Css Postitioning
HTML 5

Day 1 (HTML)

what is a markup language?

Hypertext means machine readable text and Markup means to structure it in a specific format.So,HTML is called hypertext markup language because it is a language that allows users to organize, improve the appearance of, and link text with data on the internet.

What is HTML?

HTML is a language for describing web pages.

- HTML stands for **H**yper **T**ext **M**arkup **L**anguage
- HTML is not a programming language, it is a **markup language**
- A markup language is a set of **markup tags**
- HTML uses **markup tags** to describe web pages

Hyper -> Reference Link
Text -> Data/Information
Markup->Predefined tags
Language->Communication

Html Sections

<head>
<title></title>
<style></style>
<script></script>
<meta ...../>
<link...../>
</head>

<body>
<h1>....</h1>
<h2>....</h2>
....
<a>....
.....
</body>

1. HTML Tags

The Basic HTML Tags are,

- <html> Defines an HTML document

  The <html> tag tells the browser that this is an HTML document.
  The <html> tag is the container for all other HTML elements

- <body> Defines the document's body

  The <body> element contains all the contents of an HTML document, such as text, hyperlinks, images, tables, lists, etc.

- <h1> to <h6> Defines header 1 to header 6

  <h1> defines the most important heading.
  <h6> defines the least important heading.

- Defines a paragraph

  Browsers automatically add some space (margin) before and after each element. The margins can be modified with CSS (with the margin properties).

- <pre> Tag

  The <pre> tag defines preformatted text.

  Text in a <pre> element is displayed in a fixed-width font (usually Courier), and it preserves both spaces and line breaks.

-
 Inserts a single line break

  The <br> tag is an empty tag which means that it has no end tag.
  Note: Use the <br> tag to insert line breaks, not to create paragraphs.

- <hr> Defines a horizontal line

  The <hr> tag creates a horizontal line in an HTML page.
  The <hr> element can be used to separate content in an HTML page.

HTML Element - "HTML tags" and "HTML elements" are often used to describe the same thing. But strictly speaking, an HTML element is everything between the start tag and the end tag, including the tags:

For example:
This is a paragraph.

- **Attribute** - Used to modify the value of the HTML element. Elements will often have multiple attributes.

  Below is a list of some attributes that are standard for most HTML elements:

  Hello World

3. Comments in HTML
HTML Comments is information about that page. The comment declaration are given as
For Example:
<!--This is the first page -->
An HTML comment begins with "<!--" ends with "-->"

- **Text Formatting Tags**

  Some of the text formatting tags in HTML is given below.

Tags	Description
<b>	Defines bold text
<big>	Defines big text
	Defines emphasized text
	Defines strong text

<sup> Defines superscripted text
<sub> Defines subscripted text
<i> Defines italic text
<small> Defines small text
<u> Defines underlined text

Few Examples for text formatting.

```
<html>

<body>

<p><b>This text is bold</b></p>

<p><strong>This text is strong</strong></p>

<p><big>This text is big</big></p>

<p><em>This text is emphasized</em></p>

<p><i>This text is italic</i></p>

<p><small>This text is small</small></p>

<p>This is<sub> subscript</sub> and <sup>superscript</sup></p>

<p><u>This text is underlinel</u></p>
</body>

</html>
```

This text is bold

This text is strong

This text is big

This text is emphasized

This text is italic

This text is small

This is subscript and superscript

<u>This text is underlined</u>

HTML Entities

Some characters are reserved in HTML.

It is not possible to use the less than (<) or greater than (>) signs in your text, because the

browser will mix them with tags.

To actually display reserved characters, we must use character entities in the HTML source code.

A character entity looks like this:

HTML Useful Character Entities

Note: Entity names are case sensitive!

Result	Description	Entity Name	Entity Number
	non-breaking space		
<	less than	&lt;	&#60;
>	greater than	&gt;	&#62;
¢	cent	&cent;	&#162;
£	pound	&pound;	&#163;
¥	yen	&yen;	&#165;
€	euro	&euro;	&#8364;
©	copyright	&copy;	&#169;
®	registered trademark	&reg;	&#174;
™	trademark	&trade;	&trade;
₹	INR	&#8377;	&#x20B9;

For Tab space ** **

DOCTYPE

According to HTML standards, each HTML document begins with a DOCTYPE declaration that specifies which version of HTML the document uses. Originally, the DOCTYPE declaration was used only by SGML-based tools like HTML validators, which needed to determine which version of HTML a document used (or claimed to use).

- **HTML Links**

A hyperlink (or link) is a word, group of words, or image that you can click on to jump to a new document or a new section within the current document.
When you move the cursor over a link in a Web page, the arrow will turn into a little hand.
Links are specified in HTML using the <a> tag.
The <a> tag can be used to create a link to another document, by using the href attribute

For Example:
This anchor defines a link to Google.com:
Visit Google.com!
This text

The target Attribute

The target attribute specifies where to open the linked document.
The example below will open the linked document in a new browser window or a new tab:
Click Here

Image-link:
In HTML, images are defined with the <img> tag.
The <img> tag is empty, which means that it contains attributes only, and has no closing tag.
To display an image on a page, you need to use the src attribute. Src stands for "source".
The value of the src attribute is the URL of the image you want to display.
For Example:

Mailto link: Send Mail

6. HTML Tables

With HTML you can create tables.
For Example:
Tables are defined with the <table> tag.
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.

