

Full Stack Module-I

Manual V8.3

ANUDIP FOUNDATION



ICONS AND THEIR MEANING



HINTS:
Get ready for helpful insites on difficult topics and questions.



STUDENTS:
This icon symbolize important instreutions and guides for the students.



TEACHERS/TRAINERS:
This icon symbolize important instreutions and guides for the trainers.

Lesson No	Lesson Name	Practical Duration (Minutes)	Theory Duration (Minutes)	Page No
1	HTML5 Basics	60	60	03
2	Tables	60	60	09
3	List	60	60	12
4	Working with Links	60	60	15
5	Image Handling	60	60	19
6	Frames	60	60	22
7	HTML Forms for User Input	60	60	25
8	New Form Elements	60	60	29
9	HTML5 – Client-Side Storage	60	60	36
10	The Offline Access & Drag and Drop API	60	60	40
11	HTML5 APIs	60	60	49
12	Introduction to CSS 3	60	60	53
13	Selectors, Pseudo Classes, and Pseudo Elements	60	60	60
14	Fonts and Text Effect	60	60	64
15	Color, Gradients, Background Images, and Masks	60	60	88

16	Border and Box Effects	60	60	92
17	Working with Colors	60	60	101
18	Layout	60	60	106
19	Media Queries	60	60	115
20	Implementing CSS3 in the "Real World"	60	60	118
21	Transforms, Transitions, and Animations	60	60	124
22	Getting Started with Bootstrap	60	60	130
23	Material design	60	60	154

Total Duration: __Hours

Lesson 1: HTML5 Basics (120 minutes)

Objective: After completing this lesson you will be able to : <ul style="list-style-type: none">• Understand the structure of an HTML page.• Learn to apply physical/logical character effects.• Learn to manage document spacing.	Materials Required: <ul style="list-style-type: none">• Computer With Windows XP and above• Stable Internet connection
Self- Learning Duration: 60 minutes	Practical Duration: 60 minutes
Total Duration: 120 minutes	

HTML5 Basics

HTML5 is the latest addition to the ever popular Hyper Text Markup Language. Well, the word "addition" can be a slight cliché to the part. HTML5 can be best described as an updated version of HTML. It's more featured rich and better in terms of coding or implementation part.

To have a better idea of HTML5, you need to first have a clear idea of the basics of HTML. You need to understand the structure of an HTML page, the application of logical and physical character effects, and managing of document spacing. Apart from that, you also need to understand the procedures that are required to work with tables, links, and images, frames, etc. in HTML and how to properly handle the scenario.

Let's move on:

The structure of an HTML page

Can you say: why we require HTML? What exactly is this HTML?

HTML or Hyper Text Markup Language is a markup language used for developing web pages.

Markup language is a special type of computer language that defines elements in a document through the use of tags. In simpler term, the markup language is created using human readable words and not those regular programming codes and syntaxes. HTML and XML are the most popular markup languages available till date.

Properties of HTML

- HTML primarily describes the webpage structure
- HTML comprises of a series of elements
- HTML elements instructs the browser on ways to display the content
- HTML elements are primarily represented through tags
- HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
- Browsers use HTML tags to render the webpage's content

A sample HTML structure

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Observe the above sample HTML. Let me define the tags accordingly:

- <!DOCTYPE html> is a declaration tag that indicates that the document will be HTML5
- <html> is basically the root element of this and every HTML page
- <head> is the element where the meta information related to the document will be stored
- <title> is the element used for specifying the title to be used for the document
- <body> is the element that feature the entire content of the page to be visible before the audience
- <h1> element indicates large heading
- <p> element indicates paragraph

Logical and Physical Character Tags in HTML

HTML comprises of both logical and physical character tags. Let's have a clear understanding of both these variants:

Logical Tags

The primary functionality of these tags is to define the meaning of the enclosed text to the browser.

Example of a logical tag:

```
<strong> </strong>
```

Once you place text within these tags, it clearly indicates to the browser that the text being enclosed possess greater importance. These days, search engines focus on these tags to understand the importance of the pages or even to learn the basic content of the page.

Physical Tags

These tags are used to provide specific set of instructions to the browser related to a certain way the text enclosed, to be displayed.

Example of a physical tag:

``

This tag instructs the enclosed text to be displayed in bold in the browser.

Physical tags can be described as more straightforward approach when compared to logical tags. The former ones came to the scenario to add a pinch of stylization to the HTML pages. During those early days, Style Sheets were not introduced to HTML. But after the introduction of Style Sheets, most notably the Cascading ones, use of physical tags became less obvious in web page development through HTML.

Managing Document Spacing

Document spacing is a major part of web development. Without proper space management, the document and other content of the webpage will look cumbersome and haphazard. A web designer, hence, must learn how to manage space in a webpage document.

HTML comes with a property called WHITESPACE COLLAPSE. Whether you include a single space or a hundred spaces in the HTML code, the browser will collapse all those spaces to a single space within the specific part of the document.

But as a web designer, do you want such a situation to dictate you on terms?

Definitely not!

There is a way to add whitespaces to your HTML document to manage document spacing. Using CSS, it can be done.

Cascading Style Sheets are simple web designing languages used for the formatting of a webpage and make it a lot more presentable. Adding spaces to a document becomes easier with the help of a cascading style sheet. There are padding and margin properties in CSS for adding space around the elements where the document is being enclosed.

```
p {  
  text-indent: 3em;  
}
```

If you are not in the mood to use CSS, there is another way to handle document spacing.

Say for example, you want to include an additional space or two to the text. You can opt for non-breaking space. The best part about this character is that it will never collapse within the browser but at the same time, will act similar to a standard space character.

The following example highlights the use of 5 spaces within a text:

The coding:

This text has five extra spaces inside
it

The output:

This text has five extra spaces inside it

While this can be an option, the non-breaking spaces elements do add up to extra amount of spacing to the text. It is also a lengthier procedure and requires additional input to the coding part.

On the other hand, using a style sheet to handle the styling and spacing part is a much easier option without going for extra code typing scenarios. Changing style for the entire site becomes easier since the update needs to be done within the style sheet only.

Reviewing the chapter

- HTML stands for Hyper Text Markup Language.
- HTML is the most popular markup language to develop web pages.
- HTML comprises of a series of elements that are represented through tags.
- Logical character tags are used to define the meaning of the enclosed text to the browser.
- Physical character tags are used for providing specific set of instructions to the browser related to a certain way the text enclosed, to be displayed.
- Cascading Style Sheets are simple web designing languages used for the formatting of a webpage and make it a lot more presentable. Adding spaces to a document becomes easier with the help of a cascading style sheet.

Testing your skills

1. HTML stands for

a) Hyper Text Marking Language, b) Hyper Tonic Markup Listing, c) Hyper Text Markup Language, d) Hyper Textual Markup Language

2. _____ is a special type of computer language that defines elements in a document through the use of tags.

a) Marking Language, b) Markup Language, c) Both (a) and (b), d) None of the above

3. These tags are used to define the meaning of the enclosed text to the browser

a) Logical Tags, b) Physical Tags, c) Temporary Tags, d) Enclosure Tags

4. The browser will collapse all those spaces to a single space within the specific part of the document. This type of a situation in HTML is known as

a) Sudden Collapse, b) Whitespace Collapse, c) Synchronize, d) Space Merge

5. _____ tag stores the meta information related to the document

a) <h1>, b) <body>, c) <title>, d) <head>

Lesson 2: Tables (120 minutes)

Objective: After completing this lesson you will be able to :

- Understand the structure of an HTML table.
- Learn to control table format like cell spanning, cell spacing, border

Materials Required:

- Computer With Windows XP and above
- Stable Internet connection

Self- Learning Duration: 60 minutes

Practical Duration: 60 minutes

Total Duration: 120 minutes

Tables

Tables in HTML are used for arranging data, be it the images, text, or links, into columns and rows of cells.

The structure of a HTML table

- A <table> tag is used to define the HTML table.
- Every ROW of the Table is defined using the <tr> tag.
- Every Header of the Table is defined using the <th> tag.
- Every Cell or Data of the Table is defined using the <td> tag.
- The table headings are aligned at the center and bold, by default.

A sample example:

```
<table style="width:100%">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

Output:

Basic HTML Table

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94
John	Doe	80

Cell Spanning with HTML Tables

Observe the real-world tables and you will notice the presence of cells that stretch across more than one column or rows in that table.

If you consider the words of publishing world, such a scenario can be best referred to as CELLS STRADDLING THE RELATED ROWS AND COLUMNS. Now, it is quite common for the straddling cells to contain information much relevant to the straddled columns and rows, like that of a shared data or common heading.

In common HTML terms, such straddling cells are usually referred to as Cell Spanning. Keep in mind, spanning cells across the columns and rows can be created using the attributed mentioned in the `<td>` and `<th>` tags.

The **colspan attribute** creates a cell that spans two or more columns.

The **rowspan attribute** creates a cell that spans two or more rows.

Both these attributes can be used together to create a cell that can span columns and rows simultaneously.

Cell Spacing

The cell spacing attribute in HTML defines the space between cells (in pixels). This must not be confused with the Cell Padding attribute that specifies the space between the content and wall of the cell.

An example

```
<table cellpadding="10">
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
</table>
```

Output:

Table without cellpadding:

Month	Savings
January	\$100

Table with cellpadding:

Month	Savings
January	\$100

Note: The cellpadding attribute is not supported in HTML5.

It must be noted that Cell Spacing is not supported in HTML5.

Border

The border attribute basically specifies whether a border be displayed around the table cells or not.

`<table border="1">` denotes the presence of border around the table cells.

`<table border="0">` denotes the absence of border around the table cells.

Reviewing the chapter

- Tables in HTML are used for arranging data, be it the images, text, or links, into columns and rows of cells.
- The table headings are aligned at the center and bold, by default.
- Cell spanning basically indicates conjunction of different cells into one unit to create a larger cell structure. Cells can be joined either vertically, horizontally, or even in both ways.
- Cell spacing attribute in HTML defines the space between cells (in pixels).
- Cell Padding attribute specifies the space between the content and wall of the cell.
- Border attribute is used to specify the presence of border around the cells of the table. If the attribute is zero, no border will be displayed. If it's one, border will be displayed.

Testing your skills

1. Data in tables can be

a) Image, b) Text, c) Both (a) and (b), d) None of the above

2. Border attribute used for specifying no border around the cells

a) Zero, b) Null, c) None, d) Vacant

3. Straddling of cells is commonly referred to as

a) Cell spacing, b) Cell Spanning, c) Cell Padding, d) None of the above

4. Cells in a table can be joined in which of the following ways:

a) Horizontally, b) Vertically, c) Both (a) and (b), d) Only (a)

5. What tag is used for defining the Headers of a Table?

a) <th>, b <ht>, c) <td>, d) <tr>

Lesson 3: List (120 minutes)

Objective: After completing this lesson you will be able to : <ul style="list-style-type: none"> • Have clear idea about numbered list • Create bulleted list • Understand the Type attribute 	Materials Required: <ul style="list-style-type: none"> • Computer With Windows XP and above • Stable Internet connection
Self- Learning Duration: 60 minutes	Practical Duration: 60 minutes
Total Duration: 120 minutes	

List

There are three ways through which web developers can specify list in HTML. These are:

- *Unordered list*

In this type, items will be listed in plain bullets.

- *Ordered list*

Here, items will be listed using different schemes of numbers.

- *Definition list* <dl>

When using Definition List, the items will be arranged exactly in the same way as you find them in a dictionary.

Always keep in mind: every list must feature one or even more elements.

Bulleted List

There can be 4 types of bulleted list. These are disc, circle, square, and none. These lists can be used for both ordered as well as unordered list structure.

The Type Attribute

The Type Attribute is used to specify the type of element being used.

- For button elements, this attribute will specify the button type.
- For input elements, this attribute will specify the <input> element type to display.
- For embed, link, object, script, source, and style elements, the type attribute specifies the Internet media/MIME type.

The syntax:

```
<ul type = "square">
```

```
<ul type = "disc">
```

```
<ul type = "circle">
```

Sample example where we are using the <ul type = "square">:

```
<!DOCTYPE html>
<html>

  <head>
    <title>HTML Unordered List</title>
  </head>

  <body>
    <ul type = "square">
      <li>Beetroot</li>
      <li>Ginger</li>
      <li>Potato</li>
      <li>Radish</li>
    </ul>
  </body>

</html>
```

The outcome of this code:

- Beetroot
- Ginger
- Potato
- Radish

Similarly, when using the <ul type = "disc">, the outcome will be disc shaped bullets (in comparison to what you got in the previous outcome, the square shaped ones)

Sample example for <dl type>

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

Output:

A Description List

Coffee

- black hot drink

Milk

- white cold drink

Sample example for <ol type>

```
<ol type="1">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Output:

Ordered List with Numbers

1. Coffee
2. Tea
3. Milk

Reviewing the chapter

- When using Definition List, the items will be arranged exactly in the same way as you find them in a dictionary.
- The Type Attribute is used to specify the type of element being used.

Testing your skills

1. The syntax <ul type = ____> is used for disc shaped bullets.

a) Disc, b) Circle, c) Round, d) Hole

2. Which attribute is used for specifying the type of element being used?

a) File Type, b) Input, c) Element, d) Type

3. There are _ types of bulleted lists.

a) 2, b) 4, c) 3, d) 5

4. The syntax of Ordered List is

a) , b) , c) <il>, d) <dl>

5) Which among these is not a bulleted list type?

a) circle, b) square, c) triangle, d) none

Lesson 4: Working with Links (120 minutes)

Objective: After completing this lesson you will be able to : <ul style="list-style-type: none">• Understand the working of hyperlinks in web pages.• Learn to create hyperlinks in web pages.• Have better understanding of the TARGET Attribute in link	Materials Required: <ul style="list-style-type: none">• Computer With Windows XP and above• Stable Internet connection
Self- Learning Duration: 60 minutes	Practical Duration: 60 minutes
Total Duration: 120 minutes	

Linking in a web page (Hyperlinks)

Links are part and parcel of every web page. A website may feature numerous links. Clicking these links may redirect the user to different parts of the same page or other pages. These links are commonly referred to as Hyperlinks.

These hyperlinks can be used on words, images, symbols, or even phrases. A user, simply by clicking on the linked element can get redirected to another part of the page, website, or to an entirely different site.

Linking Documents

A link is specified using HTML tag <a>. The tag <a> is popularly referred to as Anchor tag. Anything inserted within the <a> and is considered part of the link. Once the user clicks that link, it will be redirected to the linked document.

A simple syntax for the use of <a> tag:

```
<a href = "Document URL" ... attributes-list>Link Text</a>
```

Let's have a look at a sample example:

```
<head>
  <title>Hyperlink Example</title>
</head>

<body>
  <p>Click following link</p>
  <a href = "https://www.wikipedia.org" target = "_self">Wikipedia</a>
</body>
</html>
```

This will produce the following result, where you can click on the link (the blue colored underlined element) generated to reach to the home page of Wikipedia:

Click following link

[Wikipedia](#)

Different colors in HTML links

When inserting a link, you will notice different colors in a link. What do these colors signify? Let's discuss:

By default, a link, when created, will be in blue. This also suggests that the link is yet to be visited from a particulate browser.

A link, once visited, will show the purple color.

An active link usually displays red color.

Also note that all the links will be underlined once created.

Using CSS, the colors of these links can be changed.

The example below will make things clearer than before:

```
<style>
a:link {
  color: green;
  background-color: transparent;
  text-decoration: none;
}

a:visited {
  color: pink;
  background-color: transparent;
  text-decoration: none;
}

a:hover {
  color: red;
  background-color: transparent;
  text-decoration: underline;
}

a:active {
  color: yellow;
  background-color: transparent;
  text-decoration: underline;
}
</style>
```

The Target Attribute

The Target attribute is used for specifying the location to open the linked document.

Here are the values associated with Target attributes:

- `_blank`

Opens the linked document in a new window or tab

- `_self`

Opens the linked document in the same window/tab as it was clicked (this is default)

- `_parent`

Opens the linked document in the parent frame

- `_top`

Opens the linked document in the full body of the window

- `framename`

Opens the linked document in a named frame

Let's have a look at a sample example:

```
<!DOCTYPE html>
<html>

  <head>
    <title>Hyperlink Example</title>
    <base href = "https://www.wikipedia.org/">
  </head>

  <body>
    <p>Click any of the following links</p>
    <a href = "/html/index.htm" target = "_blank">Opens in New</a> |
    <a href = "/html/index.htm" target = "_self">Opens in Self</a> |
    <a href = "/html/index.htm" target = "_parent">Opens in Parent</a> |
    <a href = "/html/index.htm" target = "_top">Opens in Body</a>
  </body>

</html>
```

This is what the page will display:

Click any of the following links

[Opens in New](#) | [Opens in Self](#) | [Opens in Parent](#) | [Opens in Body](#)

Reviewing the chapter

- A link is specified using HTML tag `<a>`. The tag `<a>` is popularly referred to as Anchor tag. Anything inserted within the `<a>` and `` is considered part of the link. Once the user clicks that link, it will be redirected to the linked document.
- These links are commonly referred to as Hyperlinks.
- By default, a link, when created, will be in blue. This also suggests that the link is yet to be visited from a particulate browser.
- A link, once visited, will show the purple color.
- An active link usually displays red color.
- Also note that all the links will be underlined once created.
- Using CSS, the colors of these links can be changed.
- The Target attribute is used for specifying the location to open the linked document.

Testing your skills

1. A link is specified using which tag?

a) `<l>`, b) `<hl>`, c) `<a>`, d) `<lt>`

2. ____ opens the linked document in a new window or tab

a) `_top`, b) `_parent`, c) `_self`, d) `_blank`

3. _____ opens the linked document in a named frame

a) `frameName`, b) `linkName`, c) `docName`, d) `fullDoc`

4. What color will an active link display?

a) blue, b) green, c) red, d) Purple

5. How many types of target attributes in HTML are present?

a) 5, b) 3, c) 2, d) 4

Lesson 5: Image Handling (120 minutes)

Objective: After completing this lesson you will be able to : <ul style="list-style-type: none"> • Understand the role of images in web pages • Learn to add images to web pages • Learn to use images as hyperlinks 	Materials Required: <ul style="list-style-type: none"> • Computer With Windows XP and above • Stable Internet connection
Self- Learning Duration: 60 minutes	Practical Duration: 60 minutes
Total Duration: 120 minutes	

Images

Images have a special role in content development. Including images in the content definitely helps in improving the overall appearance of the web page. Images help in making the content of a page more informative.

Adding Images to web pages

The tag is used for displaying image on a web page. Do keep in mind that the tag is an empty tag that only features the attributes. Here, you will not have to include the closing tags. That means, there is no need for something like to close the command.

The <src> attribute specifies the URL/web address of the image

The <alt> attribute acts as an alternative identity to the image. In case the image fails to get displayed, the <alt> attribute will showcase an alternative text for the image.

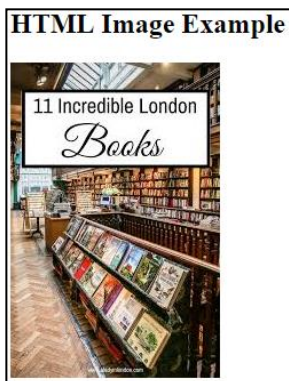
The value of the <alt> attribute should describe the image.

Let's observe an example:

```
<h2>HTML Image Example</h2>

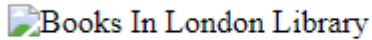
```

This will display the following outcome:



If the image is not displayed (due to some error), then the following words, "Books In London Library" will be displayed (since I have used the <alt> attribute in the coding. Have a look:

HTML Image Example



Note: I have taken this picture from the web and saved in my system with the name "books london". Now, when I have written the code with `img src = "books london.jpg"`, the outcome of the code when run in the browser was shown above. However, if the name of the image mentioned was wrong or something that way not present in the system, it would have never shown anything unless there's present the <alt> attribute within which I have already mentioned "Books In London Library". Henceforth, the above scenario appears.

Using Images as hyperlinks

To use an image as a link, put the tag inside the <a> tag. Let's observe an example to make things clearer:

```
<h2>HTML Image Example</h2>
<a href="https://www.aladyinlondon.com/2017/08/best-london-books.html">

</a>
```

Managing Image Size

Using the <style> attribute, you can easily specify and customize the height and width of the image. See below an example:

```

```

Reviewing the chapter

- Images help in making the content of a page more informative and attractive.
- The tag is used for displaying image on a web page.
- To use an image as a link, put the tag inside the <a> tag.
- Using the <style> attribute, you can easily specify and customize the height and width of the image

Testing your skills

1. _____ helps in making the content of a web page more productive, informative, and attractive
a) Image, b) graph, c) links, d) none of the above

2. The <> attribute specifies the URL for the image

a) , b) <href>, c) <src>, d) <alt>

3. If using an image link, where should the tag be placed?

a) <HREF>, b) <a>, c) <alt>, d) <src>

4. The <> tag acts as an additional identity to the image

a) <alt>, b) <add>, c) <a>, d) <ai>

5) Which attribute is used for customizing the height and width of the image?

a) <design>, b) <style>, c) <href>, <none of the above>

Lesson 6: Frames (120 minutes)

Objective: After completing this lesson you will be able to : <ul style="list-style-type: none">• Understand the need for frames in web pages.• Learn to create and work with frames.	Materials Required: <ul style="list-style-type: none">• Computer With Windows XP and above• Stable Internet connection
Self- Learning Duration: 60 minutes	Practical Duration: 60 minutes
Total Duration: 120 minutes	

Frames

Frames in HTML are used for creating multiple sections of the browser window. The separation is required to load different HTML documents in each section for better view and clarity.

Frameset refers to a collection of frames in the browser window. Take the example of a table. How are rows and columns arranged? Exactly the same process is followed by HTML to divide the frames in the window.

Pros and cons of Frames

While Frames help in creating a better clarity and reshaping the document through segregation, there are certain negative points that show some serious concern regarding the use of Frames. Smaller devices may find it difficult to display a framed page due to the resolution of their screens.

Another negative trait about using Frames is the fact that certain browsers till date not support the Frame technology and the webpage may display improperly.

Creating Frames

If you want to include a Frame in the webpage, the code must contain `<frameset>` tag and not the `<body>` tag.

The `<frameset>` tag defines the process to divide the window into frames.

The `<rows>` attribute of `<frameset>` tag defines horizontal frames.

The `<cols>` attribute defines vertical frames.

Each frame is indicated by `<frame>` tag. This instructs the HTML document that will open into the frame and so on.

Point to be noted: If you are considering HTML5, never opt for use of Frame elements. The `<frame>` tag has been deprecated in HTML5.

A sample example that creates three horizontal frames

```
<!DOCTYPE html>
<html>

  <head>
    <title>HTML Frames</title>
  </head>

  <frameset rows = "10%,80%,10%">
    <frame name = "top" src = "/html/top_frame.htm" />
    <frame name = "main" src = "/html/main_frame.htm" />
    <frame name = "bottom" src = "/html/bottom_frame.htm" />

    <noframes>
      <body>Your browser does not support frames.</body>
    </noframes>

  </frameset>

</html>
```

The <frameset> tag Attributes – Detailed List

- <cols> Specifies how many columns are contained in the frameset and the size of each column.
- <rows> Specifies how many columns are contained in the frameset and the size of each column.
- <border> Specifies the width of the border of each frame in pixels.
- <frameborder> Specifies whether a three-dimensional border should be displayed between frames.
- <framespacing> Specifies the amount of space between frames in a frameset.

The <frame> tag Attributes – Detailed List

- <src> This attribute is used to give the file name that should be loaded in the frame. Its value can be any URL.
- <name> This attribute allows you to give a name to a frame. It is used to indicate which frame a document should be loaded into.
- <frameborder> This attribute specifies whether or not the borders of that frame are shown.
- <marginwidth> This attribute allows you to specify the width of the space between the left and right of the frame's borders and the frame's content.
- <marginheight> This attribute allows you to specify the height of the space between the top and bottom of the frame's borders and its contents.
- <noresize> This attribute prevents a user from being able to resize the frame.
- <scrolling> This attribute controls the appearance of the scrollbars that appear on the frame.
- <longdesc> This attribute allows you to provide a link to another page containing a long description of the contents of the frame.

Reviewing the chapter

- Frames in HTML are used for creating multiple sections of the browser window.
- Frameset refers to a collection of frames in the browser window.
- Smaller devices may find it difficult to display a framed page due to the resolution of their screens.
- Each frame is indicated by <frame> tag. This instructs the HTML document that will open into the frame and so on.

Testing your skills

1. Which tag is used to include frame in a webpage?

a) <frameset>, b) <frame>, c) <frameborder>, d) <longdesc>

2. _____ specifies the width of the border of each frame in pixels

a) <border>, b) <frameborder>, c) <marginheight>, d) none of the above

3. A collection of frames in the browser window is known as

a) <frames> b) <frameall>, c) <frameset>, d) <longdesc>

4. This attribute specifies the amount of space between frames in a frameset

a) <space>, b) <fspace>, c) <sframe>, d) <framespacing>

5. This attribute specifies the size of a row

a) <rows>, b) <rowwidth>, c) <noresize>, d) none of the above

Lesson 7: HTML Forms For User Input (120 minutes)

Objective: After completing this lesson you will be able to :

- Understand the role of forms in web pages
- Understand various HTML elements used in forms.
- Single line text field
- Text area
- Check box
- Radio buttons
- Password fields
- Pull-down menus
- File selector dialog box

Materials Required:

- Computer With Windows XP and above
- Stable Internet connection

Self- Learning Duration: 60 minutes

Practical Duration: 60 minutes

Total Duration: 120 minutes

HTML Forms

An HTML form can be best described as a part of a document that contains controls like text fields, password fields, checkboxes, radio buttons, submit button, menus etc.

