FUTURE VISION BIE

One Stop for All Study Materials

& Lab Programs

Future Vision

By K B Hemanth Raj

Scan the QR Code to Visit the Web Page

Or

Visit : https://hemanthrajhemu.github.io

Gain Access to All Study Materials according to VTU,
CSE – Computer Science Engineering,
ISE – Information Science Engineering,
ECE - Electronics and Communication Engineering
& MORE...

Join Telegram to get Instant Updates: https://bit.ly/VTU_TELEGRAM

Contact: MAIL: futurevisionbie@gmail.com

INSTAGRAM: www.instagram.com/hemanthraj_hemu/

INSTAGRAM: www.instagram.com/futurevisionbie/

WHATSAPP SHARE: https://bit.ly/FVBIESHARE

HTML Tables and Forms

Objectives

1. Introducing Tables
2. Styling Tables
3. Introducing Forms
4. Form Control Elements
5. Table and Form Accessibility
6. Microformats

Section 1 of 6

INTRODUCING TABLES

HTML Tables

A grid of cells

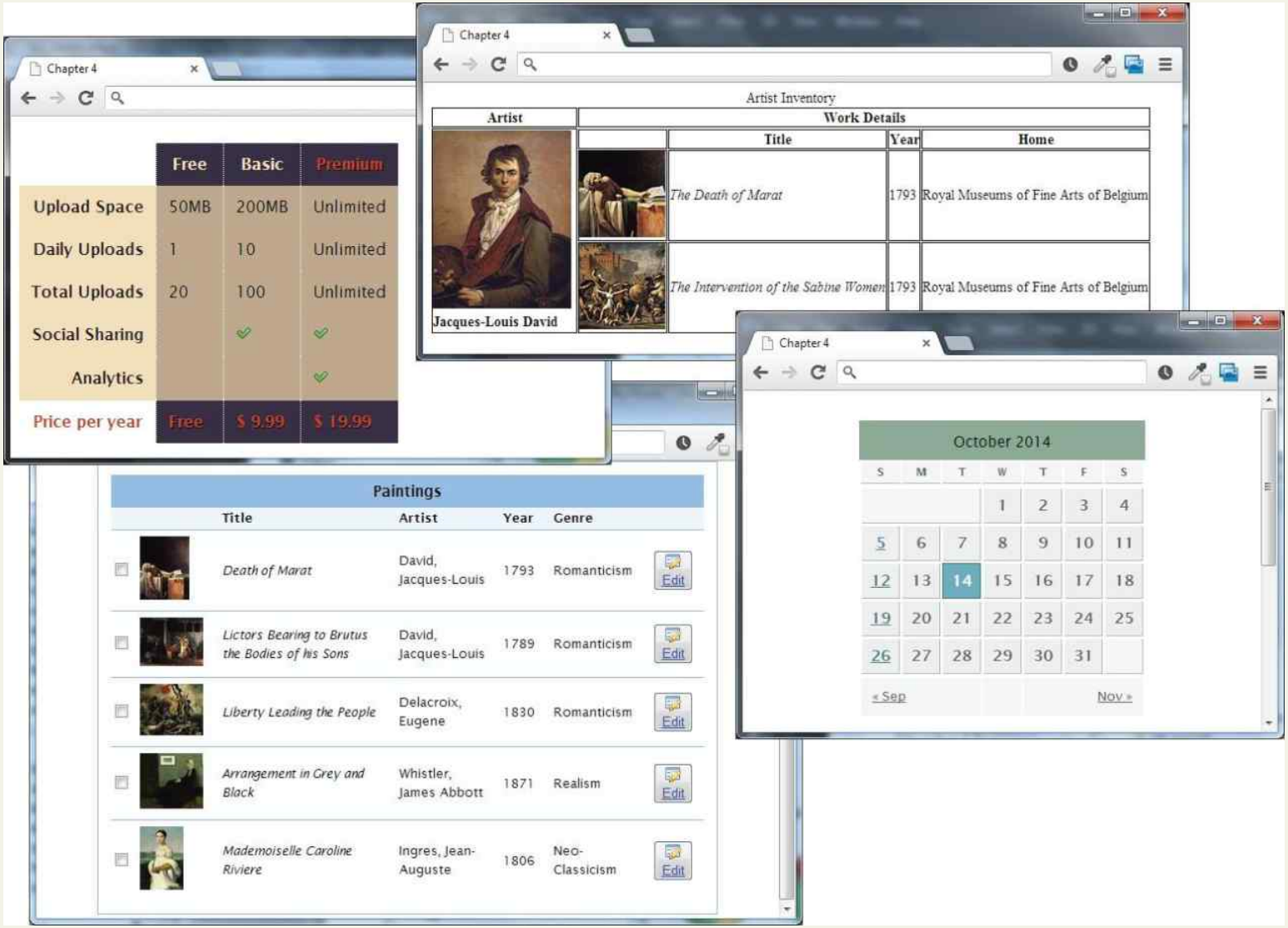
A table in HTML is created using the **<table>** element

Tables can be used to display:

- Many types of content
  - Calendars, financial data, lists, etc…
- Any type of data
  - Images
  - Text
  - Links
  - Other tables

HTML Tables

Example usages



Tables Basics

Rows and cells

- an HTML **<table>** contains any number of rows (**<tr>**)
- each row contains any number of table data cells (**<td>**)
- Content goes inside of **<td></td>** tags

```
<table>
        <tr>
                <td>The Death of Marat</td>
        </tr>
</table>
```

content

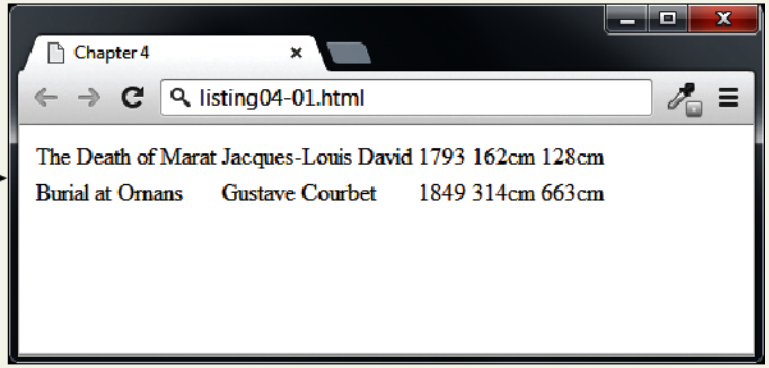
A basic Example

<table>

<tr>

The Death of Marat <td>	Jacques-Louis David <td>	1793 <td>	162cm <td>	128cm <td>
Burial at Ornans <td>	Gustave Courbet <td>	1849 <td>	314cm <td>	663cm <td>

```
<table>
    <tr>
        <td>The Death of Marat</td>
        <td>Jacques-Louis David</td>
        <td>1793</td>
        <td>162cm</td>
        <td>128cm</td>
    </tr>
    <tr>
        <td>Burial at Ornans</td>
        <td>Gustave Courbet</td>
        <td>1849</td>
        <td>314cm</td>
        <td>663cm</td>
    </tr>
</table>
```



