset class="form-group">
```

```
<div class="row">

<legend class="col-form-label col-sm-2 pt-0">Radios</legend>

<div class="col-sm-10">

<div class="form-check">

    <input class="form-check-input" type="radio" name="gridRadios" id="gridRadios1"
value="option1" checked>

    <label class="form-check-label" for="gridRadios1">
        First radio
    </label>

</div>

<div class="form-check">

    <input class="form-check-input" type="radio" name="gridRadios" id="gridRadios2"
value="option2">

    <label class="form-check-label" for="gridRadios2">
        Second radio
    </label>

</div>

<div class="form-check disabled">

    <input class="form-check-input" type="radio" name="gridRadios" id="gridRadios3"
value="option3" disabled>

    <label class="form-check-label" for="gridRadios3">
        Third disabled radio
    </label>

</div>

</div>
```

```
</div>

</fieldset>

<div class="form-group row">
    <div class="col-sm-2">Checkbox</div>
    <div class="col-sm-10">
        <div class="form-check">
            <input class="form-check-input" type="checkbox" id="gridCheck1">
            <label class="form-check-label" for="gridCheck1">
                Example checkbox
            </label>
        </div>
    </div>
</div>

<div class="form-group row">
    <div class="col-sm-10">
        <button type="submit" class="btn btn-primary">Sign in</button>
    </div>
</div>
</form>
```

The form consists of several input fields and controls:

- Email: A text input field.
- Password: A text input field.
- Radios: A group of three radio buttons. The first one is checked (blue outline) and labeled "First radio". The second one is unselected (grey outline) and labeled "Second radio". The third one is unselected (grey outline) and labeled "Third disabled radio" (disabled).
- Checkbox: A checkbox input field labeled "Example checkbox".

A blue "Sign in" button is located at the bottom left of the form area.

Horizontal form label sizing

Be sure to use .col-form-label-sm or .col-form-label-lg to your <label>s or <legend>s to correctly follow the size of .form-control-lg and .form-control-sm.

```
<form>

<div class="form-group row">

    <label for="colFormLabelSm" class="col-sm-2 col-form-label col-form-label-sm">Email</label>

    <div class="col-sm-10">
        <input type="email" class="form-control form-control-sm" id="colFormLabelSm" placeholder="col-form-label-sm">
    </div>
</div>

<div class="form-group row">
    <label for="colFormLabel" class="col-sm-2 col-form-label">Email</label>
    <div class="col-sm-10">
        <input type="email" class="form-control" id="colFormLabel" placeholder="col-form-label">
    </div>
</div>
```

```
<div class="form-group row">  
    <label for="colFormLabelLg" class="col-sm-2 col-form-label col-form-label-lg">Email</label>  
    <div class="col-sm-10">  
        <input type="email" class="form-control form-control-lg" id="colFormLabelLg" placeholder="col-form-label-lg">  
    </div>  
</div>  
</form>
```



Example 2

```
<form>  
    <div class="form-row align-items-center">  
        <div class="col-sm-3 my-1">  
            <label class="sr-only" for="inlineFormInputName">Name</label>  
            <input type="text" class="form-control" id="inlineFormInputName" placeholder="Jane Doe">  
        </div>  
        <div class="col-sm-3 my-1">  
            <label class="sr-only" for="inlineFormInputGroupUsername">Username</label>  
            <div class="input-group">
```

```
<div class="input-group-prepend">  
  <div class="input-group-text">@</div>  
</div>  
  
  <input type="text" class="form-control" id="inlineFormInputGroupUsername"  
placeholder="Username">  
  </div>  
  
</div>  
  
<div class="col-auto my-1">  
  <div class="form-check">  
    <input class="form-check-input" type="checkbox" id="autoSizingCheck2">  
    <label class="form-check-label" for="autoSizingCheck2">  
      Remember me  
    </label>  
  </div>  
</div>  
  
<div class="col-auto my-1">  
  <button type="submit" class="btn btn-primary">Submit</button>  
</div>  
</div>  
</form>
```



The image shows a screenshot of a web page with an inline form. At the top left, there is a text input field containing "Jane Doe". Next to it is another text input field with a placeholder "@ Username". Below these fields is a checkbox labeled "Remember me" followed by a blue "Submit" button.

Inline forms

Use the .form-inline class to display a series of labels, form controls, and buttons on a single horizontal row. Form controls within inline forms vary slightly from their default states.

```
<form class="form-inline">

    <label class="sr-only" for="inlineFormInputName2">Name</label>

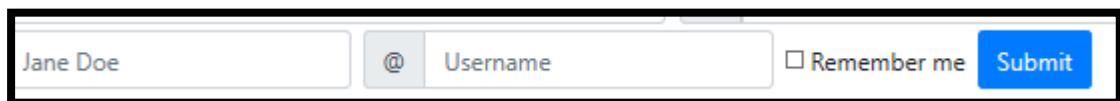
    <input type="text" class="form-control mb-2 mr-sm-2" id="inlineFormInputName2"
placeholder="Jane Doe">

    <label class="sr-only" for="inlineFormInputGroupUsername2">Username</label>

    <div class="input-group mb-2 mr-sm-2">
        <div class="input-group-prepend">
            <div class="input-group-text">@</div>
        </div>
        <input type="text" class="form-control" id="inlineFormInputGroupUsername2"
placeholder="Username">
    </div>

    <div class="form-check mb-2 mr-sm-2">
        <input class="form-check-input" type="checkbox" id="inlineFormCheck">
        <label class="form-check-label" for="inlineFormCheck">
            Remember me
        </label>
    </div>

    <button type="submit" class="btn btn-primary mb-2">Submit</button>
</form>
```



Alternatives to hidden labels

Assistive technologies such as screen readers will have trouble with your forms if you don't include a label for every input. For these inline forms, you can hide the labels using the .sr-only class. There are further alternative methods of providing a label for assistive technologies, such as the aria-label, aria-labelledby or title attribute. If none of these are present, assistive technologies may resort to using the placeholder attribute, if present, but note that use of placeholder as a replacement for other labelling methods is not advised.

Associating help text with form controls

Help text should be explicitly associated with the form control it relates to using the aria-describedby attribute. This will ensure that assistive technologies—such as screen readers—will announce this help text when the user focuses or enters the control.

Help text below inputs can be styled with .form-text. This class includes display: block and adds some top margin for easy spacing from the inputs above.

```
<label for="inputPassword5">Password</label>  
  
<input type="password" id="inputPassword5" class="form-control" aria-  
describedby="passwordHelpBlock">  
  
<small id="passwordHelpBlock" class="form-text text-muted">  
  
Your password must be 8-20 characters long, contain letters and numbers, and must not contain  
spaces, special characters, or emoji.  
  
</small>
```



Validation

Provide valuable, actionable feedback to your users with HTML5 form validation—[available in all our supported browsers](#). Choose from the browser default validation feedback, or implement custom messages with our built-in classes and starter JavaScript.

We currently recommend using custom validation styles, as native browser default validation messages are not consistently exposed to assistive technologies in all browsers (most notably, Chrome on desktop and mobile).

How it works

Here's how form validation works with Bootstrap:

- HTML form validation is applied via CSS's two pseudo-classes, :invalid and :valid. It applies to <input>, <select>, and <textarea> elements.
- Bootstrap scopes the :invalid and :valid styles to parent .was-validated class, usually applied to the <form>. Otherwise, any required field without a value shows up as invalid on page load. This way, you may choose when to activate them (typically after form submission is attempted).
- To reset the appearance of the form (for instance, in the case of dynamic form submissions using AJAX), remove the .was-validated class from the <form> again after submission.
- As a fallback, .is-invalid and .is-valid classes may be used instead of the pseudo-classes for [server side validation](#). They do not require a .was-validated parent class.
- Due to constraints in how CSS works, we cannot (at present) apply styles to a <label> that comes before a form control in the DOM without the help of custom JavaScript.
- All modern browsers support the [constraint validation API](#), a series of JavaScript methods for validating form controls.
- Feedback messages may utilize the [browser defaults](#) (different for each browser, and unstylistable via CSS) or our custom feedback styles with additional HTML and CSS.
- You may provide custom validity messages with setCustomValidity in JavaScript.

Example

Custom styles

For custom Bootstrap form validation messages, you'll need to add the novalidate boolean attribute to your <form>. This disables the browser default feedback tooltips, but still provides access to the form validation APIs in JavaScript. Try to submit the form below; our JavaScript will intercept the submit button and relay feedback to you. When attempting to submit, you'll see the :invalid and :valid styles applied to your form controls.

Custom feedback styles apply custom colors, borders, focus styles, and background icons to better communicate feedback. Background icons for <select>s are only available with .custom-select, and not .form-control.

```
<form class="needs-validation" novalidate>  
<div class="form-row">  
  <div class="col-md-4 mb-3">
```

```
<label for="validationCustom01">First name</label>

<input type="text" class="form-control" id="validationCustom01" value="Mark" required>

<div class="valid-feedback">
  Looks good!
</div>

</div>

<div class="col-md-4 mb-3">

  <label for="validationCustom02">Last name</label>

  <input type="text" class="form-control" id="validationCustom02" value="Otto" required>

  <div class="valid-feedback">
    Looks good!
  </div>

</div>

<div class="col-md-4 mb-3">

  <label for="validationCustomUsername">Username</label>

  <div class="input-group">
    <div class="input-group-prepend">
      <span class="input-group-text" id="inputGroupPrepend">@</span>
    </div>
    <input type="text" class="form-control" id="validationCustomUsername" aria-describedby="inputGroupPrepend" required>
  <div class="invalid-feedback">
    Please choose a username.
  </div>
```

```
</div>

</div>

</div>

<div class="form-row">

<div class="col-md-6 mb-3">

    <label for="validationCustom03">City</label>

    <input type="text" class="form-control" id="validationCustom03" required>

    <div class="invalid-feedback">

        Please provide a valid city.

    </div>

</div>

<div class="col-md-3 mb-3">

    <label for="validationCustom04">State</label>

    <select class="custom-select" id="validationCustom04" required>

        <option selected disabled value="">Choose...</option>

        <option>...</option>

    </select>

    <div class="invalid-feedback">

        Please select a valid state.

    </div>

</div>

<div class="col-md-3 mb-3">

    <label for="validationCustom05">Zip</label>
```

```
<input type="text" class="form-control" id="validationCustom05" required>

<div class="invalid-feedback">
    Please provide a valid zip.
</div>

</div>

<div class="form-group">
    <div class="form-check">
        <input class="form-check-input" type="checkbox" value="" id="invalidCheck" required>
        <label class="form-check-label" for="invalidCheck">
            Agree to terms and conditions
        </label>
        <div class="invalid-feedback">
            You must agree before submitting.
        </div>
    </div>
</div>

<button class="btn btn-primary" type="submit">Submit form</button>
</form>

<script>
// Example starter JavaScript for disabling form submissions if there are invalid fields
(function() {
```

```
'use strict';

window.addEventListener('load', function() {

    // Fetch all the forms we want to apply custom Bootstrap validation styles to

    var forms = document.getElementsByClassName('needs-validation');

    // Loop over them and prevent submission

    var validation = Array.prototype.filter.call(forms, function(form) {

        form.addEventListener('submit', function(event) {

            if (form.checkValidity() === false) {

                event.preventDefault();

                event.stopPropagation();

            }

            form.classList.add('was-validated');

        }, false);

    });

}, false);

})0;

</script>
```



Modal

Use Bootstrap's JavaScript modal plugin to add dialogs to your site for lightboxes, user notifications, or completely custom content.

The following are to be taken into consideration before using modals

- Modals are built with HTML, CSS, and JavaScript. They're positioned over everything else in the document and remove scroll from the <body> so that modal content scrolls instead.
- Clicking on the modal "backdrop" will automatically close the modal.
- Bootstrap only supports one modal window at a time. Nested modals aren't supported as we believe them to be poor user experiences.
- Modals use position: fixed, which can sometimes be a bit particular about its rendering. Whenever possible, place your modal HTML in a top-level position to avoid potential interference from other elements. You'll likely run into issues when nesting a .modal within another fixed element.
- Once again, due to position: fixed, there are some caveats with using modals on mobile devices. [See our browser support docs](#) for details.
- Due to how HTML5 defines its semantics, [the autofocus HTML attribute](#) has no effect in Bootstrap modals. To achieve the same effect, use some custom JavaScript:

```
$('#myModal').on('shown.bs.modal', function () {  
    $('#myInput').trigger('focus')  
})
```

Examples

Modal components

Below is a *static* modal example (meaning its position and display have been overridden). Included are the modal header, modal body (required for padding), and modal footer (optional).

Example 1

```
<div class="modal" tabindex="-1" role="dialog">
```

```
<div class="modal-dialog" role="document">

  <div class="modal-content">

    <div class="modal-header">

      <h5 class="modal-title">Modal title</h5>

      <button type="button" class="close" data-dismiss="modal" aria-label="Close">
        <span aria-hidden="true">&times;</span>
      </button>

    </div>

    <div class="modal-body">

      <p>Modal body text goes here.</p>

    </div>

    <div class="modal-footer">

      <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>

      <button type="button" class="btn btn-primary">Save changes</button>

    </div>

  </div>

</div>

<!-- Button trigger modal -->

<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal">

  Launch demo modal

</button>
```

```
<!-- Modal -->

<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">

  <div class="modal-dialog" role="document">

    <div class="modal-content">

      <div class="modal-header">

        <h5 class="modal-title" id="exampleModalLabel">Modal title</h5>

        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>

      </div>

      <div class="modal-body">
        ...
      </div>

      <div class="modal-footer">

        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>

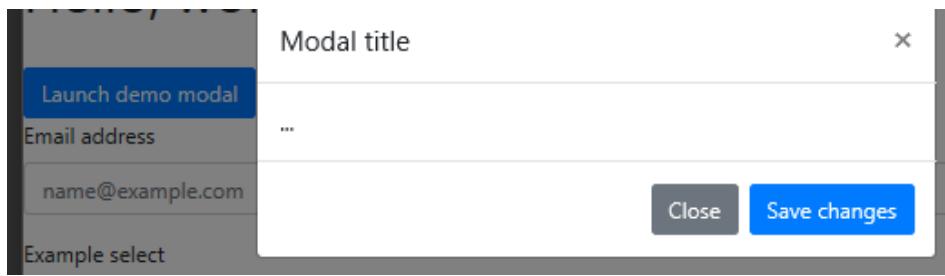
        <button type="button" class="btn btn-primary">Save changes</button>

      </div>

    </div>

  </div>

</div>
```



Varying modal content

Have a bunch of buttons that all trigger the same modal with slightly different contents? Use `event.relatedTarget` and [HTML data-* attributes](#) (possibly [via jQuery](#)) to vary the contents of the modal depending on which button was clicked.

```
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal" data-whatever="@mdo">Open modal for @mdo</button>
```

```
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal" data-whatever="@fat">Open modal for @fat</button>
```

```
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal" data-whatever="@getbootstrap">Open modal for @getbootstrap</button>
```

```
<div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
```

```
  <div class="modal-dialog" role="document">
```

```
    <div class="modal-content">
```

```
      <div class="modal-header">
```

```
        <h5 class="modal-title" id="exampleModalLabel">New message</h5>
```

```
      <button type="button" class="close" data-dismiss="modal" aria-label="Close">
```

```
        <span aria-hidden="true">&times;</span>
```

```
      </button>
```

```
    </div>
```

```
<div class="modal-body">

<form>

  <div class="form-group">
    <label for="recipient-name" class="col-form-label">Recipient:</label>
    <input type="text" class="form-control" id="recipient-name">
  </div>

  <div class="form-group">
    <label for="message-text" class="col-form-label">Message:</label>
    <textarea class="form-control" id="message-text"></textarea>
  </div>

</form>

</div>

<div class="modal-footer">

  <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
  <button type="button" class="btn btn-primary">Send message</button>
</div>
</div>
</div>

$(#exampleModal').on('show.bs.modal', function (event) {

  var button = $(event.relatedTarget) // Button that triggered the modal

  var recipient = button.data('whatever') // Extract info from data-* attributes

  // If necessary, you could initiate an AJAX request here (and then do the updating in a
  callback).
```

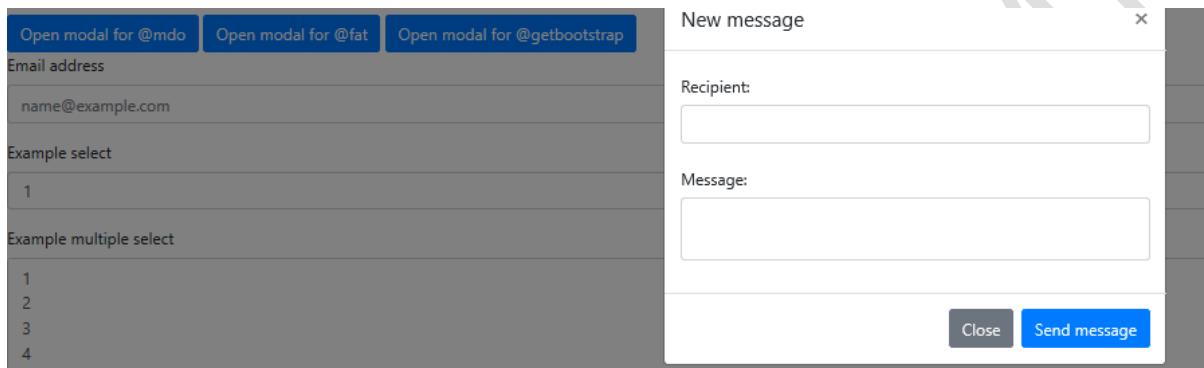
```
// Update the modal's content. We'll use jQuery here, but you could use a data binding library or other methods instead.
```

```
var modal = $(this)

modal.find('.modal-title').text('New message to ' + recipient)

modal.find('.modal-body input').val(recipient)

})
```



Navs

Navigation available in Bootstrap share general markup and styles, from the base .nav class to the active and disabled states.

The base .nav component is built with flexbox and provide a strong foundation for building all types of navigation components. It includes some style overrides (for working with lists), some link padding for larger hit areas, and basic disabled styling.

The base .nav component does not include any .active state. The following examples include the class, mainly to demonstrate that this particular class does not trigger any special styling.

Example

```
<ul class="nav">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item">
```

```
<a class="nav-link" href="#">Link</a>  
</li>  
  
<li class="nav-item">  
  <a class="nav-link" href="#">Link</a>  
</li>  
  
<li class="nav-item">  
  <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>  
</li>  
  
</ul>
```



Active Link Link Disabled

Horizontal alignment

By default, navs are left-aligned, but you can easily change them to center or right aligned.

Centered with .justify-content-center:

```
<ul class="nav justify-content-center">  
  <li class="nav-item">  
    <a class="nav-link active" href="#">Active</a>  
  </li>  
  <li class="nav-item">  
    <a class="nav-link" href="#">Link</a>  
  </li>  
  <li class="nav-item">
```

```
<a class="nav-link" href="#">Link</a>  
</li>  
  
<li class="nav-item">  
  <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>  
</li>
```

Vertical

Stack your navigation by changing the flex item direction with the .flex-column utility. Need to stack them on some viewports but not others? Use the responsive versions (e.g., .flex-sm-column).