An HTML form facilitates the user to enter data. This data is then sent over the server for processing such as name, email address, password, phone number, etc.

The Role of HTML Forms

HTML Forms are primarily required for collection of data and information from the visitor visiting the site. For example, during user registration you would like to collect information such as name, email address, credit card, etc.

A form will first take input from the visitor of the site and then post it to a back-end application like CGI, ASP Script or PHP script etc. The back-end application will perform required processing on the passed data based on defined business logic inside the application.

The Different HTML Elements Used in Forms

There are various form elements available like text fields, text area fields, drop-down menus, radio buttons, checkboxes, file selector dialog boxes, pull down menus, password fields, etc.

Syntax for HTML Form

```
<form action = "Script URL" method = "GET|POST">  
  form elements like input, textarea etc.  
</form>
```

Different HTML Form Tags

- <form> tag defines an HTML form to enter inputs by the user side.
- <input> tag defines an input control.
- <textarea> tag defines a multi-line input control.
- <label> tag defines a label for an input element.
- <fieldset> tag groups the related element in a form.
- <legend> tag defines a caption for a <fieldset> element.
- <select> tag defines a drop-down list.
- <optgroup> tag defines a group of related options in a drop-down list.
- <option> tag defines an option in a drop-down list.
- <button> tag defines a clickable button.

A sample example of Form (*only text input functionality*)

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

The Output will look similar:

First name:

Last name:

An example where the form uses Radio button

```
<form>
  <input type="radio" name="gender" value="male" checked> Male<br>
  <input type="radio" name="gender" value="female"> Female<br>
  <input type="radio" name="gender" value="other"> Other
</form>
```

The Output will look similar:

☒ Male
☐ Female
☐ Other

An example where the form uses Submit button

```
<form action="/action_page.php">
  First name:<br>
  <input type="text" name="firstname" value="Mickey"><br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse"><br><br>
  <input type="submit" value="Submit">
</form>
```

The Output will look similar:

First name:

Mickey

Last name:

Mouse

Submit

The Checkbox element with output:

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

<form action="">
<input type="checkbox" name="vehicle" value="Bike">I have a bike<br>
<input type="checkbox" name="vehicle" value="Car">I have a car
</form>

</body>
</html>
```

☐ I have a bike
☐ I have a car

The dropdown element with output:

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

<select>
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="opel">Opel</option>
  <option value="audi">Audi</option>
</select>

</body>
</html>
```

Volvo ▾
Volvo
Saab
Opel
Audi

The file selector dialogue box with output:

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

<h1>Show a file-select field:</h1>

<h3>Show a file-select field which allows only one file to be chosen:</h3>
<form action="/action_page.php">
  Select a file: <input type="file" name="myFile"><br><br>
  <input type="submit">
</form>

<h3>Show a file-select field which allows multiple files:</h3>
<form action="/action_page.php">
  Select files: <input type="file" name="myFile" multiple><br><br>
  <input type="submit">
</form>
```

Show a file-select field:

Show a file-select field which allows only one file to be chosen:

Select a file: No file chosen

Show a file-select field which allows multiple files:

Select files: No file chosen

The Password field with output:

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

<h3>A demonstration of how to access a Password Field</h3>
Password: <input type="password" id="myPsw" value="psw123">

<p>Click the button to get the password of the password field.</p>
<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {
  var x = document.getElementById("myPsw").value;
  document.getElementById("demo").innerHTML = x;
}
</script>

</body>
</html>
```

A demonstration of how to access a Password FieldPassword:

Click the button to get the password of the password field.

Reviewing the chapter

- HTML Forms are primarily required for collection of data and information from the visitor visiting the site.
- A form will first take input from the visitor of the site and then post it to a back-end application like CGI, ASP Script or PHP script etc
- <form> tag defines an HTML form to enter inputs by the used side.

Testing Your Skills

1. _____ tag defines an HTML form to enter inputs by the used side

a) <form>, b) <input>, c) <label>, d) <none of the above>

2. _____ tag defines an option in a drop-down list

a) <button>, b) <select>, c) <fieldset>, d) <option>

3. Which of the following HTML elements can be included in a form?

a) radio buttons, b) checkboxes, c) file selector dialog boxes, d) all of the above

4. Which of these followings are back-end applications?

a) CGI, b) ASP Script, c) both (a) and (b), d) none of the above

5. _____ tag groups the related element in a form

a) <optgroup>, b) <legend>, c) <label>, d) <fieldset>

Lesson 8: New Form Elements (120 minutes)

Objective: After completing this lesson you will be able to :

- Understand the new HTML form elements such as date, number, range, email, search and datalist
- Offerings of HTML5
- HTML5 – Page Layout
- HTML5: Browser Support
- Enhanced Form Elements
- An HTML5 Detection Library: Modernizr
- Audio And Video
- HTML5 Canvas

Materials Required:

- Computer With Windows XP and above
- Stable Internet connection

Self- Learning Duration: 60 minutes

Practical Duration: 60 minutes

Total Duration: 120 minutes

New form elements in HTML5

With HTML5, you get to witness several New Form Elements. These elements come bundled with new features and capabilities. One of the most notable inclusions regarding these New Form Elements is modifying part of the web page dynamically.

Here are the new form elements and what you can do with them:

- **<Datalist>** is used for attaching a list of suggestions to the text input element. Once the user starts typing within the text field, several suggestions in listed form will start appearing. The user can easily make the choice while scrolling through the mouse.

Below is a sample example:

```
<label for = "txtList">Your name
  <input type = "text"
    list = "names"
    id = "txtList"/>
<datalist id = "names">
  <option value = "Andy">
  <option value = "Andrew">
  <option value = "Androcles">
</datalist>
</label>
```

- **<meter>** primarily refers to a numeric value falling within a specific range. This tag has a number of attributes to deal with. Have a look:
 - value: If you don't specify a value, the first numeric value inside the `<meter></meter>` pair becomes the value.
 - max: The maximum possible value of the item.
 - min: The minimum possible value of the item.
 - high: If the value can be defined as a range, this is the high end of the range.
 - low: If the value can be defined as a range, this is the low end of that range.
 - optimum: The optimal value of the element.

Point to be noted: The value, whether high, low, or optimum, should all be between min and max.

Note that the meter element is used to output a numeric element. Use `<input type = "range">` for numeric input within a range.

Here is code for a simple meter range:

```
<p>
  A
  <meter min = "0"
    max = "10"
    value = "7"></meter>
</p>
```

- **<output>** is used for displaying of text output. It clearly refers to a specific section of the page that can be modified by a script (usually JavaScript). A sample example is provided below:

```
<output id = "myOutput">
  This is the original value
</output>
<button onclick = "changeOutput()">
  change the output
</button>
```

Do remember that once the button is pressed, it will call for the `changeOutput()` JavaScript function. Focus on the below example to have a better understanding of the structure:

```
function changeOutput(){  
  var myOutput = document.getElementById("myOutput");  
  myOutput.value = "The value has changed";  
} // end changeOutput
```

You will notice a change in the content of the myOutput once the function runs.

Opera is the only browser that presently supports the output element. Because of this, you can consider using the innerHTML attribute of any page element in case you want to change its content. This attribute helps in changing the content dynamically through coding.

- **<progress>** indicates the total amount of task being completed through percentage marking.

JavaScript coding can be used for modification purpose. Below is a sample HTML5 code to make things clearer:

```
<p>Now destroying the world. <br />  
<p>  
  progress:  
  <progress value = "25"  
             max = "100"></progress>  
</p>
```

The present version of HTML5 features a keygen element that generates an encryption key for passing encrypted data to a server. Have a look at its parameters:

- **keytype**: Specifies the type of encryption. (rsa is standard.)
- **challenge**: A string passed along with the public key. (This is normally specified by the server)

What HTML5 offers?

With HTML5, it becomes a lot easier to handle coding for the website. The DOCTYPE declaration for HTML5 has got a tad simpler. Have a look:

```
<!DOCTYPE html>
```

In fact, you will notice the character encoding (charset) declaration also to be much simpler than the earlier versions:

<meta charset="UTF-8">

HTML5 Page Layout

A sample example to make things easier and clearer:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Title of the document</title>
</head>

<body>
Content of the document.....
</body>

</html>
```

Do keep in mind that the default character encoding in HTML5 is UTF-8.

Enhanced Form Elements

Most of the New HTML5 elements are segregated into different groups like semantic, attribute, graphic, and multimedia. Let's have a look at the elements falling under these groups:

- **New semantic elements**

- <header>
- <footer>
- <article>
- <section>

New attributes form elements

- Number
- Date
- Time
- Calendar
- Range

type='date' - Creates a date input control, such as a pop-up calendar, for specifying a date (year, month, day). The initial value must be provided in ISO date format.

```
<input type='date' name='set_date' value='2011-10-15' />
```



type='datetime' - Creates a combined date/time input field. The value is an ISO formatted date and time that is defined and submitted as UTC time.

```
<input type='datetime' name='dt' value='2011-06-14T01:26:32:00Z' />
```



type='datetime-local' - Creates a date/time input control, the same as 'datetime', but assuming the time is in the local time zone. Initial values must be provided in ISO date/time format.

```
<input type='datetime-local' name='dtl' value='2012-05-23T13:44:16:00'>
```

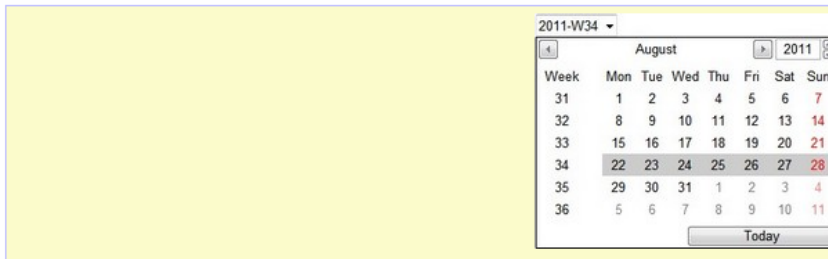
type='month' - Creates a date input control (such as a pop-up calendar), for specifying a particular month in a year.

```
<input type='month' value='2012-09' name='mnt' />
```



type='week' - Creates a date input control (such as a pop-up calendar), for specifying a particular week in a year. Values are provided in ISO week numbering format.


```
<input type='week' name='weeks' value='2011-W34' />
```



Week	Mon	Tue	Wed	Thu	Fri	Sat	Sun
31	1	2	3	4	5	6	7
32	8	9	10	11	12	13	14
33	15	16	17	18	19	20	21
34	22	23	24	25	26	27	28
35	29	30	31	1	2	3	4
36	5	6	7	8	9	10	11

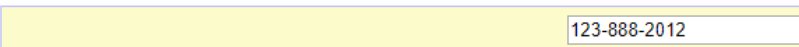
type='time' - Creates a date input control for specifying a time (hour, minute, seconds, fractional_seconds).

```
<input type='time' name='currenttime' value='18:12:00' />
```



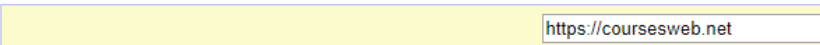
type='tel' - Creates an input field for entering and editing a telephone number.

```
<input type='tel' name='hometel' value='123-888-2012' />
```



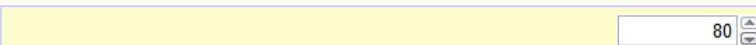
type='url' - is used for input fields that should contain a URL address. The value of the url field is automatically validated when the form is submitted (if it is not in proper URL format, return an error message).

```
<input type='url' name='site' size='25' value='https://coursesweb.net' />
```



type='number' - is used to create input fields that should contain a numeric value. You can also set restrictions on what numbers are accepted, with the 'min', 'max' and 'step' attributes.

```
<input type='number' name='points' min='5' max='80' />
```



type='range' - Creates a slider control that should contain a value from a range of numbers. The range starts at the value provided by the 'min' attribute (0 by default) and ends at the value provided by the 'max' attribute (100 by default).

```
<input type='range' name='val' min='1' max='10' />
```



type='email' - is used for input fields that should contain an e-mail address. The value of the email field is automatically validated when the form is submitted.

```
Email; <input type='email' name='email' />
```

Email:

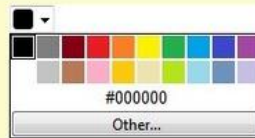
type='search' - Creates a one-line text input box for entering a search query, like a site search.

```
Search: <input type='search' name='srch' size='25' value='Search term' />
```

Search:

type='color' - Creates a color well control for selecting a color value.

```
<input type='color' name='get_color' />
```



New graphic elements

- <svg>
- <canvas>

New multimedia elements

- <audio>
- <video>

HTML5 Browser support

Opera is currently the only browser that supports this element, although you can use the <select> element inside the datalist object to get non-supportive browsers to display your code.

An HTML5 Detection Library: Modernizr

Modernizr can be best described as a MIT-licensed, open sourced JavaScript library, used for detecting support for different HTML5 and CSS3 features. This small JavaScript Library has the power to detect the availability of native implementations for different Next Gen web technologies. The latest version is 3.7.1. Make sure you are using the latest version for the projects. To use the latest version, just follow this simple trick. Just include the following `<script>` element at the top of your page in the below example:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Dive into HTML5</title>
  <script src="modernizr.min.js"></script>
</head>
<body>
  ...
</body>
</html>
```

Kindly note that Modernizr runs automatically and there are no such `modernizr_init()` functions to call. While running, it creates a global object called Modernizr featuring all Boolean properties for every feature it can detect. For example, if your browser supports the canvas API the `Modernizr.canvas` property will be `TRUE`. If the browser does not support the canvas API, the `Modernizr.canvas` property will be `FALSE`.

There are several new features which are being introduced through HTML5 and CSS3 but same time many browsers do not support these news features. Modernizr provides an easy way to detect any new feature to let the user take any kind of corresponding action. For example, if a browser does not support video feature then you may like to display a simple page.

HTML Canvas

The HTML `<canvas>` element is used to draw graphics on a web page.

However, you need to keep in mind that the `<canvas>` element is only a container for graphics. You must use JavaScript to actually draw the graphics.

Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

A sample example using `<canvas>` that will display a text within a rectangular box:

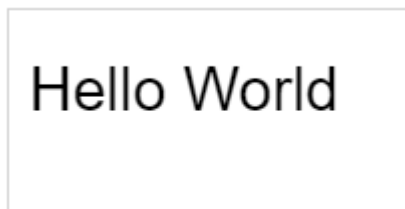

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="200" height="100" style="border:1px solid
#d3d3d3;">
Your browser does not support the HTML5 canvas tag.</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World",10,50);
</script>

</body>
</html>
```

The output will be like this:



Reviewing the chapter

- With HTML5, you get to witness several New Form Elements. These elements come bundled with new features and capabilities. One of the most notable inclusions regarding these New Form Elements is modifying part of the web page dynamically.
- The <meter> element is used to output a numeric element.
- The present version of HTML5 features a keygen element that generates an encryption key for passing encrypted data to a server.
- The default character encoding in HTML5 is UTF-8
- The New HTML5 elements are segregated into different groups like semantic, attribute, graphic, and multimedia
- Opera is currently the only browser that support New Form HTML elements
- Modernizr can be best described as a MIT-licensed, open sourced JavaScript library, used for detecting support for different HTML5 and CSS3 features. This small JavaScript Library has the power to detect the availability of native implementations for different Next Gen web technologies
- The HTML <canvas> element is used to draw graphics on a web page.

Testing Your Skills

1. The following are considered New Form HTML5 Elements:

a) <meter>, b) <progress>, c) <output>, d) all of the above

2. _____ indicates the total amount of task being completed through percentage marking

a) <output>, b) <progress>, c) <keytype>, d) none of the above

3. Which of these are <progress> parameter elements?
a) <keychain>, b) <keytype>, c) <key>, d) <pkey>
4. Which of the following are new semantic elements?
a) <header>, b) <footer>, c) <article>, d) all of the above
5. Which of the following are new graphics elements?
a) <video>, b) <svg>, c) both (a) and (b), d) none of the above
6. The latest version of Modernizr available is:
a) 1.1.1, b) 3.2.1, c) 3.7.1, d) 4.6.0
7. Which element is used for drawing graphics in HTML5?
a) <paint>, b) <digitaldraw>, c) <canvas>, d) <draw>
8. Which of the following are new multimedia elements in HTML5?
a) <audio>, b) <video>, c) <paint>, d) only (a) and (b)
9. Which of these are new attributes form elements?
a) Number, b) Date, c) Time, d) all of the above
10. The default character encoding in HTML5 is
a) UTF16, b) UTF8, c) UTF32, d) UCS4

Lesson 9: HTML5- Client Side Storage (120 minutes)

Objective: After completing this lesson you will be able to : <ul style="list-style-type: none">• Introduction to HTML5 Client-Side Storage• Types of Client-Side Storage	Materials Required: <ul style="list-style-type: none">• Computer With Windows XP and above• Stable Internet connection
Self- Learning Duration: 60 minutes	Practical Duration: 60 minutes
Total Duration: 120 minutes	

HTML5 Client Side Storage

It has been a common option for developers to use the client side cookies for storing data like order items, user preferences, etc. But that said, size of cookies seem to be a problematic scenario. Remember, cookies are limited in size and structure. Moreover, they tend to travel with every request to the web server. This results in causing unnecessary overhead scenarios.

With HTML5, developers can enjoy a couple of client-side storage facilities:

1. Web storage (includes local as well as session storage, much similar to session and persistent cookie)
2. Database storage (SQLite or IndexedDB)

Please take a note of the fact that once you use the web storage option, it becomes possible to store up to 5 mb of data at client side. Another good point is, unlike the cookies, this data will never travel with every request to the web server.

Types of Client Side Storage in HTML5

Storing data at client side in HTML5 can be done in either of the two ways:

- Local Storage
- Session Storage

Local Storage

HTML5 local storage makes it possible to store values in the browser which can survive the browser session.

Focus on the example below:

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

<div id="result"></div>

<script>
// Check browser support
if (typeof(Storage) !== "undefined") {
  // Store
  localStorage.setItem("lastname", "Smith");
  // Retrieve
  document.getElementById("result").innerHTML =
localStorage.getItem("lastname");
} else {
  document.getElementById("result").innerHTML = "Sorry, your browser does not
support Web Storage...";
}
</script>

</body>
</html>
```

Smith

Example explained:

- Create a localStorage name/value pair with name="lastname" and value="Smith"
- Retrieve the value of "lastname" and insert it into the element with id="result"

The example above could also be written like this:

```
// Store
```

```
localStorage.lastname = "Smith";
```

```
// Retrieve
```

```
document.getElementById("result").innerHTML = localStorage.lastname;
```

The syntax for removing the "lastname" localStorage item is as follows:

```
localStorage.removeItem("lastname");
```

Note: Name/value pairs are always stored as strings. Remember to convert them to another format when needed!

Session Storage

The sessionStorage property accesses a session Storage object for the current origin. sessionStorage has much similarity with the localStorage; however, the only difference is in sessionStorage, the end of a page session will clear the data present. But in localStorage, the data never expires.

```

<!DOCTYPE html>
<html>
<head>
<script>
function clickCounter() {
  if (typeof(Storage) !== "undefined") {
    if (sessionStorage.clickcount) {
      sessionStorage.clickcount = Number(sessionStorage.clickcount)+1;
    } else {
      sessionStorage.clickcount = 1;
    }
    document.getElementById("result").innerHTML = "You have clicked the button " +
    sessionStorage.clickcount + " time(s) in this session.";
  } else {
    document.getElementById("result").innerHTML = "Sorry, your browser does not support
    web storage...";
  }
}
</script>
</head>
<body>

<p><button onclick="clickCounter()" type="button">Click me!</button></p>
<div id="result"></div>
<p>Click the button to see the counter increase.</p>
<p>Close the browser tab (or window), and try again, and the counter is reset.</p>

</body>
</html>

```

Click me!

Click the button to see the counter increase.

Close the browser tab (or window), and try again, and the counter is reset.

A sample project

Let's focus on a sample example to test the use of session and local storage in HTML5. Create a HTML page with the name CustomerProfile.html. This page will store the details of the customer who would like to place an order. See below how the HTML coding will be like:

```

<!DOCTYPEhtml>

<htmllang="en">

<head>

<title>Customer Profile</title>

</head>

<body>

<sectionid="contact"class="grad">

<h1>Customer Profile Form</h1>

<form>

<table>

<tr>

<tdstyle="width:30%;">First Name</td>

<tdstyle="width:70%;"><inputid="txtFirstName"type="text"placeholder="Enter First
Name"autofocus/></td>

</tr>

<tr>

<tdstyle="width:30%;">Last Name</td>

<tdstyle="width:70%;"><inputid="txtLastName"type="text"placeholder="Enter Last Name"/></td>

```

```
</tr>
<tr>
<tdstyle="width:30%;">Address</td>
<tdstyle="width:70%;"><inputid="txtAddress"type="text"placeholder="Enter Your Address"/></td>
</tr>
<tr>
<tdstyle="width:30%;">City</td>
<tdstyle="width:70%;"><inputid="txtCity"type="text"placeholder="Enter City Name"/></td>
</tr>
<tr>
<tdstyle="width:30%;">Contact No.</td>
<tdstyle="width:70%;"><inputid="txtContactNo"type="text"placeholder="Enter Contact No."/></td>
</tr>
<tr>
<tdstyle="width:30%;">Email ID</td>
<tdstyle="width:70%;"><inputid="txtEmailID"type="text"placeholder="Enter Email ID"/></td>
</tr>
<tr>
<tdstyle="width:100%;text-align:center;colspan="2"><inputid="Submit"type="submit"value="Submit
Profile"/></td>
</tr>
</table>
</form>
</section>
</body>
</html>
```

Reviewing the chapter

With HTML5, developers can enjoy a couple of client-side storage facilities:

- Web storage (includes local as well as session storage, much similar to session and persistent cookie)
- Database storage (SQLite or IndexedDB)

Storing data at client side in HTML5 can be done in either of the two ways:

- Local Storage
- Session Storage

The sessionStorage property accesses a session Storage object for the current origin. sessionStorage has much similarity with the localStorage; however, the only difference is in sessionStorage, the end of a page session will clear the data present. But in localStorage, the data never expires.

Testing Your Skills

1. The_____ property is used for accessing session storage for current origin.
a) sessionStorage, b) sessionBundled, c) HTML5Session, d) None of the above
2. In which form of storage, data never expires?
a) session Storage, b) local storage, c) both (a) and (b), d) none of the above
3. This type of storage makes it possible to store values in the browser which can survive the browser session.
a) Local Storage, b) Session Storage, c) Hybrid Storage, d) Server Storage
4. Which of the following are considered Database Storage?
a) SQLite, b) IndexedDB, c) both (a) and (b), d) none of the above
5. The web storage option can store up to ___mb of data at client side
a) 2mb, b) 10mb, c) 25mb, d) 5mb

Lesson 10: The Offline Access & Drag and Drop API (120 minutes)

Objective: After completing this lesson you will be able to :

- Introduction to Offline Access in HTML5
- Techniques of implementing Offline Access
- Introduction to Drag & Drop API in HTML5
- Techniques of implementing Drag & Drop API
- Limitations of HTML5
- Comparison – Native SDK & HTML5

Materials Required:

- Computer With Windows XP and above
- Stable Internet connection

Self- Learning Duration: 60 minutes

Practical Duration: 60 minutes

Total Duration: 120 minutes

Introduction to Offline Access in HTML5

Offline storage is a highly critical feature in HTML5. The introduction of Application Cache utility option definitely helps in providing advantages to accessing the web and browsing the pages. The HTML5 App Cache instructs supporting browsers to perfectly cache copies of certain files. After downloading of these files, accessing them offline becomes easier than ever.