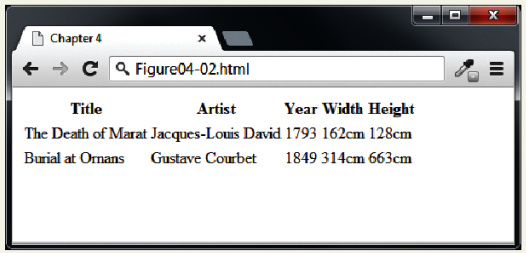
With Table Headings

<table>

<tr>	Title <th>	Artist <th>	Year <th>	Width <th>	Height <th>
<tr>	The Death of Marat <td>	Jacques-Louis David <td>	1793 <td>	162cm <td>	128cm <td>
<tr>	Burial at Ornans <td>	Gustave Courbet <td>	1849 <td>	314cm <td>	663cm <td>

th

```
<table>
   <tr>
      <th>Title</th>
      <th>Artist</th>
      <th>Year</th>
      <th>Width</th>
      <th>Height</th>
   </tr>
   <tr>
      <td>The Death of Marat</td>
      <td>Jacques-Louis David</td>
      <td>1793</td>
      <td>162cm</td>
      <td>128cm</td>
   </tr>
   <tr>
      <td>Burial at Ornans</td>
      <td>Gustave Courbet</td>
      <td>1849</td>
      <td>314cm</td>
      <td>663cm</td>
   </tr>
</table>
```



Why Table Headings

A table heading <th>

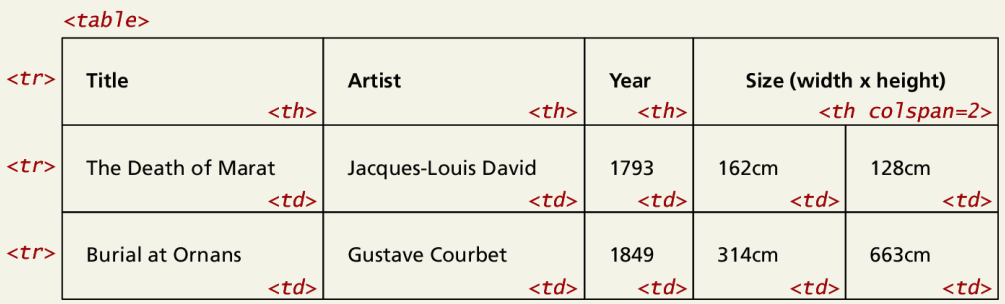
- Browsers tend to make the content within a <th> element bold
- <th> element for accessibility (it helps those using screen readers)
- Provides some semantic info about the row being a row of headers

Spanning Rows and Columns

Span Span Span a Row

Each row must have the same number of <td> or <th> containers. If you want a given cell to cover several columns or rows,

use the **colspan or rowspan attributes**



Notice that this row now only has four cell elements.

```
<table>
  <tr>
    <th>Title</th>
    <th>Artist</th>
    <th>Year</th>
    <th colspan="2">Size (width x height)</th>
  </tr>
  <tr>
    <td>The Death of Marat</td>
    <td>Jacques-Louis David</td>
    <td>1793</td>
    <td>162cm</td>
    <td>128cm</td>
  </tr>
  ...
</table>
```


https://hemanthrajhemu.github.io

Using Tables for Layout

It works in many situations

- Popular in 1990s
- Results in table bloat
- Not semantic
- Larger HTML pages
- Browser quirks

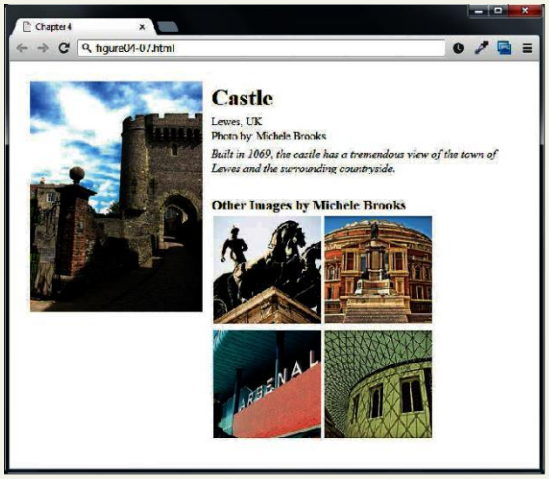
<table>

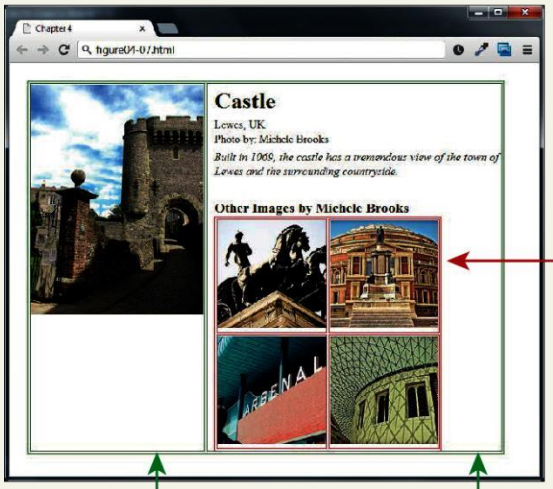
Artist <th>	Title <th>	Year <th>	<tr>
Jacques-Louis David <td rowspan=3>	The Death of Marat <td>	1793 <td>	<tr>
	The Intervention of the Sabine Women <td>	1799 <td>	<tr>
	Napoleon Crossing the Alps <td>	1800 <td>	<tr>

```
<table>
   <tr>
      <th>Artist</th>
      <th>Title</th>
      <th>Year</th>
   </tr>
   <tr>
      <td rowspan="3">Jacques-Louis David</td>
      <td>The Death of Marat</td>
      <td>1793</td>
   </tr>
   <tr>
      <td>The Intervention of the Sabine Women</td>
      <td>1799</td>
   </tr>
   <tr>
      <td>Napoleon Crossing the Alps</td>
      <td>1800</td>
   </tr>
   ...
</table>
```

Notice that these two rows now only have two cell elements.

Example Table layouts






```
<table>
  <tr>
    <td>
      <img src="images/959.jpg" alt="Castle"/>
    </td>
    <td>
      <h2>Castle</h2>
      <p>Lewes, UK</p>
      <p>Photo by: Michele Brooks</p>
      <p>Built in 1069, the castle has a tremendous
         view of the town of Lewes and the
         surrounding countryside.
      </p>

      <h3>Other Images by Michele Brooks</h3>

      <table>
        <tr>
          <td><img src="images/464.jpg" /></td>
          <td><img src="images/537.jpg" /></td>
        </tr>
        <tr>
          <td><img src="images/700.jpg" /></td>
          <td><img src="images/828.jpg" /></td>
        </tr>
      </table>
    </td>
  </tr>
</table>
```

Additional table tags

- <caption>
- <col>,<colgroup>
- <thead>
- <tfoot>
- <tbody>

A title for the table is good for accessibility.