HTML Table Headers

Header information in a table are defined with the <th> tag.
All major browsers display the text in the <th> element as bold and centered.

```
<table border="1">
<tr>
<th>Header 1</th>
<th>Header 2</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

How the HTML code above looks in your browser:

Header 1	Header 2
row 1, cell	row 1, cell 2

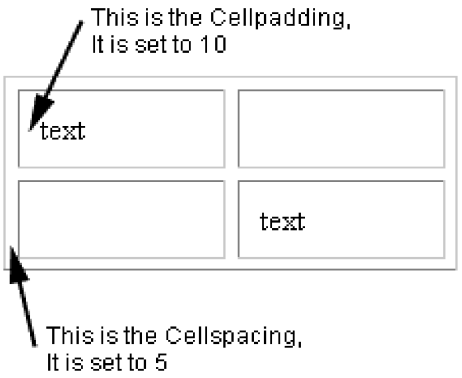
1	
row 2, cell 1	row 2, cell 2

<table> cellpadding Attribute

<table width="150" border="1" cellpadding="10">

<table> cellspacing Attribute

<table width="150" border="1" cellspacing="5">



7. Frames

With frames, you can display more than one HTML document in the same browser window. Each HTML document is called a frame, and each frame is independent of the others.

For Example:

Frame Tags are,

Tag	**Description**
<frameset>	Defines a set of frames

The <frameset> tag defines a frameset.

The <frameset> element holds one or more <frame> elements. Each <frame> element can hold a separate document.

The <frameset> element specifies HOW MANY columns or rows there will be in the frameset, and HOW MUCH percentage/pixels of space will occupy each of them.

For example.

```
<frameset cols="25%,*,25%">
 <frame src="frame_a.htm" />
 <frame src="frame_b.htm" />
 <frame src="frame_c.htm" />
```

</frameset>

<noframes>Defines a noframe section for browsers that do not handle frames
The <noframes> tag is a fallback tag for browsers that do not support frames. It can contain all the HTML elements that you can find inside the <body> element of a normal HTML page.
The <noframes> element can be used to link to a non-frameset version of the web site or to display a message to users that frames are required.
The <noframes> element goes inside the <frameset> element.
For Example :

```
<html>
<frameset cols="25%,50%,25%">
 <frame src="frame_a.htm" />
 <frame src="frame_b.htm" />
 <frame src="frame_c.htm" />
 <noframes><body>Sorry, your browser does not handle frames!</body>
 </noframes>
</frameset>
</html>
```

Few Frameset attributes

```
<frame
      name=""
      src=""
      noresize="noresize"
      scrolling="yes/No"
      frameborder=""
      title=""
      style="" />
```

- **Iframes**

The <iframe> tag specifies an inline frame.

An inline frame is used to embed another document within the current HTML document.

For Example:
The syntax of defining Iframe,

```
<iframe src ="demo_iframe.htm" width="100%">
</iframe>
```

9.DIV tag.

Definition and Usage

The tag defines a division or a section in an HTML document.
The tag is used to group block-elements to format them with styles.
Tip: The element is very often used together with CSS, to layout a web page.
Note: By default, browsers always place a line break before and after the element. However, this can be changed with CSS.
Example coding.

```
<div style="color: #0900C4; border: 1px solid black;">
  <h5>Subtitle</h5>
```

```
  <p>This paragraph would be your content paragraph...</p>
  <p>Here's another content article right here.</p>
</div>
```

Output.

Subtitle

This paragraph would be your content paragraph...
Here's another content article right here.

- **Meta tag**

  Metadata is data (information) about data.
  The <meta> tag provides metadata about the HTML document. Metadata will not be displayed on the page, but will be machine parsable.
  Meta elements are typically used to specify page description, keywords, author of the document, last modified, and other metadata.

  The <meta> tag always goes inside the <head> element.
  The metadata can be used by browsers (how to display content or reload page), search engines (keywords), or other web services.
  For Example:
  The syntax of defining Meta tag,
  <meta name="keywords" content="HTML, CSS, XML, XHTML, JavaScript" />

Day 2 (HTML)

1. HTML Images

The <img> tag defines an image in an HTML page.
The <img> tag has two required attributes: src and alt.
Note: Images are not technically inserted into an HTML page, images are linked to HTML pages. The <img> tag creates a holding space for the referenced image.
Tip: To link an image to another document, simply nest the <img> tag inside <a> tags.

Attribute	Description
alt	Specifies an alternate text for an image
src	Specifies the URL of an image

For Example:
The syntax of defining an image,

The most common HTML lists are ordered and unordered lists:

An ordered list:

- The first list item
- The second list item
- The third list item

<ol type=”upper-roman”>
Milk
Tea

Coffee
Juice

An unordered list:

- List item
- List item
- List item

<ul type=”circle”>
Milk
Tea
Coffee
Juice

HTML Definition Lists

A definition list is a list of items, with a description of each item.
The <dl> tag defines a definition list.
The <dl> tag is used in conjunction with <dt> (defines the item in the list) and <dd> (describes the item in the list):

<dl>
<dt>Coffee</dt>
<dd>- black hot drink</dd>
<dd>- black cold drink</dd>
<dt>Milk</dt>
<dd>- white cold drink</dd>
<dd>- Rose cold drink</dd>
</dl>

Coffee

- black hot drink
- black cold drink

Milk

- white cold drink
- Rose cold drink

2. HTML Forms

HTML forms are used to pass data to a server.
A form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select lists, textarea, fieldset, legend, and label elements.
The <form> tag is used to create an HTML form:
For Example:
A form is defined with the <form> tag.
The syntax of defining form,

```
<form>
First name:<input type="text" name="firstname"> <br>
Last name:<input type="text" name="lastname">
</form>
```

3. HTML Input

Input fields come in several flavors including checkboxes, text fields, radios, and form submission buttons. The <input /> tag does not require a closing tag and is thus an "all in one" tag.
For Example:
Some attributes are,

- **Text**
- **Password**
- **Checkbox**
- **Radio**
- **file**
- **Hidden**
- **Button**
- **Submit**
- **Reset**

The syntax of defining an input tag,