Techniques of implementing Offline Access

CREATE AN HTML5 PAGE

Create your HTML5 page using the following initial outline:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
</head>
```

```
<body>
```

```
<h1>Lovely Picture</h1>
```

```
<canvas id="picCanvas" width="450" height="350"></canvas>
```



```
</body>
```

```
</html>
```

The above coding will help in creating a page that will include a drawing within the canvas element that in turn will include an image. Here, the canvas element specifies dimensions. However, to do so, we need to simply use JavaScript in a separate file to draw within it, referencing it through the ID attribute.

CREATE A JAVASCRIPT FILE

So that we can test the App Cache function, let's create a separate JavaScript file to work with the page. Create a new file and save it "picdraw_functions.js".

Enter the following function:

```
function drawPic(picSRC, canvasElem) {  
  
    //get the canvas using passed element ID  
var canv = document.getElementById(canvasElem);  
    //get the context  
var cont = canv.getContext("2d");  
  
    //define a gradient to spread across the canvas diagonally  
var grad = cont.createLinearGradient(0, 0, 450, 350);  
    //define two colors  
grad.addColorStop(0, "#000033");  
grad.addColorStop(1, "#003300");  
    //set the gradient fill  
cont.fillStyle = grad;  
    //fill the entire canvas with the gradient  
cont.fillRect(0, 0, 450, 350);  
  
    //create the image  
var picImg = new Image();  
    //draw the image so that it is centered  
picImg.onload = function(){ cont.drawImage(picImg, 25, 25); };
```

```
//set the passed image source
```

```
picImg.src=picSRC;
```

```
}
```

The above code will draw the canvas element. Hence, you need not have to worry about the detailing part. Keep a note of the Function element that will receive not one but two parameters. One of the parameter will be representing the canvas element ID attribute while the other one will be representing an image file to include in the drawing.

The final canvas drawing will be the imported image file displayed against a background with a gradient fill. We are using this function to demonstrate caching the script and the image file.

INCLUDE JAVASCRIPT IN THE PAGE

Add a link to the JavaScript file in the page head section as follows:

```
<script type="text/javascript" src="picdraw_functions.js"></script>
```

Now call the function by adding the following before the closing page body tag:

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

```
drawPic("mypicture.jpg", "picCanvas");
```

```
</script>
```

Notice that the script section here passes the name of the image file to display as part of the drawing and the ID of the canvas element to draw within. You can of course alter the image filename to suit an image file you want to use with your own page. If you do use your own image file you may need to alter the dimensions – the code in the HTML and JavaScript file is designed to display an image 400px wide and 300px high in the center of the canvas.

CREATE THE MANIFEST FILE

Create a new file and save it with the name "picdraw.appcache". This is going to be your Manifest file for the page. Inside the file, you can list which items you want the browser to cache for offline use. Enter the following code:

```
CACHE MANIFEST
```

```
your_page.html
```

```
picdraw_functions.js
```

```
mypicture.jpg
```

Alter the HTML file to suit the name you gave your own page and the image filename to suit your own image file. The Manifest can optionally include a "NETWORK" section to indicate files that you do not want the browser to cache, and a "FALLBACK" section for providing alternatives to files that are not available offline.

SERVE THE MANIFEST MIME-TYPE

For HTML5 App Cache to function correctly, you must configure your server to ensure it uses the appropriate MIME-Type for your Manifest file. The process for this depends on your Web server, but for Apache servers you can simply include the following line in your ".htaccess" file:

```
AddType text/cache-manifest .appcache
```

This instructs the server to treat files with ".appcache" extension as Cache Manifest type. For other servers refer to your Web host or control panel.

INDICATE THE MANIFEST FILE

Finally, you need to indicate the name and location of your Manifest file within the HTML page markup. Alter your opening HTML tag as follows:

```
<html manifest="picdraw.appcache">
```

This must match the name of your App Cache Manifest file.

UPLOAD AND TEST

Save your files and upload them to your server, remembering to include the ".htaccess" file, the Manifest file, the Web page, the JavaScript file and the image. Browse to the page in your Web browser. Depending on which browser you are using, you may be prompted to grant permission for the site to store data for offline use. If so, grant permission. To test the cache function, disconnect your computer from the Internet and attempt to load the page again. You should see the content of the page including the JavaScript-powered canvas element with gradient and image, meaning that all three files have been saved to the App Cache.

Introduction to Drag & Drop API in HTML5

HTML Drag and Drop interfaces enable applications to use drag-and-drop features in different browsers. The user may select draggable elements with a mouse, drag those elements to a droppable element, and drop them by releasing the mouse button. A translucent representation of the draggable elements follows the pointer during the drag operation.

For web sites, extensions, and XUL applications, you can customize which elements can become draggable, the type of feedback the draggable elements produce, and the droppable elements.

Techniques of implementing Drag & Drop API

There are number of events which are fired during various stages of the drag and drop operation. These events are listed below –

- Dragstart - Fires when the user starts dragging of the object.

- Dragenter - Fired when the mouse is first moved over the target element while a drag is occurring. A listener for this event should indicate whether a drop is allowed over this location. If there are no listeners, or the listeners perform no operations, then a drop is not allowed by default.
- Dragover - This event is fired as the mouse is moved over an element when a drag is occurring. Much of the time, the operation that occurs during a listener will be the same as the dragenter event.
- Dragleave - This event is fired when the mouse leaves an element while a drag is occurring. Listeners should remove any highlighting or insertion markers used for drop feedback.
- Drag - Fires every time the mouse is moved while the object is being dragged.
- Drop - The drop event is fired on the element where the drop was occurred at the end of the drag operation. A listener would be responsible for retrieving the data being dragged and inserting it at the drop location.
- Dragend - Fires when the user releases the mouse button while dragging an object.

Note – Note that only drag events are fired; mouse events such as mousemove are not fired during a drag operation.

Let's have a look at an example with output:

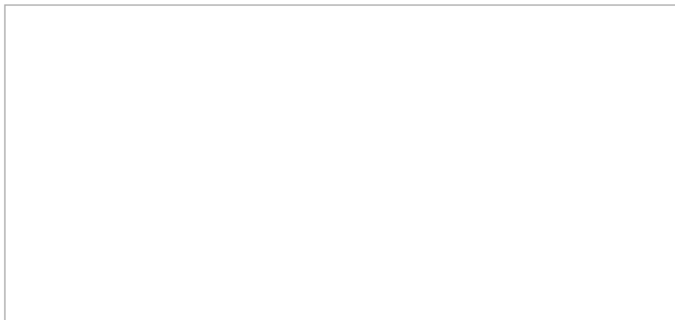
```
<!DOCTYPE HTML>
<html>
<head>
<style>
#div1 {
  width: 450px;
  height: 200px;
  padding: 10px;
  border: 1px solid #aaaaaa;
}
</style>
<script>
function allowDrop(ev) {
  ev.preventDefault();
}

function drag(ev) {
  ev.dataTransfer.setData("text", ev.target.id);
}
```

```
function drop(ev) {  
  ev.preventDefault();  
  var data = ev.dataTransfer.getData("text");  
  ev.target.appendChild(document.getElementById(data));  
}  
</script>  
</head>  
<body>  
  
<p>Drag the anudip image into the rectangle: </p>  
  
<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>  
<br>  
  
  
</body>  
</html>
```

The Output:

Drag the anudip image into the rectangle:



Just focus on the output. The Image “anudip” can be dragged to the above blank box. (Make sure you have an image downloaded with the name that you should include in the “src” part of the coding. Never

forget to mention the image extension as well. In this case, I have saved an image in the desktop with the name and extension anudip.png.

The DataTransfer Object

The event listener methods for all the drag and drop events accept Event object which has a readonly attribute called dataTransfer.

The event.dataTransfer returns DataTransfer object associated with the event as follows –

```
function EnterHandler(event) {  
    DataTransfer dt = event.dataTransfer;  
    .....  
}
```

The DataTransfer object holds data about the drag and drop operation. This data can be retrieved and set in terms of various attributes associated with DataTransfer object as explained below –

Sr.No. DataTransfer attributes and their description

dataTransfer.dropEffect [= value]

- Returns the kind of operation that is currently selected.
- This attribute can be set, to change the selected operation.
- The possible values are none, copy, link, and move.

dataTransfer.effectAllowed [= value]

- Returns the kinds of operations that are to be allowed.
- This attribute can be set, to change the allowed operations.
- The possible values are none, copy, copyLink, copyMove, link, linkMove, move, all and uninitialized.

dataTransfer.types

- Returns a DOMStringList listing the formats that were set in the dragstart event. In addition, if any files are being dragged, then one of the types will be the string "Files".

dataTransfer.clearData ([format])

- Removes the data of the specified formats. Removes all data if the argument is omitted.

dataTransfer.setData(format, data)

- Adds the specified data.

data = dataTransfer.getData(format)

- Returns the specified data. If there is no such data, returns the empty string.

dataTransfer.files

- Returns a FileList of the files being dragged, if any.

dataTransfer.setDragImage(element, x, y)

- Uses the given element to update the drag feedback, replacing any previously specified feedback.

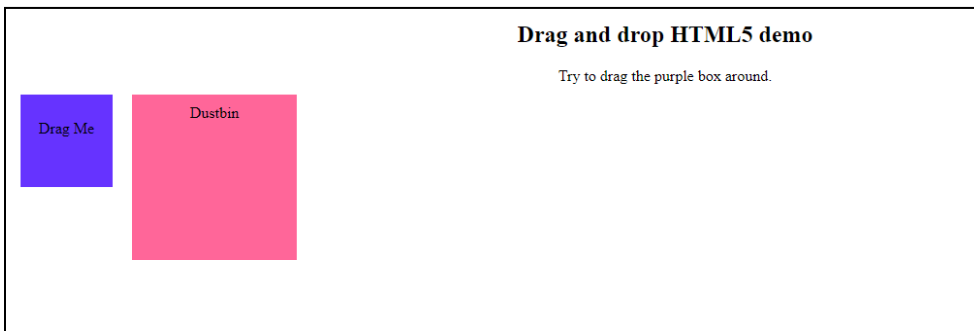
dataTransfer.addElement(element)

- Adds the given element to the list of elements used to render the drag feedback.

A sample example to understand Data Transfer Object in a better way:

```
<!DOCTYPE HTML>
<html>
  <head>
    <style>
      #boxA, #boxB { float:left;padding:10px;margin:10px; -moz-user-select:none; }
      #boxA { background-color: #6633FF; width:75px; height:75px; }
      #boxB { background-color: #FF6699; width:150px; height:150px; }
    </style>
    <script>
      function dragStart(ev) {
        ev.dataTransfer.effectAllowed='move';
        ev.dataTransfer.setData("Text", ev.target.getAttribute('id'));
        ev.dataTransfer.setDragImage(ev.target,0,0);
        return true;
      }
    </script>
  </head>
  <body>
    <center>
      <h2>Drag and drop HTML5 demo</h2>
      <div>Try to drag the purple box around.</div>
      <div id = "boxA" draggable = "true" ondragstart = "return dragStart(ev)">
        <p>Drag Me</p>
      </div>
      <div id = "boxB">Dustbin</div>
    </center>
  </body>
</html>
```


The output:



Limitations of HTML5

All of us are aware of the latest version of HTML i.e. HTML5. Yes obviously HTML5 is a considerable replacement of HTML4. The better coding system in HTML5 has enabled easy input of media files in the website now. But we know that every coin has two sides. There are few disadvantages of HTML5 which are given below and everyone should consider this points before creating new website design using HTML5-

Security – HTML5 is not must popular as such but its chances of replacing flash have increased its chances of popularity. But security is a big problem with HTML5. As far as now, HTML5 websites have to face a bit of security problems.

Features – the features of HTML5 have not fulfilled the expectations of the web developers. It was expected that it will be very much near to Flash but it doesn't even have 50% features of Flash. HTML5 is a higher version of HTML4 but doesn't offer features that are worth taking a note.

Browser compatibility – HTML5 still lacks its compatibility with all the browsers. It is still not supported by a few browsers. Thus, majority of the websites avoid the use of HTML5 for coding. Very few have used HTML5 and others have it as their alternative option. Also, the webpage developed on HTML5 sometimes appear different on different browser.

Comparison – Native SDK & HTML5

Portability

HTML5 apps are portable across different OSes and device types. A HTML5 app written with responsive design methods would also scale appropriately depending on the size of the device it's currently viewed on. When an update is required, the single app is updated and tested, and is available for all devices immediately.

Native apps, by definition, are unique to each OS, and so, to support multiple mobile OSes, a separate app must be written for each OS. When an update is required, each app must be updated independently, and tested independently. Android apps do not modify the layout to match the size of the device automatically, however, during development, different layouts can be specified for different device/screen sizes and orientations. This usually results in layouts that are more aesthetically pleasing compared to automatic responsive design HTML5 apps, albeit with more effort and planning required.

Cost to develop

HTML5 apps are generally cheaper to develop and maintain than native apps, since just the single app is required for multiple OS support. This single app can be developed by a single web developer. However, native apps for all major mobile OSes would typically require a specialized developer for each OS (Java for Android, Objective C/Swift for iOS, C# for Windows), which would be significantly more expensive than a single web developer.

Speed and Efficiency

Native apps are almost always more efficient, and faster than HTML5 apps. Despite the strides made in increasing the speed of Javascript interpreters, a HTML5 app cannot execute at the same speed as a native app of similar functionality. Apple's App Store can reject apps for being too slow or not feeling native enough, which is more likely to occur with a HTML5 app than a fully native app.

Use of Hardware

Native apps can interact with a wide range of the hardware available on a device, including location, camera, accelerometer, speakers, screen and more. HTML5 apps do not have the same ability to interact with the hardware, which can be a major deal breaker, depending on the desired functionality of the app.

If you need to deliver an app for the lowest cost at the quickest speed on multiple mobile OS types, a HTML5 app is almost always the preferred way to go. However, if your app needs to make use of device specific hardware, or needs to run very fast, you will do better with a native solution.

Reviewing the chapter

- The introduction of Application Cache utility option definitely helps in providing advantages to accessing the web and browsing the pages. The HTML5 App Cache instructs supporting browsers to perfectly cache copies of certain files.
- For HTML5 App Cache to function correctly, you must configure your server to ensure it uses the appropriate MIME-Type for your Manifest file.
- HTML Drag and Drop interfaces enable applications to use drag-and-drop features in different browsers.

Testing Your Skills

1. _____ event is fired when the user starts dragging of the object
a) Dragend, b) Dragover, c) Dragcenter, d) Dragstart
2. _____ event is fired when the mouse is first moved over the target element while a drag is occurring
a) Dragend, b) Dragcenter, c) Dragover, d) Dragstart
3. _____ event is fired as the mouse is moved over an element when a drag is occurring
a) Dragend, b) Dragcenter, c) Dragover, d) Dragstart

4. _____ event is fired when the mouse leaves an element while a drag is occurring
a) Dragend, b) Dragleave, c) Dragover, d) Dragstart
5. _____ event is fired when the user releases the mouse button while dragging an object
a) Dragend, b) Dragleave, c) Dragover, d) Dragstart
6. _____ attribute returns the kind of operation that is currently selected
a) dataTransfer.dropEffect [= value], b) dataTransfer.effectAllowed [= value], c) dataTransfer.clearData ([format]), d) data = dataTransfer.getData(format)
7. _____ attribute returns the kinds of operations that are to be allowed
a) dataTransfer.dropEffect [= value], b) dataTransfer.effectAllowed [= value], c) dataTransfer.clearData ([format]), d) data = dataTransfer.getData(format)
8. _____ attribute removes all data if the argument is omitted
a) dataTransfer.dropEffect [= value], b) dataTransfer.effectAllowed [= value], c) dataTransfer.clearData ([format]), d) data = dataTransfer.getData(format)
9. _____ attribute uses the given element to update the drag feedback
a) dataTransfer.setDragImage(element, x, y), b) dataTransfer.effectAllowed [= value], c) dataTransfer.clearData ([format]), d) data = dataTransfer.getData(format)
10. _____ attribute adds the given element to the list of elements used to render the drag feedback
a) dataTransfer.dropEffect [= value], b) dataTransfer.addElement(element), c) dataTransfer.clearData ([format]), d) data = dataTransfer.getData(format)

Lesson 11: HTML5 APIs (120 minutes)

Objective: After completing this lesson you will be able to : <ul style="list-style-type: none">• HTML5 JavaScript APIs: overview• Canvas, SVG• Web Sockets• IndexedDB• Web Notifications	Materials Required: <ul style="list-style-type: none">• Computer With Windows XP and above• Stable Internet connection
Self- Learning Duration: 60 minutes	Practical Duration: 60 minutes
Total Duration: 120 minutes	

HTML5 JavaScript APIs: overview

JavaScript is one of the most crucial web development languages. For a web developer, sound knowledge of the following three languages will definitely help in producing the finest of web pages:

1. HTML to define the content of web pages
2. CSS to specify the layout of web pages
3. JavaScript to program the behavior of web pages

However, as far as the use of JavaScript is concerned, programming the behavior of the web pages is not the only function it performs. Rather, if looked deeply, there are several server and desktop programs where you will notice the use of JavaScript. Node.js is the best known. Some databases, like MongoDB and CouchDB, also use JavaScript as their programming language.

Please keep a note of the fact that JavaScript and Java are completely different languages, both in concept and design.

What is an API?

API or Application Programming Interface is a way to develop web applications using the pre-built components. It is not unique to only web development or even to any kind of scripting languages. Websites like Twitter, YouTube and others provide APIs to the public so designers and developers can integrate features into their own websites (or for other purposes such as mobile or desktop applications). One of the main goals of an API is to standardize how certain mechanics work and to simplify otherwise complicated programming tasks. APIs are very significant in the world of HTML5 and there a number of them to explore, including Drag and Drop, Web storage, Microdata, and Geolocation, among others.

Canvas and SVG

Canvas

The HTML <canvas> element is used to draw graphics on a web page via JavaScript.

Do keep in mind that the <canvas> element is only a container for graphics. You must use JavaScript to actually draw the graphics.

Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

SVG

SVG stands for Scalable Vector Graphics and is used to define graphics for the Web. SVG is a W3C recommendation. The HTML <svg> element is a container for SVG graphics.

SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

Web Sockets

WebSockets is a next-generation bidirectional communication technology for web applications. It operates over a single socket. WebSockets can be exposed via a JavaScript interface in HTML 5 compliant browsers.

Once you get a Web Socket connection with the web server, you can send data from browser to server by calling a send() method, and receive data from server to browser by an onmessage event handler.

WebSocket Events

Event	Event Handler	Description
open	Socket.onopen	This event occurs when socket connection is established.
message	Socket.onmessage	This event occurs when client receives data from server.
error	Socket.onerror	This event occurs when there is any error in communication.
close	Socket.onclose	This event occurs when connection is closed.

Let's focus on an example:

```
<!DOCTYPE HTML>
<html>
  <head>
    <script type = "text/javascript">
      function WebSocketTest() {
        if ("WebSocket" in window) {
          alert("WebSocket is supported by your Browser!");
        }
      }
    </script>
  </head>
</html>
```

```
// Let us open a web socket
var ws = new WebSocket("ws://localhost:9998/echo");

    ws.onopen = function() {

        // Web Socket is connected, send data using send()
        ws.send("Message to send");
        alert("Message is sent...");
    };

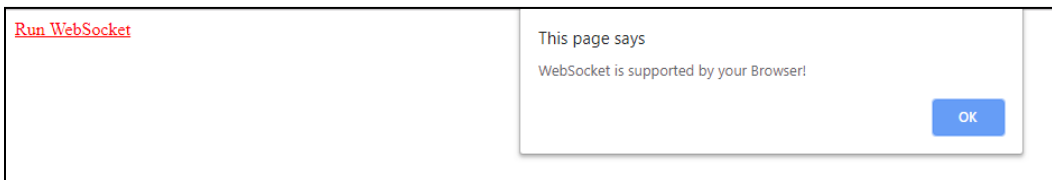
    ws.onmessage = function (evt) {
        var received_msg = evt.data;
        alert("Message is received...");
    };

    ws.onclose = function() {

        // websocket is closed.
        alert("Connection is closed...");
    };
} else {
    // The browser doesn't support WebSocket
    alert("WebSocket NOT supported by your Browser!");
}
}
</script>
</head>
<body>
    <div id = "sse">
        <a href = "javascript:WebSocketTest()">Run WebSocket</a>
    </div>

</body>
</html>
```

The output:



IndexedDB

The indexeddb is a new HTML5 concept to store the data inside user's browser. indexeddb is considered more powerful in comparison to the local storage and used for applications where there lies a requirement for bigger amount of data. IndexedDB based applications load a lot faster and can run a lot more efficiently.

Why to use indexeddb?

The W3C has announced that the Web SQL database is a deprecated local storage specification so web developer should not use this technology any more. indexeddb is an alternative for web SQL data base and more effective than older technologies.

Features of IndexedDB

- it stores key-pair values
- it is not a relational database
- IndexedDB API is mostly asynchronous
- it is not a structured query language
- it has supported to access the data from same domain

Step By Step Process To IndexedDB coding

Before enter into an indexeddb, we need to add some prefixes of implementation as shown below:

```
window.indexedDB = window.indexedDB || window.mozIndexedDB || window.webkitIndexedDB ||  
window.msIndexedDB;  
  
window.IDBTransaction = window.IDBTransaction || window.webkitIDBTransaction ||  
window.msIDBTransaction;  
  
window.IDBKeyRange = window.IDBKeyRange ||  
window.webkitIDBKeyRange || window.msIDBKeyRange  
  
if (!window.indexedDB) {  
    window.alert("Your browser doesn't support a stable version of IndexedDB.")  
}
```

Open an IndexedDB database

Before creating a database, we have to prepare some data for the data base.let's start with company employee details.

```
const employeeData = [  
  { id: "01", name: "Gopal K Varma", age: 35, email: "contact@tutorialspoint.com" },  
  { id: "02", name: "Prasad", age: 24, email: "prasad@tutorialspoint.com" }  
];
```

Adding the data

Here adding some data manually into the data as shown below –

```
function add() {  
  var request = db.transaction(["employee"], "readwrite")  
  .objectStore("employee")  
  .add({ id: "01", name: "prasad", age: 24, email: "prasad@tutorialspoint.com" });  
  
  request.onsuccess = function(event) {  
    alert("Prasad has been added to your database.");  
  };  
  
  request.onerror = function(event) {  
    alert("Unable to add data\r\nPrasad is already exist in your database! ");  
  }  
}
```

Retrieving Data

We can retrieve the data from the data base using with get()

```
function read() {  
  var transaction = db.transaction(["employee"]);  
  var objectStore = transaction.objectStore("employee");  
  var request = objectStore.get("00-03");
```



```
request.onerror = function(event) {  
    alert("Unable to retrieve daa from database!");  
};  
  
request.onsuccess = function(event) {  
  
    if(request.result) {  
        alert("Name: " + request.result.name + ", Age:  
            " + request.result.age + ", Email: " + request.result.email);  
    } else {  
        alert("Kenny couldn't be found in your database!");  
    }  
};  
}
```

Using with get(), we can store the data in object instead of that we can store the data in cursor and we can retrieve the data from cursor.

```
function readAll() {  
    var objectStore = db.transaction("employee").objectStore("employee");  
  
    objectStore.openCursor().onsuccess = function(event) {  
        var cursor = event.target.result;  
        if (cursor) {  
            alert("Name for id " + cursor.key + " is " + cursor.value.name + ",  
                Age: " + cursor.value.age + ", Email: " + cursor.value.email);  
            cursor.continue();  
        } else {  
            alert("No more entries!");  
        }  
};  
}
```

Removing the data

We can remove the data from IndexedDB with `remove()`. Here is how the code looks like

```
function remove() {  
  var request = db.transaction(["employee"], "readwrite")  
    .objectStore("employee")  
    .delete("02");  
  
  request.onsuccess = function(event) {  
    alert("prasad entry has been removed from your database.");  
  };  
}
```

HTML Code

To show all the data we need to use `onClick` event as shown below code –

```
<!DOCTYPE html>  
  
<html>  
  <head>  
    <meta http-equiv = "Content-Type" content = "text/html; charset = utf-8" />  
    <title>IndexedDb Demo | onlyWebPro.com</title>  
  </head>  
  
  <body>  
    <button onclick = "read()">Read </button>  
    <button onclick = "readAll()"></button>  
    <button onclick = "add()"></button>  
    <button onclick = "remove()">Delete </button>  
  </body>  
</html>
```

The final code should be as –

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<meta http-equiv = "Content-Type" content = "text/html; charset = utf-8" />
```

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

```
//prefixes of implementation that we want to test
```

```
window.indexedDB = window.indexedDB || window.mozIndexedDB ||
```

```
window.webkitIndexedDB || window.msIndexedDB;
```

```
//prefixes of window.IDB objects
```

```
window.IDBTransaction = window.IDBTransaction ||
```

```
window.webkitIDBTransaction || window.msIDBTransaction;
```

```
window.IDBKeyRange = window.IDBKeyRange || window.webkitIDBKeyRange ||
```

```
window.msIDBKeyRange
```

```
if (!window.indexedDB) {
```

```
    window.alert("Your browser doesn't support a stable version of IndexedDB.")
```

```
}
```

```
const employeeData = [
```

```
    { id: "00-01", name: "gopal", age: 35, email: "gopal@tutorialspoint.com" },
```

```
    { id: "00-02", name: "prasad", age: 32, email: "prasad@tutorialspoint.com" }
```

```
];
```

```
var db;
```

```
var request = window.indexedDB.open("newDatabase", 1);
```

```
request.onerror = function(event) {
```

```
    console.log("error: ");
```

```
};
```

```
request.onsuccess = function(event) {
```

```
db = request.result;
console.log("success: "+ db);
};

request.onupgradeneeded = function(event) {
  var db = event.target.result;
  var objectStore = db.createObjectStore("employee", {keyPath: "id"});

  for (var i in employeeData) {
    objectStore.add(employeeData[i]);
  }
}

function read() {
  var transaction = db.transaction(["employee"]);
  var objectStore = transaction.objectStore("employee");
  var request = objectStore.get("00-03");

  request.onerror = function(event) {
    alert("Unable to retrieve daa from database!");
  };

  request.onsuccess = function(event) {
    // Do something with the request.result!
    if(request.result) {
      alert("Name: " + request.result.name + ",
        Age: " + request.result.age + ", Email: " + request.result.email);
    } else {
      alert("Kenny couldn't be found in your database!");
    }
  };
}
```

```
function readAll() {  
    var objectStore = db.transaction("employee").objectStore("employee");  
  
    objectStore.openCursor().onsuccess = function(event) {  
        var cursor = event.target.result;  
  
        if (cursor) {  
            alert("Name for id " + cursor.key + " is " + cursor.value.name + ",  
                Age: " + cursor.value.age + ", Email: " + cursor.value.email);  
            cursor.continue();  
        } else {  
            alert("No more entries!");  
        }  
    };  
}  
  
function add() {  
    var request = db.transaction(["employee"], "readwrite")  
        .objectStore("employee")  
        .add({ id: "00-03", name: "Kenny", age: 19, email: "kenny@planet.org" });  
  
    request.onsuccess = function(event) {  
        alert("Kenny has been added to your database.");  
    };  
  
    request.onerror = function(event) {  
        alert("Unable to add data\r\nKenny is already exist in your database! ");  
    }  
}  
  
function remove() {
```

```
var request = db.transaction(["employee"], "readwrite")  
.objectStore("employee")  
.delete("00-03");
```

```
request.onsuccess = function(event) {  
    alert("Kenny's entry has been removed from your database.");  
};  
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<button onclick = "read()">Read </button>
```

```
<button onclick = "readAll()">Read all </button>
```

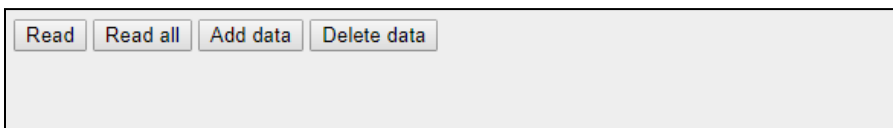
```
<button onclick = "add()">Add data </button>
```

```
<button onclick = "remove()">Delete data </button>
```

```
</body>
```

```
</html>
```

It will produce the following output –



Web notifications

Web notifications, also known as Web Messaging is the way for documents to separates browsing context to share the data without DOM (Document Object Model). It overrides the cross domain communication problem in different domains, protocols or ports

For example, you want to send the data from your page to ad container which is placed at iframe or voice-versa, in this scenario, Browser throws a security exception. With web messaging we can pass the data across as a message event.