These describe our columns, and can be used to aid in styling.

Table header could potentially also include other <tr> elements.

Yes, the table footer comes *before* the body.

Potentially, with styling the browser can scroll this information, while keeping the header and footer fixed in place.

```
<table>
<caption>19th Century French Paintings</caption>
<col class="artistName" />
<colgroup id="paintingColumns">
    <col />
    <col />
</colgroup>

<thead>
   <tr>
      <th>Title</th>
      <th>Artist</th>
      <th>Year</th>
   </tr>
</thead>

<tfoot>
   <tr>
      <td colspan="2">Total Number of Paintings</td>
      <td>2</td>
   </tr>
</tfoot>

<tbody>
   <tr>
      <td>The Death of Marat</td>
      <td>Jacques-Louis David</td>
      <td>1793</td>
   </tr>
   <tr>
      <td>Burial at Ornans</td>
      <td>Gustave Courbet</td>
      <td>1849</td>
   </tr>
</tbody>

</table>
```

Section 2 of 6

STYLING TABLES

Styling Tables

The old way'sdeprecated

In HTML5 it is left to CSS, However legacy support for deprecated HTML attributes still exist

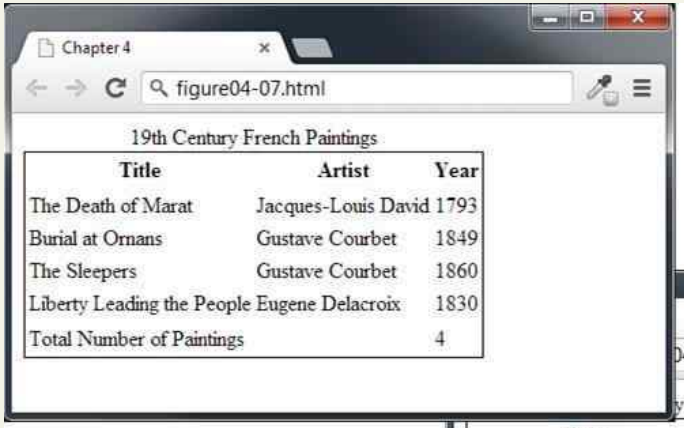
- **width, height**—for setting the width and height of cells
- **cellspacing**—for adding space between every cell in the table
- **cellpadding**—for adding space between the content of the cell

and its border

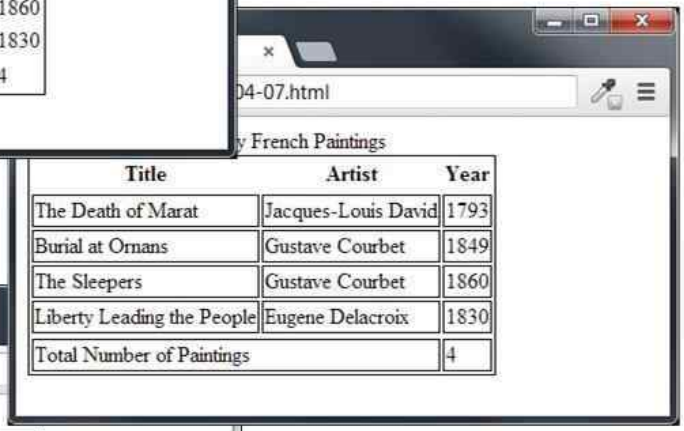
- **bgcolor**—for changing the background color of any table element
- **background**—for adding a background image to any table element
- **align**—for indicating the alignment of a table in relation to the surrounding container

Styling Tables

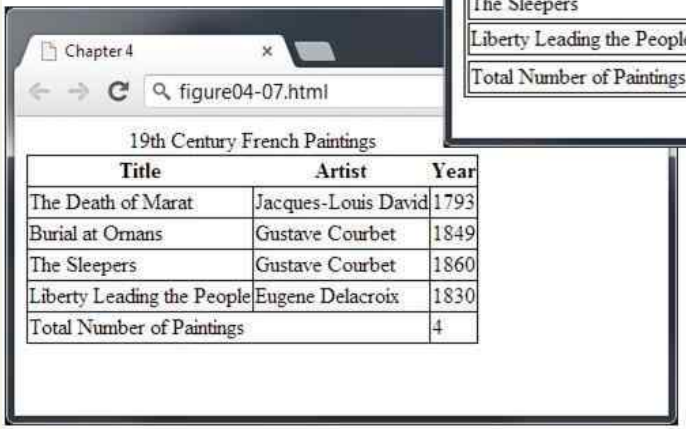
Borders




```
table {
   border: solid 1pt black;
}
```




```
table {
   border: solid 1pt black;
}
td {
   border: solid 1pt black;
}
```

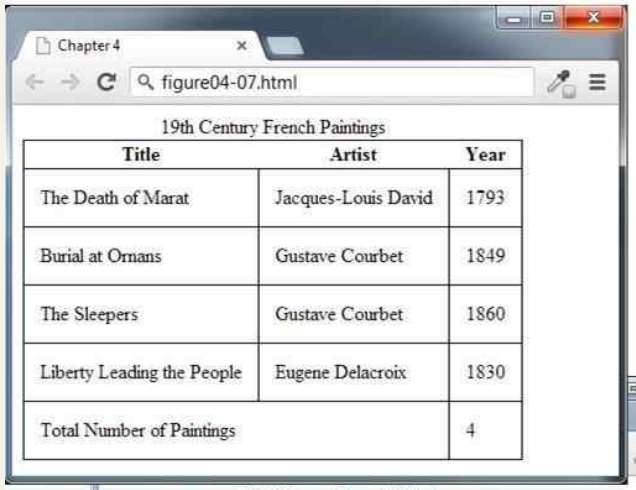



```
table {
   border: solid 1pt black;
   border-collapse: collapse;
}
td {
   border: solid 1pt black;
}
```

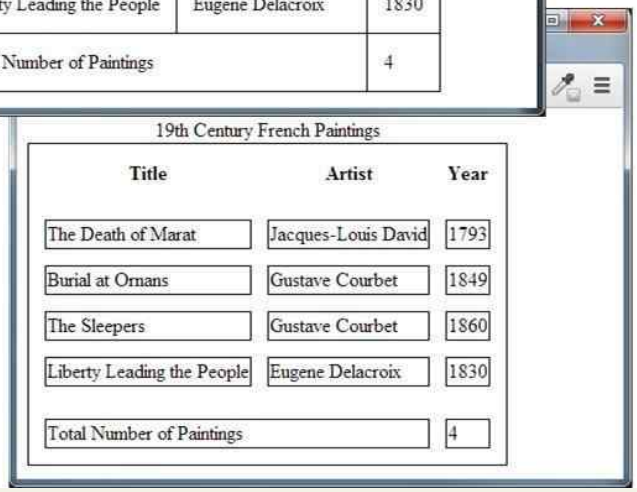

https://hemanthrajhemu.github.io

Styling Tables

Padding and spacing




```
table {
   border: solid 1pt black;
   border-collapse: collapse;
}
td {
   border: solid 1pt black;
   padding: 10pt;
}
```