```
<ul class="nav flex-column">  
  <li class="nav-item">  
    <a class="nav-link active" href="#">Active</a>  
  </li>  
  <li class="nav-item">  
    <a class="nav-link" href="#">Link</a>  
  </li>  
  <li class="nav-item">  
    <a class="nav-link" href="#">Link</a>  
  </li>  
  <li class="nav-item">  
    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>  
  </li>  
</ul>
```



Tabs

Takes the basic nav from above and adds the .nav-tabs class to generate a tabbed interface.

```
<ul class="nav nav-tabs">  
  <li class="nav-item">  
    <a class="nav-link active" href="#">Active</a>  
  </li>  
  <li class="nav-item">  
    <a class="nav-link" href="#">Link</a>  
  </li>  
  <li class="nav-item">  
    <a class="nav-link" href="#">Link</a>  
  </li>  
  <li class="nav-item">  
    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>  
  </li>  
</ul>
```



Fill and justify

Force your .nav's contents to extend the full available width one of two modifier classes. To proportionately fill all available space with your .nav-items, use .nav-fill. Notice that all horizontal space is occupied, but not every nav item has the same width.

```
<ul class="nav nav-pills nav-fill">  
  
  <li class="nav-item">  
    <a class="nav-link active" href="#">Active</a>  
  </li>  
  
  <li class="nav-item">  
    <a class="nav-link" href="#">Much longer nav link</a>  
  </li>  
  
  <li class="nav-item">  
    <a class="nav-link" href="#">Link</a>  
  </li>  
  
  <li class="nav-item">  
    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>  
  </li>  
  
</ul>
```

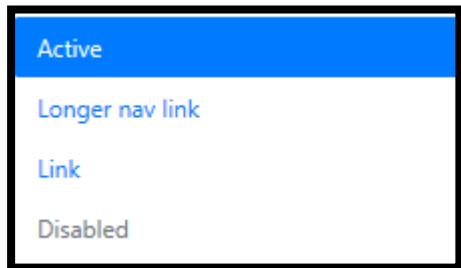


Working with flex utilities

If you need responsive nav variations, consider using a series of [flexbox utilities](#). While more verbose, these utilities offer greater customization across responsive breakpoints.

Example

```
<nav class="nav nav-pills flex-column flex-sm-row">  
  <a class="flex-sm-fill text-sm-center nav-link active" href="#">Active</a>  
  <a class="flex-sm-fill text-sm-center nav-link" href="#">Longer nav link</a>  
  <a class="flex-sm-fill text-sm-center nav-link" href="#">Link</a>  
  <a class="flex-sm-fill text-sm-center nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>  
</nav>
```



Regarding accessibility

If you're using navs to provide a navigation bar, be sure to add a role="navigation" to the most logical parent container of the ``, or wrap a `<nav>` element around the whole navigation. Do not add the role to the `` itself, as this would prevent it from being announced as an actual list by assistive technologies.

Note that navigation bars, even if visually styled as tabs with the `.nav-tabs` class, should **not** be given `role="tablist"`, `role="tab"` or `role="tabpanel"` attributes.

Using dropdowns

Add dropdown menus with a little extra HTML and the [dropdowns JavaScript plugin](#).

EXAMPLE

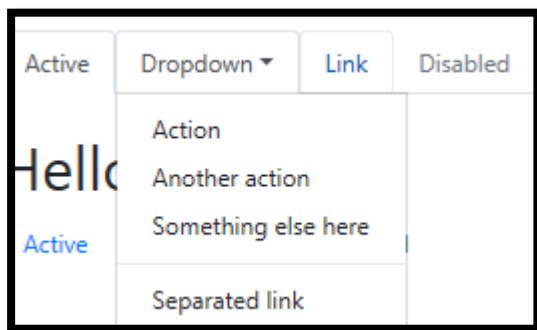
```
<ul class="nav nav-tabs">  
  <li class="nav-item">  
    <a class="nav-link active" href="#">Active</a>
```

```
</li>

<li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" data-toggle="dropdown" href="#" role="button" aria-
haspopup="true" aria-expanded="false">Dropdown</a>
    <div class="dropdown-menu">
        <a class="dropdown-item" href="#">Action</a>
        <a class="dropdown-item" href="#">Another action</a>
        <a class="dropdown-item" href="#">Something else here</a>
        <div class="dropdown-divider"></div>
        <a class="dropdown-item" href="#">Separated link</a>
    </div>
</li>

<li class="nav-item">
    <a class="nav-link" href="#">Link</a>
</li>

<li class="nav-item">
    <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
</li>
</ul>
```



Pills with dropdowns

```
<ul class="nav nav-pills">  
  
<li class="nav-item">  
  <a class="nav-link active" href="#">Active</a>  
  
</li>  
  
<li class="nav-item dropdown">  
  <a class="nav-link dropdown-toggle" data-toggle="dropdown" href="#" role="button" aria-haspopup="true" aria-expanded="false">Dropdown</a>  
  
  <div class="dropdown-menu">  
    <a class="dropdown-item" href="#">Action</a>  
    <a class="dropdown-item" href="#">Another action</a>  
    <a class="dropdown-item" href="#">Something else here</a>  
    <div class="dropdown-divider"></div>  
    <a class="dropdown-item" href="#">Separated link</a>  
  </div>  
  </li>  
  
<li class="nav-item">  
  <a class="nav-link" href="#">Link</a>  
  </li>
```

```
<li class="nav-item">  
  <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>  
</li>  
</ul>
```

Using data attributes

You can activate a tab or pill navigation without writing any JavaScript by simply specifying data-toggle="tab" or data-toggle="pill" on an element. Use these data attributes on .nav-tabs or .nav-pills as shown in the example below;

```
<!-- Nav tabs -->  
  
<ul class="nav nav-tabs" id="myTab" role="tablist">  
  <li class="nav-item">  
    <a class="nav-link active" id="home-tab" data-toggle="tab" href="#home" role="tab" aria-controls="home" aria-selected="true">Home</a>  
  </li>  
  <li class="nav-item">  
    <a class="nav-link" id="profile-tab" data-toggle="tab" href="#profile" role="tab" aria-controls="profile" aria-selected="false">Profile</a>  
  </li>  
  <li class="nav-item">  
    <a class="nav-link" id="messages-tab" data-toggle="tab" href="#messages" role="tab" aria-controls="messages" aria-selected="false">Messages</a>  
  </li>  
  <li class="nav-item">  
    <a class="nav-link" id="settings-tab" data-toggle="tab" href="#settings" role="tab" aria-controls="settings" aria-selected="false">Settings</a>  
  </li>
```

```
</ul>
```

```
<!-- Tab panes -->
```

```
<div class="tab-content">
```

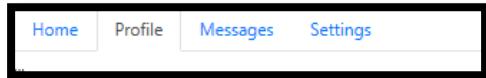
```
  <div class="tab-pane active" id="home" role="tabpanel" aria-labelledby="home-tab">...</div>
```

```
  <div class="tab-pane" id="profile" role="tabpanel" aria-labelledby="profile-tab">...</div>
```

```
  <div class="tab-pane" id="messages" role="tabpanel" aria-labelledby="messages-tab">...</div>
```

```
  <div class="tab-pane" id="settings" role="tabpanel" aria-labelledby="settings-tab">...</div>
```

```
</div>
```



Via JavaScript

Enable tabbable tabs via JavaScript (each tab needs to be activated individually):

```
$('#myTab a').on('click', function (e) {  
  e.preventDefault()  
  $(this).tab('show')  
})
```

You can activate individual tabs in several ways:

```
$('#myTab a[href="#profile"]').tab('show') // Select tab by name  
$('#myTab li:first-child a').tab('show') // Select first tab  
$('#myTab li:last-child a').tab('show') // Select last tab
```

```
$('#myTab li:nth-child(3) a').tab('show') // Select third tab
```

Methods

Asynchronous methods and transitions

All API methods are **asynchronous** and start a **transition**. They return to the caller as soon as the transition is started but **before it ends**. In addition, a method call on a **transitioning component will be ignored**.

\$().tab

Activates a tab element and content container. Tab should have either a data-target or an href targeting a container node in the DOM.

```
<ul class="nav nav-tabs" id="myTab" role="tablist">

<li class="nav-item">

  <a class="nav-link active" id="home-tab" data-toggle="tab" href="#home" role="tab" aria-controls="home" aria-selected="true">Home</a>

</li>

<li class="nav-item">

  <a class="nav-link" id="profile-tab" data-toggle="tab" href="#profile" role="tab" aria-controls="profile" aria-selected="false">Profile</a>

</li>

<li class="nav-item">

  <a class="nav-link" id="messages-tab" data-toggle="tab" href="#messages" role="tab" aria-controls="messages" aria-selected="false">Messages</a>

</li>

<li class="nav-item">

  <a class="nav-link" id="settings-tab" data-toggle="tab" href="#settings" role="tab" aria-controls="settings" aria-selected="false">Settings</a>

</li>

</ul>
```

```
<div class="tab-content">

    <div class="tab-pane active" id="home" role="tabpanel" aria-labelledby="home-tab">...</div>

    <div class="tab-pane" id="profile" role="tabpanel" aria-labelledby="profile-tab">...</div>

    <div class="tab-pane" id="messages" role="tabpanel" aria-labelledby="messages-tab">...</div>

    <div class="tab-pane" id="settings" role="tabpanel" aria-labelledby="settings-tab">...</div>

</div>

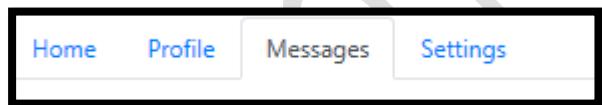
<script>

$(function () {

    $('#myTab li:last-child a').tab('show')

})

</script>
```



Navbar

- Navbars require a wrapping .navbar with .navbar-expand{-sm|-md|-lg|-xl} for responsive collapsing and [color scheme](#) classes.
- Navbars and their contents are fluid by default. Use [optional containers](#) to limit their horizontal width.
- Use our [spacing](#) and [flex](#) utility classes for controlling spacing and alignment within navbars.
- Navbars are responsive by default, but you can easily modify them to change that. Responsive behavior depends on our Collapse JavaScript plugin.
- Navbars are hidden by default when printing. Force them to be printed by adding .d-print to the .navbar. See the [display](#) utility class.

- Ensure accessibility by using a `<nav>` element or, if using a more generic element such as a `<div>`, add a `role="navigation"` to every navbar to explicitly identify it as a landmark region for users of assistive technologies.

Supported content

Navbars come with built-in support for a handful of sub-components. Choose from the following as needed:

- `.navbar-brand` for your company, product, or project name.
- `.navbar-nav` for a full-height and lightweight navigation (including support for dropdowns).
- `.navbar-toggler` for use with our collapse plugin and other [navigation toggling](#) behaviors.
- `.form-inline` for any form controls and actions.
- `.navbar-text` for adding vertically centered strings of text.
- `.collapse.navbar-collapse` for grouping and hiding navbar contents by a parent breakpoint.

Example

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">

  <a class="navbar-brand" href="#">Navbar</a>

  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

    <span class="navbar-toggler-icon"></span>

  </button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

  <ul class="navbar-nav mr-auto">

    <li class="nav-item active">

      <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>

    </li>

    <li class="nav-item">


```

```
<a class="nav-link" href="#">Link</a>

</li>

<li class="nav-item dropdown">
  <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    Dropdown
  </a>
  <div class="dropdown-menu" aria-labelledby="navbarDropdown">
    <a class="dropdown-item" href="#">Action</a>
    <a class="dropdown-item" href="#">Another action</a>
    <div class="dropdown-divider"></div>
    <a class="dropdown-item" href="#">Something else here</a>
  </div>
</li>

<li class="nav-item">
  <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
</li>
</ul>

<form class="form-inline my-2 my-lg-0">
  <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
  <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
</form>

</div>
```

```
</nav>
```



Nav

Navbar navigation links build on our .nav options with their own modifier class and require the use of [toggler classes](#) for proper responsive styling. **Navigation in navbars will also grow to occupy as much horizontal space as possible** to keep your navbar contents securely aligned.

Active states—with .active—to indicate the current page can be applied directly to .nav-links or their immediate parent .nav-items.

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">  
  <a class="navbar-brand" href="#">Navbar</a>  
  
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">  
    <span class="navbar-toggler-icon"></span>  
  </button>  
  
  <div class="collapse navbar-collapse" id="navbarNavAltMarkup">  
    <div class="navbar-nav">  
      <a class="nav-item nav-link active" href="#">Home <span class="sr-only">(current)</span></a>  
      <a class="nav-item nav-link" href="#">Features</a>  
      <a class="nav-item nav-link" href="#">Pricing</a>  
      <a class="nav-item nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>  
    </div>  
  </div>  
</nav>
```

Navbar Home Features Pricing Disabled

You may also utilize dropdowns in your navbar nav. Dropdown menus require a wrapping element for positioning, so be sure to use separate and nested elements for .nav-item and .nav-link as shown below:

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">  
  <a class="navbar-brand" href="#">Navbar</a>  
  
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavDropdown" aria-controls="navbarNavDropdown" aria-expanded="false" aria-label="Toggle navigation">  
    <span class="navbar-toggler-icon"></span>  
  </button>  
  
  <div class="collapse navbar-collapse" id="navbarNavDropdown">  
    <ul class="navbar-nav">  
      <li class="nav-item active">  
        <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>  
      </li>  
      <li class="nav-item">  
        <a class="nav-link" href="#">Features</a>  
      </li>  
      <li class="nav-item">  
        <a class="nav-link" href="#">Pricing</a>  
      </li>  
      <li class="nav-item dropdown">
```

```
<a class="nav-link dropdown-toggle" href="#" id="navbarDropdownMenuLink"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
  Dropdown link
</a>

<div class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink">
  <a class="dropdown-item" href="#">Action</a>
  <a class="dropdown-item" href="#">Another action</a>
  <a class="dropdown-item" href="#">Something else here</a>
</div>

</li>
</ul>
</div>

</nav>
```

Responsive behaviors

Navbars can utilize .navbar-toggler, .navbar-collapse, and .navbar-expand{ -sm|-md|-lg|-xl} classes to change when their content collapses behind a button. In combination with other utilities, you can easily choose when to show or hide particular elements.

Toggler

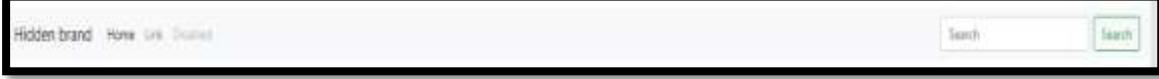
Navbar togglers are left-aligned by default, but should they follow a sibling element like a .navbar-brand, they'll automatically be aligned to the far right. Reversing your markup will reverse the placement of the toggler. Below are examples of different toggle styles.

With no .navbar-brand shown in lowest breakpoint:

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarTogglerDemo01" aria-controls="navbarTogglerDemo01" aria-expanded="false"
aria-label="Toggle navigation">
```

```
<span class="navbar-toggler-icon"></span>
</button>

<div class="collapse navbar-collapse" id="navbarTogglerDemo01">
  <a class="navbar-brand" href="#">Hidden brand</a>
  <ul class="navbar-nav mr-auto mt-2 mt-lg-0">
    <li class="nav-item active">
      <a class="nav-link" href="#">Home <span class="sr-only">(current)</span></a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">Link</a>
    </li>
    <li class="nav-item">
      <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
    </li>
  </ul>
  <form class="form-inline my-2 my-lg-0">
    <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
    <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
  </form>
</div>
</nav>
```



Pagination

Pagination is built with list HTML elements so screen readers can announce the number of available links. Use a wrapping `<nav>` element to identify it as a navigation section to screen readers and other assistive technologies.

```
<nav aria-label="Page navigation example">  
  
  <ul class="pagination">  
  
    <li class="page-item"><a class="page-link" href="#">Previous</a></li>  
  
    <li class="page-item"><a class="page-link" href="#">1</a></li>  
  
    <li class="page-item"><a class="page-link" href="#">2</a></li>  
  
    <li class="page-item"><a class="page-link" href="#">3</a></li>  
  
    <li class="page-item"><a class="page-link" href="#">Next</a></li>  
  
  </ul>  
  
</nav>
```



Working with icons

Looking to use an icon or symbol in place of text for some pagination links? Be sure to provide proper screen reader support with aria attributes.