Message Event

Message events fire Cross-document messaging, channel messaging, server-sent events and web sockets. It has been described by Message Event interface.

Attributes

- Data - Contains string data
- Origin - Contains Domain name and port
- lastEventId - Contains unique identifier for the current message event
- source - Contains to A reference to the originating document's window
- ports - Contains the data which is sent by any message port

Cross Document Messaging

Using Cross-document Messaging

Send the message from the sending page using the `postMessage()` method of the receiving page window object (in mobile, wearable, and TV applications). To receive the page, the receiving page window object must be registered to receive messages.

The `postMessage()` method supports the following parameters:

- message: Message to be sent.
- targetOrigin: Domain receiving the message. If a certain domain cannot be defined, use the wildcard ('*').
- transfer (optional): List of transferable objects.

Learning how to use cross-document messaging enhances the communication capabilities of your application:

- Create document A and B.
- Insert document B as iframe into document A:

```
<iframe id="frame1" src="/web_messaging_cross_document_messaging_iframe.htm"></iframe>
```

- In document A, use the `sendMessage()` method to send a message to document B.

Use the `postMessage()` method of the iframe window, which sends the message from the method content, to deliver the message to the iframe.

```
<script>
```

```
function sendMessage(message) {  
    var frame1 = document.getElementById('frame1');  
    frame1.contentWindow.postMessage(message, '*');
```

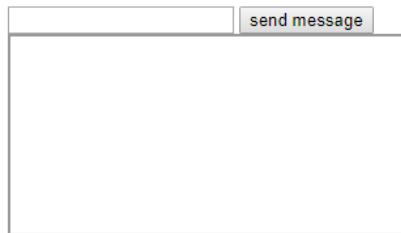
```
}  
</script>
```

Register the message event handler in document B to receive the sent message:

```
<script>  
    btnSendMessageHandler = function(e) {  
        var messageEle = document.getElementById('message');  
        sendMessage(messageEle.value);  
    };  
    /* Register event handler */  
    btnSendMessage.onclick = btnSendMessageHandler;  
</script>
```

The output:

Web messaging tutorial : cross document messaging



Reviewing the Chapter

- JavaScript is used to program the behavior of web pages
- API or Application Programming Interface is a way to develop web applications using the pre-built components.
- The HTML <canvas> element is used to draw graphics on a web page via JavaScript
- SVG stands for Scalable Vector Graphics and is used to define graphics for the Web. SVG is a W3C recommendation. The HTML <svg> element is a container for SVG graphics.
- WebSockets is a next-generation bidirectional communication technology for web applications. It operates over a single socket.
- The indexeddb is a new HTML5 concept to store the data inside user's browser. indexeddb is considered more powerful in comparison to the local storage and used for applications where there lies a requirement for bigger amount of data.
- Web notifications, also known as Web Messaging is the way for documents to separates browsing context to share the data without Document Object Model.

Testing your Skills

1. _____ is used to define the content of a web page
a) JavaScript, b) HTML, c) CSS, d) SVG

2. Which of the following is used for specifying the layout of web pages:
a) JavaScript, b) HTML, c) CSS, d) SVG

3. Which of these use JavaScript as their programming language:
a) MongoDB, b) CouchDB, c) Both (a) and (b), d) None of the above

4. _____ is a way to develop web applications using the pre-built components
a) CGI, b) API, c) HTML, d) CSS

5. _____ element is used to draw graphics on a web page via JavaScript
a) SVG, b) CGI, c) Canvas, d) None of the above

6. This event occurs when socket connection is established
a) Socket.onopen, b) Socket.onmessage, c) Socket.onerror, d) Socket.onclose

7. This event occurs when there is any error in communication
a) Socket.onopen, b) Socket.onmessage, c) Socket.onerror, d) Socket.onclose

8. DOM stands for
a) Domain, b) Domain Object Model, c) Data Over Multimedia, d) Document Object Model

9. Which of the following are fired by message events:
a) Cross-document messaging, b) channel messaging, c) server-sent events, d) all of the above

10. This event occurs when client receives data from server
a) open, b) close, c) message, d) error

Lesson 12: Introduction to CSS 3

Objective: After completing this lesson you will be able to : <ul style="list-style-type: none">• What Is CSS3?• The History of CSS• How to Add CSS: Internal, External, Inline• General Syntax: Attributes, Selectors, Cascades, etc.• Current State of Browser Support for CSS3	Materials Required: <ul style="list-style-type: none">• Computer With Windows XP and above• Stable Internet connection
Self- Learning Duration: 60 minutes	Practical Duration: 60 minutes
Total Duration: 120 minutes	

What Is CSS3?

CSS3 is the upgraded version of Cascading Style Sheets. The new upgraded version introduces new features that definitely help in improving the overall webpage appearance. When developed a page using CSS3, it will not only look attractive but the simplicity to navigate and flawless functioning definitely makes web accessibility a fascinating experience.

Designing a page using CSS3 leaves a long lasting positive impression among the visitors and online shopping enthusiasts.

The History of CSS

Hakon Wium Lie is associated with the first draft release of CSS, way back in the month of October, 1994. The primary influence behind the development of CSS was the www-style mailing list that got developed in May 1995.

It took nearly three years to develop CSS for browsers. However, in August 1996, Microsoft's browser, Internet Explorer, showed support for CSS. Next in line to support CSS was Netscape. However, some additional developments were carried out to make the CSS pages run in this browser smoothly.

In December 1996, the CSS Level 1 W3C Recommendation was made. This release supported font properties, text attributes, alignment of text, tables, images and more, colors of text and backgrounds, spacing of words, letters and lines, margins, borders, padding and positioning, and unique identification and classification of groups of attributes.

In May 1998, the CSS Level 2 was released. It was an upgraded version of the CSS Level 1. The added new capabilities included z-index, media types, bidirectional text, and absolute, relative and fixed positioning, and support for aural style sheets. Not long after the release of CSS 2, a new browser, Opera was released which also supported CSS. It wasn't until June 2011 that W3C released CSS 2.1, which fixed errors and aligned the capabilities better with browser functions.

Once the CSS Level 3 was released, capabilities were broken into modules. Between June 2011 and June 2012, the following four modules were released as formal recommendations:

1. Color
2. Selectors level 3
3. Namespaces
4. Media queries

Since the CSS Level 3 comes in separate modules, a separate release of CSS 4 is not coming up.

How to Add CSS: Internal Styles, External Styles, Inline Styles

When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet. There are basically 3 Ways to Insert CSS. These are:

- External CSS
- Internal CSS
- Inline CSS

External CSS

With an external style sheet, you can change the look of an entire website by changing just one file!

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

External styles are defined within the <link> element, inside the <head> section of an HTML page. There will be two separate files: 1) HTML, 2) CSS.

Let's look at a sample example:

The HTML File

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
<body>

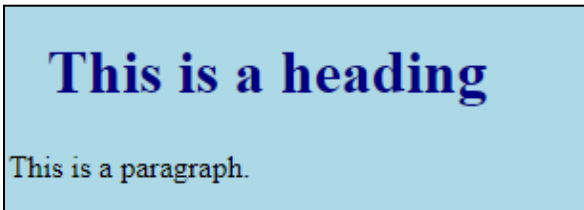
<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

The CSS File

```
body {  
  background-color: lightblue;  
}  
  
h1 {  
  color: navy;  
  margin-left: 20px;  
}
```

The output will be:



Internal CSS

An internal style sheet may be used if one single HTML page has a unique style.

The internal style is defined inside the <style> element, inside the head section.

Internal styles are defined within the <style> element, inside the <head> section of an HTML page. Let's observe an example:

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
body {  
  background-color: linen;  
}  
  
h1 {  
  color: maroon;  
  margin-left: 40px;  
}  
</style>  
</head>  
<body>  
  
<h1>This is a heading</h1>  
<p>This is a paragraph.</p>  
  
</body>  
</html>
```

Inline CSS

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

Inline styles are defined within the "style" attribute of the relevant element:

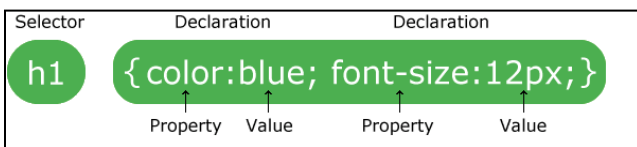
```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```

General Syntax

A CSS rule-set consists of a selector and a declaration block:



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

CSS Comments

Comments are used to explain the code, and may help when you edit the source code at a later date.

Comments are ignored by browsers.

A CSS comment starts with `/*` and ends with `*/`. Comments can also span multiple lines. Let's observe an example:

```
p {  
  color: red;  
  /* This is a single-line comment */  
  text-align: center;  
}  
  
/* This is  
a multi-line  
comment */
```

CSS Selectors

CSS selectors are used to "find" (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

1. Simple selectors (select elements based on name, id, class)
2. Combinator selectors (select elements based on a specific relationship between them)
3. Pseudo-class selectors (select elements based on a certain state)
4. Pseudo-elements selectors (select and style a part of an element)
5. Attribute selectors (select elements based on an attribute or attribute value)

Cascade

The CSS Cascade is the algorithm by which the browser decides which CSS styles to apply to an element—a lot of people like to think of this as the style that “wins”. The cascade takes an unordered list of declared values for a given property on a given element, sorts them by their declaration’s precedence, and outputs a single cascaded value.

To understand the CSS cascade better, it’s helpful to think of a CSS declaration as having “attributes”. These attributes could be various parts of the declaration—like the selector or the CSS properties—or they can be related to where the CSS declaration exists (like its origin or the position in the source code).

The CSS cascade takes a few of these attributes and assigns each of them a weight. If a CSS rule wins at a higher-priority level, that’s the rule that gets wins.

Browser support for CSS3

Most of the modern-day browsers support CSS3. Popular browsers like Internet Explorer, Mozilla, Firefox, Chrome, and Safari never have shown problems in running CSS3 files.

Reviewing the Chapter

- CSS3 is the upgraded version of Cascading Style Sheets. The new upgraded version introduces new features that definitely help in improving the overall webpage appearance.
- When developed a page using CSS3, it will not only look attractive but the simplicity to navigate and flawless functioning definitely makes web accessibility a fascinating experience.
- There are basically 3 Ways to Insert CSS. These are External CSS, Internal CSS, and Inline CSS
- With an external style sheet, you can change the look of an entire website by changing just one file!
- An internal style sheet may be used if one single HTML page has a unique style.
- An inline style may be used to apply a unique style for a single element.
- The selector points to the HTML element you want to style.
- A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.
- Comments are used to explain the code, and may help when you edit the source code at a later date.
- The cascade takes an unordered list of declared values for a given property on a given element, sorts them by their declaration's precedence, and outputs a single cascaded value.
- Most of the modern-day browsers support CSS3

Testing your Skills

1. CSS Level 2 was released on which month of 1998
a) February, b) May, c) September, d) November
2. Which of the following modules are parts of CSS Level 3
a) Namespace, b) Color, c) Selectors Level 3, d) All of the above
3. In which form of CSS3 you can change the look of an entire website by changing just one file
a) External b) Internal c) Both (a) and (b), d) None of the above
4. The External CSS styles are defined within the <> element
a) Style, b) Link, c) EXT, d) ExtStyle
5. The CSS declaration blocks are always surrounded by
a) curly braces, b) square braces, c) regular braces, d) None of the above

6. A comment starts with _ and ends with _

a) <> </>, b) ()* *(), c) /* *//, d) #<> <>#

7. ____ are used to "find" (or select) the HTML elements you want to style

a) Cascade, b) Selectors, c) Inline CSS, d) None of the above

8. Which of the following are type of CSS Selectors?

a) Pseudo Class, b) Pseudo Element, c) Attribute, d) All of the above

9. Select elements based on a certain state is known as _____ selector

a) Combinator, b) Simple, c) Pseudo Class, d) Attribute

10. The _____ takes an unordered list of declared values for a given property on a given element

a) Cascade, b) Pseudo Element Selector, c) Combinator, d) None of the above

Lesson 13: Selectors, Pseudo Classes, and Pseudo Elements

Objective: After completing this lesson you will be able to : <ul style="list-style-type: none">• Understand Selectors and its various types• Learn about Pseudo Classes• Learn about Pseudo Elements	Materials Required: <ul style="list-style-type: none">• Computer With Windows XP and above• Stable Internet connection
Self- Learning Duration: 60 minutes	Practical Duration: 60 minutes
Total Duration: 120 minutes	

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

There are several different types of selectors in CSS.

- CSS Element Selector
- CSS Id Selector
- CSS Class Selector
- CSS Universal Selector
- CSS Group Selector

1) CSS Element Selector

The element selector selects the HTML element by name.

2) CSS Id Selector

The id selector selects the id attribute of an HTML element to select a specific element. An id is always unique within the page so it is chosen to select a single, unique element.

It is written with the hash character (#), followed by the id of the element.

3) CSS Class Selector

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

CSS Class Selector for specific element

If you want to specify that only one specific HTML element should be affected then you should use the element name with class selector.

4) CSS Universal Selector

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

5) CSS Group Selector

The grouping selector is used to select all the elements with the same style definitions.

Grouping selector is used to minimize the code. Commas are used to separate each selector in grouping.

Pseudo Classes

What are Pseudo-classes?

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

Pseudo Elements

A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

- Style the first letter, or line, of an element
- Insert content before, or after, the content of an element

CSS - The ::before Pseudo-element

The ::before pseudo-element can be used to insert some content before the content of an element.

The following example inserts an image before the content of each <h1> element:

Example

```
h1::before {  
  content: url(smiley.gif);  
}
```

CSS - The ::after Pseudo-element

The ::after pseudo-element can be used to insert some content after the content of an element.

The following example inserts an image after the content of each <h1> element:

Example

```
h1::after {  
  content: url(smiley.gif);  
}
```

CSS - The ::selection Pseudo-element

The ::selection pseudo-element matches the portion of an element that is selected by a user.

The following CSS properties can be applied to ::selection: color, background, cursor, and outline.

The following example makes the selected text red on a yellow background:

Example

```
::selection {  
  color: red;  
  background: yellow;  
}
```

CSS :nth-child() Selector

The :nth-child(n) selector matches every element that is the nth child, regardless of type, of its parent.

n can be a number, a keyword, or a formula.

Syntax:

```
p:nth-child(2) {  
  background: red;  
}
```

CSS :root Selector

The :root selector matches the document's root element.

In HTML, the root element is always the html element.

Syntax:

```
:root {  
  background: #ff0000;  
}
```

Reviewing the Chapter

- CSS selectors are used to select the content you want to style. Selectors are the part of CSS rule set. CSS selectors select HTML elements according to its id, class, type, attribute etc.
- A pseudo-class is used to define a special state of an element.
- A CSS pseudo-element is used to style specified parts of an element.
- The ::before pseudo-element can be used to insert some content before the content of an element
- The ::after pseudo-element can be used to insert some content after the content of an element
- The ::selection pseudo-element matches the portion of an element that is selected by a user
- The :nth-child(n) selector matches every element that is the nth child, regardless of type, of its parent
- The :root selector matches the document's root element

Testing Your Skills

1. _____ is used to style specified parts of an element
a) Pseudo Element, b) Pseudo Class, c) CSS ID Selector, d) None of the above
2. _____ matches the portion of an element that is selected by a user
a) after pseudo-element, b) selection pseudo-element, c) before pseudo-element, d) nth child
3. _____ is used to define a special state of an element.
a) Pseudo Element, b) Pseudo Class, c) CSS ID Selector, d) None of the above
4. _____ Selector selects all the elements on the pages
a) Group Selector, b) Class, c) Universal Selector, d) ID
5. Which of the following are CSS Selectors?
a) Group Selector, b) Class, c) Universal Selector, d) All of the above

Lesson 14: Fonts and Text Effect

Objective: After completing this lesson you will be able to : <ul style="list-style-type: none">• Understand @font-face• Learn about Web Fonts: Google, Typekit, etc.• Learn about Text Shadow, Text Outline, Text Stroke, Word Wrapping	Materials Required: <ul style="list-style-type: none">• Computer With Windows XP and above• Stable Internet connection
Self- Learning Duration: 60 minutes	Practical Duration: 60 minutes
Total Duration: 120 minutes	

@font-face

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.

Without the rule, our designs are limited to the fonts that are already loaded on a user's computer, which vary depending on the system being used.

@font-face method comes with the deepest support possible right now. The @font-face rule should be added to the stylesheet before any styles. Have a look at the sample example:

```
CSS

@font-face {
  font-family: 'MyWebFont';
  src: url('webfont.eot'); /* IE9 Compat Modes */
  src: url('webfont.eot?#iefix') format('embedded-opentype'), /* IE6-IE8 */
  url('webfont.woff2') format('woff2'), /* Super Modern Browsers */
  url('webfont.woff') format('woff'), /* Pretty Modern Browsers */
  url('webfont.ttf') format('truetype'), /* Safari, Android, iOS */
  url('webfont.svg#svgFontName') format('svg'); /* Legacy iOS */
}
```

Once you create this coding, you can easily use this for styling elements. Have a look at the type of coding you need to develop:

```
CSS

body {
  font-family: 'MyWebFont', Fallback, sans-serif;
}
```

Understanding Typekit

Typekit is a web font service that serves thousands of high class fonts by various foundries to your website.

Typekit is great for you because it scales easily. Typekit allows you to pay for how much you use. So for instance, it works with a small site with thousands of page views per month to a much bigger site with millions of page views per month. Typekit is trusted by WordPress, New York Times and many more. Secondly, it has a great collection of fonts by high caliber font foundries.

Google Fonts

"Google Fonts first launched in 2010 as an engineering initiative to move the web forward and make it faster." — Google Design

Google Fonts launched in 2010, quickly becoming the Internet's largest, free, open-source selection of fonts. All Google Fonts are free for commercial and personal use. The Google Fonts website makes it easy for anyone to quickly select and utilize different fonts for their own design needs.

Who uses Google Fonts?

Everyone does! Graphic designers, UX designers, researchers, developers, web designers, bloggers, social media managers, entrepreneurs, artists, students, teachers, photographers, and many more. I've seen Google Fonts used on billboards, posters, presentation decks, wedding invitations, websites, and books.

Font Properties

The font property is a shorthand property for:

- font-style
- font-variant
- font-weight
- font-size/line-height
- font-family

The font-size and font-family values are required. If one of the other values is missing, their default values are used.

The line-height property sets the space between lines.

Property Values

- font-style - Specifies the font style. Default value is "normal"
- font-variant - Specifies the font variant. Default value is "normal"
- font-weight - Specifies the font weight. Default value is "normal"
- font-size/line-height - Specifies the font size and the line-height. Default value is "normal"
- font-family - Specifies the font family. Default value depends on the browser
- caption - Uses the font that are used by captioned controls (like buttons, drop-downs, etc.)

- icon - Uses the font that are used by icon labels
- menu - Uses the fonts that are used by dropdown menus
- message-box - Uses the fonts that are used by dialog boxes
- small-caption - A smaller version of the caption font
- status-bar - Uses the fonts that are used by the status bar
- initial - Sets this property to its default value. Read about initial
- inherit - Inherits this property from its parent element. Read about inherit

Text Shadow

The text-shadow property adds shadow to text.

This property accepts a comma-separated list of shadows to be applied to the text.

The Outline Property

An outline is a line that is drawn around elements, outside the borders, to make the element "stand out".

The outline property is a shorthand property for:

- outline-width
- outline-style (required)
- outline-color

If outline-color is omitted, the color applied will be the color of the text.

Note: The outline is not a part of the element's dimensions, therefore the element's width and height properties do not contain the width of the outline.

Text Stroke

text-stroke is an experimental property that provides text decoration options similar to those found in Adobe Illustrator or other vector drawing applications. It is not currently included in any W3C or WHATWG specification. As of June 2013, it is only implemented behind a -webkit vendor prefix, though future versions of Firefox and Internet Explorer may support the property (likely under their own prefixes).

The text-stroke property is actually shorthand for two other properties:

1. text-stroke-width, which takes unit value (1px, 0.125em, 4in, etcetera) and describes the thickness of the stroke effect.
2. text-stroke-color, which takes a color value (hex, rgb/rgba, hsl/hsla, etcetera).

text-stroke also has a companion property, text-fill-color, which will override the color property, allowing for a graceful fallback to a different text color in browsers that do not support text-stroke.

Word Wrapping

The word-wrap property allows long words to be able to be broken and wrap onto the next line.

Reviewing the Chapter

- The @font-face rule allows custom fonts to be loaded on a webpage
- Typekit is a web font service that serves thousands of high class fonts by various foundries to your website. Typekit is great for you because it scales easily
- The Google Fonts website makes it easy for anyone to quickly select and utilize different fonts for their own design needs
- The text-shadow property adds shadow to text
- The word-wrap property allows long words to be able to be broken and wrap onto the next line

Testing your Skills

1. _____ allows custom fonts to be loaded on a webpage
a) Typekit, b) @Font-face, c) font style, d) None of the above
2. Google Fonts was first launched in the year _____
a) 2001, b) 2005, c) 2010, d) 1998
3. _____ has a great collection of fonts by high caliber font foundries
a) Typekit, b) @Font-Face, c) Google Fonts, d) None of the above
4. _____ specifies the font family
a) font-style, b) font-variant, c) font-size, d) font-family
5. _____ uses the fonts that are used by dialog boxes
a) menu, b) message box, c) small caption, d) None of the above
6. This property allows long words to be able to be broken and wrap onto the next line
a) word wrapping, b) text stroke, c) outline, d) caption

Lesson 15: Color, Gradients, Background Images, and Masks

Objective: After completing this lesson you will be able to learn about :

- Opacity
- New Color Names
- Nonimage Gradients
- Multiple Background Images
- Background-size, Background-origin, Background-clip
- Image Masks

Materials Required:

- Computer With Windows XP and above
- Stable Internet connection

Self- Learning Duration: 60 minutes

Practical Duration: 60 minutes

Total Duration: 120 minutes

Opacity

The opacity property specifies the opacity/transparency of an element. The opacity property can take a value from 0.0 - 1.0. The lower value, the more transparent it will become.

The opacity property is often used together with the :hover selector to change the opacity on mouse-over:

```
img {
  opacity: 0.5;
  filter: alpha(opacity=50); /* For IE8 and earlier */
}

img:hover {
  opacity: 1.0;
  filter: alpha(opacity=100); /* For IE8 and earlier */
}
```

New Color Names

Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.

Focus on the chart below:

Hex	#FFFFFF
RGB	rgb(255, 255, 255)
HSL	hsl(0, 0%, 100%)
HSV	hsv(0, 0%, 100%)
HWB	hwb(0, 100%, 0%)
CMYK	cmyk(0%, 0%, 0%, 0%)

Each color can be represented in many different ways. For example blue can also be represented as #0000ff, #00f, rgb(0,0,255) and many other ways. It doesn't matter which one you choose as long as it's a valid color.

You can apply any of these colors to a website or blog by using the relevant CSS code.

- To set a background color, use background-color.
- To set the text color, use color
- To set a border color, use border-color.

Gradients

CSS gradients let you display smooth transitions between two or more specified colors.

CSS defines two types of gradients:

- Linear Gradients (goes down/up/left/right/diagonally)
- Radial Gradients (defined by their center)

CSS Linear Gradients

To create a linear gradient you must define at least two color stops. Color stops are the colors you want to render smooth transitions among. You can also set a starting point and a direction (or an angle) along with the gradient effect.

CSS Radial Gradients

A radial gradient is defined by its center.

To create a radial gradient you must also define at least two color stops.

Multiple Backgrounds

CSS allows you to add multiple background images for an element, through the background-image property.

The different background images are separated by commas, and the images are stacked on top of each other, where the first image is closest to the viewer.