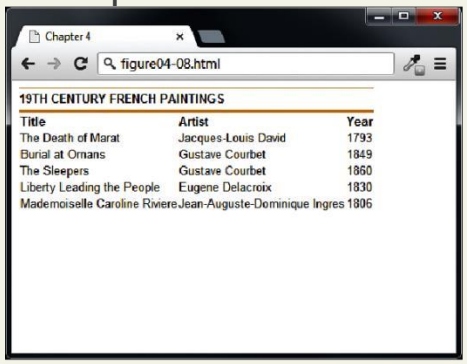



```
table {
   border: solid 1pt black;
   border-spacing: 10pt;
}
td {
   border: solid 1pt black;
}
```

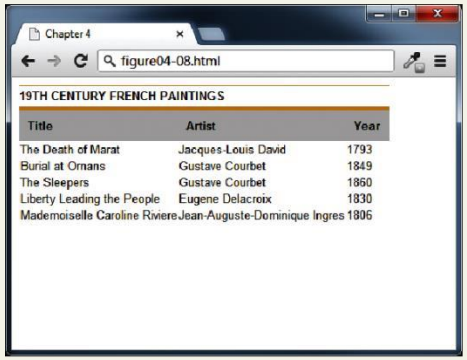

https://hemanthrajhemu.github.io

Styling Tables

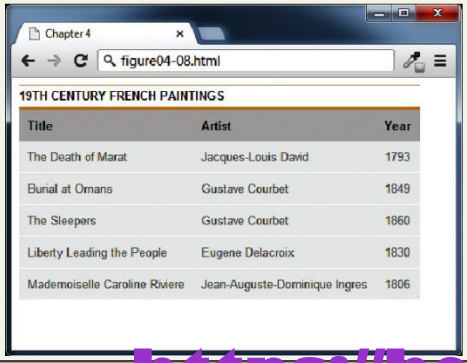
Examples




```
table {
   font-size: 0.8em;
   font-family: Arial, Helvetica, sans-serif;
   border-collapse: collapse;
   border-top: 4px solid #DCA806;
   border-bottom: 1px solid white;
   text-align: left;
}
caption {
   font-weight: bold;
   padding: 0.25em 0 0.25em 0;
   text-align: left;
   text-transform: uppercase;
   border-top: 1px solid #DCA806;
}
```



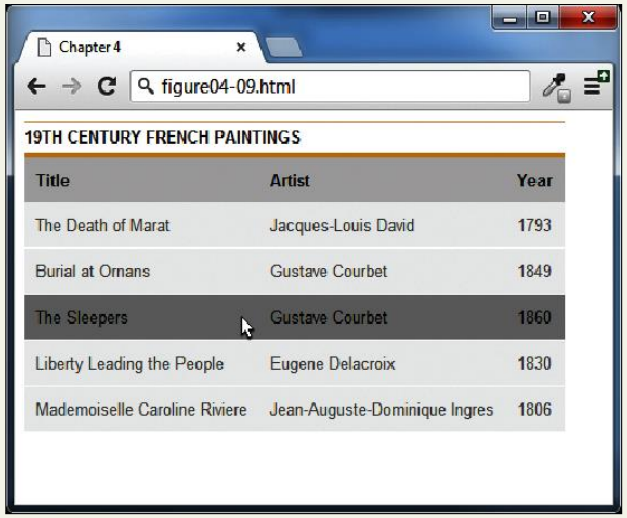

```
thead tr {
   background-color: #CACACA;
}
th {
   padding: 0.75em;
}
```




```
tbody tr {
   background-color: #F1F1F1;
   border-bottom: 1px solid white;
   color: #6E6E6E;
}
tbody td {
   padding: 0.75em;
}
```

Nth-Child

Nifty Table styling tricks: hover effect and zebra-stripes




```
tbody tr:hover {
    background-color: #9e9e9e;
    color: black;
}
```


```
tbody tr:nth-child(odd) {
    background-color: white;
}
```

Section 3 of 6

INTRODUCING FORMS

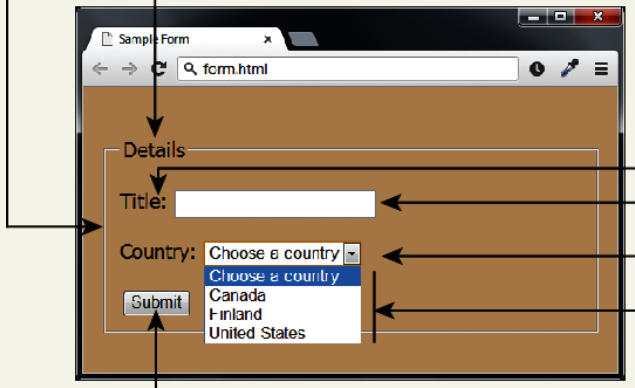
HTML Forms

Richer way to interact with server

Forms provide the user with an alternative way to interact with a web server.