```
<nav aria-label="Page navigation example">  
  
  <ul class="pagination">  
  
    <li class="page-item">  
      <a class="page-link" href="#" aria-label="Previous">  
        <span aria-hidden="true">&laquo;</span>  
      </a>  
    </li>
```

```
</li>

<li class="page-item"><a class="page-link" href="#">1</a></li>

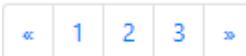
<li class="page-item"><a class="page-link" href="#">2</a></li>

<li class="page-item"><a class="page-link" href="#">3</a></li>

<li class="page-item">
    <a class="page-link" href="#" aria-label="Next">
        <span aria-hidden="true">&raquo;</span>
    </a>
</li>

</ul>

</nav>
```



Spinners

Indicate the loading state of a component or page with Bootstrap spinners, built entirely with HTML, CSS, and no JavaScript.

You will, however, need some custom JavaScript to toggle their visibility. Their appearance, alignment, and sizing can be easily customized with our amazing utility classes.

Border spinner

Use the border spinners for a lightweight loading indicator.

```
<div class="spinner-border" role="status">
    <span class="sr-only">Loading...</span>
</div>
```

C

Color

```
<div class="spinner-border text-primary" role="status">  
  <span class="sr-only">Loading...</span>  
</div>  
  
<div class="spinner-border text-secondary" role="status">  
  <span class="sr-only">Loading...</span>  
</div>  
  
<div class="spinner-border text-success" role="status">  
  <span class="sr-only">Loading...</span>  
</div>  
  
<div class="spinner-border text-danger" role="status">  
  <span class="sr-only">Loading...</span>  
</div>  
  
<div class="spinner-border text-warning" role="status">  
  <span class="sr-only">Loading...</span>  
</div>  
  
<div class="spinner-border text-info" role="status">  
  <span class="sr-only">Loading...</span>  
</div>  
  
<div class="spinner-border text-light" role="status">  
  <span class="sr-only">Loading...</span>
```

```
</div>

<div class="spinner-border text-dark" role="status">
  <span class="sr-only">Loading...</span>
</div>
```



Growing spinner

If you don't fancy a border spinner, switch to the grow spinner. While it doesn't technically spin, it does repeatedly grow!

```
<div class="spinner-grow" role="status">
  <span class="sr-only">Loading...</span>
</div>
```



Buttons

Use spinners within buttons to indicate an action is currently processing or taking place. You may also swap the text out of the spinner element and utilize button text as needed.

```
<button class="btn btn-primary" type="button" disabled>
  <span class="spinner-border spinner-border-sm" role="status" aria-hidden="true"></span>
  <span class="sr-only">Loading...</span>
</button>

<button class="btn btn-primary" type="button" disabled>
  <span class="spinner-border spinner-border-sm" role="status" aria-hidden="true"></span>
  Loading...
</button>
```

```
<button class="btn btn-primary" type="button" disabled>  
  <span class="spinner-grow spinner-grow-sm" role="status" aria-hidden="true"></span>  
  <span class="sr-only">Loading...</span>  
</button>  
  
<button class="btn btn-primary" type="button" disabled>  
  <span class="spinner-grow spinner-grow-sm" role="status" aria-hidden="true"></span>  
  Loading...  
</button>
```



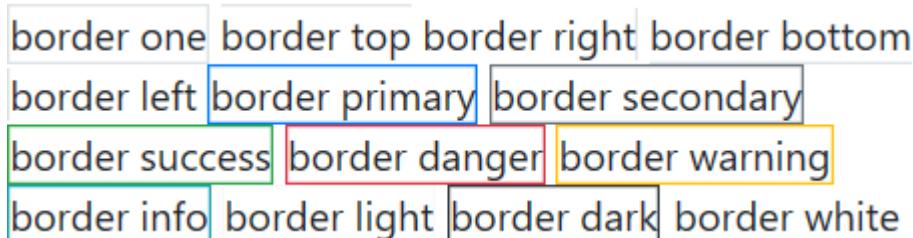
Borders/Border color

Use border utilities to quickly style the border and border-radius of an element. Great for images, buttons, or any other element.

Change the border color using utilities built on our theme colors.

```
<span class="border">border one</span>  
  
<span class="border-top">border top</span>  
  
<span class="border-right">border right</span>  
  
<span class="border-bottom">border bottom</span>  
  
<span class="border-left">border left</span>  
  
<span class="border border-primary"> border primary</span>  
  
<span class="border border-secondary">border secondary</span>  
  
<span class="border border-success">border success</span>
```

```
<span class="border border-danger">border danger</span>  
  
<span class="border border-warning">border warning</span>  
  
<span class="border border-info">border info</span>  
  
<span class="border border-light">border light</span>  
  
<span class="border border-dark">border dark</span>  
  
<span class="border border-white">border white</span>
```



Border-radius

Add classes to an element to easily round its corners.

```
  
  
  
  
  
  
  
  
  
  
  
  
  
  

```

Sizes

Use .rounded-lg or .rounded-sm for larger or smaller border-radius.

```

```

```

```

Background color

Similar to the contextual text color classes, easily set the background of an element to any contextual class. Anchor components will darken on hover, just like the text classes. Background utilities **do not set color**, so in some cases you'll want to use .text-* utilities.

```
.bg-primary
```

```
.bg-secondary
```

```
.bg-success
```

```
.bg-danger
```

```
.bg-warning
```

```
.bg-info
```

```
.bg-light
```

```
.bg-dark
```

```
.bg-white
```

```
.bg-transparent
```

```
<div class="p-3 mb-2 bg-primary text-white">.bg-primary</div>
```

```
<div class="p-3 mb-2 bg-secondary text-white">.bg-secondary</div>
```

```
<div class="p-3 mb-2 bg-success text-white">.bg-success</div>
```

```
<div class="p-3 mb-2 bg-danger text-white">.bg-danger</div>
```

```
<div class="p-3 mb-2 bg-warning text-dark">.bg-warning</div>
```

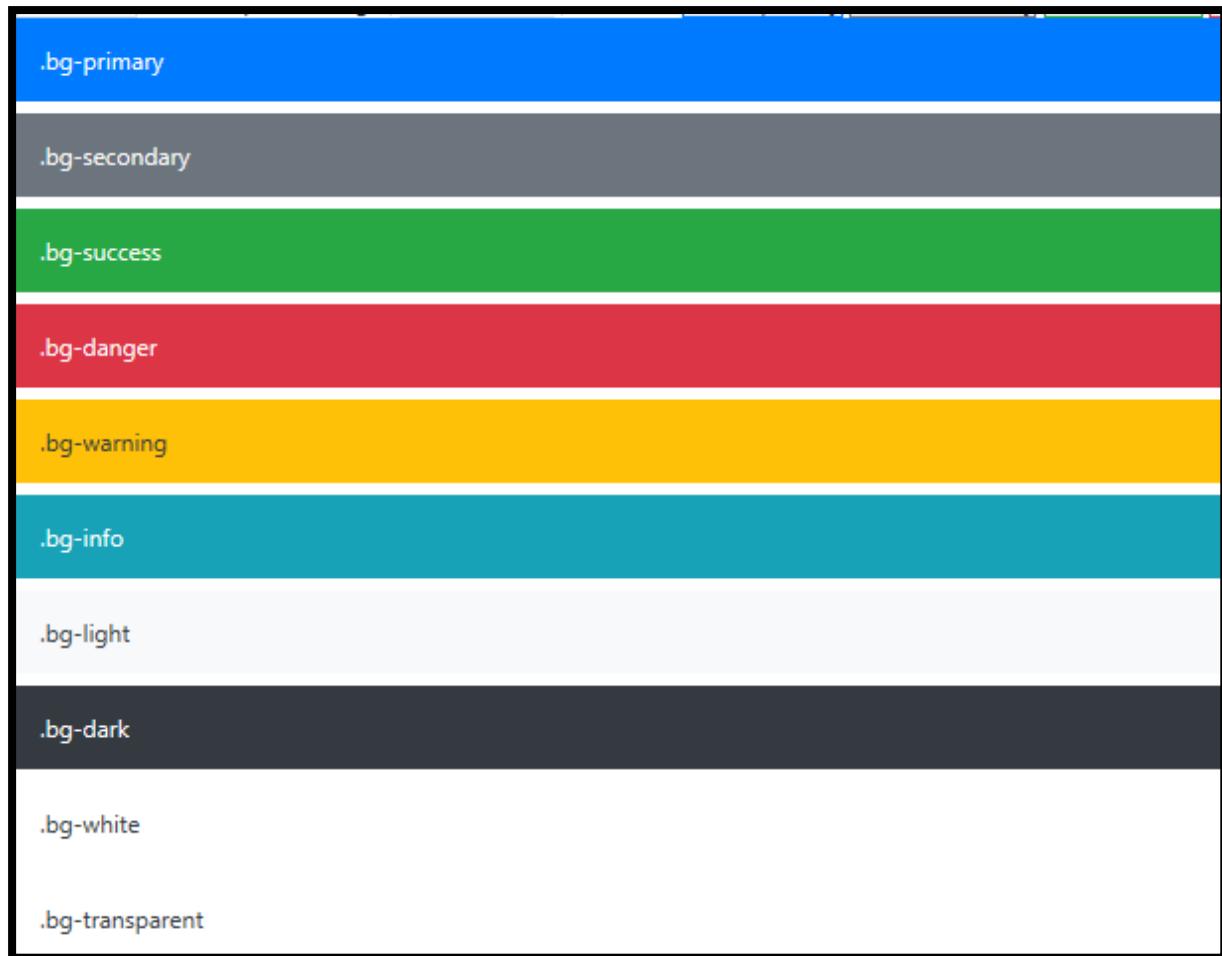
```
<div class="p-3 mb-2 bg-info text-white">.bg-info</div>
```

```
<div class="p-3 mb-2 bg-light text-dark">.bg-light</div>
```

```
<div class="p-3 mb-2 bg-dark text-white">.bg-dark</div>
```

```
<div class="p-3 mb-2 bg-white text-dark">.bg-white</div>
```

```
<div class="p-3 mb-2 bg-transparent text-dark">.bg-transparent</div>
```



Background gradient

When \$enable-gradients is set to true (default is false), you can use .bg-gradient- utility classes.

- .bg-gradient-primary
- .bg-gradient-secondary
- .bg-gradient-success
- .bg-gradient-danger
- .bg-gradient-warning
- .bg-gradient-info
- .bg-gradient-light
- .bg-gradient-dark

Display property

Quickly and responsively toggle the display value of components and more with our display utilities. Includes support for some of the more common values, as well as some extras for controlling display when printing.

As such, the classes are named using the format:

- .d-{value} for xs
- .d-{breakpoint}-{value} for sm, md, lg, and xl.

Where *value* is one of:

- none
- inline
- inline-block
- block
- table
- table-cell
- table-row
- flex
- inline-flex

Text transform

Transform text in components with text capitalization classes.

```
<p class="text-lowercase">Lowercased text.</p>
```

```
<p class="text-uppercase">Uppercased text.</p>
```

```
<p class="text-capitalize">CapiTaliZed text.</p>
```

lowercased text.

UPPERCASED TEXT.

CapiTaliZed Text.

Font weight and italics

Quickly change the weight (boldness) of text or italicize text.

Bold text.

Bolder weight text (relative to the parent element).

Normal weight text.

Light weight text.

Lighter weight text (relative to the parent element).

Italic text.

```
<p class="font-weight-bold">Bold text.</p>
```

```
<p class="font-weight-bolder">Bolder weight text (relative to the parent element).</p>
```

```
<p class="font-weight-normal">Normal weight text.</p>
```

```
<p class="font-weight-light">Light weight text.</p>
```

```
<p class="font-weight-lighter">Lighter weight text (relative to the parent element).</p>
```

```
<p class="font-italic">Italic text.</p>
```

Bold text.

Bolder weight text (relative to the parent element).

Normal weight text.

Light weight text.

Lighter weight text (relative to the parent element).

Italic text.

Text decoration

Remove a text decoration with a .text-decoration-none class.

Non-underlined link