```
<input type="text" />
<input type="password" />
<input type="checkbox" />
<input type="radio" />
<input type="file" />
<input type="submit" value="Submit" />
<input type="Reset" value="Clear!" />
```

4. Text Fields

Text fields are small rectangles that allow a user to simply input some text and submit that information to the web server.
For Example:

The syntax of defining the text fields,
<form>
First name: <input type="text" name="firstname" />
Last name: <input type="text" name="lastname" />
</form>

5. Radio Buttons

In HTML, Radio Buttons are used when you want the user to select one of a limited number of choices.

For Example:

The syntax of defining Radio buttons,
<form>
<input type="radio" name="sex" value="male" /> Male
<input type="radio" name="sex" value="female" /> Female
</form>

6. Checkboxes

In HTML, Checkboxes are used when you want the user to select one or more options of a limited number of choices.

For Example:

The syntax of defining check boxes,
<form>
<input type="checkbox" name="vehicle" value="Bike" /> I have a bike
<input type="checkbox" name="vehicle" value="Car" /> I have a car
</form>

7. Label

The <label> tag defines a label for an <input> element.
The <label> element does not render as anything special for the user. However, it provides a usability improvement for mouse users, because if the user clicks on the text within the <label> element, it toggles the control.
The for attribute of the <label> tag should be equal to the id attribute of the related element to bind them together.

For Example:

The syntax of defining Label,
<LABEL FOR="moreinfo">
send more information
<INPUT NAME="moreinfo" TYPE=CHECKBOX ID="moreinfo">
</LABEL>

8. Text-area

The text area is a multi-line text input control. A user can write text in the text-area. In a text-area you can write an unlimited number of characters.

For Example:

The syntax of defining Text - areas
<textarea rows="2" cols="20">The cat was playing in the garden.
Suddenly a dog showed up.....
</textarea>

10. Drop-down box

The <select> tag is used to create a drop-down list.

The <option> tags inside the <select> element define the available options in the list.

Tip: The <select> element is a form control and is used to collect user input.

For Example:

```
<select>
 <option value="volvo">Volvo</option>
 <option value="saab">Saab</option>
 <option value="mercedes">Mercedes</option>
 <option value="audi">Audi</option>
</select>
```

9. Submit Button

A submit button is used to send form data to a server. The data is sent to the page specified in the form's action attribute. The file defined in the action attribute usually does something with the received input:

<input type="submit" /> defines a submit button.

For Example:

The syntax of defining Submit button,

```
<form name="input" action="html_form_action.asp" method="get">
Username: <input type="text" name="user" />
<input type="submit" value="Submit" />
</form>
```

Exercise.

Creating a table
Creating a Registration form html inputs tags.
Creating a webpage using frames
Creating a webpage using DIV tag

CSS

What is CSS?

- **CSS** stands for **C**ascading **S**tyle **S**heets
- Styles define **how to display** HTML elements
- Styles were added to HTML 4.0 **to solve a problem**
- **External Style Sheets** can save a lot of work
- External Style Sheets are stored in **CSS files**

Three Ways to Insert CSS

There are three ways of inserting a style sheet:

- Inline style
- Internal style sheet
- External style sheet

Inline Styles

An inline style loses many of the advantages of style sheets by mixing content with presentation. Use this method sparingly!
To use inline styles you use the style attribute in the relevant tag. The style attribute can contain any CSS property. The example shows how to change the color and the left margin of a paragraph:

```
<p style="color:sienna;margin-left:20px">This is a paragraph.</p>
```

Internal Style Sheet

An internal style sheet should be used when a single document has a unique style. You define internal styles in the head section of an HTML page, by using the <style> tag, like this:

```
<head>
<style type="text/css">
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
</style>
</head>
```

External Style Sheet

An external style sheet is ideal when the style is applied to many pages. With an external style sheet, you can change the look of an entire Web site by changing one file. Each page must link to the style sheet using the <link> tag. The <link> tag goes inside the head section:

```
<head>
<link rel="stylesheet" type="text/css" href="abdul.css" />
</head>
```

An external style sheet can be written in any text editor. The file should not contain any html tags. Your style sheet should be saved with a .css extension. An example of a style sheet file is shown
``
below:

```
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
```

The ID and CLASS Selectors

In addition to setting a style for a HTML element, CSS allows you to specify your own selectors called "id" and "class".

The id Selector.

The id selector is used to specify a style for a single, unique element.
The id selector uses the id attribute of the HTML element, and is defined with a "#".
The style rule below will be applied to the element with id="para1":

Example

```
#para1
{
text-align:center;
color:red;
}
```

The class Selector

The class selector is used to specify a style for a group of elements. Unlike the id selector, the class selector is most often used on several elements.
This allows you to set a particular style for many HTML elements with the same class.
The class selector uses the HTML class attribute, and is defined with a "."
In the example below, all HTML elements with class="center" will be center-aligned:

Example

```
.center
 {
text-align:center;
font-family:"Times New Roman";
font-size:20px;
 }
```

You can also specify that only specific HTML elements should be affected by a class.
In the example below, all p elements with class="center" will be center-aligned:Example

```
p.center
{
text-align:center;
background-color:#b0c4de;
}
```

ID's are unique

Each element can have only one ID
Each page can have only one element with that ID

When I was first learning this stuff, I heard over and over that you should only use ID's once, but you can use classes over and over. It basically went in one ear and out the other because it sounded more like a good "rule of thumb" to me rather than something extremely important. If you are purely an HTML/CSS person, this attitude can persist because to you, they really don't seem to do anything different.

Here is one: your code will not pass validation if you use the same ID on more than one element. Validation should be important to all of us, so that alone is a big one. We'll go over more reasons for uniqueness as we go on.

Classes are NOT unique

You can use the same class on multiple elements.
You can use multiple classes on the same element.

Any styling information that needs to be applied to multiple objects on a page should be done with a class. Take for example a page with multiple "widgets":

```
<div class="widget"></div>
<div class="widget"></div>
<div class="widget"></div>
```

You can now use the class name "widget" as your hook to apply the same set of styling to each one of these. But what if you need one of them to be bigger than other other, but still share all the other attributes. Classes has you covered there, as you can apply more than one class:

```
<div class="widget"></div>
<div class="widget big"></div>
<div class="widget"></div>
```

No need to make a brand new class name here, just apply a new class right in the class attribute. These classes are space delimited and most browsers support any number of them (actually, it's more like thousands, but way more than you'll ever need).
Styling Links (Changing the color of Hyperlinks)

Here is a few ways to change and control the color of hyperlinks. Hyperlinks can be made to change colors by adding a style sheet to your HTML and editing the following Anchor Pseudo-classes:

- **Link**

  : A normal, unvisited link
- **Visited:** The color of a visited link.
- **Active :** The color of the link when it is clicked. The link will remain that color when users

  hit the Back Button after visiting the link destination.
- **Hover:** The color of the link when the mouse hovers over it.

Change Hyperlink Color with Internal Style Sheet.

Adding CSS to a HTML Document without Referencing an External Style Sheet

If you need to change colors on one page only, this code can be used. It can become a big job if you use this code to create many pages, then discover that you need to change the color again.

If you intend to build many web pages, it is suggested to create an external style sheet and

link your HTML documents to that. This way, if you need to change colors again, all you have to do is change the values to just one file - the external style sheet, and changes will take effect across an unlimited number of pages.

Add the following CSS code into the **head** section of your HTML document: <head>**code**</head> and change the color values as required:

Code