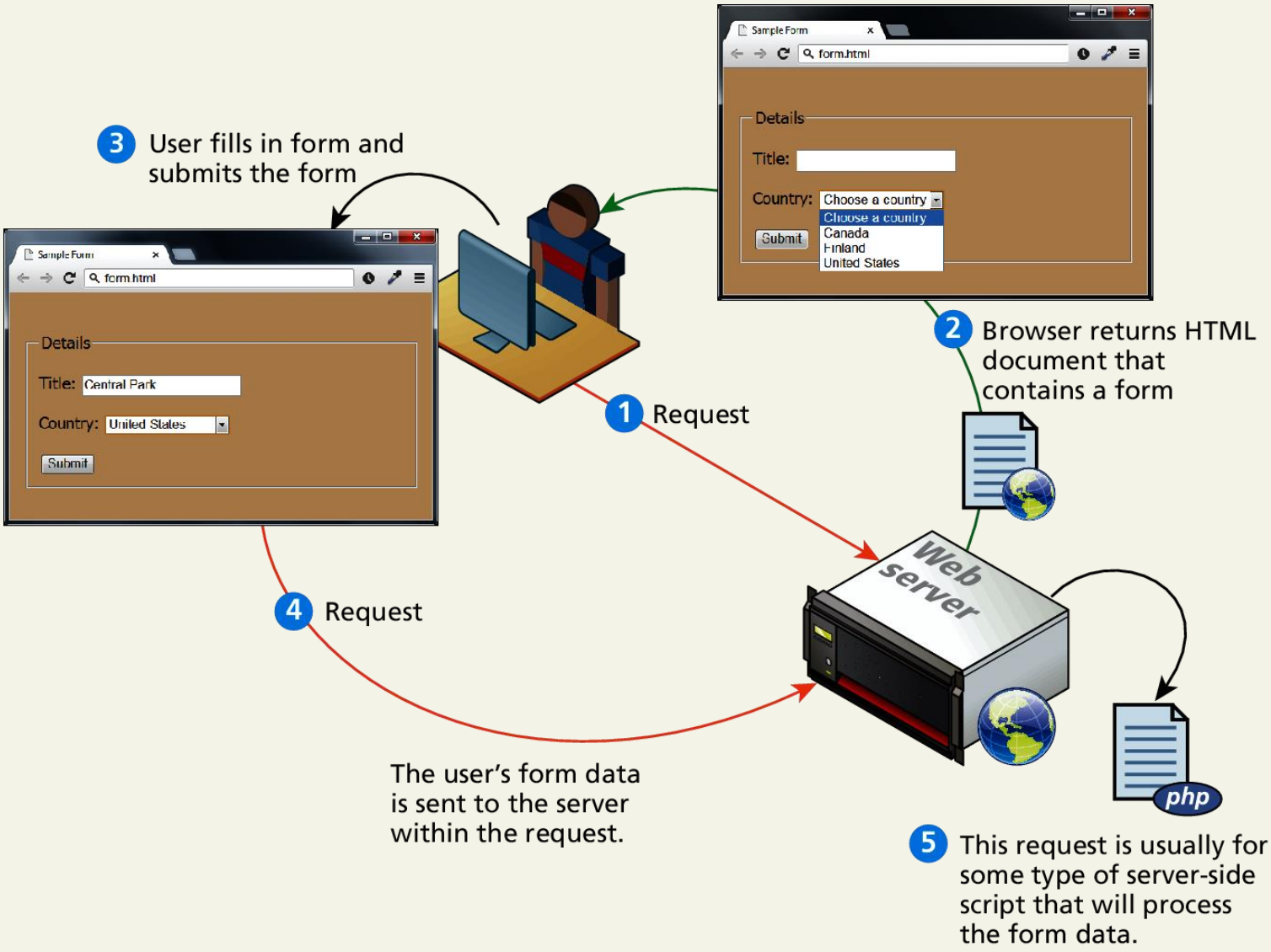
- Forms provide rich mechanisms like:
  - Text input
  - Password input
  - Options Lists
  - Radio and check boxes

Form Structure




```
<form method="get" action="process.php">
  <fieldset>
    <legend>Details</legend>
    <p>
      <label>Title: </label>
      <input type="text" name="title" />
    </p>
    <p>
      <label>Country: </label>
      <select name="where">
        <option>Choose a country</option>
        <option>Canada</option>
        <option>Finland</option>
        <option>United States</option>
      </select>
    </p>
    <input type="submit" />
  </fieldset>
</form>
```

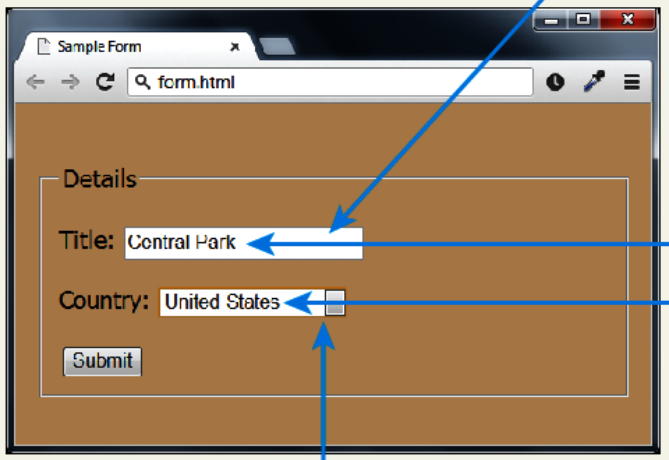
How forms interact with servers



Query Strings

At the end of the day, another string

```
<input type="text" name="title" />
```



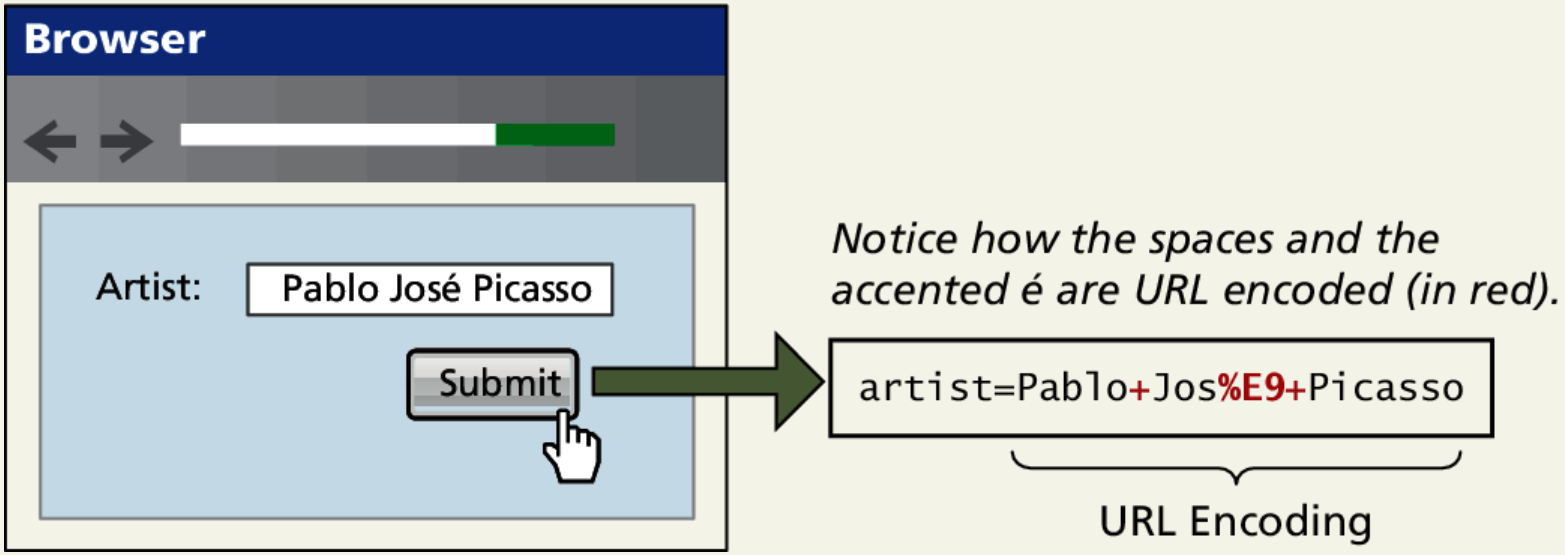

```
title=Central+Park&where=United+States
```

```
<select name="where">
```


https://hemanthrajhemu.github.io

URL encoding

Special symbols



<form> element

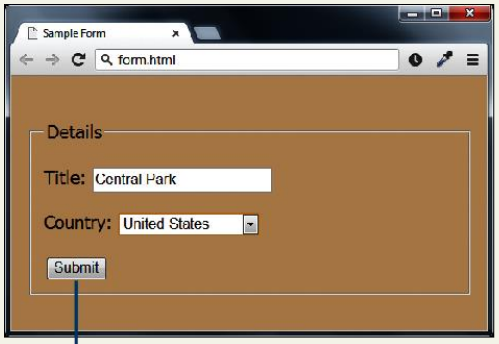
Two essential features of any form, namely the **action**
and the **method** attributes.