```
<a href="#" class="text-decoration-none">Non-underlined link</a>
```

18.2 Working with Git and Github

Git is version control software which manages changes to a project without overwriting any part of that project; it is software that runs at the heart of GitHub.

Before we start our discussion let us look at commonly used terms/commands in Git/GitHub operations

18.2.1 Git and GitHub Specific Commands/Terms

Command Line: The computer program we use to input Git commands. On a Mac, it is called Terminal. On a PC, it is a non-native program that comes with Git when you download it for the first time. In both cases, you type text-based commands, known as prompts, into the screen, instead of using a mouse.

Repository: A directory or storage space where your projects can live. Sometimes GitHub users shorten this to “repo.” It can be local to a folder on your computer, or it can be a storage space on GitHub or another online host. You can keep code files, text files, image files inside it.

Version Control: Basically, Git was designed to serve this purpose. When you have a Microsoft Word file, you either overwrite every saved file with a new save, or you save multiple versions.

With Git, you do not have to. It keeps “snapshots” of every point in time in the project’s history, so you can never lose or overwrite it.

Commit: This is a record of what files you have changed since the last time you made a commit. Essentially, you make changes to your repo (for example, adding a file or modifying one) and then tell git to put those files into a commit.

Branch: This is creating of a “branch off” of the main project with its own versions full of changes made by people working on the project. After they are done, it’s time to “merge” that branch back with the “master” which is the main directory of the project.

git init: Initializes a new Git repository. Until you run this command inside a repository or directory, it is just a regular folder. Only after you input this does it accept further Git commands.

git config: Short for “configure,” this is most useful when you are setting up Git for the first time.

git help: Forgot a command? Type this into the command line to bring up the 21 most common git commands.

git status: Check the status of your repository. See which files are inside it, which changes still need to be committed, and which branch of the repository you are currently working on.

git add: This does *not* add new files to your repository. Instead, it brings new files to Git’s attention. After you add files, they are included in Git’s “snapshots” of the repository.

git commit: Git’s most important command. After you make any sort of change, you input this in order to take a “snapshot” of the repository. Usually it goes `git commit -m "Message here."` The `-m` indicates that the following section of the command should be read as a message and should not be more than 72 characters.

git branch: Branching is one of the fundamental aspects of working with Git. Say you want to make a new feature but are worried about making changes to the main project while developing the feature. This is where **git branches** come in. Branches are essential for being able to safely experiment with concepts and ideas. Git makes it painless to create your own branch, experiment with or implement features, and then merge those changes back into the development branch, when you have finished. Branches also allow you to move back and forth between ‘states’ of a project. For instance, if you want to add a new page to your website you can create a new branch just for that page without affecting the main part of the project. Once you are done with the page, you can **merge** your changes from your branch into the master branch and delete the branch.

git checkout: Literally allows you to “check out” a repository that you are not currently inside. This is a navigational command that lets you move to the repository you want to check. You can use this command as **git checkout master** to look at the master branch, or **git checkout cats** to look at another branch.

git merge: When you are done working on a branch, you can merge your changes back to the master branch, which is visible to all collaborators. `git merge cats` would take all the changes you made to the “cats” branch and add them to the master.

git push: If you are working on your local computer, and want your commits to be visible online GitHub as well, you “push” the changes up to GitHub with this command.

git pull: If you are working on your local computer and want the most up-to-date version of your repository to work with, you “pull” the changes down from GitHub with this command.

.gitignore: this command is explicitly used to tell Git to ignore files. There are files you do not want Git to check in to GitHub.

Fork a repo : Forking a repository allows you to freely experiment with changes without affecting the original project created by another person or to use someone else's project as a starting point for your own idea.

18.2.1 Initializing a Repository

Before you can work with Git, you have to initialize a project repository, setting it up so that Git will manage it. Open up your terminal, and in your project directory run the command `git init`. A new hidden directory called `.git` will now be present in your project directory after git is initialized.

Cloning a Repository

Cloning is Similar to checking out a repository in other systems, run `git clone <repository URL>` to pull in a complete copy of the remote repository to your local system. This enables you to locally make changes, stage them, commit them, and push the changes back to the remote repository.

18.2.3 Steps on how to set up git in windows and Github account

Install git and create a GitHub account

The first two things to do are to install git and create a free GitHub account before you start enjoying the benefits of git to manage your projects.

➔ Installing Git

You can download the setup file and install it from the link below;

<http://git-scm.com/download/win> and the download will start automatically.

Another easy way to get Git installed is by installing GitHub for Windows. The installer includes a command line version of Git as well as the GUI. You can download this from the GitHub for Windows website, at <http://windows.github.com>.



Fig. 18.3

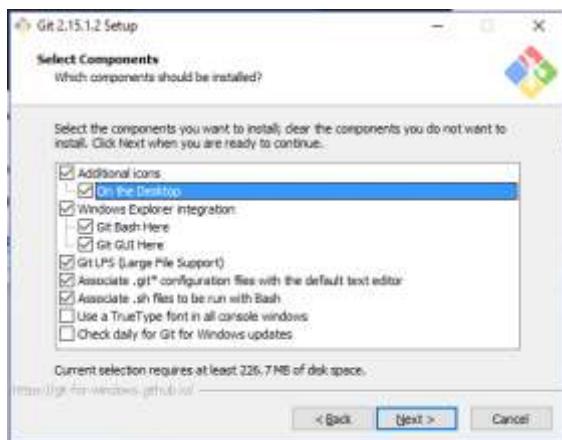
But in this tutorial, we will learn how to install git and work with it on the command prompt.

The following are installation steps;

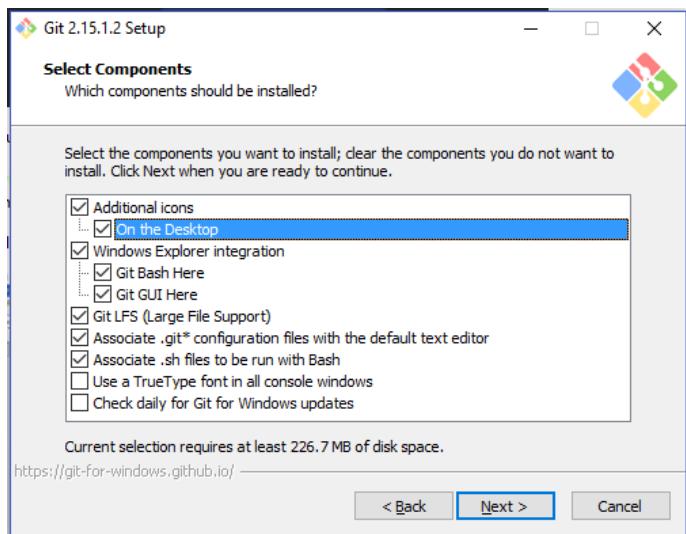
- ⇒ Click the setup file



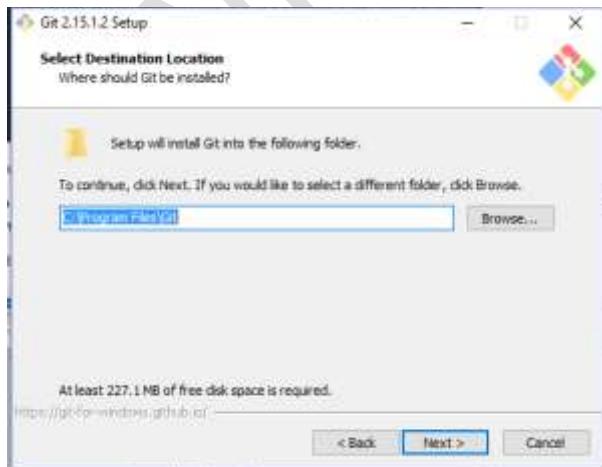
- ⇒ Select destination and click on Next



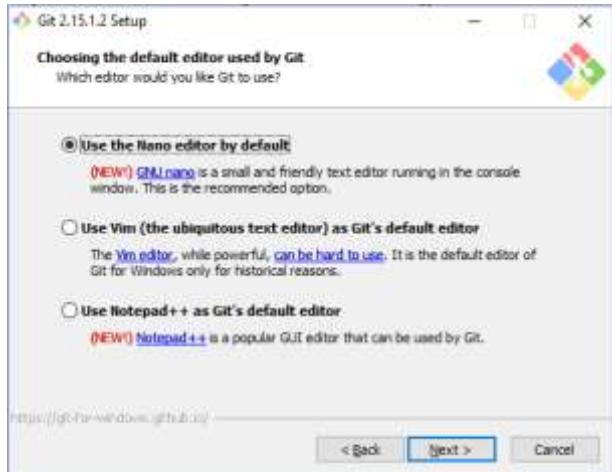
⇒ Select components and click on Next



⇒ Select Location to install git and click on Next. (At default, Git is installed in the Program Files Location)

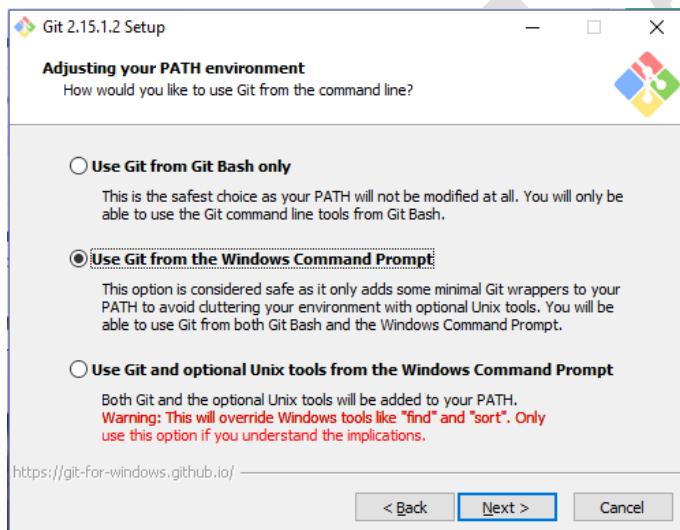


⇒ Select text Editor to use with git(you can leave it at the default text editor)

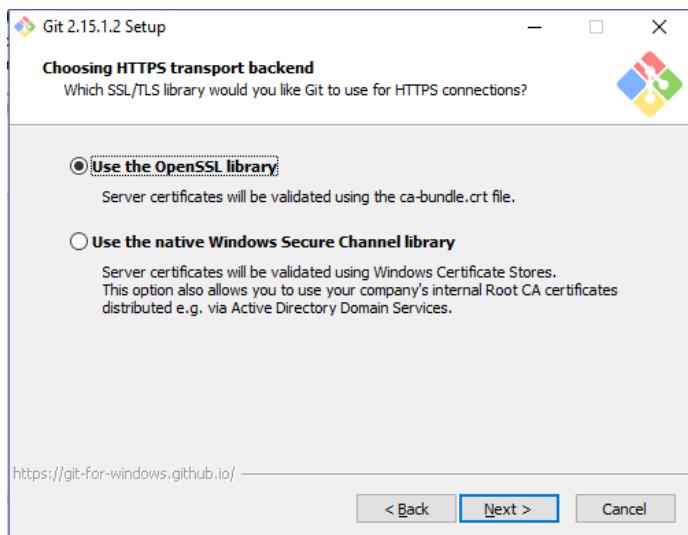