The following example has two background images, the first image is a flower (aligned to the bottom and right) and the second image is a paper background (aligned to the top-left corner):

```
#example1 {  
  background-image: url(img_flwr.gif), url(paper.gif);  
  background-position: right bottom, left top;  
  background-repeat: no-repeat, repeat;  
}
```

Multiple background images can be specified using either the individual background properties (as above) or the background shorthand property.

The following example uses the background shorthand property (same result as example above):

```
#example1 {  
  background: url(img_flwr.gif) right bottom no-repeat, url(paper.gif) left top repeat;  
}
```

Background Size

The CSS background-size property allows you to specify the size of background images.

The size can be specified in lengths, percentages, or by using one of the two keywords: contain or cover.

The contain keyword scales the background image to be as large as possible (but both its width and its height must fit inside the content area). As such, depending on the proportions of the background image and the background positioning area, there may be some areas of the background which are not covered by the background image.

The cover keyword scales the background image so that the content area is completely covered by the background image (both its width and height are equal to or exceed the content area). As such, some parts of the background image may not be visible in the background positioning area.

Background-origin

The CSS background-origin property specifies where the background image is positioned.

The property takes three different values:

- border-box - the background image starts from the upper left corner of the border
- padding-box - (default) the background image starts from the upper left corner of the padding edge
- content-box - the background image starts from the upper left corner of the content

Background-clip

The CSS background-clip property specifies the painting area of the background.

The property takes three different values:

- border-box - (default) the background is painted to the outside edge of the border
- padding-box - the background is painted to the outside edge of the padding
- content-box - the background is painted within the content box

Image Masks

Masking is much similar to a clipping property, but with some differences. A mask image is used as some kind of "color mesh", to filter the visual portions of an element. In the following paragraphs I'll explain the difference between two kind of masks: the luminance mask and the alpha mask.

Reviewing the chapter

- The opacity property specifies the opacity/transparency of an element.
- Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values
- CSS gradients let you display smooth transitions between two or more specified colors
- Color stops are the colors you want to render smooth transitions among
- CSS allows you to add multiple background images for an element, through the background-image property.
- Multiple background images can be specified using either the individual background properties (as above) or the background shorthand property.
- The CSS background-size property allows you to specify the size of background images.
- The cover keyword scales the background image so that the content area is completely covered by the background image
- The CSS background-origin property specifies where the background image is positioned
- The CSS background-clip property specifies the painting area of the background

Testing your Skills

1. _____ property specifies the transparency of an element

a) Opacity, b) Gradient, c) Background, d) Image Mask

2. _____ selector is used to change the opacity on mouse-over

a) :hover, b) :opacity, c) :transparency, d) none of the above

3. CSS defines _____ types of gradients

a) 4, b) 2, c) 3, d) 5

4. The size of a background image can be specified by

a) length, b) percentage, c) contain/cover keywords, d) all of the above

5. Here the background image starts from the upper left corner of the content

a) border box, b) padding box, c) content box, d) None of the above

Lesson 16: Border and Box Effects**Objective:** After completing this lesson you will be able to learn about :

- Different border properties in CSS
- Rounded corners
- Resizing, outline offsets, image borders, and box shadow

Materials Required:

- Computer With Windows XP and above
- Stable Internet connection

Self- Learning Duration: 60 minutes**Practical Duration:** 60 minutes**Total Duration:** 120 minutes**CSS Border Properties**

The CSS border properties allow you to specify the style, width, and color of an element's border.

Rounded Corners

The border-radius property is used to add rounded borders to an element.

A sample example:

```
<!DOCTYPE html>

<html>

<head>

<style>

p.normal {

    border: 2px solid red;

}

p.round1 {

    border: 2px solid red;

    border-radius: 5px;

}

p.round2 {

    border: 2px solid red;

    border-radius: 8px;

}

p.round3 {

    border: 2px solid red;

    border-radius: 12px;
```

```
}  
</style>  
</head>  
<body>  
<h2>The border-radius Property</h2>  
<p>This property is used to add rounded borders to an element:</p>  
<p class="normal">Normal border</p>  
<p class="round1">Round border</p>  
<p class="round2">Rounder border</p>  
<p class="round3">Roundest border</p>  
<p><b>Note:</b> The "border-radius" property is not supported in IE8 and earlier versions.</p>  
</body>  
</html>
```

The output:

The border-radius Property

This property is used to add rounded borders to an element:

Normal border

Round border

Rounder border

Roundest border

Note: The "border-radius" property is not supported in IE8 and earlier versions.

Resizing

The resize property defines if (and how) an element is resizable by the user.

The resize property does not apply to inline elements or to block elements where overflow="visible". So, make sure that overflow is set to "scroll", "auto", or "hidden".

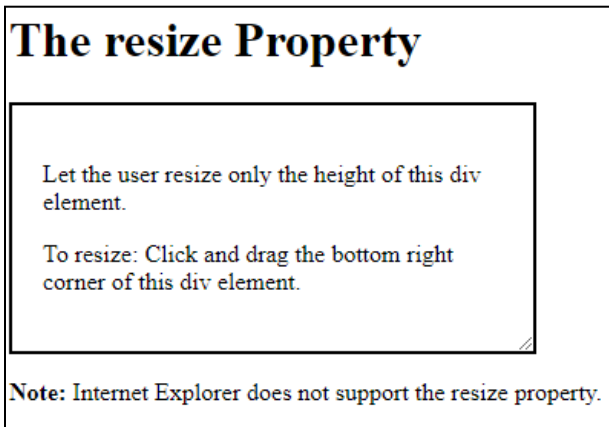
Property values:

- None - Default value where the user cannot resize the element
- Both - The user can resize both the height and width of the element
- Horizontal - The user can resize the width of the element
- Vertical - The user can resize the height of the element
- Initial - Sets this property to its default value.
- Inherit - Inherits this property from its parent element.

A sample example:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 2px solid;
  padding: 20px;
  width: 300px;
  resize: vertical;
  overflow: auto;
}
</style>
</head>
<body>
<h1>The resize Property</h1>
<div>
  <p>Let the user resize only the height of this div element.</p>
  <p>To resize: Click and drag the bottom right corner of this div element.</p>
</div>
<p><b>Note:</b> Internet Explorer does not support the resize property.</p>
</body>
</html>
```

The output:



Outline Offsets

The outline-offset property adds space between an outline and the edge or border of an element.

The space between an element and its outline is transparent.

Outlines differ from borders in three ways:

- An outline is a line drawn around elements, outside the border edge
- An outline does not take up space
- An outline may be non-rectangular

A sample example:

```
<!DOCTYPE html>
<html>
<head>
<style>
div.ex1 {
  margin: 20px;
  border: 1px solid black;
  background-color: yellow;
  outline: 4px solid red;
  outline-offset: 15px;
}
div.ex2 {
  margin: 10px;
  border: 1px solid black;
  background-color: yellow;
  outline: 5px dashed blue;
  outline-offset: 5px;
}
</style>
</head>
<body>
<h1>The outline-offset Property</h1>
<div class="ex1">This div has a 4 pixels solid red outline 15 pixels outside the border edge.</div>
```



```
<br>  
<div class="ex2">This div has a 5 pixels dashed blue outline 5 pixels outside the border edge.</div>  
</body>  
</html>
```

The output:

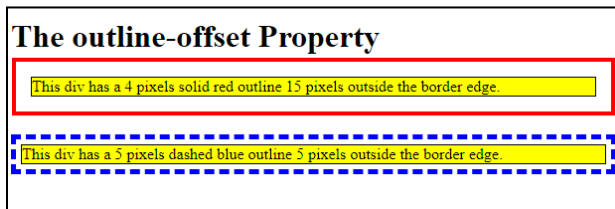


Image Borders

The CSS border-image property allows you to specify an image to be used instead of the normal border around an element.

The property has three parts:

1. The image to use as the border
2. Where to slice the image
3. Define whether the middle sections should be repeated or stretched

A sample example:

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
#borderimg {  
    border: 10px solid transparent;  
    padding: 15px;  
    border-image: url(border.png) 30 round;  
}  
</style>  
</head>  
<body>  
<h1>The border-image Property</h1>
```

<p>Here, the middle sections of the image are repeated to create the border:</p>

<p id="borderimg">border-image: url(border.png) 30 round;</p>

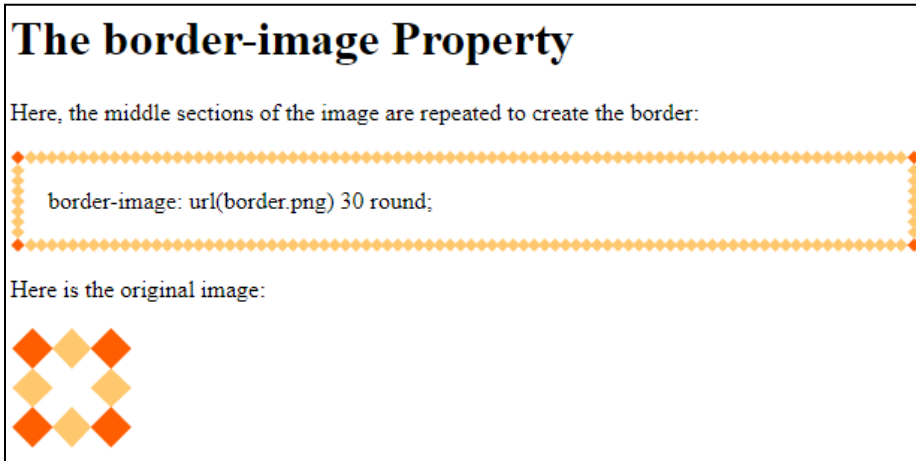
<p>Here is the original image:</p>

<p>Note: Internet Explorer 10, and earlier versions, do not support the border-image property.</p>

</body>

</html>

The output:



Box Shadow

The box-shadow property attaches one or more shadows to an element.

To attach more than one shadow to an element, add a comma-separated list of shadows

A sample example:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
#example1 {
```

```
  border: 1px solid;
```

```
  padding: 10px;
```

```
  box-shadow: 5px 10px;
```

```
}
```

```
#example2 {
```

```
  border: 1px solid;
```

```
padding: 10px;
box-shadow: 5px 10px #888888;
}
#example3 {
border: 1px solid;
padding: 10px;
box-shadow: 5px 10px red;
}
</style>
</head>
<body>
<h1>The box-shadow Property</h1>
<p>The box-shadow property defines the shadow of an element:</p>
<h2>box-shadow: 5px 10px:</h2>
<div id="example1">
  <p>A div element with a shadow. The first value is the horizontal offset and the second value is the vertical offset. The shadow color will be inherited from the text color.</p>
</div>
<h2>box-shadow: 5px 10px #888888:</h2>
<div id="example2">
  <p>You can also define the color of the shadow. Here the shadow color is grey.</p>
</div>
<h2>box-shadow: 5px 10px red:</h2>
<div id="example3">
  <p>A red shadow.</p>
</div>
</body>
</html>
```

The output:

The box-shadow Property

The box-shadow property defines the shadow of an element:

box-shadow: 5px 10px;

A div element with a shadow. The first value is the horizontal offset and the second value is the vertical offset. The shadow color will be inherited from the text color.

box-shadow: 5px 10px #888888;

You can also define the color of the shadow. Here the shadow color is grey.

box-shadow: 5px 10px red;

A red shadow.

Reviewing the chapter

- The CSS border properties allow you to specify the style, width, and color of an element's border.
- The border-radius property is used to add rounded borders to an element.
- The resize property defines if (and how) an element is resizable by the user.
- The resize property does not apply to inline elements or to block elements where overflow="visible". So, make sure that overflow is set to "scroll", "auto", or "hidden".
- The outline-offset property adds space between an outline and the edge or border of an element.
- The space between an element and its outline is transparent.
- The CSS border-image property allows you to specify an image to be used instead of the normal border around an element.
- The box-shadow property attaches one or more shadows to an element.

Testing your skills

1. _____ sets this Resizing Property to its default value

a) Initial, b) Inherent, c) Both, d) None

2. _____ property adds space between an outline and the edge or border of an element

a) outline-offset, b) inline-offset, c) offset, d) none of the above

3. _____ property allows you to specify an image to be used instead of the normal border around an element

a) box shadow, b) border image, c) outline offset, d) none of the above

4. To attach more than one shadow to an element, add a _____-separated list of shadows

a) semicolon, b) colon, c) hash, d) comma

5. Which resizing property is used to resize the width of the element?

a) width, b) vertical, c) horizontal, d) initial

Lesson 17: Working with Colors

Objective: After completing this lesson you will be able to learn about : <ul style="list-style-type: none"> • Different border properties in CSS • Rounded corners • Resizing, outline offsets, image borders, and box shadow 	Materials Required: <ul style="list-style-type: none"> • Computer With Windows XP and above • Stable Internet connection
Self- Learning Duration: 60 minutes	Practical Duration: 60 minutes
Total Duration: 120 minutes	

Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.

CSS/HTML supports 140 standard color names.

Background

You can set the background color for HTML elements. Let's observe an example:

```
<!DOCTYPE html>
<html>
<body>
<h1 style="background-color:DodgerBlue;">Hello World</h1>
```

```
<p style="background-color:Tomato;">
```

Lorem ipsum dolor sit amet, consectetur 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>
```

The output:

Hello World

Lorem ipsum dolor sit amet, consectetur 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.

Multiple Background

CSS allows you to add multiple background images for an element, through the background-image property.

The different background images are separated by commas, and the images are stacked on top of each other, where the first image is closest to the viewer.

The following example has two background images, the first image is a flower (aligned to the bottom and right) and the second image is a paper background (aligned to the top-left corner):

```
<!DOCTYPE html>
<html>
<head>
<style>
#example1 {
  background-image: url(img_flwr.gif), url(paper.gif);
  background-position: right bottom, left top;
  background-repeat: no-repeat, repeat;
  padding: 15px;
}
</style>
</head>
<body>

<h1>Multiple Backgrounds</h1>
<p>The following div element has two background images:</p>

<div id="example1">
  <h1>Lorem Ipsum Dolor</h1>
```

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. </p>

<p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. </p>

</div>

</body>

</html>

The output:

Multiple Backgrounds

The following div element has two background images:



Colors

RGB and RGBA

RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.

An RGBA color value is specified with: `rgba(red, green, blue, alpha)`. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

HSL and HSLA

HSL stands for Hue, Saturation and Lightness.

An HSL color value is specified with: `hsl(hue, saturation, lightness)`.

Hue is a degree on the color wheel (from 0 to 360):

- 0 (or 360) is red
- 120 is green
- 240 is blue

Saturation is a percentage value: 100% is the full color.

Lightness is also a percentage; 0% is dark (black) and 100% is white.

HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.

An HSLA color value is specified with: `hsla(hue, saturation, lightness, alpha)`, where the alpha parameter defines the opacity. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

Opacity & Transparency

The CSS opacity property sets the opacity for the whole element (both background color and text will be opaque/transparent). The opacity property value must be a number between 0.0 (fully transparent) and 1.0 (fully opaque).

Using `currentColor`

`currentColor` is a CSS keyword that enabled more DRY (Don't Repeat Yourself) stylesheets. It can be used wherever a color value is expected (i.e. `background-color`, `border-color`, `outline-color`, `box-shadow`, etc).

Working with gradients

CSS gradients let you display smooth transitions between two or more specified colors.

CSS defines two types of gradients:

- Linear Gradients (goes down/up/left/right/diagonally)
- Radial Gradients (defined by their center)

Reviewing the chapter

- Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.
- CSS/HTML supports 140 standard color names
- CSS allows you to add multiple background images for an element, through the `background-image` property.
- The different background images are separated by commas, and the images are stacked on top of each other, where the first image is closest to the viewer.
- RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.
- HSL stands for Hue, Saturation and Lightness.

- HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.
- The opacity property value must be a number between 0.0 (fully transparent) and 1.0 (fully opaque).
- currentColor is a CSS keyword that enabled more DRY (Don't Repeat Yourself) stylesheets
- CSS gradients let you display smooth transitions between two or more specified colors.

Testing your skills

1. Which of these are form of colors:

a) RGB, b) HEX, c) Both (a) and (b), d) Only (a)

2. _____ property is used to add multiple background images to an element

a) multi-image, b) background-image, c) multiple-background, d) none of the above

3. _____ specifies opacity of a color

a) RGBA, b) HSLA, c) HSL, d) none of the above

4. currentColor is a CSS keyword that enabled more _____ stylesheets

a) HSL, b) CRY, c) DRY, d) Inherent

5. Which gradient is defined by the center?

a) Radial, b) Vertical, c) Linear, d) Horizontal

Lesson 18: Layout

Objective: After completing this lesson you will be able to learn about :

- Different layouts
- Positioning, Box Layout, Table Layout
- Vendor Prefixes
- Working with Columns

Materials Required:

- Computer With Windows XP and above
- Stable Internet connection

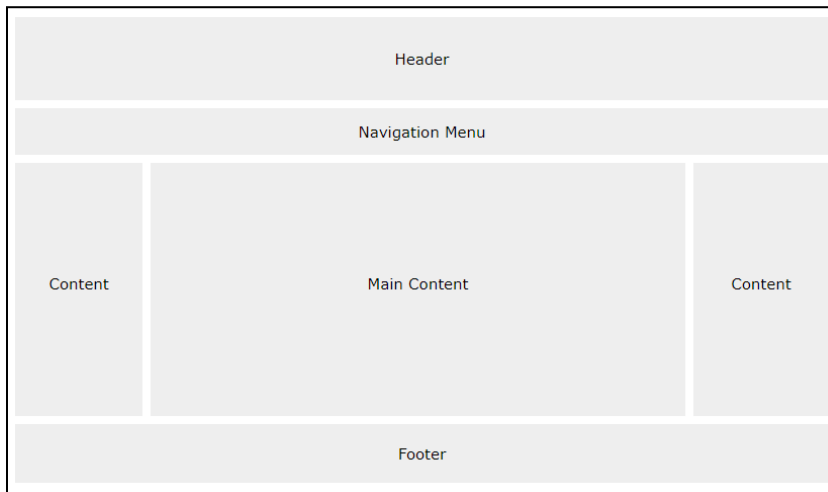
Self- Learning Duration: 60 minutes

Practical Duration: 60 minutes

Total Duration: 120 minutes

Introduction

A website is often divided into headers, menus, content and footer. There are tons of different layout designs to choose from. However, the most common layout format is provided below. Have a look:



Positioning

The position Property

The position property specifies the type of positioning method used for an element.

There are five different position values:

- static
- relative
- fixed
- absolute
- sticky

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

The Static position

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page. Have a look at a sample example with output:

<pre><!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></pre>	<p>position: static;</p> <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>This div element has position: static;</div>
--	--

The Relative position

An element with position: relative; is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element. Have a look at a sample example with output:

<pre><!DOCTYPE html> <html> <head> <style> div.relative { position: relative; left: 30px; border: 3px solid #73AD21; } </style> </head> <body> <h2>position: relative;</h2> <p>An element with position: relative; is positioned relative to its normal position:</p> <div class="relative"> This div element has position: relative; </div> </body> </html></pre>	<p>position: relative;</p> <p>An element with position: relative; is positioned relative to its normal position:</p> <div>This div element has position: relative;</div>
--	---

The Fixed position

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

A fixed element does not leave a gap in the page where it would normally have been located. Have a look at a sample example with output:

```

<!DOCTYPE html>
<html>
<head>
<style>
div.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>

<h2>position: fixed;</h2>

<p>An element with position: fixed; is positioned relative to the
viewport, which means it always stays in the same place even if the
page is scrolled:</p>

<div class="fixed">
This div element has position: fixed;
</div>

</body>
</html>

```

position: fixed;

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled:

This div element has position: fixed;

The Absolute position

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed). However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

Keep in mind, A "positioned" element is one whose position is anything except static. Have a look at a sample example with output:

```

<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
  position: relative;
  width: 400px;
  height: 200px;
  border: 3px solid #73AD21;
}
div.absolute {
  position: absolute;
  top: 80px;
  right: 0;
  width: 200px;
  height: 100px;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>

<h2>position: absolute;</h2>

<p>An element with position: absolute; is positioned relative to the nearest positioned
ancestor (instead of positioned relative to the viewport, like fixed):</p>

<div class="relative">This div element has position: relative;
<div class="absolute">This div element has position: absolute;</div>
</div>

</body>
</html>

```

position: absolute;

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed):

This div element has position: relative;

This div element has position: absolute;

The Sticky position

An element with position: sticky; is positioned based on the user's scroll position.

A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed). Have a look at a sample example with output:

```

<!DOCTYPE html>
<html>
<head>
<style>
div.sticky {
  position: -webkit-sticky;
  position: sticky;
  top: 0;
  padding: 5px;
  background-color: #c6e0b4;
  border: 2px solid #4caf50;
}
</style>
</head>
<body>

<p>Try to <b>scroll</b> inside this frame to understand how sticky positioning works.</p>
<p>Note: IE/Edge 15 and earlier versions do not support sticky position.</p>

<div class="sticky">I am sticky</div>

<div style="padding-bottom:200px">
<p>In this example, the sticky element sticks to the top of the page (top: 0), when you reach its
scroll position.</p>
<p>Scroll back up to remove the stickiness.</p>
<p>Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, malisest concludaturque et eum,
altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et.
Incidentint efficiantur his ad. Eum no molestiae voluptatibus.</p>
<p>Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo,
malisest concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam
omnis evertitur eum. Affert laboramus repudiandae nec et. Incidentint efficiantur his ad. Eum no
molestiae voluptatibus.</p>
<p>Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo,
malisest concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam
omnis evertitur eum. Affert laboramus repudiandae nec et. Incidentint efficiantur his ad. Eum no
molestiae voluptatibus.</p>
</div>

</body>
</html>

```

Try to scroll inside this frame to understand how sticky positioning works.

Note: IE/Edge 15 and earlier versions do not support sticky position.

I am sticky!

In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.

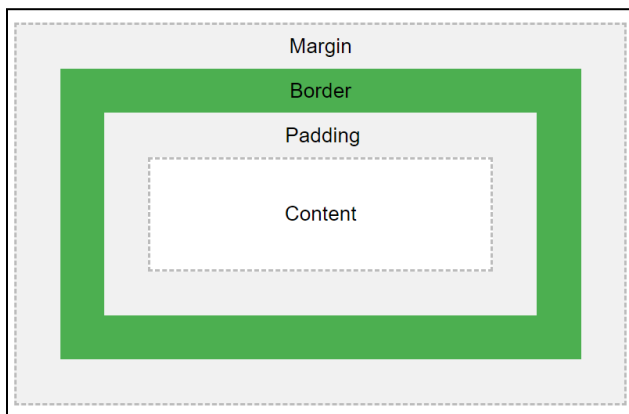
Scroll back up to remove the stickiness.

Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, malisest concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Incidentint efficiantur his ad. Eum no molestiae voluptatibus.

Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, malisest concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Incidentint efficiantur his ad. Eum no molestiae voluptatibus.

Box Layout

The CSS box layout is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:



A clearer look into these parts:

- Content - The content of the box, where text and images appear
- Padding - Clears an area around the content. The padding is transparent
- Border - A border that goes around the padding and content
- Margin - Clears an area outside the border. The margin is transparent

The box model allows us to add a border around elements, and to define space between elements. Let's have a look at an example with output:

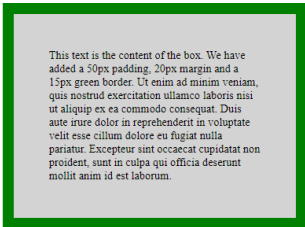
<pre><!DOCTYPE html> <html> <head> <style> div { background-color: lightgrey; width: 300px; border: 15px solid green; padding: 50px; margin: 20px; } </style> </head> <body> <h2>Demonstrating the Box Model</h2> <p>The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.</p> <div>This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</div> </body> </html></pre>	<p>Demonstrating the Box Model</p> <p>The CSS box model is essentially a box that wraps around every HTML element. It consists of: borders, padding, margins, and the actual content.</p>  A visual representation of the CSS box model. It shows a light grey rectangular box with a thick green border. Inside the box, there is a paragraph of text. The text is centered and wraps around to multiple lines. The box is set against a white background.
---	---

Table Layout

The table-layout property defines the algorithm used to lay out table cells, rows, and columns. There are four Table Layout properties. Have a look:

1. **Auto** - Browsers use an automatic table layout algorithm. The column width is set by the widest unbreakable content in the cells. The content will dictate the layout

2. **Fixed** - Sets a fixed table layout algorithm. The table and column widths are set by the widths of table and col or by the width of the first row of cells. Cells in other rows do not affect column widths. If no widths are present on the first row, the column widths are divided equally across the table, regardless of content inside the cells
3. **Initial** - Sets this property to its default value.
4. **Inherit** - Inherits this property from its parent element.

Vendor Prefixes

CSS vendor prefixes, aka CSS browser prefixes, are a way for browser makers to add support for new CSS features before those features are fully supported in all browsers. This may be done during a sort of testing and experimentation period where the browser manufacturer is determining exactly how these new CSS features will be implemented.



Do keep in mind that CSS vendors prefixes are not hacks, however, and you should have no reservations about using them in your work. A vendor prefix isn't a hack because it allows the specification to set up rules for how a property might be implemented, while at the same time allowing browser makers to implement a property in a different way without breaking everything else.

Furthermore, these prefixes are working with CSS properties that will eventually be a part of the specification. We are simply adding some code in order to get access to the property early. This is another reason why you end the CSS rule with the normal, non-prefixed property. That way you can drop the prefixed versions once full browser support is achieved.