- The **action** attribute specifies the URL of the server-side resource that will process the form data

- The **method** attribute specifies how the query string data will be transmitted from the browser to the server.

  - GET
  - POST

GET vs POST




```
<form method="get" action="process.php">
```

```
GET /process.php?title=Central+Park&where=United+States http/1.1
```

querystring

```
<form method="post" action="process.php">
```

```
POST /process.php http/1.1
Date: Sun, 20 May 2012 23:59:59 GMT
Host: www.mysite.com
User-Agent: Mozilla/4.0
Content-Length: 47

title=Central+Park&where=United+States
```

HTTP Header

querystring

GET vs POST

Advantages and Disadvantages

- Data can be clearly seen in the address bar.
- Data remains in browser history and cache.
- Data can be bookmarked
- Limit on the number of characters in the form data returned.

POST

- Data can contain binary data.
- Data is hidden from user.
- Submitted data is not stored in cache, history, or bookmarks.

Section 4 of 6

FORMS CONTROL ELEMENTS

Form-Related HTML Elements

Type	Description
<button>	Defines a clickable button.
<datalist>	An HTML5 element form defines lists to be used with other form elements.
<fieldset>	Groups related elements in a form together.
<form>	Defines the form container.
<input>	Defines an input field. HTML5 defines over 20 different types of input.
<label>	Defines a label for a form input element.
<legend>	Defines the label for a fieldset group.
<option>	Defines an option in a multi-item list.
<optgroup>	Defines a group of related options in a multi-item list.
<select>	Defines a multi-item list.
<textarea>	Defines a multiline text entry box.

Text Input Controls

Type	Description
text	Creates a single line text entry box. **<input type="text" name="title" />**
textarea	Creates a multiline text entry box. **<textarea rows="3" ... />**
password	Creates a single line text entry box for a password **<input type="password" ... />**
search	Creates a single-line text entry box suitable for a search string. This is an HTML5 element. **<input type="search" …/>**
email	Creates a single-line text entry box suitable for entering an email address. This is an HTML5 element. **<input type="email" …/>**
tel	Creates a single-line text entry box suitable for entering a telephone. This is an HTML5 element. **<input type="tel" …/>**
url	Creates a single-line text entry box suitable for entering a URL. This is an HTML5 element. **<input type="url" … />**

Text Input Controls

Classic

```
<input type="text" ... />
```

Text:

```
<textarea>
  enter some text
</textarea>
```

```
<textarea placeholder="enter some text">
</textarea>
```

TextArea: enter some text

TextArea: Enter some text

```
<input type="password" ... />
```

Password:

Password: ••••

Text Input Controls

HTML5

```
<input type="search" placeholder="enter search text" ... />
```


```
<input type="email" ... />
```


```
<input type="url" ... />
```

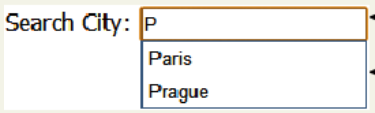

```
<input type="tel" ... />
```

HTML5 advanced controls

Pattern attribute

```
<input type="text" ... placeholder="L#L #L#" pattern="[a-z][0-9][a-z] [0-9][a-z][0-9]"  />
```

datalist




```
<input type="text" name="city" list="cities"  />

<datalist id="cities">
   <option>Calcutta</option>
   <option>Calgary</option>
   <option>London</option>
   <option>Los Angeles</option>
   <option>Paris</option>
   <option>Prague</option>
</datalist>
```

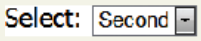
Select Lists

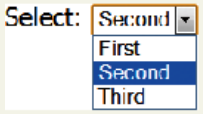
Chose an option, any option.

- **<select>** element is used to create a multiline box for selecting one or more items
  - The options are defined using the **<option>** element
  - can be hidden in a dropdown or multiple rows of the list can be visible
  - Option items can be grouped together via the **<optgroup>** element.

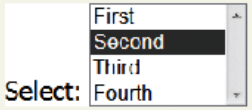
Select Lists

Select List Examples






```
<select name="choices">
  <option>First</option>
  <option selected>Second</option>
  <option>Third</option>
</select>
```




```
<select size="3" ... >
```

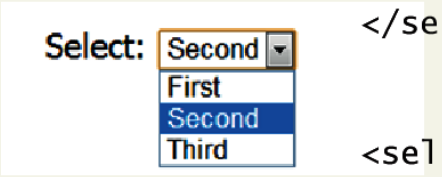

```
<select … >
 <optgroup label="North America">
   <option>Calgary</option>
   <option>Los Angeles</option>
 </optgroup>
 <optgroup label="Europe">
   <option>London</option>
   <option>Paris</option>
   <option>Prague</option>
 </optgroup>
</select>
```

Which Value to send

Select Lists Cont.

The **value** attribute of the <option> element is used to specify what value will be sent back to the server.

The value attribute is optional; if it is not specified, then the text within the container is sent instead




```
<select name="choices">
  <option>First</option>
  <option>Second</option>
  <option>Third</option>
</select>
```

?choices=Second

```
<select name="choices">
  <option value="1">First</option>
  <option value="2">Second</option>
  <option value="3">Third</option>
</select>
```

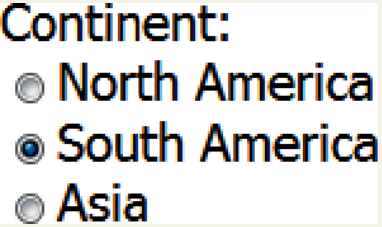
?choices=2

Radio Buttons

Radio buttons are useful when you want the user to select a single item from a small list of choices and you want all the choices to be visible

- radio buttons are added via the **<input type="radio">** element
- The buttons are mutually exclusive (i.e., only one can be chosen) by sharing the same name attribute
- The checked attribute is used to indicate the default choice
- the value attribute works in the same manner as with the <option> element

Radio Buttons




```
<input type="radio" name="where" value="1">North America<br/>
<input type="radio" name="where" value="2" checked>South America<br/>
<input type="radio" name="where" value="3">Asia
```

Checkboxes

Checkboxes are used for getting yes/no or on/off responses from the user.