```
<html>
<head>
<style type="text/css">
a:link {
COLOR: blue;
}
a:visited {
COLOR: pink;
}
a:hover {
COLOR: red;
}
a:active {
COLOR: green;
}
</style>
<headd>
<body>
<a href="http://www.goldcoastwebdesigns.com/ase/" >Link Text</a>
</body>
</html>
```

Code

```
<a style="color: #800000; text-decoration: none; background: #80FF80"
href="http://www.goldcoastwebdesigns.com/" >Link Text</a>
```

Notes

background Changes the background color of the hyperlink.
text-decoration The none value removes the underline.

CSS background-position Properties.

Styling Backgrounds

Background Image.

```
<!DOCTYPE html>
<html>
```

```
<head>
<style>
body
{
background-image:url('paper.gif');
}
</style>
</head>
<body >
<h1>Hello World!</h1>
</body>
</html>
```

Background Image size

```
<!DOCTYPE html>
<html>
<head>
<style>
body
{
background-image :url("paper.gif");
background-size:180px 160px;
background-repeat:no-repeat;
padding-top:40px;
}
</style>
</head>
<body>
<p>
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.
</p>
</body>
</html>
```

Background Center

```
<!DOCTYPE html>
<html>
<head>
<style>
body
{
background-image:url('smiley.gif');
background-repeat:no-repeat;
background-attachment:fixed;
background-position:center;
```

```
}
</style>
</head>
<body>
<p><b>Note:</b> For this to work in Firefox and Opera, the background-attachment property must be set to "fixed".</p>
</body>
</html>
```

Example for background repeat

http://www.w3schools.com/cssref/playit.asp?filename=playcss_background-repeat HYPERLINK "http://www.w3schools.com/cssref/playit.asp?filename=playcss_background-repeat&preval=no-repeat"& HYPERLINK "http://www.w3schools.com/cssref/playit.asp?filename=playcss_background-repeat&preval=no-repeat"preval=no-repeat

Different List Item Markers

```
<!DOCTYPE html>
<html>
<head>
<style>
ul.a {
    list-style-type: circle;
}

ul.b {
    list-style-type: square;
}

ol.c {
    list-style-type: upper-roman;
}

ol.d {
    list-style-type: lower-alpha;
}
</style>
</head>
<body>
<p>Example of unordered lists:</p>
<ul class="a">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>

<ul class="b">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
<p>Example of ordered lists:</p>
```

```
<ol class="c">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>
<ol class="d">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>
</body>
</html>
```

Border Radius Property

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
    border: 2px solid #a1a1a1;
    padding: 10px 40px;
    background: #dddddd;
    width: 300px;
    border-radius: 25px;
}
</style>
</head>
<body>

<div>The border-radius property allows you to add rounded corners to elements.</div>

</body>
</html>
```

Text Formatting

```
p {
    letter-spacing: 0.5em;
    word-spacing: 2em;
    line-height: 1.5;
    text-align: center;
    font-style: italic;
    text-transform: uppercase;


}
```

Positioning Elements with CSS

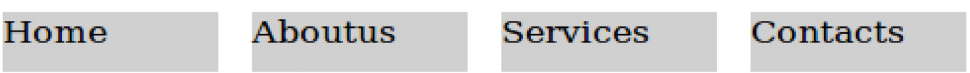
```
<!DOCTYPE html>
<html>
<head>
<style>
.rel {
border: 1px solid green;
position: relative;
top: 200px;
}
</style>
</head>
<body>

<P class="border1">Paragraph One<P>
<P class="rel">Paragraph Two<P>
<P class="border1">Paragraph Three<P>

</body>
</html>
```

Display Inline property

Example :

```
<style>

        #display1 li
        {
                display: inline-block;
                width: 100px;
                height: 30px;
                background: #ccc;
        }
</style>

<ul id="display1">
        <li>Home</li>
        <li>Aboutus</li>
        <li>Services</li>
         <li>Contacts</li>
</ul>
```

Output:



CSS: Text over Image

HTML Coding

```
<div class="image">

  <img alt="" src="http://www.kavoir.com/img/text-over-image.jpg" />
  <div class="text">
    <p>This lovely image gladly demonstrates how I (this text) can be flying over her face.</p>
    <p>It's very kind of her but don't you agree that this place needs a camp fire and a little love?</p>
</div>
</div>
```

CSS Coding

```
.image {
position:relative;
}
 .text {
position:absolute;
top:10px; /* in conjunction with left property, decides the text position */
left:10px;
width:300px; /* optional, though better have one */
}
```

Few Css Properties list

text-decoration

This states whether the text has got a line running under, over, or through it.

- `text-decoration: underline`, does what you would expect.
- `text-decoration: overline` places a line above the text.
- `text-decoration: line-through` puts a line through the text (“strike-through”).

This property is usually used to decorate links and you can specify no underline with `text-decoration: none`.

zooming:

.div { transition: [transition-property] [transition-duration] [transition-timing-function] [transition-delay]; }

Example:

```
div {
        transition: all 2s ease;
        border: 5px solid red;
}
```

```
div:hover {
        border: 5px solid green;
}
```
- See more at: http://www.corelangs.com/css/box/zoom.html#sthash.ARblieeE.dpuf

color:orange;
background-color:#d0e4fe;
font-size : 12pt
background-repeat: repeat | repeat-x | repeat-y | no-repeat
background-position:[[top | center | bottom] || [left | center | right]]
background-image:url("images/back40.gif");
font-family:"verdana";
font-weight:Bold;
font-size:14px;
word-spacing: 0.4em
letter-spacing: 0.2em
text-align: [[left | right | justify |right]]
margin-top: <length> | <percentage> | auto]
margin-right: <length> | <percentage> | auto]
margin-bottom: <length> | <percentage> | auto]
margin-left: <length> | <percentage> | auto]
margin: [<length> | <percentage> | auto]
border-color: [black | red]
border-style: [none | dotted | dashed | solid | double]
text-decoration:[none | overline | line-through | underline | blink];
text-transform:uppercase;[capitalize | lowercase | Uppercase];

**********_______********_______**********

Design a web page using Div with CSS

Design a menu with CSS

Styling Forms with CSS

Design a Image gallery with CSS

Javascript

What is javascript?