Working with Columns

The columns property is a shorthand property for:

- column-width
- column-count

The **column-width** part will define the minimum width for each column, while the **column-count** part will define the maximum number of columns. By using this property, the multi-column layout will automatically break down into a single column at narrow browser widths, without the need of media queries or other rules.

Let's focus on a sample example with output:

```

<!DOCTYPE html>
<html>
<head>
<style>
.newspaper {
  -webkit-columns: 100px 3; /* Chrome, Safari, Opera */
  -ms-columns: 100px 3; /* Firefox */
  columns: 100px 3;
}

.newspaper2 {
  -webkit-columns: 50px 4; /* Chrome, Safari, Opera */
  -ms-columns: 50px 4; /* Firefox */
  columns: 50px 4;
}
</style>
</head>
<body>

<h1>The columns Property</h1>
<p>The columns property is a shorthand property for column-width and column-count.</p>

<div class="newspaper">
  Lorem ipsum dolor sit amet, consectetur 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 irure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et justo odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugiat nulla facilisis. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius.</div>

<div class="newspaper2">
  Lorem ipsum dolor sit amet, consectetur 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 irure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et justo odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugiat nulla facilisis. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius.</div>

<strong>Note:</strong> The columns property is not supported in Internet Explorer 9 and earlier versions.</div>
</body>
</html>

```

The columns Property

The columns property is a shorthand property for column-width and column-count.

Set the minimum width for each column to 100px, and the maximum number of columns to 3:

Lorem ipsum dolor sit amet, consectetur 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 irure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et justo odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugiat nulla facilisis. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius.

Set the minimum width for each column to 50px, and the maximum number of columns to 4:

Lorem ipsum dolor sit amet, consectetur 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 irure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et justo odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugiat nulla facilisis. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius.

Note: The columns property is not supported in Internet Explorer 9 and earlier versions.

Layout: Columns, Flexible Box

Column-count, Column-gap, Column-rule

The **column-count** property specifies the number of columns an element should be divided into.

Let's focus on a sample example with output:

```

<!DOCTYPE html>
<html>
<head>
<style>
.newspaper {
  -webkit-column-count: 3; /* Chrome, Safari, Opera */
  -ms-column-count: 3; /* Firefox */
  column-count: 3;
}
</style>
</head>
<body>

<h1>The column-count Property</h1>
<p>The column-count property defines the number of columns an element is divided into.</p>

<div class="newspaper">
  Lorem ipsum dolor sit amet, consectetur 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 irure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et justo odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugiat nulla facilisis. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius.</div>

<p><strong>Note:</strong> The column-count property is not supported in Internet Explorer 9 and earlier versions.</p>
</body>
</html>

```

The column-count Property

The column-count property defines the number of columns an element is divided into:

Lorem ipsum dolor sit amet, consectetur 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 irure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et justo odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugiat nulla facilisis. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius.

Note: The column-count property is not supported in Internet Explorer 9 and earlier versions.

The **column-gap** property specifies the gap between the columns.

Let's focus on a sample example with output:

```

<!DOCTYPE html>
<html>
<head>
<style>
.newspaper {
  -webkit-column-count: 3; /* Chrome, Safari, Opera */
  -moz-column-count: 3; /* Firefox */
  column-count: 3;

  -webkit-column-gap: 40px; /* Chrome, Safari, Opera */
  -moz-column-gap: 40px; /* Firefox */
  column-gap: 40px;
}
</style>
</head>
<body>

<h1>The column-gap Property</h1>
<p>The column-gap property defines the gap between the columns of the element:</p>

<div class="newspaper">
  Lorem ipsum dolor sit amet, consectetur 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 dui dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius.
</div>

<p><strong>Note:</strong> The column-gap property is not supported in Internet Explorer 9 and earlier versions.</p>
</body>
</html>

```

The column-gap Property

The column-gap property defines the gap between the columns of the element:

Lorem ipsum dolor sit amet, consectetur 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 dui dolore te feugait nulla facilisi. Nam liber tempor cum soluta	nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius.
---	---	---

Note: The column-gap property is not supported in Internet Explorer 9 and earlier versions.

The **column-rule** property sets the width, style, and color of the rule between columns.

Let's focus on a sample example with output:

```

<!DOCTYPE html>
<html>
<head>
<style>
.newspaper {
  -webkit-column-count: 3; /* Chrome, Safari, Opera */
  -moz-column-count: 3; /* Firefox */
  column-count: 3;

  -webkit-column-gap: 40px; /* Chrome, Safari, Opera */
  -moz-column-gap: 40px; /* Firefox */
  column-gap: 40px;

  -webkit-column-rule: 4px double #ff00ff; /* Chrome, Safari, Opera */
  -moz-column-rule: 4px double #ff00ff; /* Firefox */
  column-rule: 4px double #ff00ff;
}
</style>
</head>
<body>

<h1>The column-rule Property</h1>
<p>The column-rule property sets the width, style, and color of the rule between the columns of the element:</p>

<div class="newspaper">
  Lorem ipsum dolor sit amet, consectetur 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 dui dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius.
</div>

<p><strong>Note:</strong> The column-rule property is not supported in Internet Explorer 9 and earlier versions.</p>
</body>
</html>

```

The column-rule Property

The column-rule property sets the width, style, and color of the rule between the columns of the element:

Lorem ipsum dolor sit amet, consectetur 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 dui dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend	option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius.
--	---	--

Note: The column-rule property is not supported in Internet Explorer 9 and earlier versions.

Multiple Column Layouts

The CSS multi-column layout allows easy definition of multiple columns of text - just like in newspapers. The Column Count Property is used for achieving multiple column layouts. Please refer to the Column Count Section for example.

Gaps between Columns

Gaps between columns are specified through the **Column Gap** Property. Let's focus on an example with output:

```

<!DOCTYPE html>
<html>
<head>
<title>
</title>
</head>
<body>
<div class="column">
</div>
<div class="column">
</div>
<div class="column">
</div>
</body>
</html>

```

The column-gap Property

The column-gap property defines the gap between the columns of the element:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, qui 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 justo odio dignissim qui blandit praesent lacinia nisl et blandit augue dui dolore feugiat nulla facilisis. Nam liber tempore eum soluta.

nobis eleifend option congue nihil imperdiet doming id quod mazam placerat fames proin aenean. Tya non habent claritatem insulam, ut tunc legendum in la qua facit eorum claritatem. In vestigations demonstraverunt lectores legere me lius quod ii legunt saepius.

Note: The column-gap property is not supported in Internet Explorer 9 and earlier versions.

Box-orient, Box-pack, Box-align, Box-flex, etc

- The **box-orient CSS** property sets whether an element lays out its contents horizontally or vertically.
- The **box-pack CSS** property specifies how a box packs its contents in the direction of its layout. The effect of this is only visible if there is extra space in the box.
- The **box-align CSS** property specifies how an element aligns its contents across its layout in a perpendicular direction. The effect of the property is only visible if there is extra space in the box.
- The **box-flex CSS** property specifies how a box grows to fill the box that contains it, in the direction of the containing box's layout.

Reviewing the chapter

- The position property specifies the type of positioning method used for an element.
- HTML elements are positioned static by default.
- An element with position: relative; is positioned relative to its normal position
- An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled.
- A sticky element toggles between relative and fixed, depending on the scroll position
- The CSS box layout is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content.
- CSS vendor prefixes, aka CSS browser prefixes, are a way for browser makers to add support for new CSS features before those features are fully supported in all browsers
- The CSS multi-column layout allows easy definition of multiple columns of text - just like in newspapers. The Column Count Property is used for achieving multiple column layouts

Testing your skills

1. There are ____ different position values

a) 6, b) 3, c) 5, d) 4

2. HTML elements are positioned _____by default.

a) static, b) relative, c) fixed, d) absolute

3. A _____element does not leave a gap in the page where it would normally have been located

a) relative, b) fixed, c) static, d) none of the above

4. A "positioned" element is one whose position is anything except _____

a) static, b) relative, c) fixed, d) absolute

5. Which of the following are part of box layout:

a) margin, b) border, c) padding, d) all of the above

6. The effect of this property is only visible if there is extra space in the box

a) box flex, b) box pack, c) box align, d) box orient

7. The effect of this property is only visible if there is extra space in the box

a) box flex, b) box pack, c) box align, d) box orient

Lesson 19: Media Queries

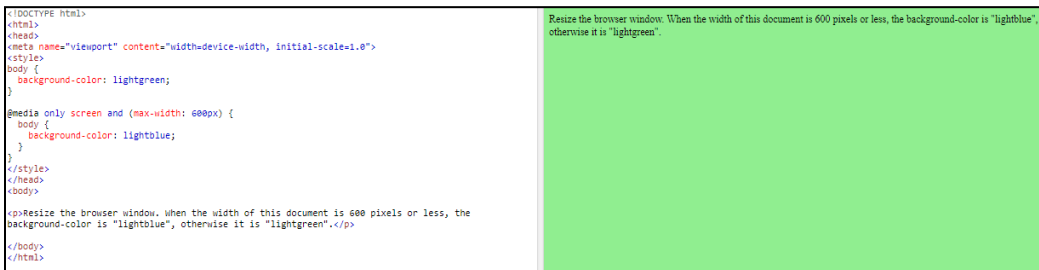
Objective: After completing this lesson you will be able to learn about : <ul style="list-style-type: none"> • The basics of media queries • Targeting Device Capabilities using media queries • Working with Columns 	Materials Required: <ul style="list-style-type: none"> • Computer With Windows XP and above • Stable Internet connection
Self- Learning Duration: 60 minutes	Practical Duration: 60 minutes
Total Duration: 120 minutes	

Media Queries

Media query is a CSS technique introduced in CSS3.

It uses the @media rule to include a block of CSS properties only if a certain condition is true.

Let's take an example where if the browser window is 600px or smaller, the background color will be light blue:



Targeting Device Capabilities using media queries

Media types describe the general category of a given device. Although websites are commonly designed with screens in mind, you may want to create styles that target special devices such as printers or audio-based screen readers. For example, this CSS targets printers:

```
@media print { ... }
```

You can also target multiple devices. For instance, this @media rule uses two media queries to target both screen and print devices:

```
@media screen, print { ... }
```

Media types describe the general category of a device. Except when using the not or only logical operators, the media type is optional and the all type will be implied.

all

Suitable for all devices.

print

Intended for paged material and documents viewed on a screen in print preview mode. **screen**

Intended primarily for screens.

speech

Intended for speech synthesizers.

Because they describe devices in only very broad terms, just a few are available; to target more specific attributes, use *media features* instead.

Media features describe specific characteristics of the user agent, output device, or environment. Media feature expressions test for their presence or value, and are entirely optional. Each media feature expression must be surrounded by parentheses.

Sample example to demonstrate the use of media queries to change the background color for different devices (observe the output on the right hand side):

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
background-color: tan;
color: black;
}

/* On screens that are 992px wide or less, the background color is blue */
@media screen and (max-width: 992px) {
body {
background-color: blue;
color: white;
}
}

/* On screens that are 600px wide or less, the background color is olive */
@media screen and (max-width: 600px) {
body {
background-color: olive;
color: white;
}
}
</style>
</head>
<body>

<h1>Resize the browser window to see the effect!</h1>
<p>By default, the background color of the document is "tan". If the screen size is 992px or less, the
color will change to "blue". If it is 600px or less, it will change to "olive".</p>

</body>
</html>
```

Resize the browser window to see the effect!

By default, the background color of the document is "tan". If the screen size is 992px or less, the color will change to "blue". If it is 600px or less, it will change to "olive".

Now, once I resize the browser and make it smaller, see the change in color:



Reviewing the chapter

- Media query is a CSS technique introduced in CSS3.
- It uses the @media rule to include a block of CSS properties only if a certain condition is true
- Media types describe the general category of a given device. Although websites are commonly designed with screens in mind, you may want to create styles that target special devices such as printers or audio-based screen readers.
- Media features describe specific characteristics of the user agent, output device, or environment

Testing your skills

1. Media query is introduced in which version of CSS:

a) CSS, b) CSS2, c) CSS3, d) CSS4

2. This media type is intended primarily for screens

a) print, b) all, c) speech, d) none of the above

3. The media rule that targets

a) @media screen, print { ... }, b) @media print { ... }, c) @print { ... }, d) none of the above

4. The media type intended for speech synthesizers

a) all, b) print, c) speech, d) media

5. Media features describe specific characteristics of:

a) user agent, b) output device, c) environment, d) all of the above

Lesson 20: Implementing CSS3 in the "Real World"

Objective: After completing this lesson you will be able to learn about : <ul style="list-style-type: none">• Modernizr• HTML5 Shims, jQuery, LESS, and SASS• CSS Grid Systems	Materials Required: <ul style="list-style-type: none">• Computer With Windows XP and above• Stable Internet connection
Self- Learning Duration: 60 minutes	Practical Duration: 60 minutes
Total Duration: 120 minutes	

Modernizr

Modernizr is a small JavaScript Library that detects the availability of native implementations for next-generation web technologies

There are several new features which are being introduced through HTML5 and CSS3 but same time many browsers do not support these news features.

Modernizr provides an easy way to detect any new feature so that you can take corresponding action. For example, if a browser does not support video feature then you would like to display a simple page.

You can create CSS rules based on the feature availability and these rules would apply automatically on the webpage if browser does not support a new feature.

Before you start using Modernizr, you would have to include its javascript library in your HTML page header as follows –

```
<script src="modernizr.min.js" type="text/javascript"></script>
```

As mentioned above, you can create CSS rules based on the feature availability and these rules would apply automatically on the webpage if browser does not support a new feature.

Detection testing (Modernizr)

Modernizr creates a global Modernizr JavaScript object, which allows us to query different properties of that object to perform feature detection by calling Modernizr.<featurename>.

So to test for canvas support, you can write the following code to see how the page reacts:

```
<script>

  if (Modernizr.canvas) {

    alert("This browser supports HTML5 canvas!");

  }

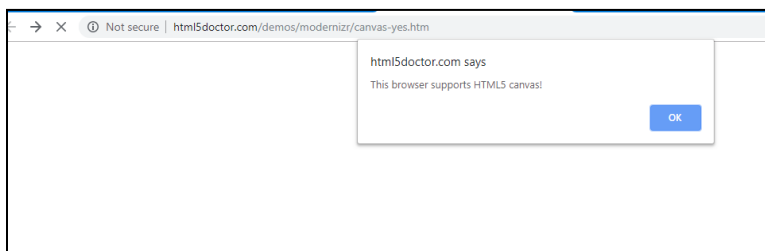
</script>
```

If you are confused, this is the complete code for creation of the page:

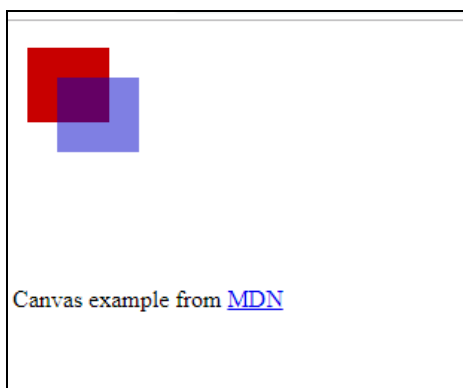
```
<!DOCTYPE html>
<html>
<head>
<script src="http://cdnjs.cloudflare.com/ajax/libs/modernizr/2.0.6/modernizr.min.js"></script>
<meta charset=utf-8 />
<title>Canvas?</title>
<script>
if (Modernizr.canvas) {
  alert("This browser supports HTML5 canvas!")
}

function draw() {
  var canvas = document.getElementById("canvas");
  if (canvas.getContext) {
    var ctx = canvas.getContext("2d");
    ctx.fillStyle = "rgb(200,0,0)";
    ctx.fillRect (10, 10, 55, 50);
    ctx.fillStyle = "rgba(0, 0, 200, 0.5)";
    ctx.fillRect (30, 30, 55, 50);
  }
}
</script>
</head>
<body onload="draw();" >
  <canvas id="canvas" width="150" height="150">
    <p>This example requires a browser that supports the
    <a href="http://www.w3.org/html/wg/html5/">HTML5</a>
    &lt;canvas&gt; feature.</p>
  </canvas>
  <p>Canvas example from <a href="https://developer.mozilla.org/en/Canvas_tutorial/Basic_usage">MDN</a></p>
</body>
</html>
```

The output will be something like this:



Once you click the 'ok', it will display similar page content as below:



Here it is notable that you need to prefix "no-" to every feature/property you want to handle for the browser which does not support them.

Following is the syntax to detect a particular feature through JavaScript –

```
/* In your CSS: */
.no-audio #music {
    display: none; /* Don't show Audio options */
}
.audio #music button {
    /* Style the Play and Pause buttons nicely */
}

<!-- In your HTML: -->
<div id="music">

    <audio>
        <source src="audio.ogg" />
        <source src="audio.mp3" />
    </audio>

    <button id="play">Play</button>
    <button id="pause">Pause</button>
</div>
```

HTML5 Shims

The shim is to enable HTML5 elements to be able to be styled through CSS in Internet Explorer versions less than 9. You may well consider keeping the header.php clean and then insert the shim from functions.php file

```
PHP
// add ie conditional html5 shim to header
function add_ie_html5_shim () {
    echo '<!--[if lt IE 9]>';
    echo '<script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>';
    echo '<![endif]-->';
}
add_action('wp_head', 'add_ie_html5_shim');
```

jQuery

jQuery is a lightweight, "write less, do more", JavaScript library.

The purpose of jQuery is to make it much easier to use JavaScript on your website.

jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.

jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

There are lots of other JavaScript frameworks out there, but jQuery seems to be the most popular, and also the most extendable.

A sample example with output:

<pre><!DOCTYPE html> <html> <head> <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script> <script> \$(document).ready(function(){ \$("p").click(function(){ \$(this).hide(); }); }); </script> </head> <body> <p>If you click on me, I will disappear.</p> <p>Click me away!</p> <p>Click me too!</p> </body> </html></pre>	<p>If you click on me, I will disappear.</p> <p>Click me away!</p> <p>Click me too!</p>
---	---

LESS and SASS

Pre-processors extend CSS with variables, operators, interpolations, functions, mixins and many more other usable assets. SASS, LESS and Stylus are the well known ones.

SASS is the most mature, stable, and powerful professional grade CSS extension language in the world. Sass is completely compatible with all versions of CSS.

LESS (which stands for Leaner Style Sheets) is a backwards-compatible language extension for CSS. LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for website.

LESS is a dynamic style sheet language that extends the capability of CSS. LESS is also cross browser friendly.

CSS Grid Systems

The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

A grid layout consists of a parent element, with one or more child elements.

Have a look at a sample example with output:

```

<!DOCTYPE html>
<html>
<head>
<title>
</title>
</head>
<body>
<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>
</body>
</html>

```

Grid Elements

A Grid Layout must have a parent element with the `display` property set to `grid` or `inline-grid`.

Direct child element(s) of the grid container automatically becomes grid items.

1	2	3
4	5	6
7	8	9

Reviewing the chapter

- Modernizr is a small JavaScript Library that detects the availability of native implementations for next-generation web technologies
- Modernizr provides an easy way to detect any new feature so that you can take corresponding action. For example, if a browser does not support video feature then you would like to display a simple page.
- Modernizr creates a global Modernizr JavaScript object, which allows us to query different properties of that object to perform feature detection by calling `Modernizr.<featurename>`.
- The shim is to enable HTML5 elements to be able to be styled through CSS in Internet Explorer versions less than 9.
- jQuery is a lightweight, "write less, do more", JavaScript library. The purpose of jQuery is to make it much easier to use JavaScript on your website.
- SASS is the most mature, stable, and powerful professional grade CSS extension language in the world. Sass is completely compatible with all versions of CSS.
- LESS (which stands for Leaner Style Sheets) is a backwards-compatible language extension for CSS. LESS is a CSS pre-processor that enables customizable, manageable and reusable style sheet for website.
- The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

Testing your skills

1. _____ provides an easy way to detect any new feature so that you can take corresponding action

a) modernizr b) LESS, c) SASS, d) HTML5 SHIMS

2. jQuery is a lightweight, "_____", JavaScript library

a) write less, do less, b) write more, do more, c) write less, do more, d) write more, do less

3. jQuery simplifies which of the following from JavaScript, like and.

a) AJAX calls, b) DOM manipulation, c) Both (a) and (b), d) none of the above

4. Which of the following features is part of JQuery Library

a) HTML Event Methods, b) CSS manipulations, c) DOM manipulations, d) all of the above

5. _____ is a backwards-compatible language extension for CSS

a) SASS, b) LESS, c) SHIM, d) Modernizr

Lesson 21: Transforms, Transitions, and Animations

Objective: After completing this lesson you will be able to learn about :

- different CSS functions like `translate()`, `rotate()`, `rotate3D` functions etc.
- Changing Image on Hover through CSS
- `@keyframes`

Materials Required:

- Computer With Windows XP and above
- Stable Internet connection

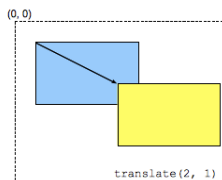
Self- Learning Duration: 60 minutes

Practical Duration: 60 minutes

Total Duration: 120 minutes

`translate()`

The **`translate()`** CSS function repositions an element in the horizontal and/or vertical directions. Its result is a `<transform-function>` data type.



This transformation is characterized by a two-dimensional vector. Its coordinates define how much the element moves in each direction.

The `translate()` function is specified as either one or two values:

1. `translate(tx)`
2. `translate(tx, ty)`

“tx” - a `<length>` representing the abscissa (x-coordinate) of the translating vector.

“ty” - a `<length>` representing the ordinate of the translating vector (or y-coordinate). If unspecified, its default value is 0. For example, `translate(2)` is equivalent to `translate(2, 0)`.

`rotate()`

The **`rotate()`** CSS function defines a transformation that rotates an element around a fixed point on the 2D plane, without deforming it. Its result is a `<transform-function>` data type.

The fixed point that the element rotates around — mentioned above — is also known as the transform origin. This defaults to the center of the element, but you can set your own custom transform origin using the `transform-origin` property.

The amount of rotation created by `rotate()` is specified by an `<angle>`. If positive, the movement will be clockwise; if negative, it will be counter-clockwise. A rotation by 180° is called point reflection.

Let's take an example:

`rotate(a)` where "a" signifies an `<angle>` representing the angle of the rotation. A positive angle denotes a clockwise rotation, a negative angle a counter-clockwise one.

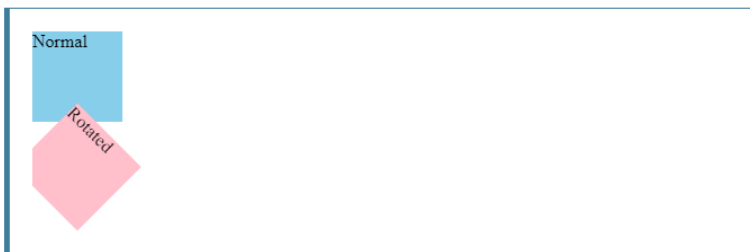
HTML

```
1 <div>Normal</div>
2 <div class="rotated">Rotated</div>
```

CSS

```
1 div {
2   width: 80px;
3   height: 80px;
4   background-color: skyblue;
5 }
6
7 .rotated {
8   transform: rotate(45deg); /* Equal to rotateZ(45deg) */
9   background-color: pink;
10 }
```

The output:



Rotate3D() function

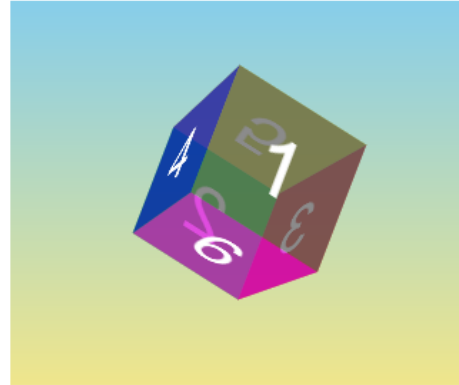
The `rotate3d()` CSS function defines a transformation that rotates an element around a fixed axis in 3D space, without deforming it. Its result is a `<transform-function>` data type.