=> Adjusting Path Environment

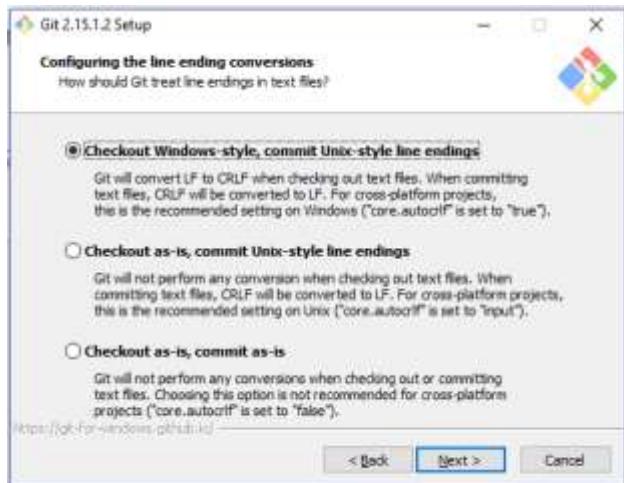
This is asking your choice that whether you like to Git from the **Windows Command Prompt** or you like to use some other program like **Git Bash**. As of now just select the **Windows Cmd** for simplicity .



=> Select **Use the OpenSSL Binary** option and click **Next**.

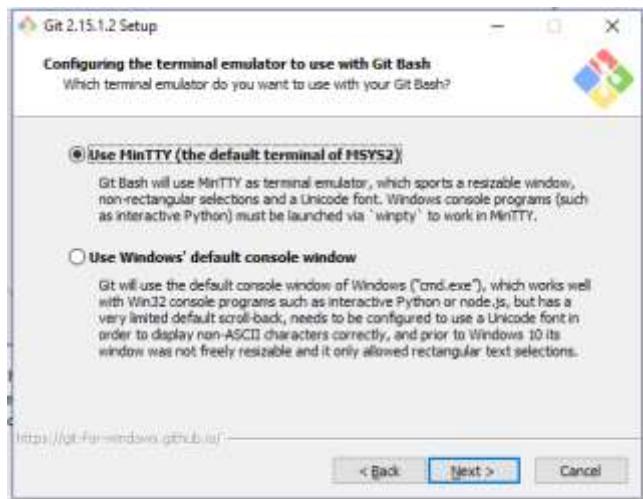


- ⇒ On the next step of the installation process it is recommended to choose the **Checkout Windows-style, commit Unix-style line endings** for cross platform project.



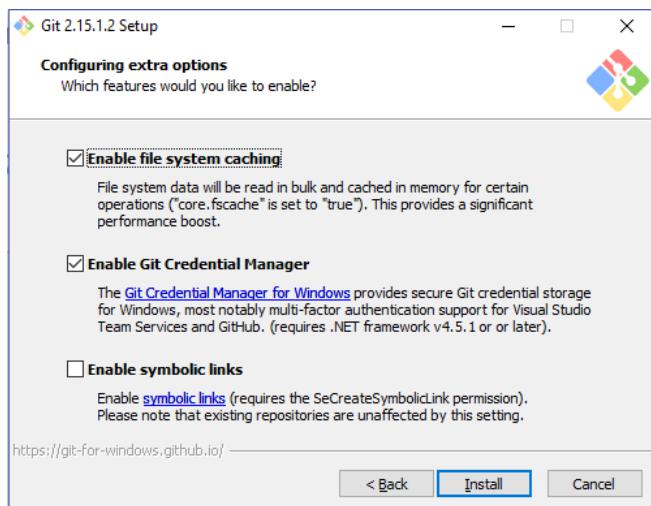
- ⇒ Configuring the terminal emulator to use with Git Bash

Choose the **Use MinTTY** option in order to use GIT with the built-in MinTTY terminal. Choosing the other option will allow you to use Git commands with the Windows command prompt tool, but this is not recommended as it is not designed to be used with GIT.

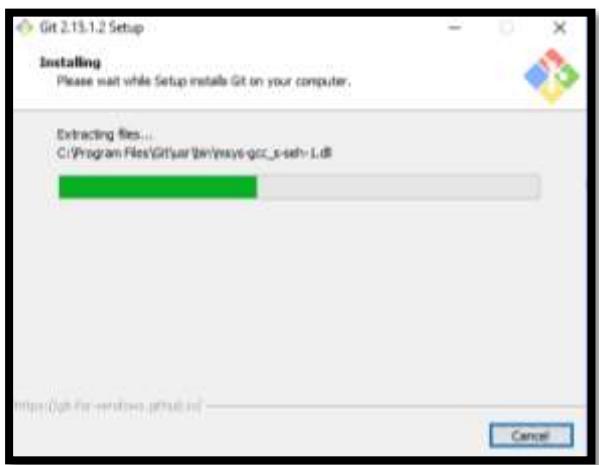


⇒ Configuring extra options

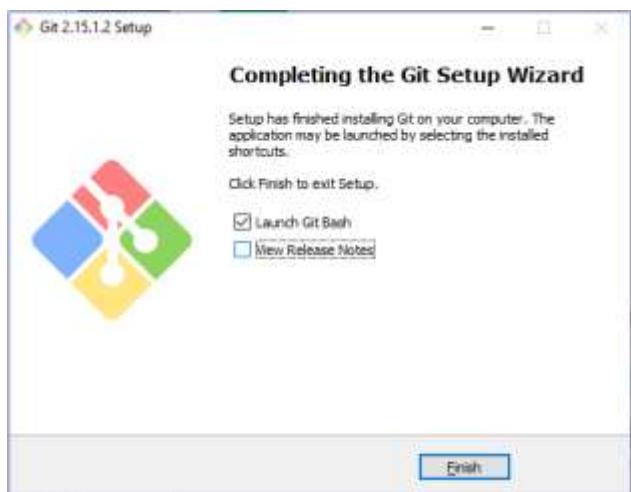
On the last step leave the default options enabled and click the **Install** button to start the installation process of the software.



Installation Windows



Once the installation is completed you will see a confirmation message.



⇒ Click on finish

18.3 Github Account

GitHub is now the largest online storage space of collaborative works that exists in the world. You build a profile, upload projects to share and connect with other users by “following” their accounts. While many users store programs and code projects, there is nothing preventing you from keeping text documents or other file types in your project folders to show off.

What is GitHub?

It is a web based git repository hosting service, easy management of code, open-source software for version control, effective collaboration and bug tracker.

Difference between git and github

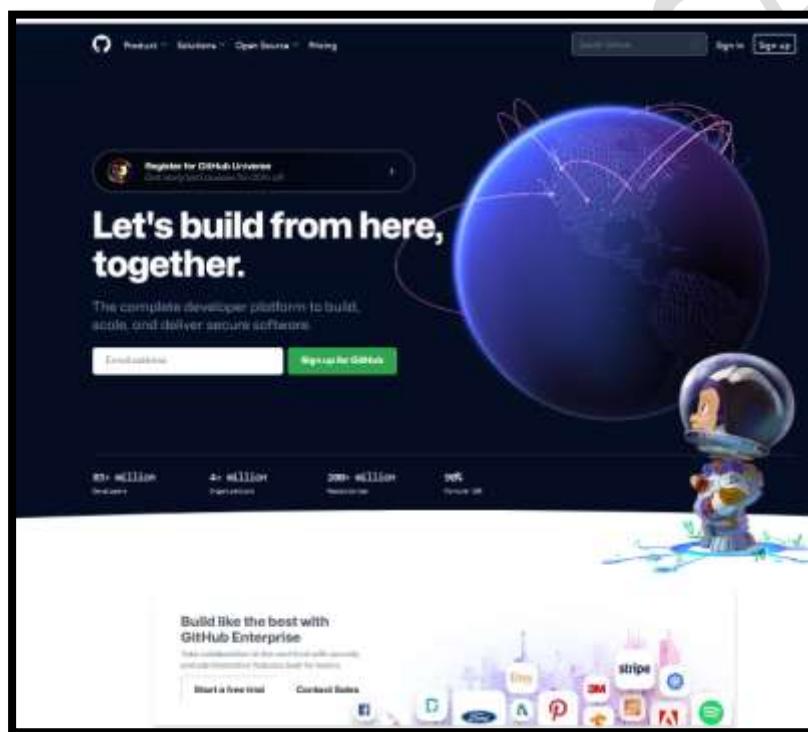
- Git is a revision control system, a tool to manage your code history while github is a hosting service for git repositories.
- Installed and maintained in your local system while github is exclusively cloud-based
- Git is the tool while github is the service for projects that use git.

Steps on creating Github account

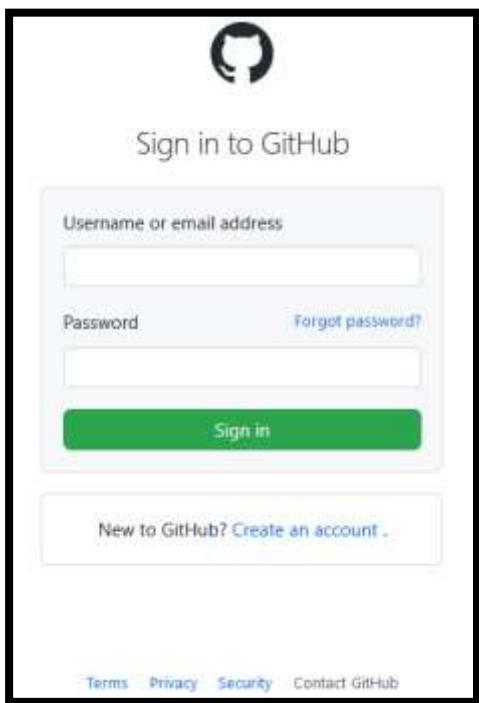
Create a GitHub account via this link <https://github.com/> (Accounts are free for public repositories, but there is a charge for private repositories, though now you have a private Github account with maximum of three users)

Going to the link will display the page below;

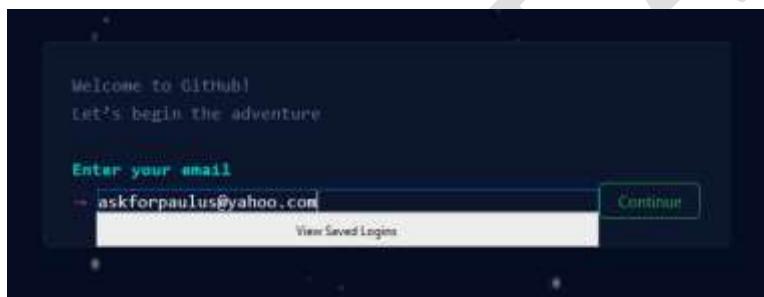
- ⇒ Step 1 is to create personal account with username, email address and password



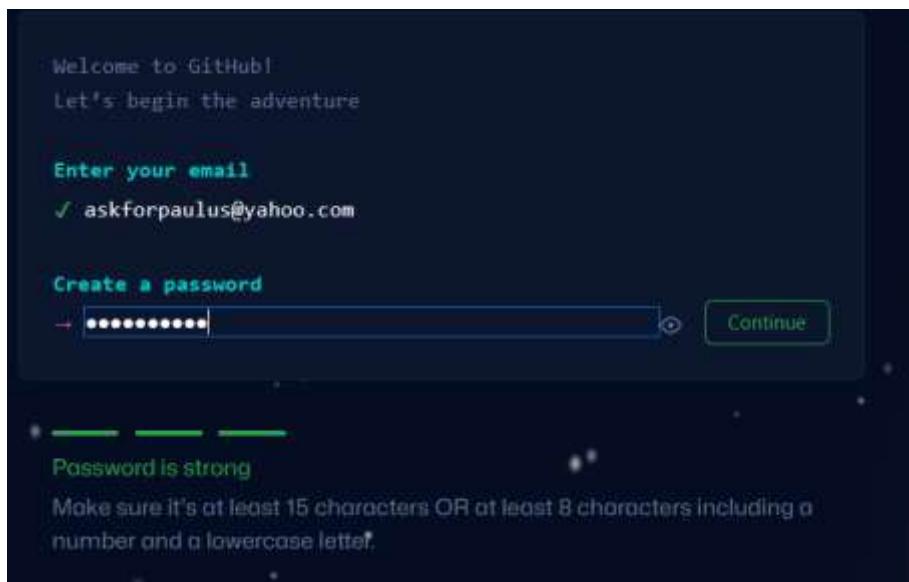
If you already have an account with github, click on Sign in to display a page as shown below;



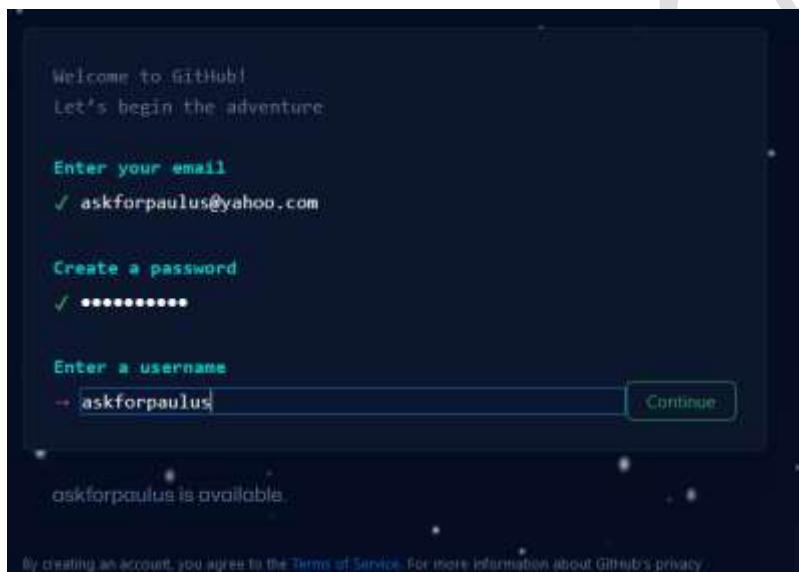
If you are new to the account, either you click on **Create an account** from the above image or from previous page click on sign up which will display a page as shown below;



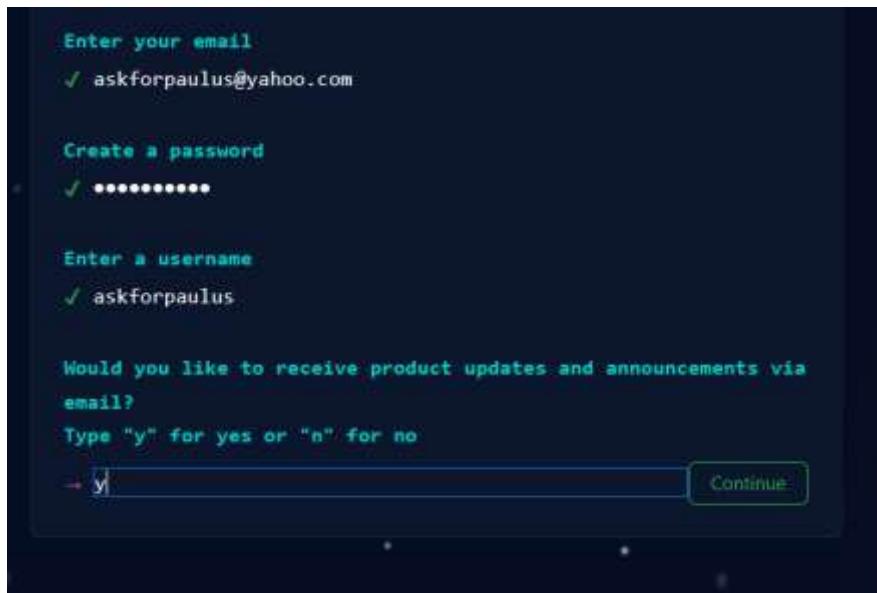
Enter your valid email address and then click on **Continue** button as shown in the diagram above, this will take you to the diagram as shown below to enter the password to be used and click on Continue button;



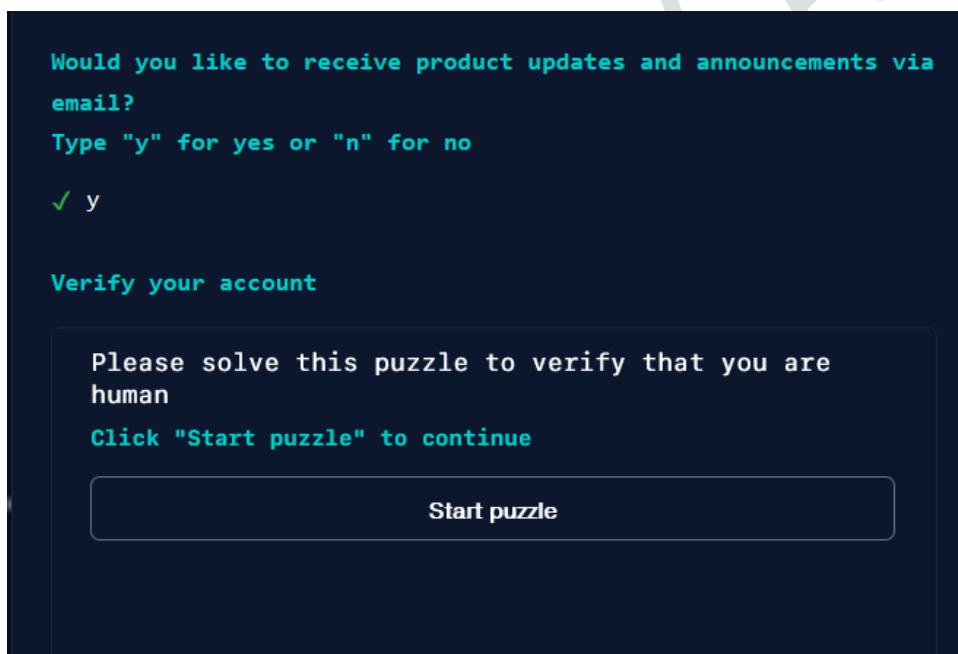
As soon as this is done, a field will be displayed enter your username which is not used by anyone in github as shown in the diagram below;



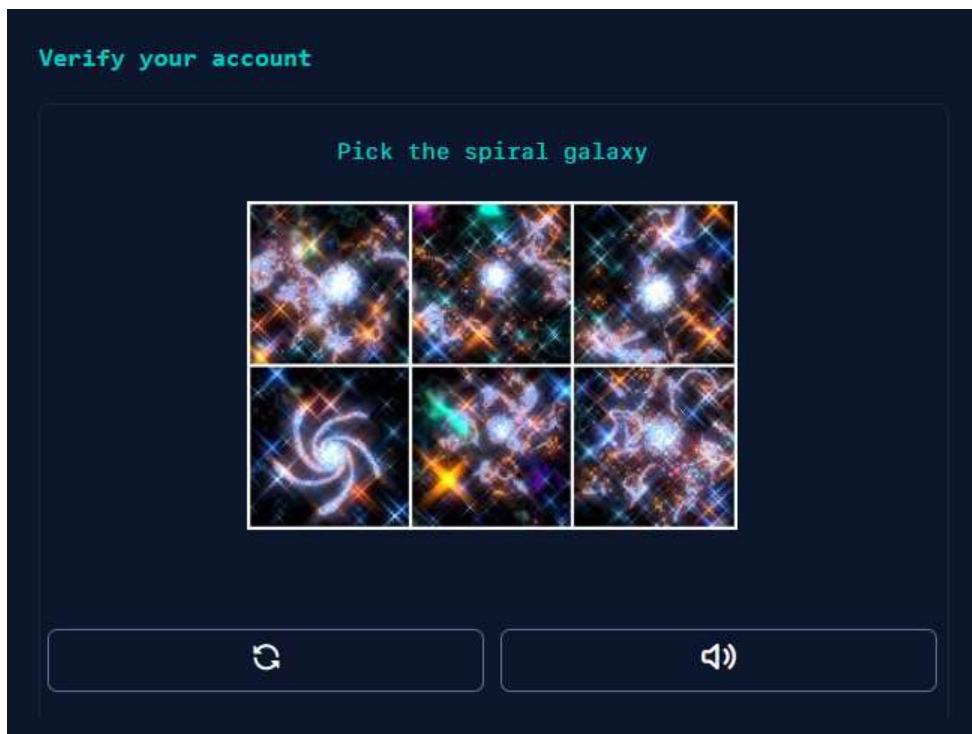
As soon as you enter your username, click on Continue button to display a review page as shown below;



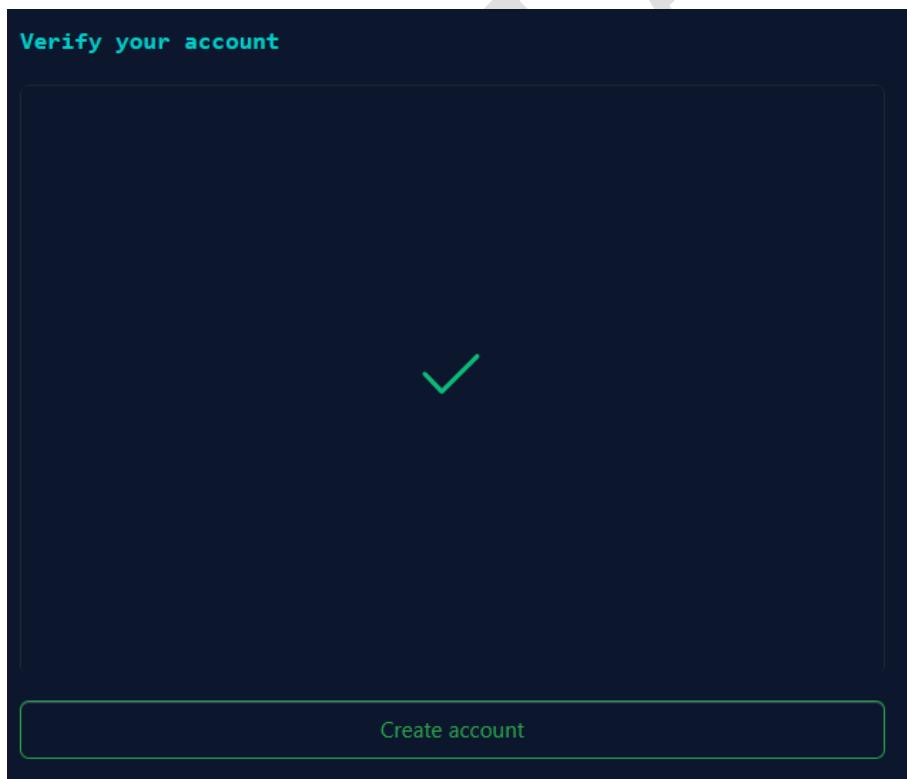
In the above diagram, enter y for yes or n for no as shown above, then click on **Continue** button to display a page for puzzle as shown below;



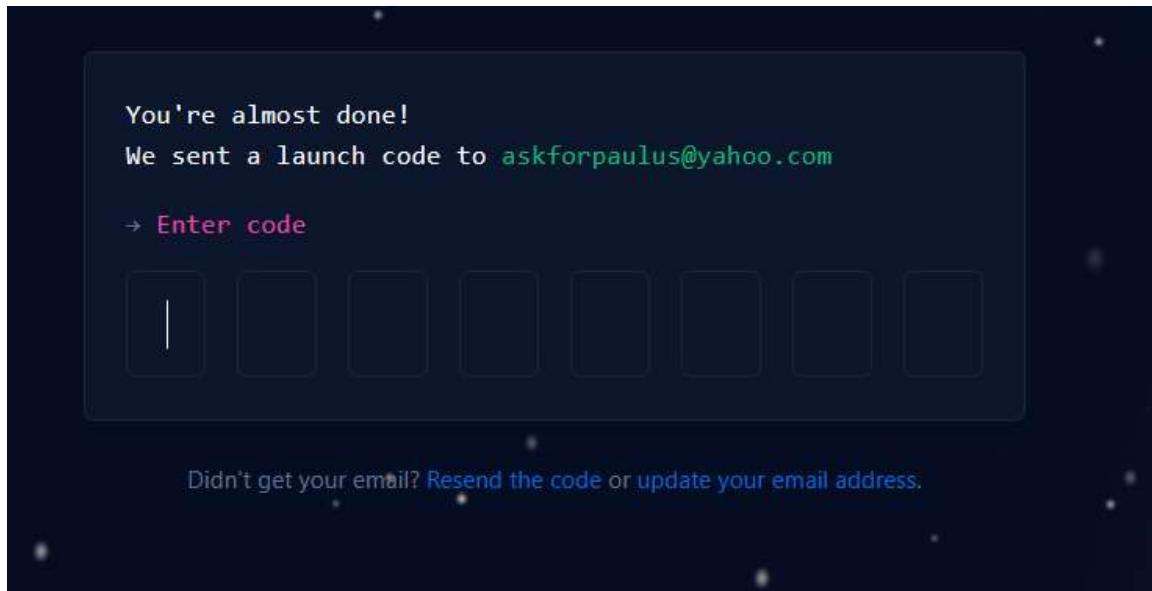
Click on start puzzle button to display a puzzle page as shown below;



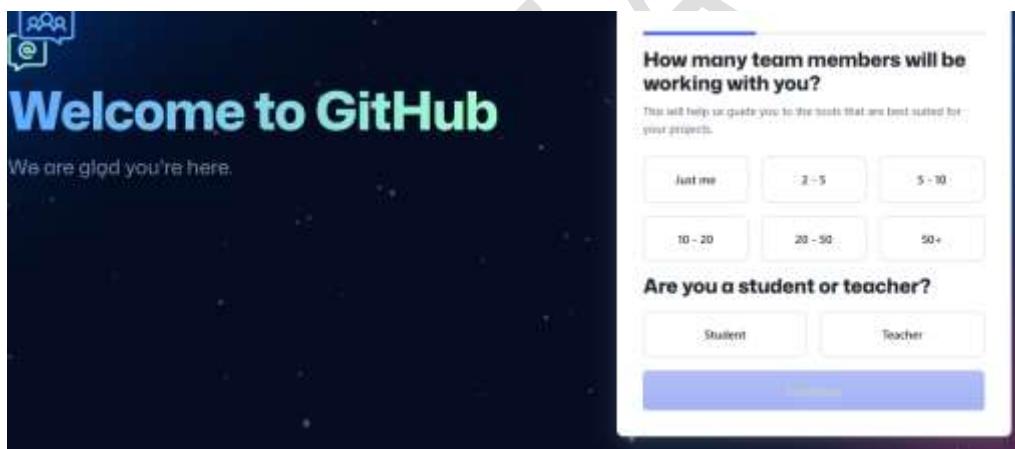
This puzzle is to authenticate that you are human. This is to be done twice then click on create account after you pass the puzzle test as shown in the diagram below;



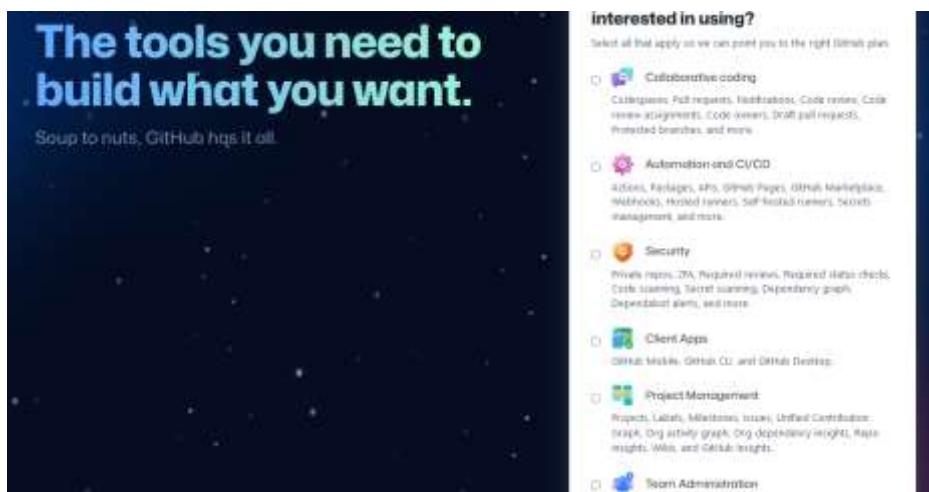
As soon as you click on **Create account** button, a code will be sent to the register email address and a page is displayed as shown below to enter the code for authentication



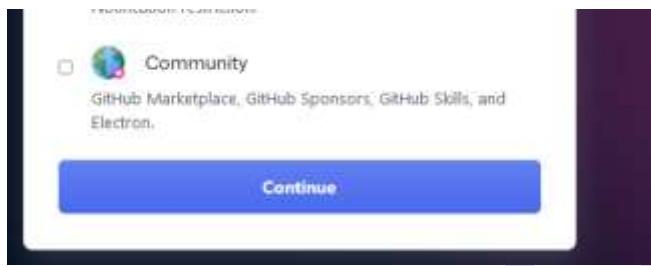
When you correctly enter the verification, a page will be displayed to select the team members and status as shown in the diagram below;



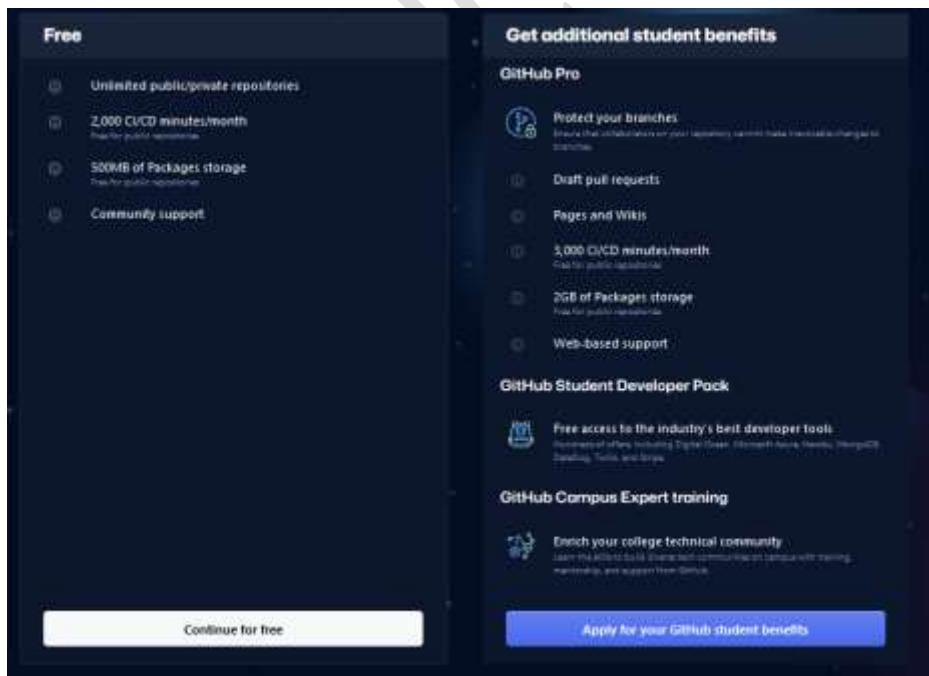
Select the necessary options and click on Continue button, this will take you to a page with interesting tools to select from that will enhance your project as shown below;



When you finish selecting the tools, click on continue button as it is shown below;

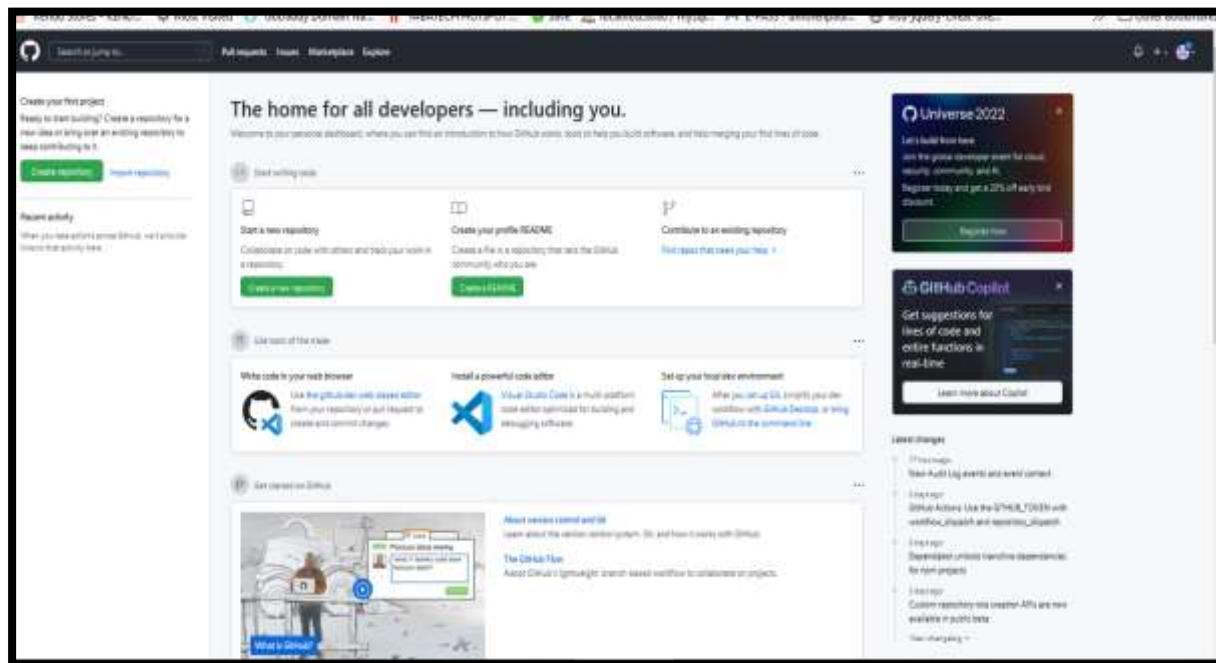


It will now take you to a page to select the type of account to use such as free or pro, so at this point you can make the choice of your account to be used;

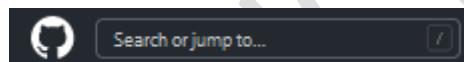


Web Development Fundamentals, Umoren Paul Udo

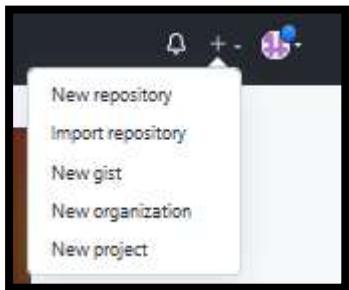
As soon as this is done, your account is created and you are login to the dashboard of github as shown below;



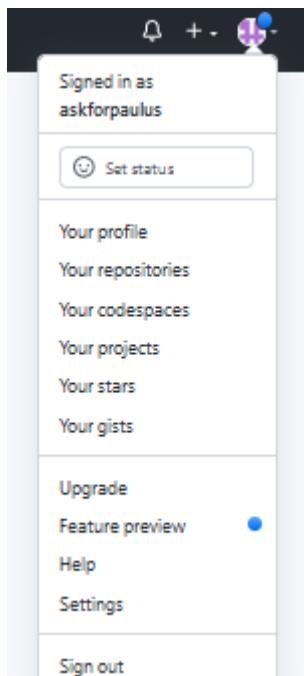
Getting started with github environment



Plus icon: when you click on the Plus icon, it displays a dropdown below that contains the following items: New repository, Import repository, New gist, New organization and New project.



Profile icon: this is where you edit your account information as well as Sign out from your account.



18.4 Local Git Repository

Next is to start creating a new project using git. Please note that all your activities will be done the terminal that is a command prompt for those using windows. Like common navigation to file/folder location commands on the command prompt environment

Step 1: Create a local git repository

Creating a new project on your local machine using git, first create a new repository often call 'repo', for short.

- ⇒ So open a command prompt in windows Operating System
- ⇒ Navigate to Desktop and create a project folder the following commands will navigate you to Desktop Environment C:\Users\GTCO CALSCAN>**cd Desktop** and the following will create a project folder call mfonurua C:\Users\GTCO CALSCAN\Desktop> mkdir mfonurua

Note: you may use any location of your choice, mine I use Desktop, you may decide to use Document by type cd Document when you start up your command prompt environment in windows OS

- ⇒ Then navigate into the project folder using the following commands C:\Users\GTCO CALSCAN\Desktop>**cd mfonurua**- then you will see a command in this format on your command prompt environment C:\Users\GTCO CALSCAN\Desktop\mfonurua>
- Note: mofnurua is my project folder; yours can be any name of your choice.

Step 2 Initialize Git Repository

To initialize a git repository in the root of the folder, run the **git init** command as shown below;