JavaScript is a scripting language that enables web developers/designers to build more functional and interactive websites.

Common uses of JavaScript include:

- Form validation
- Popup windows
- Dynamic dropdown menus
- Displaying date/time

JavaScript usually runs on the client-side (the browser's side), as opposed to server-side (on the web server). One benefit of doing this is performance. On the client side, JavaScript is loaded into the browser and can run as soon as it is called. Without running on the client side, the page would need to refresh each time you needed a script to run.

JavaScript syntax

```
<script type="text/javascript">
document.write("JavaScript is not Java");
</script>
```

Javascript popup box

Till now we have used only one kind of output statement - document.write. Javascript has few more "interactive" output statements in the form of Pop up Boxes. There are three types of Pop up Boxes - Alert box, Confirm Box and Prompt Box. Let us take a look at the Alert Box.

Javascript Alert Box

A javascript alert box pops-up with a message and an 'OK' button. It displays an alert box with a string passed to it. For example: alert() will display an empty dialog box with just the 'OK' button. The alert("Hello world") will display a dialog box with the message, 'Hello world' and an 'OK' button.

Let us look at the example that simply displays the "Hello World" pop up box when loaded.

```
<html>
<body>
<script type="text/javascript">

alert("Hello world");
```

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

Javascript Confirm Box

The Javascript confirm box differs from a regular alert box in that, it provides two choices for the user: 'OK' and 'CANCEL' to confirm or reject the request. The confirm statement takes a string that will be displayed with the text box along with the OK and the Cancel Button. In addition it will return a value. The returned value will be true if OK button is pressed. The returned value will be false if Cancel button is pressed. We can use a variable and an if statement to determine if the 'OK' or 'CANCEL' button is clicked..

Let us look at the example that simply displays a confirm box and diplays an alert box with a message "You said :OK" if the user presses OK button. It will display an alert box with a "You said : Cancel", if the user presses Cancel Button.

```
<html>
<body>
<script type="text/javascript">
var a = confirm("Do you want continue ?");
if (a)
alert("You said: Ok");
else
alert("You said: Cancel");
</script>
</body>
</html>
```

Javascript Prompt Box

Prompt uses a text field to enter a value. It also has 'OK' and 'CANCEL' buttons.

The text input to the prompt can be stored in a variable.

Let us look at the example that simply displays a prompt box and diplays an alert box with a message. The message displayed in the alert box will depen upn the user input in the prompt box.

```
<html>
<body>
<script type="text/javascript">
var name = prompt("What is your name?", "Type your name here");
alert("Your name is: "+name)
</script>
</body>
</html>
```

Example function in Javascript

A function that does not execute when a page loads should be placed inside the *head* of

your HTML document. Creating a function is really quite easy. All you have to do is tell the browser you're making a function, give the function a name, and then write the JavaScript like normal. Below is the example alert function from the previous lesson.

Html and Javascript code.
>

In the previous lesso

```
<html>
<head>
<script type="text/javascript">

function popup()
{
alert("Hello World");
}

</script>
</head>
<body>
<script>
popup();
</script>
</body>
</html>
```

n, we used an event handler to trigger off a call to our function. There are 18 event handlers that you can use to link your HTML elements to a piece of JavaScript.When you write a JavaScript function, you will need to determine when it will run. Often, this will be when a user does something like click or hover over something, submit a form, double clicks on something etc.

These are examples of events.

Using JavaScript, you can respond to an event using event handlers. You can attach an event handler to the HTML element for which you want to respond to when a specific event occurs.
For example, you could attach JavaScript's onMouseover event handler to a button and specify some JavaScript to run whenever this event occurs against that button.
The HTML 4 specification refers to these as intrinsic events and defines 18 as listed below:

Event Handler	**Event that it handles**
onBlur	User has left the focus of the object. For example, they clicked away from a text field that was previously selected.
onChange	User has changed the object, then attempts to leave that field (i.e. clicks elsewhere).
onClick	User clicked on the object.
onDblClick	User clicked twice on the object.
onFocus	User brought the focus to the object (i.e. clicked on it/tabbed to it)
onKeydown	A key was pressed over an element.
onKeyup	A key was released over an element.
onKeypress	A key was pressed over an element then released.
onLoad	The object has loaded.
onMousedown	The cursor moved over the object and mouse/pointing device was

	pressed down.
onMouseup	The mouse/pointing device was released after being pressed down.
onMouseover	The cursor moved over the object (i.e. user hovers the mouse over the object).
onMousemove	The cursor moved while hovering over an object.
onMouseout	The cursor moved off the object
onReset	User has reset a form.
onSelect	User selected some or all of the contents of the object. For example, the user selected some text within a text field.
onSubmit	User submitted a form.
onUnload	User left the window (i.e. user closes the browser window).