```
transform: rotate3d(0);
```

```
transform: rotate3d(1, 1, 1, 45deg);
```

```
transform: rotate3d(2, -1, -1, -0.2turn);
```

```
transform: rotate3d(0, 1, 0.5, 3.142rad);
```



In 3D space, rotations have three degrees of liberty, which together describe a single axis of rotation. The axis of rotation is defined by an $[x, y, z]$ vector and pass by the origin (as defined by the transform-origin property).

If, as specified, the vector is not normalized (i.e., if the sum of the square of its three coordinates is not 1), the user agent will normalize it internally.

A non-normalizable vector, such as the null vector, $[0, 0, 0]$, will cause the rotation to be ignored, but without invalidating the whole CSS property.

Understanding the Syntax

The amount of rotation created by `rotate3d()` is specified by three `<number>`s and one `<angle>`. The `<number>`s represents the x-, y-, and z-coordinates of the vector denoting the axis of rotation. The `<angle>` represents the angle of rotation; if positive, the movement will be clockwise; if negative, it will be counter-clockwise.

Let's take an example, `rotate3d(x, y, z, a)`, where

"x" - Is a `<number>` describing the x-coordinate of the vector denoting the axis of rotation which could be between 0 and 1.

"y" - Is a `<number>` describing the y-coordinate of the vector denoting the axis of rotation which could be between 0 and 1.

"z" - Is a `<number>` describing the z-coordinate of the vector denoting the axis of rotation which could be between 0 and 1.

"a" - Is an `<angle>` representing the angle of the rotation. A positive angle denotes a clockwise rotation, a negative angle a counter-clockwise one.

An example:

Rotating of the Y axis

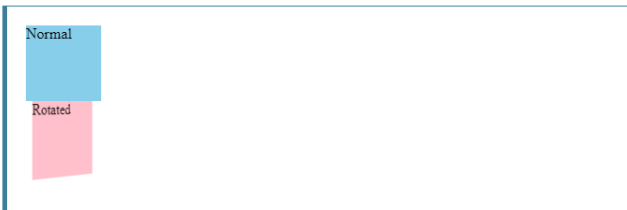
HTML

```
1 <div>Normal</div>
2 <div class="rotated">Rotated</div>
```

CSS

```
1 body {
2   perspective: 800px;
3 }
4
5 div {
6   width: 80px;
7   height: 80px;
8   background-color: skyblue;
9 }
10
11 .rotated {
12   transform: rotate3d(0, 1, 0, 60deg);
13   background-color: pink;
14 }
```

The output:



Changing Image on Hover through CSS

You can simply use the CSS background-image property in combination with the :hover pseudo-class to replace or change the image on mouse-over.

Let's try out the following example to understand how it basically works:



Focus on the output on the right side. Once you take your mouse cursor over the image, it will transform into the front side of the card. Try it!

@Keyframes

The @keyframes rule specifies the animation code. The animation is created by gradually changing from one set of CSS styles to another. During the animation, you can change the set of CSS styles many times.

Specify when the style change will happen in percent, or with the keywords "from" and "to", which is the same as 0% and 100%. 0% is the beginning of the animation, 100% is when the animation is complete.

The syntax:

@keyframes animationname {keyframes-selector {css-styles;}}

An example that uses @keyframes animation to move an element up and down:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: red;
  position: relative;
  -webkit-animation: mymove 5s infinite; /* Safari 4.0 - 8.0 */
  animation: mymove 5s infinite;
}

/* Safari 4.0 - 8.0 */
@-webkit-keyframes mymove {
  from {top: 0px;}
  to {top: 200px;}
}

/* Standard syntax */
@keyframes mymove {
  from {top: 0px;}
  to {top: 200px;}
}
</style>
</head>
<body>

<h1>The @keyframes Rule</h1>


<p><strong>Note:</strong> The @keyframes rule is not supported in Internet Explorer 9 and
earlier versions.</p>

<div></div>

</body>
</html>
```

The @keyframes Rule

Note: The @keyframes rule is not supported in Internet Explorer 9 and earlier versions.



Reviewing the chapter

- The translate() CSS function repositions an element in the horizontal and/or vertical directions
- The translate() function is specified as either one or two values: translate(tx) and translate(tx, ty)
- The rotate() CSS function defines a transformation that rotates an element around a fixed point on the 2D plane, without deforming it
- The rotate3d() CSS function defines a transformation that rotates an element around a fixed axis in 3D space, without deforming it
- You can simply use the CSS background-image property in combination with the :hover pseudo-class to replace or change the image on mouse-over
- The @keyframes rule specifies the animation code. The animation is created by gradually changing from one set of CSS styles to another

Testing your skills

1. The translate() CSS function repositions an element in which direction:

a) horizontal b) vertical, c) Both (a) and (b), d) none of the above

2. _____function defines a transformation that rotates an element around a fixed point on the 2D plane, without deforming it

a) rotate(), b) translate(), c) rotate3D(), d) none of the above

3. The :hover pseudo-class is used for

a) replacing the image on mouse over, b) changing the image on mouse over, c) Both (a) and (b), d) none of the above

4. The @keyframe rule specifies the

a) bar code, b) animation code, c) transformation code, d) key code

5. In rotate3D() function, the <angle> represents

a) angle of rotation, b) axis of rotation, c) angle of coordinates, d) none of the above

Lesson 22: Introduction to Bootstrap

Objective: After completing this lesson you will be able to learn about :

- Bootstrap basics, advantages, setup and overview
- Mobile first strategy
- how to set up and install Bootstrap and its components, create a simple landing page include some basic content and style it.
- You can create navigation menus, set background images, include buttons, columns and contact forms.

Materials Required:

- Computer With Windows XP and above
- Stable Internet connection

Self- Learning Duration: 60 minutes

Practical Duration: 60 minutes

Total Duration: 120 minutes

Bootstrap Introduction

Bootstrap is a front-end framework that helps you build mobile responsive websites more quickly and easily. First developed by Twitter, Bootstrap is by now used for anything from developing web applications to WordPress themes. It is also completely free, versatile and intuitive.

Why Bootstrap?

Bootstrap was originally created by Twitter. A small group of developers in the company put it together as an internal tool to help create consistent web interfaces. The project kept growing until at last they decided to release it to the public for free usage in 2011. Mark Otto and Jacob Thornton are considered to be the primary developers of Bootstrap technology.

Since then, support for Bootstrap has continued both by a number of the original developers as well as a large group of contributors. It has seen several major updates over the years (the last one, Bootstrap 4, in January 2018) that have, among other things, added its famous grid system, flat design, a mobile-first approach and modern CSS.

By now, the framework is among the most popular web development tools on the net. However, what exactly does it do?

Bootstrap Advantages

Bootstrap helps you set them up by offering a large number of CSS classes you can easily apply to HTML elements to create the site components you need. That way, you can conjure complex web pages from standard HTML and customize them to your needs. In addition to that, Bootstrap comes with a number of jQuery plugins that can provide additional functionality such as carousels, buttons, tooltips and more.

What's new in Bootstrap 4?

Bootstrap 4 is built on these standards: HTML5, CSS3, JavaScript and Sass. During recent years, front-end developers have shown a preference for Sass instead of Less, widely adopting it in their projects. In all the previously released versions, Bootstrap used Less as its preprocessor. Below is a list of updated features that Bootstrap 4 has to offer:

- The `.panel`, `thumbnail`, and `.well` classes have been replaced by `.card` in Bootstrap 4.
- Buttons have a brand new style. They definitely went "flat" and removed gradients.
- The `.btn-outline-*` classes have been introduced.
- One of the greatest innovations in this version of Bootstrap is the support for Flexbox, which provides simpler and more flexible layout options in CSS.
- New Spacing Utility Classes have been introduced, allowing you to have better control of horizontal and vertical space. You can now add some space in any direction using a utility class (with margins or paddings).
- In Bootstrap 4, 16px is the new default font size (it was 14px previously), which means that the size of the text is bigger and more visible. Moreover, in this new version everything is intended to be more dynamic thanks to an important enhancement: the font sizing and the grid system are now based on rems.
- The use of rems has been employed also to introduce a new typographic component: display heading.

Bootstrap 4 supports four different sizes for display heading (class `display-*`). Focus below to have a clear view:

```
<h1 class="display-1">Heading 1</h1>
<h1 class="display-2">Heading 2</h1>
<h1 class="display-3">Heading 3</h1>
<h1 class="display-4">Heading 4</h1>
```

This is how they look:

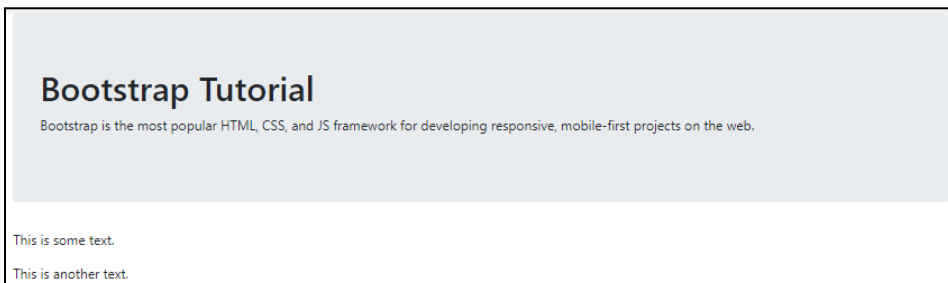
Bootstrap 4 (display-1)
Bootstrap 4 (display-2)
Bootstrap 4 (display-3)
Bootstrap 4 (display-4)

Bootstrap 4 Jumbotron

A jumbotron indicates a big grey box for calling extra attention to some special content or information. Inside a jumbotron you can put nearly any valid HTML, including other Bootstrap elements/classes.

Let's have a look at a sample code example with output (*Use a `<div>` element with class `.jumbotron` to create a jumbotron*):

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
</head>
<body>
<div class="container">
  <div class="jumbotron">
    <h1>Bootstrap Tutorial</h1>
    <p>Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile-
first projects on the web.</p>
  </div>
  <p>This is some text.</p>
  <p>This is another text.</p>
</div>
</body>
</html>
```

The output:

The Grid System in Bootstrap

Bootstrap includes a responsive, mobile first fluid grid system that appropriately scales up to 12 columns as the device or viewport size increases. It includes predefined classes for easy layout options, as well as powerful mixins for generating more semantic layouts.

Mobile First Strategy

Without any doubt, the mobile-first approach is the core new feature in Bootstrap 4. What can be greater than having a responsive site by default? And with its 4 grids system it brings the extension to a completely new horizons and allows you to create different design for any of the supported devices.

Here are the features:

- Fully visual and responsive smart design - With the Bootstrap 4 Grid floating panel it's all fully visually done and no hand-coding is required for any action while you create your design. From the greatly useful floater you can choose all the options you need and in a mouse click they'll appear in the grid.
- Fluid design by default - No more adaptive designs for a large number of devices and screen sizes, now they come fluid and you don't have to worry about upcoming screen sizes and resolutions. Your design will flow and look great on every device.
- Design for small devices - With the mobile-first approach you can decide which is the most important part of the content that you want your users to see when browsing your website on small devices such as phones.
- Progressive enhancement - With Bootstrap 4 you can build your layout simultaneously for all four sizes and keep a constant eye on the process, directly in Dreamweaver. As the device size grows you can add additional content until you reach your desired desktop website layout.

Getting Started with Bootstrap

Bootstrap gives you a lot of shortcuts for creating web pages that will save you time and energy. All you need is a basic understanding of HTML and CSS to creates that are responsive, mobile first and compatible with all modern browsers.

Setup and Overview

In order to use Bootstrap, you first need to integrate it into your development environment aka web page. For that, you have two different possibilities: load it remotely or download and use Bootstrap locally. However, for both, you first need something to load it into.

1. Create an HTML Page

As a first step, we will create a simple HTML template as a base where we will use Bootstrap. For that, the first thing you want to do is create a folder on your computer or server for the project files. In this case, I will simply call it bootstrap. Here, create a new text file and call it index.html. Open it with a text editor of your choice (e.g. Notepad++) and then paste the code below into it.

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
  <title>Bootstrap Tutorial Sample Page</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>
</body>
</html>
```

Don't forget to save your file before moving on!

2a. Load Bootstrap via CDN

As already explained, Bootstrap consists mainly of style sheets and scripts. As such, they can be loaded in the header and footer of your web page like other assets such as custom fonts. The framework offers a CDN (content delivery network) access path for that.

To get Bootstrap into your page, simply paste the code below into the `<head>` section of your template.

```
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
crossorigin="anonymous">
```

When you now save the file, any browser that opens it will automatically load the Bootstrap assets.

Using the remote method is a good idea as many users will already have the framework in the cache of their browser. If that is the case, they won't have to reload it when coming to your site, leading to faster page loading time. As a consequence, this is the recommended method for live sites.

However, for experimenting and development, or if you want to be independent of an Internet connection, you can also get your own copy of Bootstrap. This is what I am doing for this tutorial because it also results in less code to post.

2b. Host Bootstrap Locally

An alternative way to set up Bootstrap is to download it to your hard drive and use the files locally. You find download options in the same place as the links to the remote version. Here, be sure to get the compiled CSS and JS files. You don't need the source files.

Once you have done so, unzip the file and copy its contents into the same directory as `index.html`. After that, you can load the Bootstrap CSS into your project like this:

```
<link rel="stylesheet" href="bootstrap/css/bootstrap.min.css">
```

You will notice that this includes the file path at which to find the Bootstrap file. In your case, make sure your path corresponds to your actual setup. For example, the names of the directories might differ if you downloaded a different version of Bootstrap.

3. Include jQuery

In order to get the full functionality of Bootstrap, you need to also load the jQuery library. Here, too, you have the possibility to load it remotely or host it locally.

In the first case, you find the link to latest version of jQuery here. You can use it to load the library into your page by putting the line of code below right before where it says `</body>` in your page.

```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
```

Alternatively, download jQuery, unzip and put it into the project folder. Then, include it in the same place of your file in this way:

```
<script src="jquery-3.3.1.min.js"></script>
```

Again, make sure the path corresponds to your setup.

4. Load Bootstrap JavaScript

The last step in setting up Bootstrap is to load the Bootstrap JavaScript library. These are included in the downloaded version of the framework and you also find links to remote sources in the same place as before. However, we will load it in a different place than the style sheet. Instead of the header, it goes into the page footer, right after the call for jQuery.

You can call it remotely like this:

```
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js" integrity="sha384-ChfqquxZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
crossorigin="anonymous"></script>
```

Or locally like so:

```
<script src="bootstrap/js/bootstrap.min.js"></script>
```

5. Putting it All Together

If you have followed the steps above correctly, you should end up with a file that looks like this for the remote solution:

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



```
<head>

<title>Bootstrap Tutorial Sample Page</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
crossorigin="anonymous">

</head>

<body>

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"> </script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js" integrity="sha384-
ChfqquxZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
crossorigin="anonymous"> </script>

</body>

</html>
```

Alternatively, if you are hosting locally, your page template should resemble the code below:

```
<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Tutorial Sample Page</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="bootstrap/css/bootstrap.min.css">

</head>

<body>

<script src="jquery-3.3.1.min.js"> </script>

<script src="bootstrap/js/bootstrap.min.js"> </script>

</body>

</html>
```

Design Your Landing Page

Alright, that was, admittedly, a lot of preparation work. However, it wasn't very hard, was it? Plus, now the fun begins.

At the moment, when you navigate to your sample site, you should simply see a blank page. Time to change that.

1. Add a Navigation Bar

The first thing we want to do is add a navigation bar to the top of the page. This allows your visitors to get around your site and discover the rest of your pages.

For that, we will use the navbar class. This is one of the default elements of Bootstrap. It creates a navigation element that is responsive by default and will automatically collapse on smaller screens. It also offers built-in support for adding branding, color schemes, spacing and other components.

We will use it like below and post it just below the <body> tag:

```
<nav class="navbar navbar-expand-md">
  <a class="navbar-brand" href="#">Logo</a>
  <button class="navbar-toggler navbar-dark" type="button" data-toggle="collapse" data-
  target="#main-navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="main-navigation">
    <ul class="navbar-nav">
      <li class="nav-item">
        <a class="nav-link" href="#">Home</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">About</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Contact</a>
      </li>
    </ul>
  </div>
</nav>
```

Some explanation of the code:

- **navbar-expand-md** — This denotes at which point the navigation bar expands from vertical or hamburger icon to a full-size horizontal bar. In this case, we have set it to medium screens, which, in Bootstrap, is anything greater than 768px.
- **navbar-brand** — This is used for your website branding. You can also include a logo image file here.
- **navbar-toggler** — Denotes the toggle button for the collapsed menu. The piece `data-toggle="collapse"` defines that this will turn to a hamburger menu, not to drop-down, which is the other option. It's important that you define a data-target with a CSS id (defined by the #) and wrap a div with the same name around the actual navbar element.
- **navbar-toggler-icon** — As you can probably guess, this creates the icon users click on to open the menu on smaller screens.
- **navbar-nav** — The class for the `` list element that holds the menu items. The latter are denoted with `nav-item` and `nav-link`.

Why am I explaining this so much?

Because that is the point of Bootstrap. You have all of these standards that allow you to quickly create elements with some HTML and CSS classes. You don't have to write any CSS to provide styling, everything is already set up within Bootstrap. Plus, everything is mobile responsive out of the box! Are you starting to see how helpful this is?

The above is enough to add a navigation bar to your site. However, at the moment, it still looks like very little.



That's because it doesn't have a lot of styling attached to it. While you are able to add default colors (for example, by giving the navbar a class like `bg-dark navbar-dark`), we instead want to add our own.

2. Include Custom CSS

Fortunately, if you want to change the default styling, you don't have to wade through a large library of style sheets and make the changes by hand. Instead, just like with a WordPress child theme, you are able to add your own CSS files which you can use to overwrite existing styling.

For that, simply create a blank file with your text editor and call it `main.css`. Save it, then add it to the head section of your Bootstrap site like this:

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

This is the code for a style sheet that resides in the main directory. If you decide to place your inside the `css` folder, be sure to include the correct path in the link.

From here, you are able to add custom CSS to your site. For example, to style the navigation bar and its elements, you could use markup like this:

```
body {  
  padding: 0;  
  margin: 0;  
  background: #f2f6e9;  
}  
  
/*--- navigation bar ---*/  
  
.navbar {  
  background: #6ab446;  
}  
  
.nav-link,  
.navbar-brand {  
  color: #fff;  
  cursor: pointer;  
}  
  
.nav-link {  
  margin-right: 1em !important;  
}  
  
.nav-link:hover {
```

```
color: #000;
```

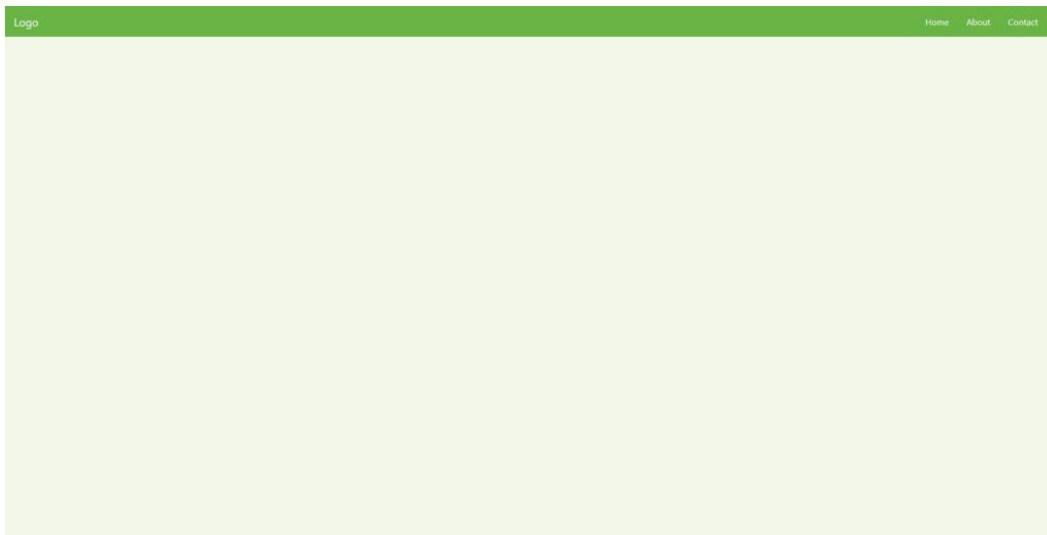
```
}
```

```
.navbar-collapse {
```

```
justify-content: flex-end;
```

```
}
```

And here is the result:



Looks better than before, doesn't it?

3. Create a Page Content Container

After the navigation bar, the next thing you want is a container for the page content. This is really easy in Bootstrap as all you need for it is this underneath the navbar tag:

```
<header class="page-header header container-fluid">
```

```
</header>
```

Notice the container-fluid class. This is another one of those default Bootstrap classes. Applying it to the div element automatically applies a bunch of CSS to it.

The -fluid part makes sure the container stretches across the entire width of the screen. There's also just container, which has fixed widths applied to it, so there will always be space on both sides of the screen.

However, if you now reload the page, you still won't see anything (unless you use the developer tools). That's because you only created an empty HTML element. This will start changing now.

4. Background Image and Custom JavaScript

As the next step in this Bootstrap tutorial, we want to include a full-screen background image for our landing page header. For that, we will have to use some jQuery to make the image stretch all the way across the screen.

You do that the same way you include custom CSS. First, create a text file of the name `main.js` and place it inside your site folder. Then, call it before the closing `</body>` tag inside `index.html`.

```
<script src="main.js"></script>
```

After that, you can copy and paste this piece of code to make the `<header>` element stretch across the entire screen:

```
$(document).ready(function(){  
    $('.header').height($(window).height());  
})
```

Then, the only thing that's left is to actually set a background image. You can do this like so inside `main.css`:

```
.header {  
    background-image: url('images/header-background.jpg');  
    background-size: cover;  
    background-position: center;  
    position: relative;  
}
```

If you place an image of sufficient size at the location specified by the path above, you will achieve a result similar to this:



5. Add an Overlay

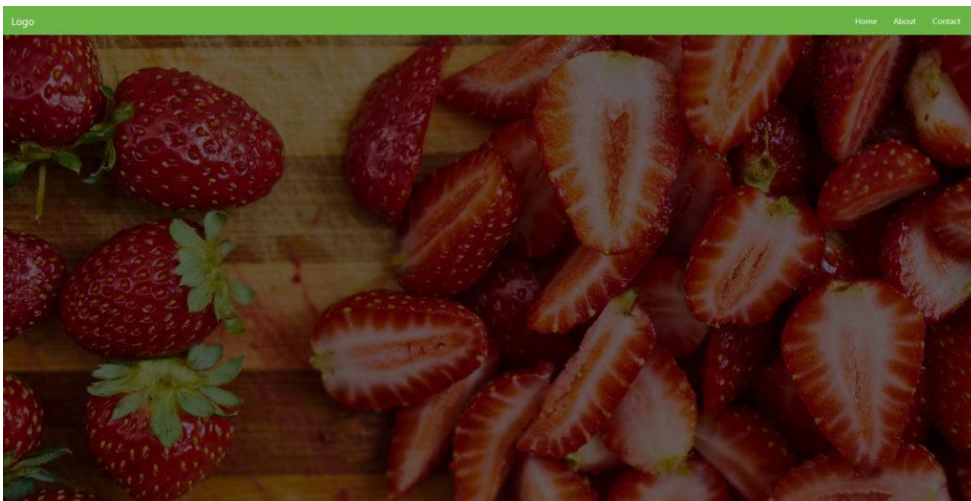
To make the background image extra stylish, we will also add an overlay. For that, you will create another div element inside the one you just included earlier.

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

Then, you can add the following in your custom CSS file:

1. `.overlay {`
2. `position: absolute;`
3. `min-height: 100%;`
4. `min-width: 100%;`
5. `left: 0;`
6. `top: 0;`
7. `background: rgba(0, 0, 0, 0.6);`
8. `}`

This will create this nice overlay for the image you input earlier:



6. Include a Page Title and Body Text

Now you probably want to add a page title in the form of a heading plus some body text. That way, your visitors will know immediately which site they are on and what they can expect from it.

To create those, simply add this snippet inside the container you set up in the last step, below the overlay:

```
<div class="description">  
<h1>Welcome to the Landing Page!</h1>
```

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque interdum quam odio, quis placerat ante luctus eu. Sed aliquet dolor id sapien rutrum, id vulputate quam iaculis. Suspendisse consectetur mi id libero fringilla, in pharetra sem ullamcorper.</p>

</div>

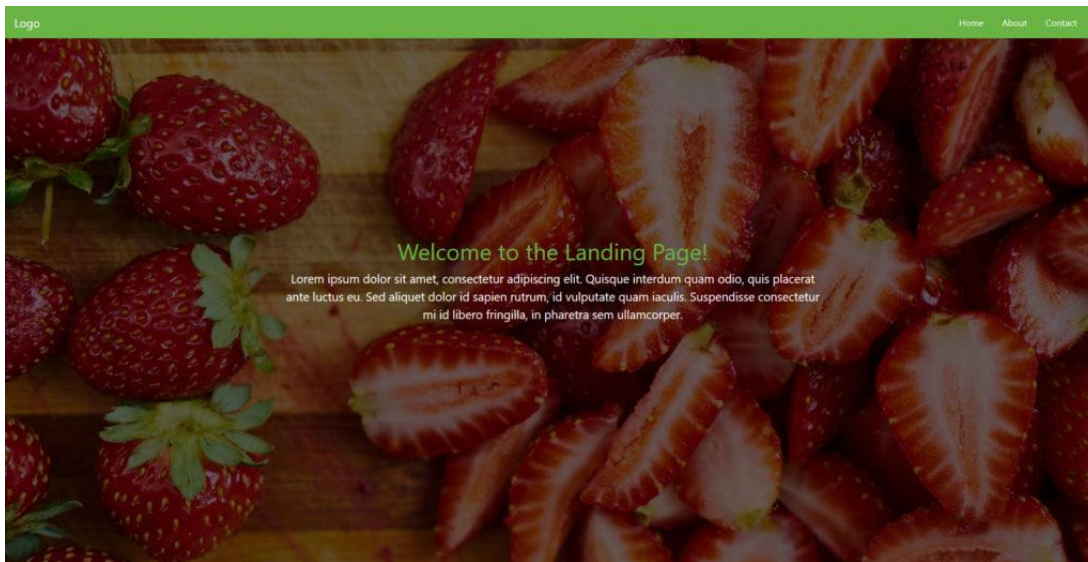
After that, add the following markup to main.css.

```
.description {  
  left: 50%;  
  position: absolute;  
  top: 45%;  
  transform: translate(-50%, -55%);  
  text-align: center;  
}
```

```
.description h1 {  
  color: #6ab446;  
}
```

```
.description p {  
  color: #fff;  
  font-size: 1.3rem;  
  line-height: 1.5;  
}
```

When you do, the landing page now looks like this:



It's really starting to come together, isn't it?