```
C:\Users\GTCO CALSCAN\Desktop\mfonurua>git init
```

When the above command is run, your prompt command environment will display the following;

```
C:\Users\GTCO CALSCAN\Desktop\mfonurua>git init
Initialized empty Git repository in C:/Users/GTCO CALSCAN/Desktop/mfonurua/.git/
C:\Users\GTCO CALSCAN\Desktop\mfonurua>
```

A new hidden directory called .git will now be present in your project directory. This is where Git stores its database and configuration information, so that it can track your project.

Step 3 Add a new file to the project folder created (repository)

This can be done using a text editor you are familiar with and saving the file to your project repository. Mine is mfonurua, I will save my file to mfonurua project repository.

Step 4 git status

After creating the new file, you can use the git status command to see which files git knows exist.

```
C:\Users\GTCO CALSCAN\Desktop\mfonurua>git status
```

When the above command is executed on the command prompt, the following is displayed on your command prompt environment

```
C:\Users\GTCO CALSCAN\Desktop\mfonurua>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    contents.txt

nothing added to commit but untracked files present (use "git add" to track)
C:\Users\GTCO CALSCAN\Desktop\mfonurua>
```

What the above screenshot is saying is that you created a new file called mnelson.txt, but unless you use the 'git add' nothing will be done with the file by git.

To add a file to a commit, you first need to add it to the staging environment. To do this, you can use the **git add <filename>** command

Once you have used the git add command to add all the files to the staging environment, you can then tell git to package them into a commit using the **git commit** command.

Note: The staging environment is also called 'staging', is the new preferred term for this, but you can also see it referred to as the 'index'.

Step 5 adding our changes to git by using git add command as shown below;

```
C:\Users\GTCO CALSCAN\Desktop\mfonurua>git add contents.txt
```

After this, run git status again to see the changes added to the git as shown below;

```
C:\Users\GTCO CALSCAN\Desktop\mfonurua>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   contents.txt

C:\Users\GTCO CALSCAN\Desktop\mfonurua>
```

Step 6 Create a commit

It's time to create your first commit! Run the command

```
git commit -m "Your message about the commit"
```

If it is your first commit and you have not configured your git environment to set user email and username, you encounter the following error warning:

```
C:\Users\GTCO CALSCAN\Desktop\mfonurua>git commit -m "this is my first commit to git"
*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'GTCO CALSCAN@LAPTOP-49BJ0V4M.(none)')
C:\Users\GTCO CALSCAN\Desktop\mfonurua>
```

Then run the following commands;

1. C:\Users\GTCO CALSCAN\Desktop\mfonurua> **git config --global user.email "umorenpaul@gmail.com"**

```
C:\Users\GTCO CALSCAN\Desktop\mfonurua> git config --global user.email "umorenpaul@gmail.com"
```

2. C:\Users\GTCO CALSCAN\Desktop\mfonurua> **git config --global user.name "umoren"**

```
C:\Users\GTCO CALSCAN\Desktop\mfonurua> git config --global user.name "umoren"
```

The above commands will then set up your user email and username in your git you can now commit your changes with git commit command as shown below;**Git commit -m “this is my first commit”**

If your file is successfully committed, your system will display the screen below;

```
C:\Users\GTCO CALSCAN\Desktop\mfonurua>git commit
[master (root-commit) 939eed3] this is my first commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 contents.txt
```

Hints: To add multiple files to git repository, just use a wildcard to get all files set:

```
git add *
```

And then run the git commit command like this:

```
git commit -a -m "all files"
```

18.4.1 Viewing Change History with git log command

Just running git log in your project repository will show you a list of changes in reverse chronological order. With no further arguments, you will see the commit hash, the author name and email, a timestamp for the commit, and the commit message as shown below;

```
C:\Users\GTCO CALSCAN\Desktop\mfonurua>git log
commit 939eed3193f34fe751a3593462a6de3f7f7b9b65 (HEAD -> testing, origin/testing, origin/master, master)
Author: umoren <umorenpaul@gmail.com>
Date:   Sat Jan 13 20:41:58 2018 +0000

    this is my first commit

C:\Users\GTCO CALSCAN\Desktop\mfonurua> .git
```

Now this is fine, but what if you want to customize what you see? What if you just want to view the commit hash and the commit message? To see only the commit hash and the comment message, you pass the --oneline switch to git log, like this:

```
git log --oneline.
```

This will output history information, as shown in the screenshot below.,

```
C:\Users\GTCO CALSCAN\Desktop\mfonurua>git log --oneline
939eed3 (HEAD -> testing, origin/testing, origin/master, master) this is my first commit
```

18.4.2 Create a new branch and merging

To create new branches, use the code below;

```
git checkout -b workflow
```

If the command is executed successfully it will display the diagram below;

```
C:\Users\GTCO CALSCAN\Desktop\mfonurua>git checkout -b testing
Switched to a new branch 'testing'

C:\Users\GTCO CALSCAN\Desktop\mfonurua>
```

After running the above command, you can use the **git branch** command to confirm that your branch was created: when a new branch is created, git automatically switch you to it.

To switch back to the master branch, check it out as follows:

```
git checkout master
```

If you make changes to the master branch, you will want to merge them into your exploratory one as follows:

git checkout workflow

git merge master

If you are satisfied with your changes in the workflow branch, merge them into the master:

git checkout master

git merge workflow

If you are done with the branch and want to delete it:

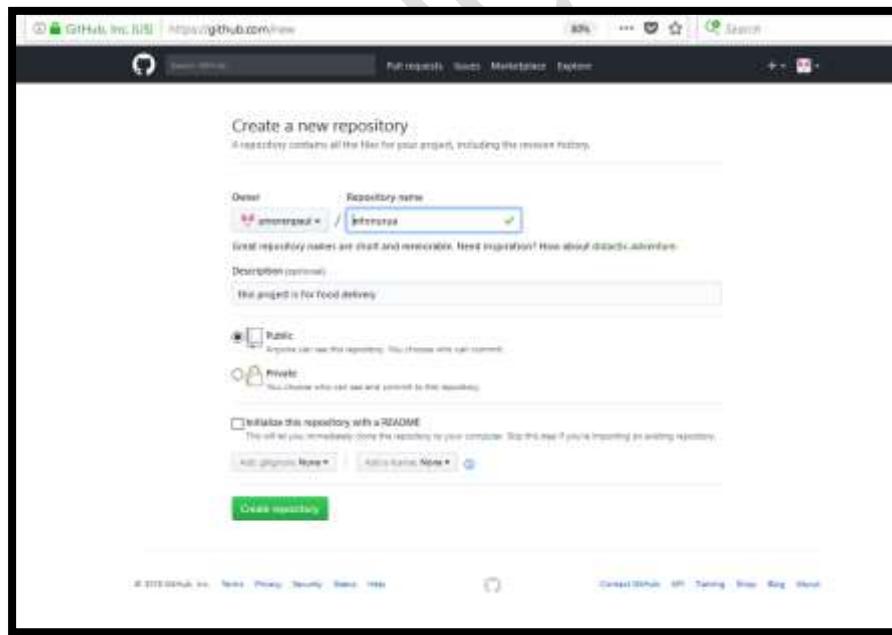
git branch -d workflow

18.4.3 Create a new repository on GitHub

If you only want to keep track of your code locally, you don't need to use GitHub. But if you want to work with a team, you can use GitHub to collaboratively modify the project's code.

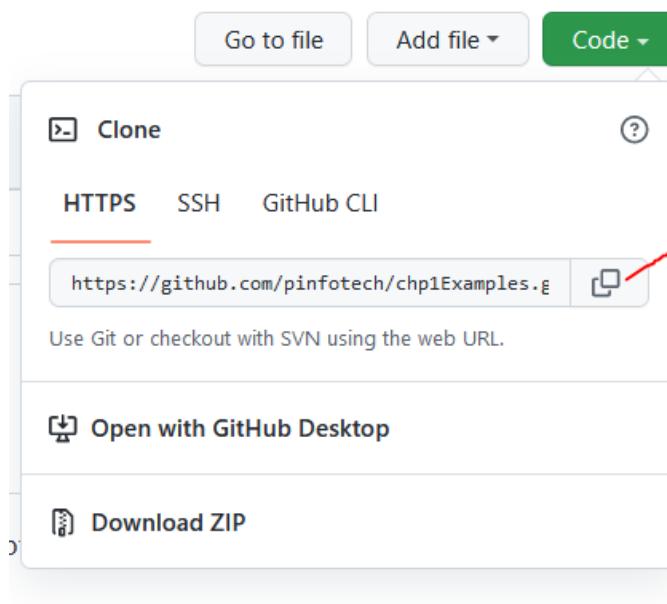
Steps on how to create a new repo on GitHub

Step 1: log in and go to the GitHub home page. You should see a green '+' New repository' button, click on it and the flowing screen will show;



When you are done filling out the information, press the 'Create repository' button to make your new repo.

Then go to the newly created repository, click on code button to display option as shown below, the click on the HTTPS option, then click on copy icon to copy the link to git environment.



18.4.4 Push a branch to GitHub

Now we will **push** the commit in your branch to your new GitHub repo. This makes your project to be available on your GitHub remote repository and allows other people to see the changes you have made. If you are pushing to someone else's repository,

they are to be approved by the repository's owner, the changes can then be merged into the master branch.

To push changes onto a new branch on GitHub, you run the command below;