Example for onblur event

```
<!DOCTYPE html>
<html>
<body>

Enter your name: <input type="text" id="fname" onblur="myFunction()">

<p>When you leave the input field, a function is triggered which transforms the input text to upper case.</p>

<script>
function myFunction() {
    var x = document.getElementById("fname");
    x.value = x.value.toUpperCase();
}
</script>

</body>
</html>
```

Onkeypress Event

```
<!DOCTYPE html>
<html>
<head>
<script>
function my()
{
alert("You pressed a key inside the input field");
}
</script>
</head>
<body>
<p>A function is triggered when the user is pressing a key in the input field.</p>
<input type="text" onkeypress="my()">
</body>
</html>
```

Onmouseover and Onmouseout

```
<html>
<head>
<style>
div
{
border:1px solid black;
}
</style>
<script type="text/javascript">
function over()
{
alert("Mouse Over");
}
function out()
{
   alert("Mouse Out");
}
</script>
</head>
<body>
<div onmouseover="over()" onmouseout="out()">
<h2> This is inside the division </h2>
</div>
</body>
</html>
```

Addition program

Through Name

```
<html>
<head>
<SCRIPT  type="text/javascript">
function add()
{
 a = parseInt(document.form1.first.value);
 b = parseInt(document.form1.second.value);
 c = (a+b);
 alert(c);
}
</script>
</head>
<body>
<form name="form1" >
Enter the First Value    <input type="text" name="first" > <br>
Enter the Second Value   <input type="text" name="second"  >
<br> <br>
```

```
<input type="button" value="Add"  onclick="add()">
<input type="RESET" value="Clear" > <br>
</form>
</body>
</html>
```

Same program using Get element by id function

```
<html>
<head> </head>
<script type="text/javascript">
function myfunction()
{
var first = parseInt(document.getElementById("textbox1").value);
var second = parseInt(document.getElementById("textbox2").value);
var answer =first+second;

var a = document.getElementById('textbox3');
a.value=answer;
 }
</script>
 <body>
Value 1:<input type="text" id="textbox1" /> <br>
 Value 2:<input type="text"  id="textbox2" /> <br>
<input type="submit" name="button" id="button1" onclick="myfunction()" value="==" />
<br/>
Your answer is:--
<input type="text" name="textbox3" id="textbox3" readonly="true"/>
</body>
</html>
```

Page Redirection

```
<html>
  <head>

    <script type="text/javascript">

        function Redirect()
        {
          window.location="http://www.tutorialspoint.com";
        }

    </script>

  </head>

  <body>
    <p>Click the following button, you will be redirected to home page.</p>

    <form>
      <input type="button" value="Redirect Me" onclick="Redirect();" />
    </form>

  </body>
</html>
```

Displaying the Current Date

Typing this code...

```
var a = new Date()
var day = a.getDate()
var month = a.getMonth() + 1



var year = a.getFullYear()
document.write("<b>" + day + "/" + month + "/" + year + "</b>")
```

Notice that we added 1 to the *month* variable to correct the problem with January being 0 and December being 11. After adding 1, January will be 1, and December will be 12.

Displaying the Current Time

```
var b = new Date()
var hours = b.getHours()
var minutes = b.getMinutes()
if (minutes < 10)
minutes = "0" + minutes
document.write("<b>" + hours + ":" + minutes + " " + "</b>")
```

Basic innerHTML Example

Here's a basic example to demonstrate how innerHTML works.
This code includes two functions and two buttons. Each function displays a different message and each button triggers a different function.

In the functions, the getElementById refers to the HTML element by using its ID. We give the HTML element an ID of "myText" using id="myText".
So in the first function for example, you can see that

document.getElementById('myText').innerHTML = 'Thanks!'; is setting the innerHTML of the "myText" element to "Thanks!".

Code:

```
<script type="text/javascript">
function Msg1()
{
  document.getElementById('myText').innerHTML = 'Thanks!';
}
function Msg2()
{
  document.getElementById('myText').innerHTML = 'Try message 1 again...';
}
</script>
<input type="button" onclick="Msg1()" value="Show Message 1" />
<input type="button"  onclick="Msg2()" value="Show Message 2" />
<p id="myText"></p>
```

JavaScript Comparison and Logical Operators

Comparison and Logical operators are used to test for *true* or *false*.

Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values.
Given that **x=5**, the table below explains the comparison operators:

Operator	Description	Example
==	is equal to (value) Coparison Operator	x==8 is false x==5 is true
===	is exactly equal to (value and type) Identical Operator	x===5 is true x==="5"is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

How Can it be Used

Comparison operators can be used in conditional statements to compare values and take action depending on the result:

```
if (age<18) x="Too young";
```

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<script type="text/javascript">
var a=10;
var b = "10";
if(a===b)
{
document.write("Both Are Same");
}
else
{
document.write("Both are not same");
}
</script>
</body>
</html>
```

Logical Operators

Logical operators are used to determine the logic between variables or values.
Given that **x=6 and y=3**, the table below explains the logical operators:

Operator	Description	Example
&&	and	(x <10 &&y >1) is true
||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

Conditional Operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

Syntax

variablename=(*condition*)?*value1*:*value2*

Example

If the variable *age* is a value below 18, the value of the variable *voteable* will be "Too young, otherwise the value of *voteable* will be "Old enough":
voteable=(age<21)?"Too young":"Old enough";

Exception Handling

JavaScript Try...Catch Statement

The try...catch statement allows you to test a block of code for errors.

When browsing Web pages on the internet, we all have seen a JavaScript alert box telling us there is a runtime error and asking "Do you wish to debug?". Error message like this may be useful for developers but not for users. When users see errors, they often leave the Web page.

This chapter will teach you how to catch and handle JavaScript error messages, so you don't lose your audience.

The try...catch Statement

The try...catch statement allows you to test a block of code for errors. The try block contains the code to be run, and the catch block contains the code to be executed if an error occurs.

Syntax

```
try
 {
 //Run some code here
 }
catch(err)
```

```
{
//Handle errors here
 }
```
Note that try...catch is written in lowercase letters. Using uppercase letters will generate a JavaScript error!

Examples

The example below is supposed to alert "Welcome guest!" when the button is clicked. However, there's a typo in the message() function. alert() is misspelled as adddlert(). A JavaScript error occurs. The catch block catches the error and executes a custom code to handle it. The code displays a custom error message informing the user what happened:

Example

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
var txt="";
function message()
{
try
 {
 alert("Welcome guest!");
 }
catch(err)
 {
 txt="There was an error on this page.\n\n";
 txt+="Error description: " + err.message + "\n\n";
 txt+="Click OK to continue.\n\n";
 alert(txt);
 }
}
</script>
</head>

<body>
<input type="button" value="View message" onclick="message()" />
</body>
</html>
```

The next example uses a confirm box to display a custom message telling users they can click OK to continue viewing the page or click Cancel to go to the homepage. If the confirm method returns false, the user clicked Cancel, and the code redirects the user. If the confirm method returns true, the code does nothing:

Example

```
<!DOCTYPE html>
<html>
<head>
```

```
<script type="text/javascript">
var txt="";

function message()
{
try
 {
 adddlert("Welcome guest!");
 }
catch(err)
 {
 txt="There was an error on this page.\n\n";
 txt+="Click OK to continue viewing this page,\n";
 txt+="or Cancel to return to the home page.\n\n";
 if(!confirm(txt))
 {
 document.location.href="http://www.w3schools.com/";
 }
 }
}
</script>
</head>

<body>
<input type="button" value="View message" onclick="message()" />
</body>

</html>
```

The throw Statement

The throw statement can be used together with the try...catch statement, to create an exception for the error. Learn about the throw statement in the next chapter.