- checkboxes are added via **the <input type="checkbox”>** Element
- You can also group checkboxes together by having them share the same name attribute
- Each checked checkbox will have its value sent to the server
- Like with radio buttons, the checked attribute can be used to set the default value of a checkbox

Checkboxes

I accept the software license ☑

```
<label>I accept the software license</label>
<input type="checkbox" name="accept" >
```

Where would you like to visit?
☑ Canada
☐ France
☑ Germany

```
<label>Where would you like to visit? </label><br/>
<input type="checkbox" name="visit" value="canada">Canada<br/>
<input type="checkbox" name="visit" value="france">France<br/>
<input type="checkbox" name="visit" value="germany">Germany
```

?accept=on&visit=canada&visit=germany

Button Controls

Type	Description
<input type="submit">	Creates a button that submits the form data to the server.
<input type="reset">	Creates a button that clears any of the user's already entered form data.
<input type="button">	Creates a custom button. This button may require Javascript for it to actually perform any action.
<input type="image">	Creates a custom submit button that uses an image for its display.
<button>	Creates a custom button. The <button> element differs from <input type="button"> in that you can completely customize what appears in the button; using it, you can, for instance, include both images and text, or skip server-side processing entirely by using hyperlinks. You can turn the button into a submit button by using the type="submit" attribute.

Button Controls

```
<input type="submit"  />
```


```
<input type="reset"  />
```

```
<input type="button" value="Click Me" />
```


```
<input type="image" src="appointment.png" />
```

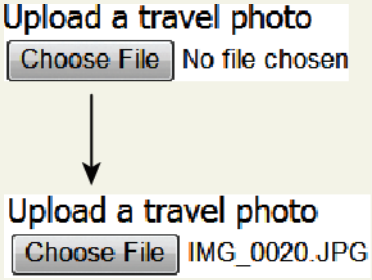

```
<button>
   <a href="email.html">
      <img src="images/email.png" alt=""/>
      Email
   </a>
</button>
```

```
<button type="submit"  >
   <img src="images/edit.png" alt=""/>
   Edit
</button>
```

Specialized Controls

I'm so special

- **<input type=hidden>**
- **<input type=file>**



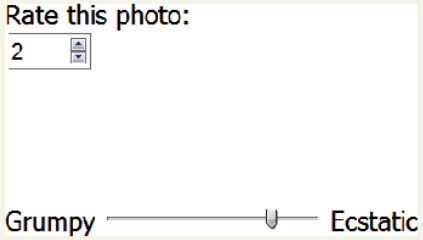

```
<form method="post" enctype="multipart/form-data" ... >
   ...
   <label>Upload a travel photo</label>
   <input type="file" name="photo" />
   …
</form>
```

Number and Range

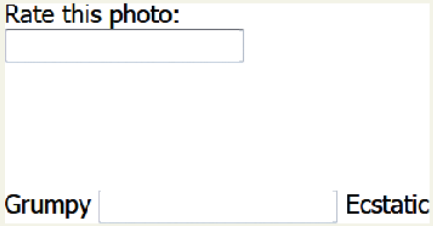
Typically input values need be **validated**. Although server side validation is required, optional client side pre-validation is good practice.

The number and range controls Added in HTML5 provide a way to input numeric values that **eliminates the need for JavaScript numeric validation!!!**

Number and Range




```
<label>Rate this photo: <br/>
<input type="number" min="1" max="5" name="rate" />


Grumpy
<input type="range" min="0" max="10" step="1" name="happiness" />
Ecstatic
```



Controls as they appear in browser
that doesn't support these input types

Color


```
<label>Background Color: <br/>
<input type="color" name="back" />
```

Background Color:

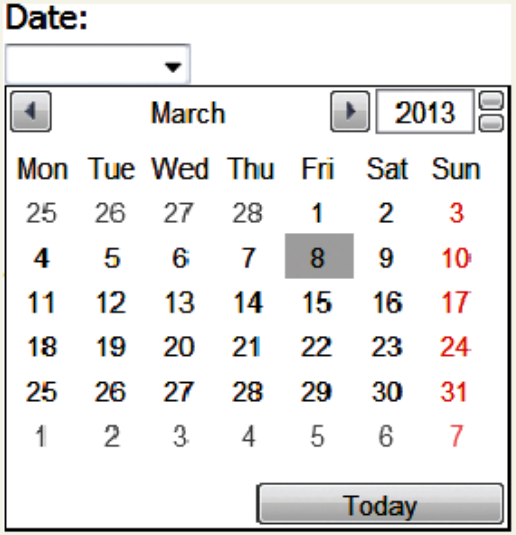
Control as it appears in browser that doesn't support this input type

Date and Time Controls

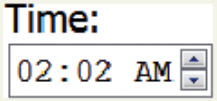
Dates and times often need validation when gathering this information from a regular text input control.

From a user's perspective, entering dates can be tricky as well: you probably have wondered at some point in time when entering a date into a web form, what format to enter it in, whether the day comes before the month, whether the month should be entered as an abbreviation or a number, and so on.

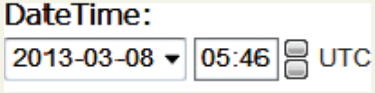
HTML5 Date and Time Controls



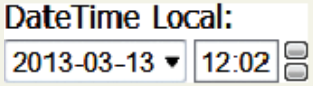

```
<label>Date: <br/>
<input type="date" ... />
```




```
<input type="time" ... />
```

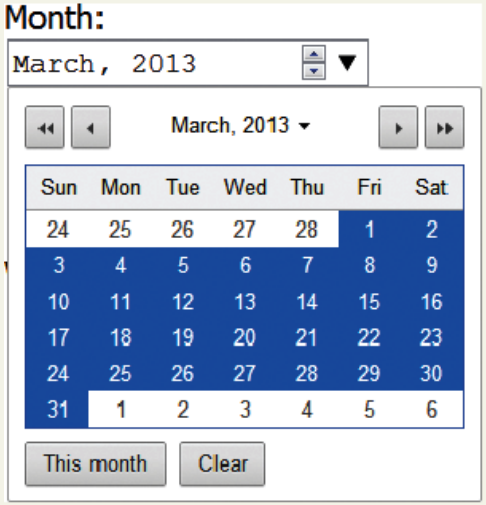



```
<input type="datetime" ... />
```

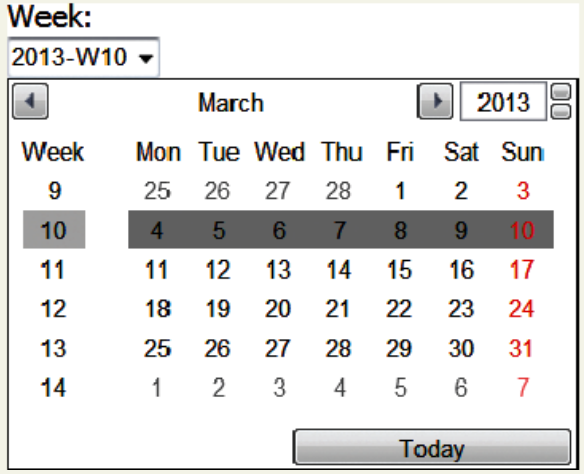



```
<input type="datetime-local" ... />
```

HTML5 Date and Time Controls




```
<input type="month" ... />
```




```
<input type="week" ... />
```

HTML Controls

Type	Description
date	Creates a general date input control. The format for the date is "yyyy-mm-dd".
time	Creates a time input control. The format for the time is "HH:MM:SS", for hours:minutes:seconds.
datetime	Creates a control in which the user can enter a date and time.
datetime-local	Creates a control in which the user can enter a date and time without specifying a time zone.
month	Creates a control in which the user can enter a month in a year. The format is "yyyy-mm".
week	Creates a control in which the user can specify a week in a year. The format is "yyyy-W##".