```
(base) umoren@umoren-HP-Pavilion-x360-m3-Convertible:~/Desktop/advanceCSS$ git remote add origin https://github.com/pinfotech/webupdate.git
```

- ⇒ git remote add origin <https://github.com/pinfotech/chp1Examples.git>
- ⇒ then run this command git push origin yourbranchname

during this, a username and a password will be required from you for the authentication of your account.

Note: Support for password authentication was removed on August 13, 2021. What is used now is Personal Access Token (PAT). We now look at how to create PAT.

Creating a personal access token

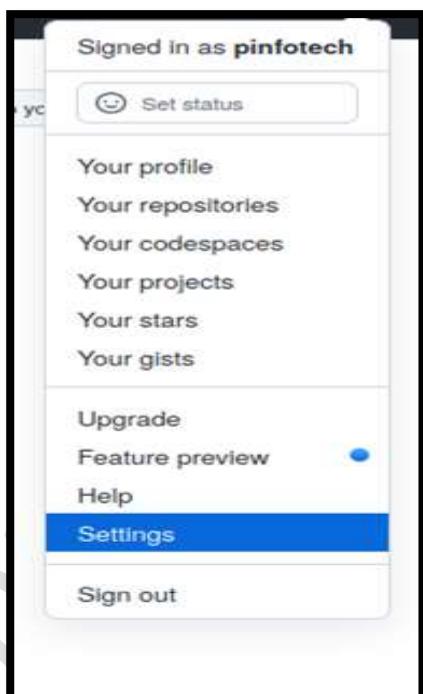
You can create a personal access token to use in place of a password with the command line or with the API.

Personal access tokens (PATs) are an alternative to using passwords for authentication to GitHub when using the GitHub API or the command line.

As a security precaution, GitHub automatically removes personal access tokens that haven't been used in a year. To provide additional security, we highly recommend adding an expiration to your personal access tokens.

The following are steps to create Personal Access Token

1. Verify your email address, if it hasn't been verified yet
2. Click on your profile from the dropdown list, click on Settings as shown in the diagram below;



3. In the left sidebar, click **Developer settings**.
4. In the left sidebar, click **Personal access tokens** as shown in the diagram below;
5. Click **Generate new token** as shown in the diagram below;

Personal access tokens

Generate new token

- Then enter the necessary information as shown in the diagram below;

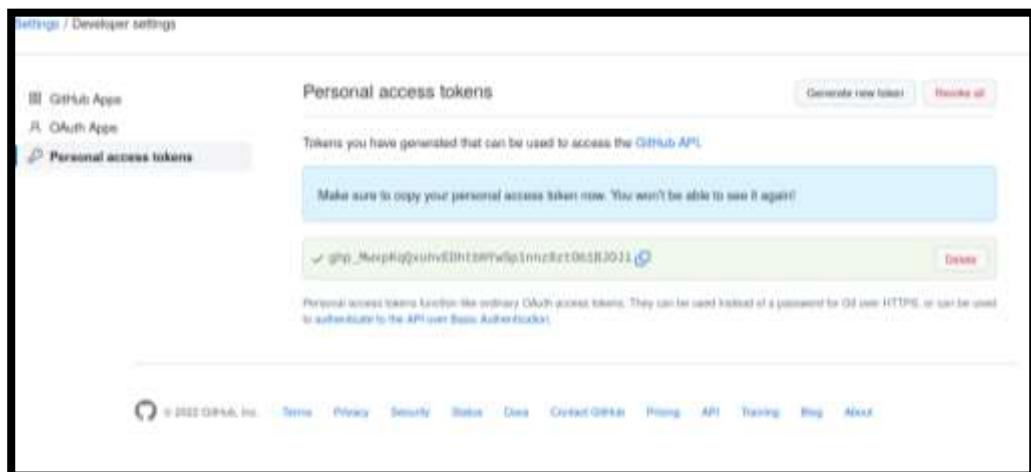
The screenshot shows the 'New personal access token' creation page. It includes fields for 'Name' (set to 'update of projects'), 'Expiration' (set to '30 days'), and a 'Select scopes' section. The 'Select scopes' section lists various GitHub API permissions with checkboxes:

| Scope | Description |
|--|--------------------------------------|
| <input type="checkbox"/> repo | Full control of private repositories |
| <input type="checkbox"/> repo:status | Access commit status |
| <input type="checkbox"/> repo:deployment | Access deployment status |
| <input type="checkbox"/> public_repo | Access public repositories |
| <input type="checkbox"/> repo:invite | Access repository invitations |
| <input type="checkbox"/> security_events | Read and write security events |
| <input type="checkbox"/> workflow | Update GitHub Action workflows |

- After selecting the necessary options to be used with the PAT, click on the Generate token at the end of the page as shown in the diagram below;

The screenshot shows the 'Generate token' confirmation page. It displays the selected scopes and two buttons at the bottom: 'Generate token' (highlighted in green) and 'Cancel'.

- From the PAT keys generated page display as shown below, copy out the Token and save it, because you can not see it again after the page is closed.



The Personal Access Token(PAT) will be used in place of password during the pushing to the github repository.

You will have noted the word "origin" in the command below, what happens is that when you clone a remote repository to your local machine, git creates an **alias** for you. In nearly all cases this alias is called "origin." It's essentially shorthand for the remote repository's URL. So, to push your changes to the remote repository, you could've used either the command:

git push git@github.com:git/git.git yourbranchname

or

git push origin yourbranchname

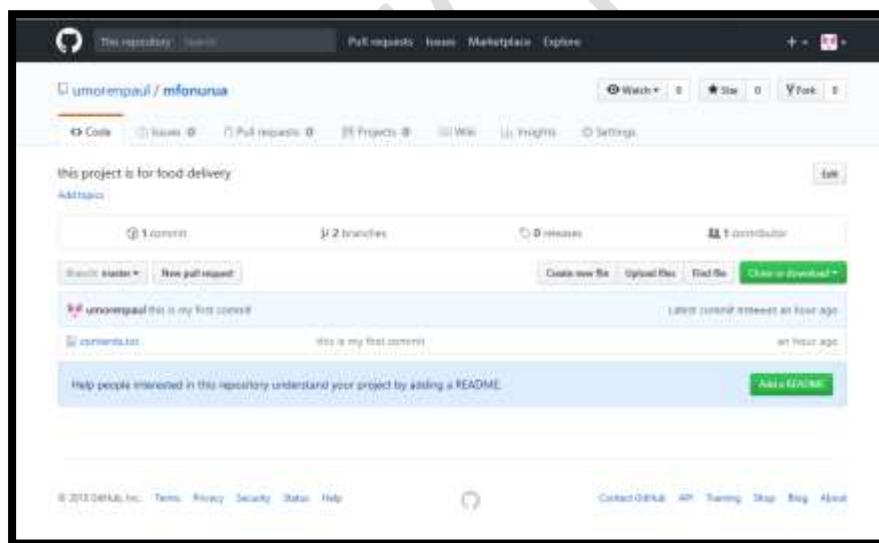
If this is your first time using GitHub locally, it might prompt you to log in with your GitHub username and password- which is the PAT in this case as shown in the diagram below;