JavaScript Throw Statement.

The throw statement allows you to create an exception.

The Throw Statement

The throw statement allows you to create an exception. If you use this statement together with the try...catch statement, you can control program flow and generate accurate error messages.

Syntax

throw *exception*

The exception can be a string, integer, Boolean or an object.

Note that *throw* is written in lowercase letters. Using uppercase letters will generate a JavaScript error!

Example

The example below determines the value of a variable called x. If the value of x is higher than 10, lower than 5, or not a number, we are going to throw an error. The error is then caught by the catch argument and the proper error message is displayed:

Example

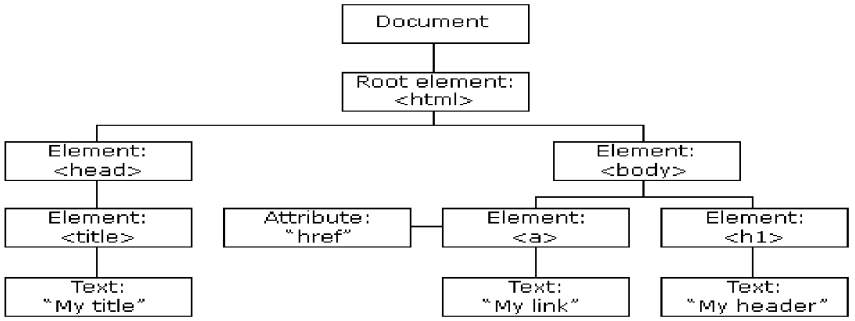
```
<!DOCTYPE html>
<html>
<body>
<script type="text/javascript">
var x=prompt("Enter a number between 5 and 10:","");
try
 {
 if(x>10)
 {
 throw "Err1";
 }
 else if(x<5)
 {
 throw "Err2";
 }
 else if(isNaN(x))
 {
 throw "Err3";
 }
 }

catch(err)
 {
 if(err=="Err1")
 {
 document.write("Error! The value is too high.");
 }
 if(err=="Err2")
 {
 document.write("Error! The value is too low.");
 }
 if(err=="Err3")
 {
 document.write("Error! The value is not a number.");
 }
 }
</script>
</body>
</html>
```

Java Script DOM

The HTML DOM (Document Object Model)

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

The **HTML DOM** model is constructed as a tree of **Objects**:



With a programmable object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can react to all the events in the page

Finding HTML Elements

Often, with JavaScript, you want to manipulate HTML elements.

To do so, you have to find the elements first. There are a couple of ways to do this:

- Finding HTML elements by id
- Finding HTML elements by tag name

Finding HTML Elements by Id

The easiest way to find HTML elements in the DOM, is by using the element id.

This example finds the element with id="intro":

```
<!DOCTYPE html>

<html>

<body>

<p id="intro">Hello World!</p>

<script>

x=document.getElementById("intro").innerHTML;

document.write("<p>The text from the intro paragraph: " + x+ "</p>");

</script>

</body>

</html>
```

Finding HTML Elements by Tag Name

This example finds the element with id="main", and then finds all <p> elements inside "main":

```
<html>
<body>
<div id="main">
<p id="intro">The DOM is very useful.</p>
<p>This example demonstrates the <b>getElementsByTagName</b> method</p>
<h5>Hello World</h5>
<pre>Where are you</pre>
</div>
<script>
var x=document.getElementById("main");
var y=x.getElementsByTagName("p");
document.write(y[0].innerHTML);
</script>
</body>
</html>
```

Changing HTML Content

The easiest way to modify the content of an HTML element is by using the **innerHTML**

property.

To change the content of an HTML element, use this syntax:

```
<!DOCTYPE html>
<html>
<body>
<p id="p1">Hello World!</p>
<script>
document.getElementById("p1").innerHTML="New text!";
</script>
<p>The paragraph above was changed by a script.</p>
</body>
</html>
```

Changing an HTML Attribute

To change the attribute of an HTML element, use this syntax:

```
<!DOCTYPE html>
<html>
<body>
<img id="image" src="smiley.gif" width="160" height="120">
<script>
document.getElementById("image").src="pic_mountain.jpg";
</script>
<p>The original image was smiley.gif, but the script changed it to landscape.jpg</p>
</body>
</html>
```

Changing HTML Style

To change the style of an HTML element, use this syntax:

The following example changes the style of a <p> element:

```
<!DOCTYPE html>

<html>

<body>

<p id="p1">Hello World!</p>

<p id="p2">Hello World!</p>

<script>

document.getElementById("d1").style.border="1px solid black";

document.getElementById("p2").style.color="blue";
```

```
document.getElementById("p2").style.fontFamily="Arial";

document.getElementById("p2").style.fontSize="larger";

</script>

<p>The paragraph above was changed by a script.</p>

</body>

</html>
```

DOM USING JAVASCRIPT EVENTS:

Reacting to Events

A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element.

To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute:

```
<!DOCTYPE html>
<html>
<body>

<h1 onclick="this.innerHTML='Ooops!' ">Click on this text!</h1>

</body>
</html>
```

HTML Event Attributes

To assign events to HTML elements you can use event attributes.

```
<!DOCTYPE html>
<html>
<body>
<button onclick="displayDate()">Try it</button>
<script>
function displayDate()
{
document.getElementById("demo").innerHTML=Date();
}
</script>
<p id="demo"></p>
</body>
</html>
```

The onload and onunload Events

The onload and onunload events are triggered when the user enters or leaves the page.