7. Create a CTA Button

No landing page is complete without a call to action, most often in the form of a button. For that reason, we would be amiss not to include how to create one in this Bootstrap tutorial.

The framework offers plenty of tools to create buttons quickly and easily. You can find a lot of examples here. In my case, I add the following markup right below the page content inside the `<description>` container:

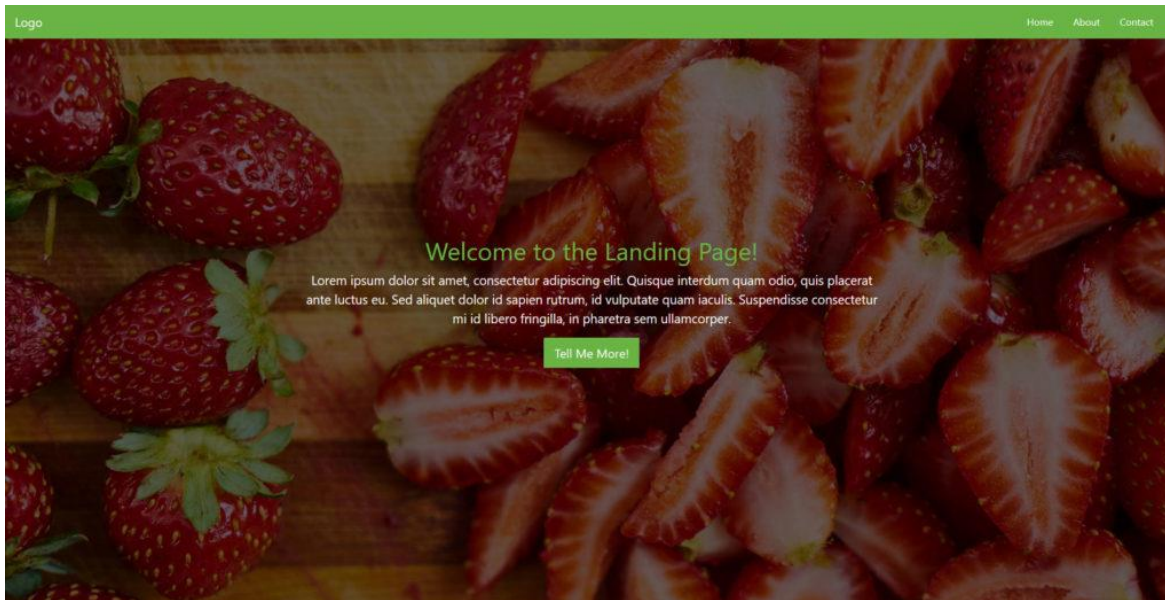
```
<button class="btn btn-outline-secondary btn-lg">Tell Me More!</button>
```

In addition to that, I add this CSS to main.css:

```
.description button {  
  border: 1px solid #6ab446;  
  background: #6ab446;  
  border-radius: 0;  
  color: #fff;  
}
```

```
.description button:hover {  
  border: 1px solid #fff;  
  background: #fff;  
  color: #000;  
}
```

After saving and reloading, it looks like this:



8. Set up a Three-column Section

I am already quite satisfied with how things are shaping up. However, we are not done with the page yet. Next up, we want to create three columns below the main content for additional information. This is a specialty of Bootstrap since it plays to its strength: creating a grid. Here's how to do that in this case:

```
<div class="container features">
  <div class="row">
    <div class="col-lg-4 col-md-4 col-sm-12">
      <h3 class="feature-title">Lorem ipsum</h3>
      
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque interdum quam odio, quis
      placerat ante luctus eu. Sed aliquet dolor id sapien rutrum, id vulputate quam iaculis.</p>
    </div>
    <div class="col-lg-4 col-md-4 col-sm-12">
      <h3 class="feature-title">Lorem ipsum</h3>
      
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque interdum quam odio, quis
      placerat ante luctus eu. Sed aliquet dolor id sapien rutrum, id vulputate quam iaculis.</p>
    </div>
    <div class="col-lg-4 col-md-4 col-sm-12">
  </div>
```

</div>

</div>

The first thing you will notice is the row element. You need this whenever creating columns to act as a container for the grid.

As for the columns, they all have several classes: col-lg-4, col-md-4 and col-sm-12. These denote that we are dealing with columns and the size they will have on different screens.

To understand this, you need to know that in a Bootstrap grid, all columns in one row always add up to the number 12. So, giving them the classes above means that they will take up one third of the screen on large and medium screens (12 divided by 3 is 4) but the entire screen on small devices (12 out of 12 columns).

Makes sense, doesn't it?

You will also notice that I included images and added the .image-fluid class to them. This is to make them responsive so that they scale along with screen that the page is viewed on.

In addition to that, you have the following styling included in the usual place:

```
.features {  
  margin: 4em auto;  
  padding: 1em;  
  position: relative;  
}
```

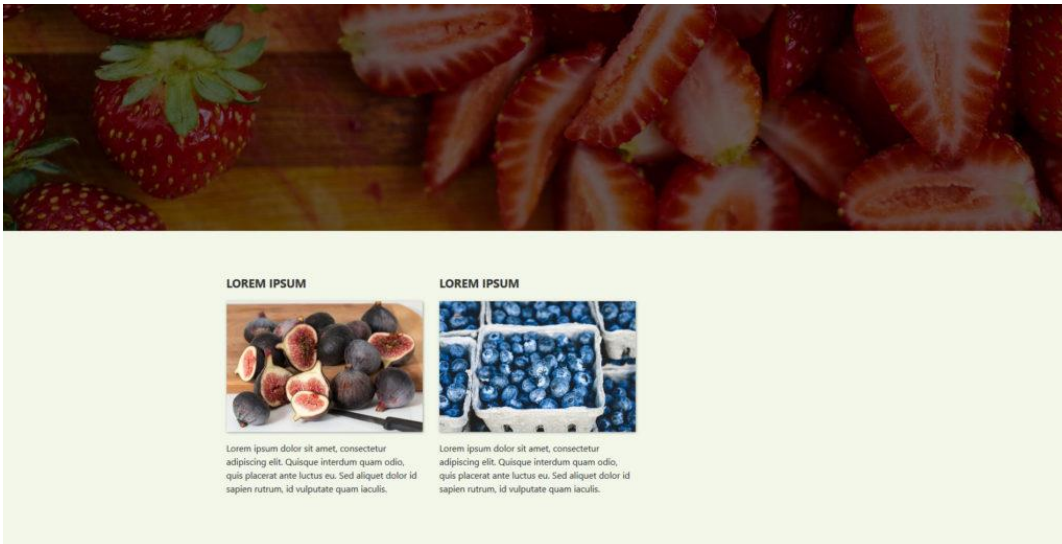
```
.feature-title {  
  color: #333;  
  font-size: 1.3rem;  
  font-weight: 700;  
  margin-bottom: 20px;  
  text-transform: uppercase;  
}
```

```
.features img {  
  -webkit-box-shadow: 1px 1px 4px rgba(0, 0, 0, 0.4);  
  -moz-box-shadow: 1px 1px 4px rgba(0, 0, 0, 0.4);  
  box-shadow: 1px 1px 4px rgba(0, 0, 0, 0.4);  
}
```

```
margin-bottom: 16px;
```

```
}
```

When added below the main content and saved, it looks like this:



9. Add a Contact Form

You will notice that one of the new fields is still empty. This was on purpose because we want to add a contact form to it. This is a very normal practice for landing pages to allow visitors to get in touch.

Creating a contact form in Bootstrap is quite easy:

```
<h3 class="feature-title">Get in Touch!</h3>
<div class="form-group">
  <input type="text" class="form-control" placeholder="Name" name="">
</div>
<div class="form-group">
  <input type="email" class="form-control" placeholder="Email Address" name="email">
</div>
<div class="form-group">
  <textarea class="form-control" rows="4"></textarea>
</div>
<input type="submit" class="btn btn-secondary btn-block" value="Send" name="">
```

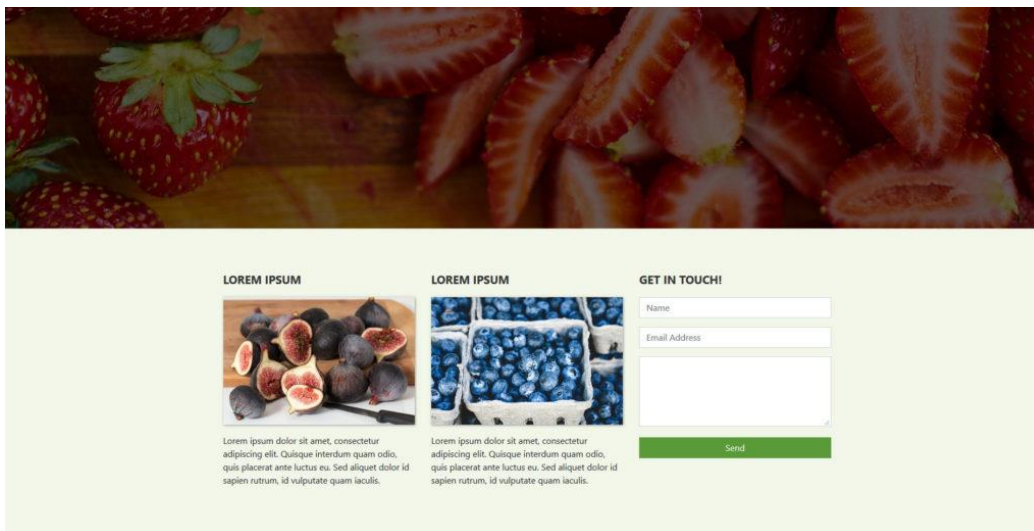
By now, I shouldn't have to explain the markup to create columns anymore. Here's what the rest of the markup means:

- form-group — Used to wrap around form fields for formatting.
- form-control — Denotes form fields such as inputs, text areas etc.

There is a lot more you can do with forms, but for demonstration purposes, the above is enough. Place it inside the remaining empty column and then add this styling in main.css:

```
.features .form-control,  
.features input {  
border-radius: 0;  
}  
  
.features .btn {  
background-color: #589b37;  
border: 1px solid #589b37;  
color: #fff;  
margin-top: 20px;  
}  
  
.features .btn:hover {  
background-color: #333;  
border: 1px solid #333;  
}
```

When you do, you get a form like this:



10. Create a Two-column Footer

Alright, we are now getting towards the end of the Bootstrap tutorial. The last thing you want to add to your landing page is a footer section with two columns. By now, this shouldn't pose much of a problem for you anymore.


```

<footer class="page-footer">
<div class="container">
<div class="row">
<div class="col-lg-8 col-md-8 col-sm-12">
<h6 class="text-uppercase font-weight-bold">Additional Information</h6>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque interdum quam odio, quis
placemat ante luctus eu. Sed aliquet dolor id sapien rutrum, id vulputate quam iaculis.</p>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque interdum quam odio, quis
placemat ante luctus eu. Sed aliquet dolor id sapien rutrum, id vulputate quam iaculis.</p>
</div>
<div class="col-lg-4 col-md-4 col-sm-12">
<h6 class="text-uppercase font-weight-bold">Contact</h6>
<p>1640 Riverside Drive, Hill Valley, California
<br/>info@mywebsite.com
<br/>+ 01 234 567 88
<br/>+ 01 234 567 89</p>
</div>
</div>
<div class="footer-copyright text-center">© 2019 Copyright: MyWebsite.com</div>
</footer>

```

Besides the usual grid markup, this section highlights a few possibilities to modify typography with Bootstrap:

- text-uppercase
- font-weight-bold
- text-center

It should be pretty clear from the names of the classes what they do.

In addition to the above, you can use styling like this:

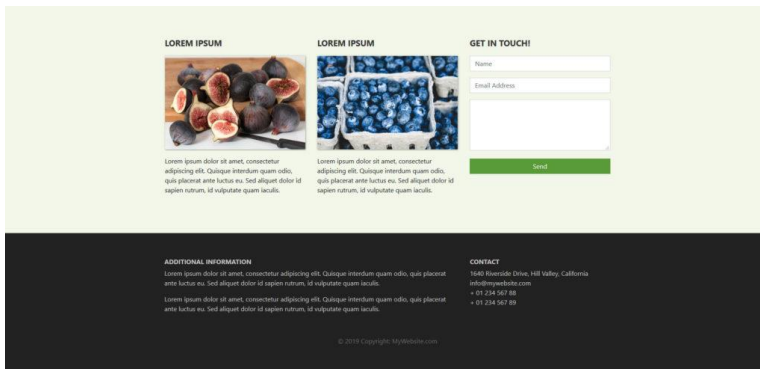
```

.page-footer {
background-color: #222;
color: #ccc;
padding: 60px 0 30px;
}

```

```
.footer-copyright {
  color: #666;
  padding: 40px 0;
}
```

This results in a beautiful footer that looks like so:



11. Add Media Queries

The page is basically ready by now. It is also fully responsive. However, in the mobile view of the browser, the upper section doesn't come out quite right yet.



However, no worries, you can correct that quite easily with a simple media query. Unless you are using SASS for compiling your Bootstrap site, these work the same way as in other instances. You just need to keep in mind the preset break points included in Bootstrap.

As a consequence, to correct the above problem, you can simply include a piece of code like this:

```
@media (max-width: 575.98px) {
```

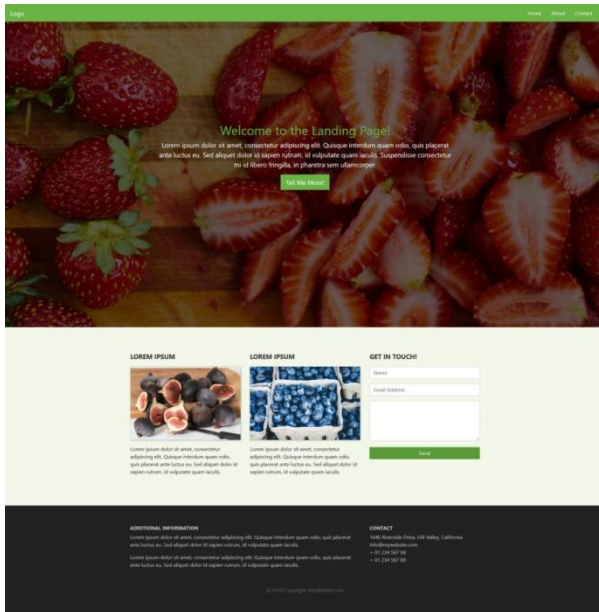
```
.description {  
  left: 0;  
  padding: 0 15px;  
  position: absolute;  
  top: 10%;  
  transform: none;  
  text-align: center;  
}  
.description h1 {  
  font-size: 2em;  
}  
.description p {  
  font-size: 1.2rem;  
}  
.features {  
  margin: 0;  
}  
}
```

After that, everything is as it should be:



13. Putting it Altogether

If you have been following along, you should now be set with page very similar to the one below.



Looks great, doesn't it? Plus, it also works on mobile and is completely responsive.

Not bad for a few lines of code, right? With this, you have all you need to create a landing page with Bootstrap.

Glyphicons

Glyphicons are icon fonts which you can use in your web projects. Bootstrap includes 260 glyphs from the Glyphicon Halflings set. Glyphicons Halflings are normally not available for free, but their creator has made them available for Bootstrap free of cost. As a thank you, you should include a link back to Glyphicons whenever possible.

Use glyphicons in text, buttons, toolbars, navigation, or forms:

Reviewing the chapter

- Bootstrap is an open source, front-end development framework anyone can use for free. It allows you to quickly prototype designs, create web pages and generally hit the ground running.
- Bootstrap is a front-end framework that helps you build mobile responsive websites more quickly and easily
- Bootstrap helps you set them up by offering a large number of CSS classes you can easily apply to HTML elements to create the site components you need
- Bootstrap includes a responsive, mobile first fluid grid system that appropriately scales up to 12 columns as the device or viewport size increases.
- A jumbotron indicates a big grey box for calling extra attention to some special content or information.

- Bootstrap comes with a number of jQuery plugins that can provide additional functionality such as carousels, buttons, tooltips and more
- Bootstrap 4 is built on these standards: HTML5, CSS3, JavaScript and Sass
- The use of rems has been employed also to introduce a new typographic component: display heading
- An alternative way to set up Bootstrap is to download it to your hard drive and use the files locally.
- Glyphicons are icon fonts which you can use in your web projects. Bootstrap includes 260 glyphs from the Glyphicon Halflings set.

Testing your skills

1. Glyphicons is mainly used for

a) Slideshow, b) Graphic Images, c) Animation, d) Providing Different icons

2. Which Of The Following Is Correct About Bootstrap Jumbotron?

a) A big grey box for calling extra attention to some special content or information, b) To Use The Jumbotron: Create A Container <div> With The Class Of .jumbotron, c) Both (a) And (b), d) None Of The Above

3. Bootstrap Is Developed By

a) James Gosling, b) Mark Otto And Jacob Thornton, c) Mark Jukervich, d) None Of Them

4. A Standard Navigation Tab Is Created With:

a) <ul Class="navigation-tabs">, b) <ul Class="nav Tabs">, c) <ul Class="navnav-tabs">, d) <ul Class="navnav-navbar">

5. Bootstrap's Grid System Allows Up To_____ Columns Across The Page

a) 6, b) 12, c) 24, d) 8

Lesson 23: Material design

Objective: After completing this lesson you will be able to learn about :

- Containers
- Responsive Breakpoints
- Grid System
- Media Object

Materials Required:

- Computer With Windows XP and above
- Stable Internet connection

Self- Learning Duration: 60 minutes

Practical Duration: 60 minutes

Total Duration: 120 minutes

Containers

Containers are the most basic layout element in Bootstrap and are required when using our default grid system. Choose from a responsive, fixed-width container (meaning its max-width changes at each breakpoint) or fluid-width (meaning it's 100% wide all the time).

While containers can be nested, most layouts do not require a nested container.



```
<div class="container">  
  <!-- Content here -->  
</div>
```

Use .container-fluid for a full width container, spanning the entire width of the viewport.



```
<div class="container-fluid">  
  ...  
</div>
```

Responsive breakpoints

Since Bootstrap is developed to be mobile first, we use a handful of media queries to create sensible breakpoints for our layouts and interfaces. These breakpoints are mostly based on minimum viewport widths and allow us to scale up elements as the viewport changes.

Bootstrap primarily uses the following media query ranges—or breakpoints—in our source Sass files for our layout, grid system, and components.

```
// Extra small devices (portrait phones, less than 576px)
```

```
// No media query since this is the default in Bootstrap
```

```
// Small devices (landscape phones, 576px and up)
```

```
@media (min-width: 576px) { ... }
```

```
// Medium devices (tablets, 768px and up)
```

```
@media (min-width: 768px) { ... }
```

```
// Large devices (desktops, 992px and up)
```

```
@media (min-width: 992px) { ... }
```

```
// Extra large devices (large desktops, 1200px and up)
```

```
@media (min-width: 1200px) { ... }
```

Since we write our source CSS in Sass, all our media queries are available via Sass mixins:

```
@include media-breakpoint-up(xs) { ... }
```

```
@include media-breakpoint-up(sm) { ... }
```

```
@include media-breakpoint-up(md) { ... }
```

```
@include media-breakpoint-up(lg) { ... }
```

```
@include media-breakpoint-up(xl) { ... }
```

```
// Example usage:
```

```
@include media-breakpoint-up(sm) {
```

```
  .some-class {  
    display: block;
```

```
  }
```

```
}
```

We occasionally use media queries that go in the other direction (the given screen size or smaller):

```
// Extra small devices (portrait phones, less than 576px)
```

```
@media (max-width: 575.98px) { ... }
```

```
// Small devices (landscape phones, less than 768px)
@media (max-width: 767.98px) { ... }

// Medium devices (tablets, less than 992px)
@media (max-width: 991.98px) { ... }

// Large devices (desktops, less than 1200px)
@media (max-width: 1199.98px) { ... }

// Extra large devices (large desktops)
// No media query since the extra-large breakpoint has no upper bound on its width
```

Grid System

Bootstrap's grid system uses a series of containers, rows, and columns to layout and align content. It's built with flexbox and is fully responsive. Below is an example and an in-depth look at how the grid comes together.

```
<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>
```

The output:

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.

A clear and more comprehensive look into the process (how it actually works):

Breaking it down, here's how it works

- Containers provide a means to center and horizontally pad your site's contents. Use `.container` for a responsive pixel width or `.container-fluid` for width: 100% across all viewport and device sizes.
- Rows are wrappers for columns. Each column has horizontal padding (called a gutter) for controlling the space between them. This padding is then counteracted on the rows with negative margins. This way, all the content in your columns is visually aligned down the left side.
- In a grid layout, content must be placed within columns and only columns may be immediate children of rows.
- Thanks to flexbox, grid columns without a specified width will automatically layout as equal width columns. For example, four instances of `.col-sm` will each automatically be 25% wide from the small breakpoint and up. See the auto-layout columns section for more examples.
- Column classes indicate the number of columns you'd like to use out of the possible 12 per row. So, if you want three equal-width columns across, you can use `.col-4`.
- Column widths are set in percentages, so they're always fluid and sized relative to their parent element.
- Columns have horizontal padding to create the gutters between individual columns, however, you can remove the margin from rows and padding from columns with `.no-gutters` on the `.row`.
- To make the grid responsive, there are five grid breakpoints, one for each responsive breakpoint: all breakpoints (extra small), small, medium, large, and extra large.
- Grid breakpoints are based on minimum width media queries, meaning they apply to that one breakpoint and all those above it (e.g., `.col-sm-4` applies to small, medium, large, and extra large devices, but not the first xs breakpoint).
- You can use predefined grid classes (like `.col-4`) or Sass mixins for more semantic markup.

Media Object

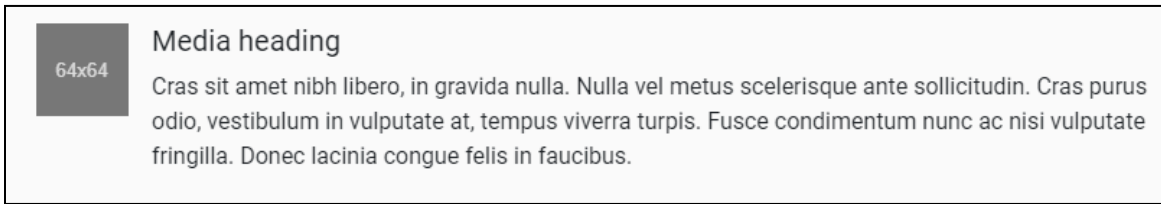
The media object helps build complex and repetitive components where some media is positioned alongside content that doesn't wrap around said media. Plus, it does this with only two required classes thanks to flexbox.

Below is an example of a single media object. Only two classes are required—the wrapping `.media` and the `.media-body` around your content. Optional padding and margin can be controlled through spacing utilities.

```
<div class="media">
  
  <div class="media-body">
    <h5 class="mt-0">Media heading</h5>

    Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio,
    vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla.
    Donec lacinia congue felis in faucibus.
  </div>
</div>
```

The output:



Media objects can be infinitely nested, though we suggest you stop at some point. Place nested .media within the .media-body of a parent media object.

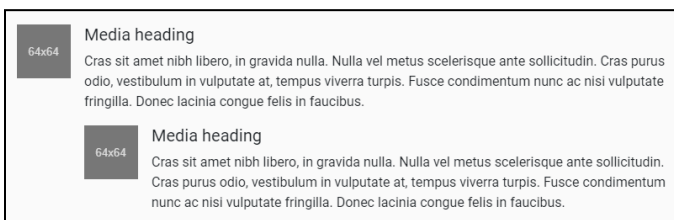
Below is an example to understand the proceedings:

```
<div class="media">
  
  <div class="media-body">
    <h5 class="mt-0">Media heading</h5>

    Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.
    <div class="media mt-3">
      <a class="pr-3" href="#">
        
      </a>
      <div class="media-body">
        <h5 class="mt-0">Media heading</h5>

        Cras sit amet nibh libero, in gravida nulla. Nulla vel metus scelerisque ante sollicitudin. Cras purus odio, vestibulum in vulputate at, tempus viverra turpis. Fusce condimentum nunc ac nisi vulputate fringilla. Donec lacinia congue felis in faucibus.
      </div>
    </div>
  </div>
</div>
```

The output:



Reviewing the chapter

- Containers are the most basic layout element in Bootstrap and are required when using our default grid system. They can be fluid-width, fixed-width, or responsive.
- While containers can be nested, most layouts do not require a nested container
- Responsive breakpoints are mostly based on minimum viewport widths and allow us to scale up elements as the viewport changes.
- Bootstrap's grid system uses a series of containers, rows, and columns to layout and align content. It's built with flexbox and is fully responsive.
- The media object helps build complex and repetitive components where some media is positioned alongside content that doesn't wrap around said media
- Media objects can be infinitely nested, though we suggest you stop at some point

Testing your skills

1. Which of the following are type of containers

a) fluid width, b) fixed-width, c) responsive, **d) all of the above**

2. In which type of container its max-width changes at each breakpoint

a) fluid width, **b) fixed-width**, c) responsive, d) none of the above

3. _____ allow us to scale up elements as the viewport changes

a) Responsive Breakpoints, b) Bootstrap, c) Containers, d) None of the above

4. Grid System uses which of the following to align and layout content

a) Containers, b) rows and columns c) only (b), **d) both (a) and (b)**

5. Column widths are set in _____

a) Decimal, **b) Percentage**, c) cm, d) mm