```
(base) umoren@umoren-HP-Pavilion-x360-m3-Convertible:~/Desktop/advanceCSS$ git r
emote add origin https://github.com/pinfotech/webupdate.git
fatal: remote origin already exists.
(base) umoren@umoren-HP-Pavilion-x360-m3-Convertible:~/Desktop/advanceCSS$ git p
ush -u origin master
Username for 'https://github.com': pinfotech
Password for 'https://pinfotech@github.com':
Counting objects: 13, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (12/12), done.
Writing objects: 100% (13/13), 1.19 MiB | 41.00 KiB/s, done.
Total 13 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:     https://github.com/pinfotech/webupdate/pull/new/master
remote:
To https://github.com/pinfotech/webupdate.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
(base) umoren@umoren-HP-Pavilion-x360-m3-Convertible:~/Desktop/advanceCSS$
```

If you refresh the GitHub page, you'll see note saying a branch with your name has just been pushed into the repository. You can also click the 'branches' link to see your branch listed there.



Now click the green button in the screenshot above. We're going to make a **pull request!**



Handling merge conflicts

One of the best features of git is its ability to easily merge multiple changes by different people.

Say you and a friend have both made changes to the same file at the same time. When you pull your friend's changes, git will often be able to combine them without any problem.

Sometimes, though, after you do

git pull yourfriend master

You will get a message like

Auto-merging README.md

CONFLICT (content): Merge conflict in README.md

Automatic merge failed; fix conflicts and then commit the result.

If you open the offending file in a text editor, you'll find an indication of the bits that are different, something like this:

<<<<< HEAD

A line in my file.

=====

A line in your friend's file

>>>>> 031389f2cd2acde08e32f0beb084b2f7c3257fff

Edit the bits from <<<<< to >>>>>, to make the file just as you want it.

Then do git add, git commit, and git push.

18.4.6 Open a Pull Request

Pull Requests are the heart of collaboration on GitHub. When you open a *pull request*, you're proposing your changes and requesting that someone review and pull in your contribution and merge them into their branch. Pull requests show *diffs*, or differences, of the content from both branches. The changes, additions, and subtractions are shown in green and red.

As soon as you make a commit, you can open a pull request and start a discussion, even before the code is finished.

By using GitHub's @mention system in your pull request message, you can ask for feedback from specific people or teams, whether they're down the hall or 10 time zones away.

You can even open pull requests in your own repository and merge them yourself. It's a great way to learn the GitHub Flow before working on larger projects.

Step 1 **Pull Request** tab, then from the Pull Request page, click the green **New pull request** button.

Step 2 In the **Example Comparisons** box, select the branch you made, readme-edits, to compare with master (the original).

Step 3 Look over your changes in the diffs on the Compare page, make sure they're what you want to submit.

Step 4 When you're satisfied that these are the changes you want to submit, click the big green **Create Pull Request** button.

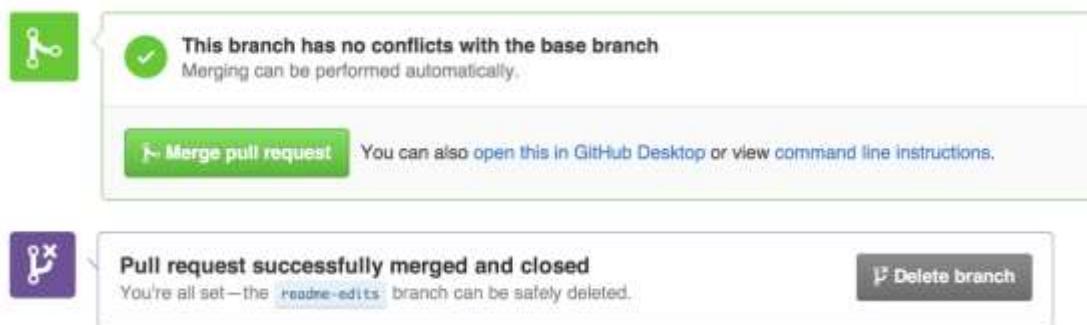
Step 5 Give your pull request a title and write a brief description of your changes.

When you're done with your message, click **Create pull request!**

Step 5. Merge your Pull Request

In this final step, it's time to bring your changes together – merging your readme-edits branch into the master branch.

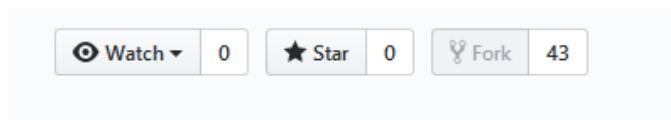
1. Click the green **Merge pull request** button to merge the changes into master.
2. Click **Confirm merge**.
3. Go ahead and delete the branch, since its changes have been incorporated, with the **Delete branch** button in the purple box.



Fork an example repository

Forking a repository is a simple two-step process. Create a repository project, example:- react_works

1. On GitHub, navigate to the umoren paul/react_works repository.
2. In the top-right corner of the page, click **Fork**.

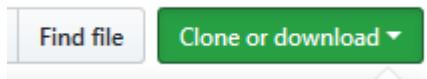


That's it! Now, you have a *fork* of the original umoren paul/react_works repository you can now work with.

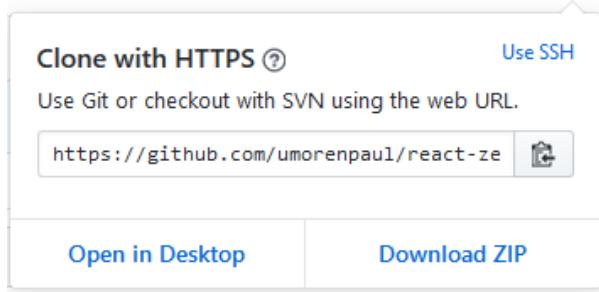
Create a local clone of your fork

Right now, you have a fork of the Spoon-Knife repository, but you don't have the files in that repository on your computer. Let's create a *clone* of your fork locally on your computer.

1. On GitHub, navigate to **your fork** of the react_works repository.
- 2.



3. Under the repository name, click **Clone or download**.



4. In the Clone with HTTPS section, click to copy the clone URL for the repository.
5. Open Git Bash. Type `git clone`, and then paste the URL you copied in Step 2. It will look like this, with your GitHub username instead of YOUR-USERNAME:
6. Press **Enter**. Your local clone will be created.

Resource Materials

<https://www.sitepoint.com/clear-html-attribute/>

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element>

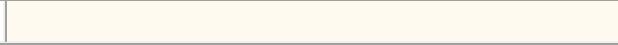
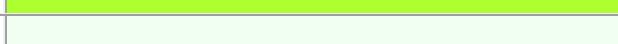
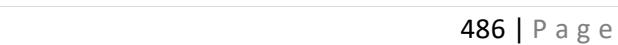
https://www.w3schools.com/tags/att_img_vspace.asp

<https://www.w3docs.com/learn-html/html-marquee-tag.html>

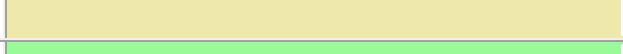
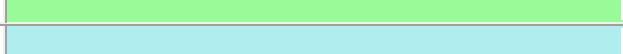
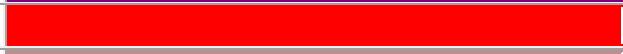
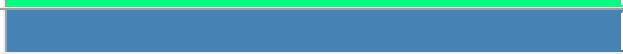
Appendix A

Table 4; HTML colour codes and their names

| Color Name | Color HEX | Color |
|----------------|-----------|-------|
| AliceBlue | #F0F8FF | |
| AntiqueWhite | #FAEBD7 | |
| Aqua | #00FFFF | |
| Aquamarine | #7FFFAD | |
| Azure | #F0FFFF | |
| Beige | #F5F5DC | |
| Bisque | #FFE4C4 | |
| Black | #000000 | |
| BlanchedAlmond | #FFEBCD | |
| Blue | #0000FF | |
| BlueViolet | #8A2BE2 | |
| Brown | #A52A2A | |
| BurlyWood | #DEB887 | |
| CadetBlue | #5F9EA0 | |
| Chartreuse | #7FFF00 | |
| Chocolate | #D2691E | |
| Coral | #FF7F50 | |
| CornflowerBlue | #6495ED | |
| Cornsilk | #FFF8DC | |
| Crimson | #DC143C | |

| | | |
|----------------|---------|--|
| Cyan | #00FFFF |  |
| DarkBlue | #00008B |  |
| DarkCyan | #008B8B |  |
| DarkGoldenRod | #B8860B |  |
| DarkGray | #A9A9A9 |  |
| DarkGreen | #006400 |  |
| DarkKhaki | #BDB76B |  |
| DarkMagenta | #8B008B |  |
| DarkOliveGreen | #556B2F |  |
| DarkOrange | #FF8C00 |  |
| DarkOrchid | #9932CC |  |
| DarkRed | #8B0000 |  |
| DarkSalmon | #E9967A |  |
| DarkSeaGreen | #8FBC8F |  |
| DarkSlateBlue | #483D8B |  |
| DarkSlateGray | #2F4F4F |  |
| DarkTurquoise | #00CED1 |  |
| DarkViolet | #9400D3 |  |
| DeepPink | #FF1493 |  |
| DeepSkyBlue | #00BFFF |  |
| DimGray | #696969 |  |
| DodgerBlue | #1E90FF |  |
| Feldspar | #D19275 |  |
| FireBrick | #B22222 |  |
| FloralWhite | #FFFFAF |  |
| ForestGreen | #228B22 |  |
| Fuchsia | #FF00FF |  |
| Gainsboro | #DCDCDC |  |
| GhostWhite | #F8F8FF |  |
| Gold | #FFD700 |  |
| GoldenRod | #DAA520 |  |
| Gray | #808080 |  |
| Green | #008000 |  |
| GreenYellow | #ADFF2F |  |
| HoneyDew | #F0FFF0 |  |
| HotPink | #FF69B4 |  |
| IndianRed | #CD5C5C |  |
| Indigo | #4B0082 | |

| | | |
|-----------------------------|----------|--|
| <u>Ivory</u> | #FFFFFF0 | |
| <u>Khaki</u> | #F0E68C | |
| <u>Lavender</u> | #E6E6FA | |
| <u>LavenderBlush</u> | #FFF0F5 | |
| <u>LawnGreen</u> | #7CFC00 | |
| <u>LemonChiffon</u> | #FFFACD | |
| <u>LightBlue</u> | #ADD8E6 | |
| <u>LightCoral</u> | #F08080 | |
| <u>LightCyan</u> | #E0FFFF | |
| <u>LightGoldenRodYellow</u> | #FAFAD2 | |
| <u>LightGrey</u> | #D3D3D3 | |
| <u>LightGreen</u> | #90EE90 | |
| <u>LightPink</u> | #FFB6C1 | |
| <u>LightSalmon</u> | #FFA07A | |
| <u>LightSeaGreen</u> | #20B2AA | |
| <u>LightSkyBlue</u> | #87CEFA | |
| <u>LightSlateBlue</u> | #8470FF | |
| <u>LightSlateGray</u> | #778899 | |
| <u>LightSteelBlue</u> | #B0C4DE | |
| <u>LightYellow</u> | #FFFFE0 | |
| <u>Lime</u> | #00FF00 | |
| <u>LimeGreen</u> | #32CD32 | |
| <u>Linen</u> | #FAF0E6 | |
| <u>Magenta</u> | #FF00FF | |
| <u>Maroon</u> | #800000 | |
| <u>MediumAquaMarine</u> | #66CDAA | |
| <u>MediumBlue</u> | #0000CD | |
| <u>MediumOrchid</u> | #BA55D3 | |
| <u>MediumPurple</u> | #9370D8 | |
| <u>MediumSeaGreen</u> | #3CB371 | |
| <u>MediumSlateBlue</u> | #7B68EE | |
| <u>MediumSpringGreen</u> | #00FA9A | |
| <u>MediumTurquoise</u> | #48D1CC | |
| <u>MediumVioletRed</u> | #C71585 | |
| <u>MidnightBlue</u> | #191970 | |
| <u>MintCream</u> | #F5FFFA | |
| <u>MistyRose</u> | #FFE4E1 | |
| <u>Moccasin</u> | #FFE4B5 | |

| | | |
|----------------------|---------|--|
| <u>NavajoWhite</u> | #FFDEAD |  |
| <u>Navy</u> | #000080 |  |
| <u>OldLace</u> | #FDF5E6 |  |
| <u>Olive</u> | #808000 |  |
| <u>OliveDrab</u> | #6B8E23 |  |
| <u>Orange</u> | #FFA500 |  |
| <u>OrangeRed</u> | #FF4500 |  |
| <u>Orchid</u> | #DA70D6 |  |
| <u>PaleGoldenRod</u> | #EEE8AA |  |
| <u>PaleGreen</u> | #98FB98 |  |
| <u>PaleTurquoise</u> | #AFEEEE |  |
| <u>PaleVioletRed</u> | #D87093 |  |
| <u>PapayaWhip</u> | #FFEFD5 |  |
| <u>PeachPuff</u> | #FFDAB9 |  |
| <u>Peru</u> | #CD853F |  |
| <u>Pink</u> | #FFC0CB |  |
| <u>Plum</u> | #DDA0DD |  |
| <u>PowderBlue</u> | #B0E0E6 |  |
| <u>Purple</u> | #800080 |  |
| <u>Red</u> | #FF0000 |  |
| <u>RosyBrown</u> | #BC8F8F |  |
| <u>RoyalBlue</u> | #4169E1 |  |
| <u>SaddleBrown</u> | #8B4513 |  |
| <u>Salmon</u> | #FA8072 |  |
| <u>SandyBrown</u> | #F4A460 |  |
| <u>SeaGreen</u> | #2E8B57 |  |
| <u>SeaShell</u> | #FFF5EE |  |
| <u>Sienna</u> | #A0522D |  |
| <u>Silver</u> | #C0C0C0 |  |
| <u>SkyBlue</u> | #87CEEB |  |
| <u>SlateBlue</u> | #6A5ACD |  |
| <u>SlateGray</u> | #708090 |  |
| <u>Snow</u> | #FFFFFA |  |
| <u>SpringGreen</u> | #00FF7F | |
| <u>SteelBlue</u> | #4682B4 | |
| <u>Tan</u> | #D2B48C | |
| <u>Teal</u> | #008080 | |
| <u>Thistle</u> | #D8BFD8 | |

| | | |
|--------------------|---------|--|
| <u>Tomato</u> | #FF6347 |  |
| <u>Turquoise</u> | #40E0D0 |  |
| <u>Violet</u> | #EE82EE |  |
| <u>VioletRed</u> | #D02090 |  |
| <u>Wheat</u> | #F5DEB3 |  |
| <u>White</u> | #FFFFFF |  |
| <u>WhiteSmoke</u> | #F5F5F5 |  |
| <u>Yellow</u> | #FFFF00 |  |
| <u>YellowGreen</u> | #9ACD32 |  |

PINFOICT ACADEMY

Appendix B

ASCII Entities in HTML

Table 4. ASCII Entities with new Entity Names

| Result | Description | Entity Name | Entity Number |
|--------|----------------|-------------|---------------|
| " | quotation mark | " | " |
| & | Ampersand | & | & |
| < | less-than | < | < |
| > | greater-than | > | > |

ISO 8859-1 Symbol Entities

| Result | Description | Entity Name | Entity Number |
|--------|-----------------------------|-------------|---------------|
| | non-breaking space | | |
| ¡ | inverted exclamation mark | ¡ | ¡ |
| ¤ | Currency | ¤ | ¤ |
| ¢ | Cent | ¢ | ¢ |
| £ | Pound | £ | £ |
| ¥ | Yen | ¥ | ¥ |
| ¦ | broken vertical bar | ¦ | ¦ |
| § | Section | § | § |
| .. | spacing diaeresis | ü | ¨ |
| © | Copyright | © | © |
| ª | feminine ordinal indicator | ª | ª |
| « | angle quotation mark (left) | « | « |
| ¬ | Negation | ¬ | ¬ |
| ‐ | soft hyphen | ­ | ­ |
| ® | registered trademark | ® | ® |
| ™ | Trademark | ™ | |
| ‐ | spacing macron | ¯ | ¯ |
| ° | Degree | ° | ° |
| ± | plus-or-minus | ± | ± |
| ² | superscript 2 | ² | ² |
| ³ | superscript 3 | ³ | ³ |
| ‘ | spacing acute | ´ | ´ |
| µ | Micro | µ | µ |
| ¶ | Paragraph | ¶ | ¶ |
| · | middle dot | · | · |

| | | | |
|---|------------------------------|----------|--------|
| , | spacing cedilla | ¸ | ¸ |
| ¹ | superscript 1 | ¹ | ¹ |
| º | masculine ordinal indicator | º | º |
| » | angle quotation mark (right) | » | » |
| ¼ | fraction ¼ | ¼ | ¼ |
| ½ | fraction ½ | ½ | ½ |
| ¾ | fraction ¾ | ¾ | ¾ |
| ¿ | inverted question mark | ¿ | ¿ |
| × | Multiplication | × | × |
| ÷ | Division | ÷ | ÷ |

ISO 8859-1 Character Entities

| Result | Description | Entity Name | Entity Number |
|--------|------------------------------|-------------|---------------|
| À | capital a, grave accent | À | À |
| Á | capital a, acute accent | Á | Á |
| Â | capital a, circumflex accent | Â | Â |
| Ã | capital a, tilde | Ã | Ã |
| Ä | capital a, umlaut mark | Ä | Ä |
| Å | capital a, ring | Å | Å |
| Æ | capital ae | Æ | Æ |
| Ç | capital c, cedilla | Ç | Ç |
| È | capital e, grave accent | È | È |
| É | capital e, acute accent | É | É |
| Ê | capital e, circumflex accent | Ê | Ê |
| Ë | capital e, umlaut mark | Ë | Ë |
| Ì | capital i, grave accent | Ì | Ì |
| Í | capital i, acute accent | Í | Í |
| Î | capital i, circumflex accent | Î | Î |
| Ï | capital i, umlaut mark | Ï | Ï |
| Ð | capital eth, Icelandic | Ð | Ð |
| Ñ | capital n, tilde | Ñ | Ñ |
| Ò | capital o, grave accent | Ò | Ò |
| Ó | capital o, acute accent | Ó | Ó |
| Ô | capital o, circumflex accent | Ô | Ô |
| Õ | capital o, tilde | Õ | Õ |
| Ö | capital o, umlaut mark | Ö | Ö |
| Ø | capital o, slash | Ø | Ø |
| Ù | capital u, grave accent | Ù | Ù |
| Ú | capital u, acute accent | Ú | Ú |

| | | | |
|---|------------------------------|----------|--------|
| Û | capital u, circumflex accent | Û | Û |
| Ü | capital u, umlaut mark | Ü | Ü |
| Ý | capital y, acute accent | Ý | Ý |
| Þ | capital THORN, Icelandic | Þ | Þ |
| ß | small sharp s, German | ß | ß |
| À | small a, grave accent | à | à |
| Á | small a, acute accent | á | á |
| Â | small a, circumflex accent | â | â |
| Ã | small a, tilde | ã | ã |
| Ä | small a, umlaut mark | ä | ä |
| Å | small a, ring | å | å |
| Æ | small ae | æ | æ |
| Ç | small c, cedilla | ç | ç |
| È | small e, grave accent | è | è |
| É | small e, acute accent | é | é |
| Ê | small e, circumflex accent | ê | ê |
| Ë | small e, umlaut mark | ë | ë |
| Ì | small i, grave accent | ì | ì |
| Í | small i, acute accent | í | í |
| Î | small i, circumflex accent | î | î |
| Ï | small i, umlaut mark | ï | ï |
| Ð | small eth, Icelandic | ð | ð |
| Ñ | small n, tilde | ñ | ñ |
| Ò | small o, grave accent | ò | ò |
| Ó | small o, acute accent | ó | ó |
| Ô | small o, circumflex accent | ô | ô |
| Õ | small o, tilde | õ | õ |
| Ö | small o, umlaut mark | ö | ö |
| Ø | small o, slash | ø | ø |
| Ù | small u, grave accent | ù | ù |
| Ú | small u, acute accent | ú | ú |
| Û | small u, circumflex accent | û | û |
| Ü | small u, umlaut mark | ü | ü |
| Ý | small y, acute accent | ý | ý |
| Þ | small thorn, Icelandic | þ | þ |
| ÿ | small y, umlaut mark | ÿ | ÿ |

Some Other Entities supported by HTML

| Result | Description | Entity Name | Entity Number |
|--------|-------------|-------------|---------------|
|--------|-------------|-------------|---------------|

| | | | |
|----|---------------------------------------|----------|---------|
| Œ | capital ligature OE | Œ | Œ |
| œ | small ligature oe | œ | œ |
| Š | capital S with caron | Š | Š |
| š | small S with caron | š | š |
| Ÿ | capital Y with diaeres | Ÿ | Ÿ |
| ^ | modifier letter circumflex accent | ˆ | ˆ |
| ~ | small tilde | ˜ | ˜ |
| | en space |   |   |
| | em space |   |   |
| | thin space |   |   |
| | zero width non-joiner | ‌ | ‌ |
| | zero width joiner | ‍ | ‍ |
| | left-to-right mark | ‎ | ‎ |
| | right-to-left mark | ‏ | ‏ |
| – | en dash | – | – |
| — | em dash | — | — |
| ‘ | left single quotation mark | ‘ | ‘ |
| ’ | right single quotation mark | ’ | ’ |
| , | single low-9 quotation mark | ‚ | ‚ |
| “ | left double quotation mark | “ | “ |
| ” | right double quotation mark | ” | ” |
| „ | double low-9 quotation mark | „ | „ |
| † | Dagger | † | † |
| ‡ | double dagger | ‡ | ‡ |
| %o | per mille | ‰ | ‰ |
| ⟨ | single left-pointing angle quotation | ‹ | ‹ |
| ⟩ | single right-pointing angle quotation | › | › |
| € | Euro | € | € |
| ™ | Trademark | | ™ |

Indexes

<address> Tag · 16

**** Tag · 12

<body> Tag · 10

<caption> Tag · 36

<caption>tags · 38

<center> Tag · 15

<form> Tag · 40

<head> tag · 7, 180, 181

<html> tag · 7, 180

<i> Tag · 12

<marquee> tag · 29, 30

<meta> tags · 8

<optgroup> tag · 54

<option> tag · 53

<pre> Tag · 14

<sub> Tag · 12

<sup> Tag · 12

<table> Tag · 10, 35

A

accept · 32, 40, 41, 42, 45, 48, 108

accept attribute · 41

action · 31, 40, 41, 43, 44, 45, 46, 48, 49, 82

Alignment attributes · 27

Alternate Text · 24

Anchor · 55, 57

Animated Text · 29

APIs · 5, 6

article · 5, 136, 139, 141, 142, 143, 144, 145, 146, 174, 175, 176

ASCII Entities · 249

aside · 5, 46, 137, 139, 141, 143, 144, 145, 147, 174, 177

attribute substring selectors · 81

Attributes · 10, 30, 40, 180

B

Background Audio · 28

Background colours · 4

background-attachment · 99

background-position · 99, 117

basic html structure · 7

bgcolor attribute · 10, 25, 31, 35

border · 10, 13, 35, 36, 37, 38, 42, 46, 56, 63, 64, 80, 81, 82, 93, 94, 104, 106, 107, 108, 109, 111, 112, 114, 116, 117, 124, 132, 134, 138, 139, 140, 150, 151, 163, 165, 166, 168, 169, 170

border attribute · 10, 35

Borders · 106, 107, 112, 134

Box shadow · 169

browsing · 2

C

Caching · 61

cascade · 6

cellpadding · 36, 46

cellspacing · 35

Checkbox · 47

Class · 66

clear attribute · 13

Coding Conventions · 178

Color · 60, 62, 108, 245

comment tag · 12

common gateway interface · 40

CSS · 1, 2, 4, 6, 14, 15, 19, 21, 59, 60, 61, 62, 64, 65, 71, 72, 76, 80, 83, 89, 90, 91, 92, 93, 94, 95, 97, 103, 106, 113, 115, 122, 124, 125, 126, 132, 136, 140, 146, 147, 150, 153, 154, 165, 168, 175

D

declaration · 9, 60, 61, 62, 88, 134, 149

Direct Child Selectors · 76

Disabled · 46

Divs · 20

DOCTYPE · 7, 8, 9, 10, 17, 18, 19, 24, 25, 26, 27, 31, 43, 44, 45, 48, 72, 90, 91, 103, 107, 113, 114, 116, 122, 126, 128, 130,

132, 134, 136, 142, 147, 150, 154, 160, 165, 168, 178, 180, 181

Document Type · 9, 178

DOM · 6, 180

drop cap · 74

Dynamic Pseudo-classes · 72

E

enctype · 40, 41, 44, 45, 48

Enctype · 41

External · 61

F

Fieldset · 55

File · 29, 44

float · 27, 74, 93, 113, 114, 115, 116, 117, 142, 143, 144, 145, 146, 147, 150, 151, 165, 166

Float · 27, 113, 114, 116

Font-family · 83

Font-size · 87

Font-style · 85

Font-weight · 60, 85

footer · 5, 20, 21, 94, 136, 141, 142, 143, 144, 145, 146, 174, 176, 177

formatting · 5, 6, 12

Forms · 4, 40

frames · 4, 5, 9, 159, 160, 161, 162, 163

frameset · 5, 9, 159, 160, 161, 162, 163

G

GET · 40, 41

gradient · 153

grouped · 60

H

hackers · 1

Headings · 10, 65

Hexadecimal · 28, 63

hidden · 42, 43, 46, 94, 146

Hidden · 46

Horizontal Rules · 22

href · 7, 9, 15, 33, 55, 56, 57, 73, 81, 103, 115, 116, 125, 136, 137, 142, 143, 145, 146, 147, 152, 161, 162, 165, 166, 167, 175, 177, 180, 181

hspace · 27, 28, 30, 32, 33

HTML · 1, 2, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 17, 20, 22, 23, 24, 33, 34, 35, 40, 47, 48, 55, 60, 61, 66, 71, 80, 93, 103, 114, 125, 126, 132, 140, 163, 168, 174, 175, 178, 179, 180, 181, 245, 249, 251

HTML5 · 1, 5, 6, 9, 116, 174, 175, 176, 177, 178, 179, 180, 181

HTTP · 2, 4, 8, 165

Hyperlink Styling · 72

hypermedia · 2

Hypertext · 2

I

ID · 66

images · 4, 13, 22, 24, 25, 26, 27, 33, 35, 41, 56, 94, 95, 97, 113, 130, 166, 167, 168, 180

Images · 22, 23, 24, 25, 26, 27, 33, 93

Input · 42, 43, 46

Internet · 1, 2, 4, 6, 18, 28, 163, 168

L

Length · 61

letter-spacing · 88

line break · 11, 12, 42

line-height · 88

Links · 4, 72, 125

Listing · 17, 19

List-style-position · 131

List-style-type · 126

loop attribute · 31

M

Margin · 106

MARKUP Language · 1

marquee · 29, 30, 31, 32, 33

measurement · 61

Media · 29, 165

media query · 165

Meta · 7, 8, 181

MIME · 41

Mouse Cursor · 154

Multiple <option> · 53

multiple shadows · 170

Multiple-column layouts · 172

N

nav · 5, 137, 138, 143, 146, 174, 175

Navigation · 114, 126

Nesting Lists · 18

Netscape · 4

noshade attribute · 23

O

ordered list · 17, 85, 126, 128

Ordered Lists · 17

overflow · 94, 144, 146

P

Paragraphs · 11

Password · 43, 44, 45

pixels · 25, 28, 32, 36, 61, 145, 160, 165, 169

POST · 40, 41

property · 60, 83, 85, 88, 90, 91, 95, 104, 106, 107, 108, 111, 113, 115, 126, 130, 132, 144, 154, 168, 169, 171, 181

Pseudo-Classes · 72

Pull-down Menus · 52

R

Radio button · 47

Read-Only · 46

Referenced content · 33

RGB · 25, 31, 64, 109

RGBA · 64

rounded corners · 168

Rule · 60

S

scrollamount · 30, 32, 33

scrolldelay · 30, 32, 33

selector · 60, 66, 71, 76, 78, 80, 81, 82, 181

servers · 3, 33

Shorthand hexadecimal · 64

sidebar · 124, 136, 177

spans · 20, 21, 132

src attribute · 24

Styling tables · 132

Submit Button · 50

surfing · 2

T

Table Tags · 35

Target · 56

text case · 89

Text Decorations · 91

Text editor · 6

Text Indentation · 90, 91

text input · 42, 43, 44, 46

Text Shadows · 171

Text-align · 91

textarea · 46, 50, 51, 52, 82

three columns layout · 146

thumbnails · 103

Tim Berners-Lee · 4

U

Universal Selector · 71

Unordered List · 17, 18, 19, 22, 24

URL · 8, 26, 40, 41, 44, 146

V

vspace · 27, 28, 30, 32, 33

W

W3C · 2, 4, 5, 9, 10, 24, 114, 154

web · 1, 2, 5, 6, 8, 15, 22, 23, 25, 28, 42, 52, 55, 60, 61, 72, 83, 85, 93, 95, 147, 153, 154, 168, 174, 178

Web browser · 3

Web client · 4

Web page · 2, 4, 160

Web site · 5

width attributes · 32

wrap · 14, 49, 50, 51, 52, 118, 137, 142, 146, 168

X

XHTML · 1, 2, 5, 9, 10, 13, 14, 22, 40, 42, 47, 52, 154, 178, 179

XML · 2, 4, 5, 10, 179, 180