The onload event can be used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

The onload and onunload events can be used to deal with cookies.

Example ONE

```
<!DOCTYPE html>
<html>
<body onload="checkCookies()">

<script>
function checkCookies()
{
if (navigator.cookieEnabled==true)
	{
	alert("Cookies are enabled")
	}
else
	{
	alert("Cookies are not enabled")
	}
}
</script>

<p>An alert box should tell you if your browser has enabled cookies or not.</p>
</body>
</html>
```

Example Two

```
<!DOCTYPE html>
<html>
<body>
<div id="example"></div>

<script>

txt = "<p>Browser CodeName: " + navigator.appCodeName + "</p>";
txt+= "<p>Browser Name: " + navigator.appName + "</p>";
txt+= "<p>Browser Version: " + navigator.appVersion + "</p>";
txt+= "<p>Cookies Enabled: " + navigator.cookieEnabled + "</p>";
txt+= "<p>Browser Language: " + navigator.language + "</p>";
txt+= "<p>Browser Online: " + navigator.onLine + "</p>";
txt+= "<p>Platform: " + navigator.platform + "</p>";
txt+= "<p>User-agent header: " + navigator.userAgent + "</p>";


document.getElementById("example").innerHTML=txt;

</script>
```

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

The onchange Event

The onchange event are often used in combination with validation of input fields.

Below is an example of how to use the onchange. The upperCase() function will be called when a user changes the content of an input field.

```
<!DOCTYPE html>
<html>
<body>
<div id="example"></div>

<script>
function val()
{

x=document.getElementById("menu").value;
alert(x);
}


</script>

<select id="menu" onchange="val()">
<option value="0">select</option>
<option>India</option>
<option>USA</option>
<option>UK</option>
</select><br><br>
<input type="button" value="click" onclick="val()">
</body>
</html>
```

Creating New HTML Elements

To add a new element to the HTML DOM, you must create the element (element node) first, and then append it to an existing element.

```
<!DOCTYPE html>
<html>
<body>

<div id="d1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>

</div>

<script>
```

```
var para=document.createElement("p");
var node=document.createTextNode("This is new.");
para.appendChild(node);

var element=document.getElementById("d1");
element.appendChild(para);
</script>

</body>
</html>
```

Removing Existing HTML Elements

To remove an HTML element, you must know the parent of the element:

```
<!DOCTYPE html>
<html>
<body>
<div id="d1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
var parent=document.getElementById("d1");
var child=document.getElementById("p1");
parent.removeChild(child);
</script>

</body>
</html>
```

Javascript Exercise

1.Normal validation.
2.Mobile number validation
3.Email id validation
4.Password validation.
5.Showign multiple images in a div like ebay for product.
6.Get the value from a textbox and display in a paragraph element
7.Calculating values using parseint function in javascript
8.Show goodmorning,goodafternoon,goodevening and goodnight message as per system time
9.Slide show in javascript
10.autorefresh for time and seconds
11.Page redirection in javascript
12.Word count, character count word search
13.dropdown box
14. Pop up window disabling Background
15. Pickzy Image task
16. Show password text and hide password text

http://www.htmliseasy.com/javascript/sample_20-02.html

<!DOCTYPE html>

```
<html>
<head>
<script>
function open_win()
{
window.open("http://www.w3schools.com");
}
</script>
</head>

<body>
<form>
<input type="button" value="Open Window" onclick="open_win()">
</form>
</body>
</html>
```

Events

Check this link for Text Decoration
http://www.w3schools.com/cssref/tryit.asp?filename=trycss_text-decoration

Exercise

CSS background Exercise

1.Setting a background color

2.Setting a background color for a table

3.Setting a background image for a page

4.Setting a background image for a table

5.Setting a background image that does not move

6.Setting a background image that repeats horizontally

7.Setting a background image that repeats vertically

8.Setting a background image that repeats for the whole page

9.Setting a background image that is center positioned

10.Setting a background image that is center positioned and does not move

11.Setting all the background features with one declaration

CSS text Exercise

1.Setting text color

2.Setting text background color

3.Setting text color and background color

4.Setting text decoration

5.Setting text direction

6.Indenting text

7.Aligning text

8.Setting text spacing

9.Setting text capitalization

10.Setting the spacing between words

11.Wrapping text

CSS list Exercise

1.Unordered lists with different bullets

2.Ordered lists with different bullets

3.Using an image for bullets in a list

4.List placement

5.Setting all the list features with one declaration

CSS border Exercise

1.Setting the style of the left border

2.Setting the width of the left border

3.Setting the color of the left border

4.Setting all the properties of the left border

5.Setting the style of the right border

6.Setting the width of the right border

7.Setting the color of the right border

8.Setting all the properties of the right border

9.Setting the style of the top border

10.Setting the width of the top border

11.Setting the color of the top border

12.Setting all the properties of the top border

13.Setting the style of the bottom border

14.Setting the width of the bottom border

15.Setting the color of the bottom border

16.Setting all the properties of the bottom border

17.Setting the style of the entire border

18.Setting the width of the entire border

19Setting the color of the entire border

20.Setting all the border properties in one declaration

CSS margin Exercise

1.Setting the left margin

2.Setting the right margin

3.Setting the top margin

4.Setting the bottom margin

5.All the margin properties in one declaration

CSS padding Exercise

1.Setting the left padding

2.Setting the right padding

3.Setting the top padding

4.Setting the bottom padding

5.All the padding properties in one declaration

CSS outline Exercise

1.
Setting the style of an outline

2.Setting the width of an outline

3.Setting the color of an outline

4.Outline and border together

5.All the outline properties in one declaration

CSS table Exercise

1.Setting a table border

2.Setting the space between border cells

3.Setting the position of a table caption

4.Setting the visibility of empty cells

5.Setting the method of display for table cells, rows, and columns

CSS dimension Exercise

1.Setting the height of an element

2.Setting the minimum height of an element

3.Setting the maximum height of an element

4.Setting the width of an element

5.Setting the minimum width of an element

6.Setting the maximum width of an element

7.Setting the space between lines