Other Controls

You mean there's more

- The <progress> and <meter> elements can be used to provide feedback to users,
  - but requires JavaScript to function dynamically.
- The <output> element can be used to hold the output from a calculation.
- The <keygen> element can be used to hold a private key for public-key encryption

Section 5 of 6

TABLE AND FORM ACCESSIBILITY

Web Accessibility

Not all web users are able to view the content on web pages in the same manner.

The term **web accessibility** refers to the assistive technologies, various features of HTML that work with those technologies, and different coding and design practices that can make a site more usable for people with visual, mobility, auditory, and cognitive disabilities.

In order to improve the accessibility of websites, the W3C created the **Web Accessibility Initiative (WAI)**

- Web Content Accessibility Guidelines

Web Content Accessibility Guidelines

- Provide text alternatives for any nontext content so that it can be changed into other forms people need, such as large print, braille, speech, symbols, or simpler language.
- Create content that can be presented in different ways (for example simpler layout) without losing information or structure.
- Make all functionality available from a keyboard.
- Provide ways to help users navigate, find content, and determine where they are.

Accessible Tables

1. Describe the table's content using the <caption> element
2. Connect the cells with a textual description in the header

```
<table>
   <caption>Famous Paintings</caption>
   <tr>
      <th scope="col">Title</th>
      <th scope="col">Artist</th>
      <th scope="col">Year</th>
      <th scope="col">Width</th>
      <th scope="col">Height</th>
   </tr>
   <tr>
      <td>The Death of Marat</td>
      <td>Jacques-Louis David</td>
      <td>1793</td>
```

Accessible Forms

Recall the <fieldset>, <legend>, and <label> elements.

Each <label> element should be associated with a single input element.

```
<label for="f-title">Title: </label>

<input type="text" name="title" id="f-title"/>


<label for="f-country">Country: </label>

<select name="where" id="f-country">
    <option>Choose a country</option>
    <option>Canada</option>
    <option>Finland</option>
    <option>United States</option>
  </select>
```

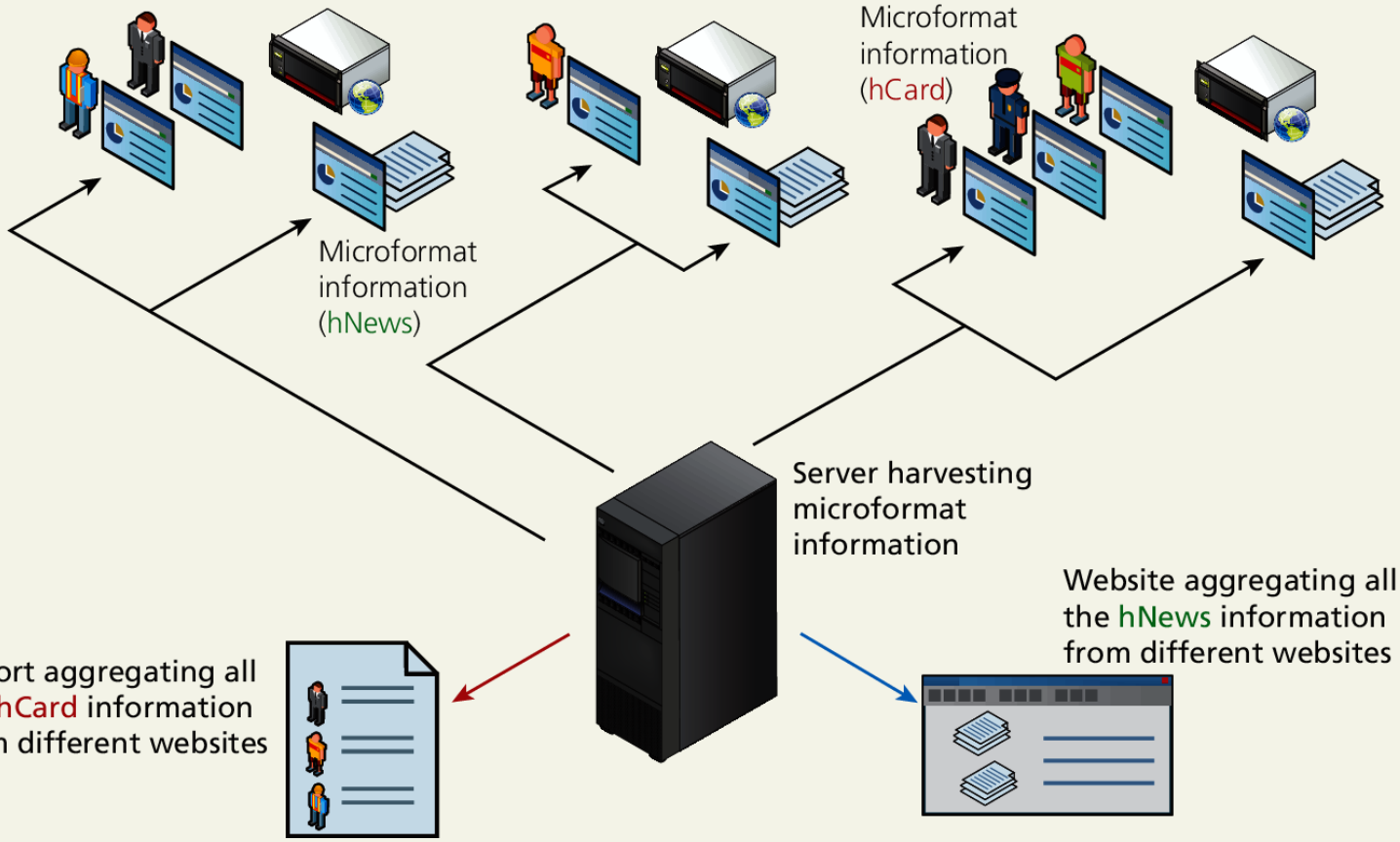
Section 6 of 6

MICROFORMATS

Microformats

A **microformat** is a small pattern of HTML markup and attributes to represent common blocks of information such as people, events, and news stories so that the information in them can be extracted and indexed by software agents

Microformat



What you've learned

1. Introducing Tables
2. Styling Tables
3. Introducing Forms
4. Form Control Elements
5. Table and Form Accessibility
6. Microformats

Chapter 7

ADVANCED CSS: LAYOUT

Approaches to CSS Layout

Fixed Layout

In a fixed layout , the basic width of the design is set by the designer, typically corresponding to an “ideal” width based on a “typical” monitor resolution.

The advantage of a fixed layout is that it is easier to produce and generally has a predictable visual result.

Approaches to CSS Layout

Fixed Layout


```
div#wrapper {
  width: 960px;
  background-color: blue;
}
```

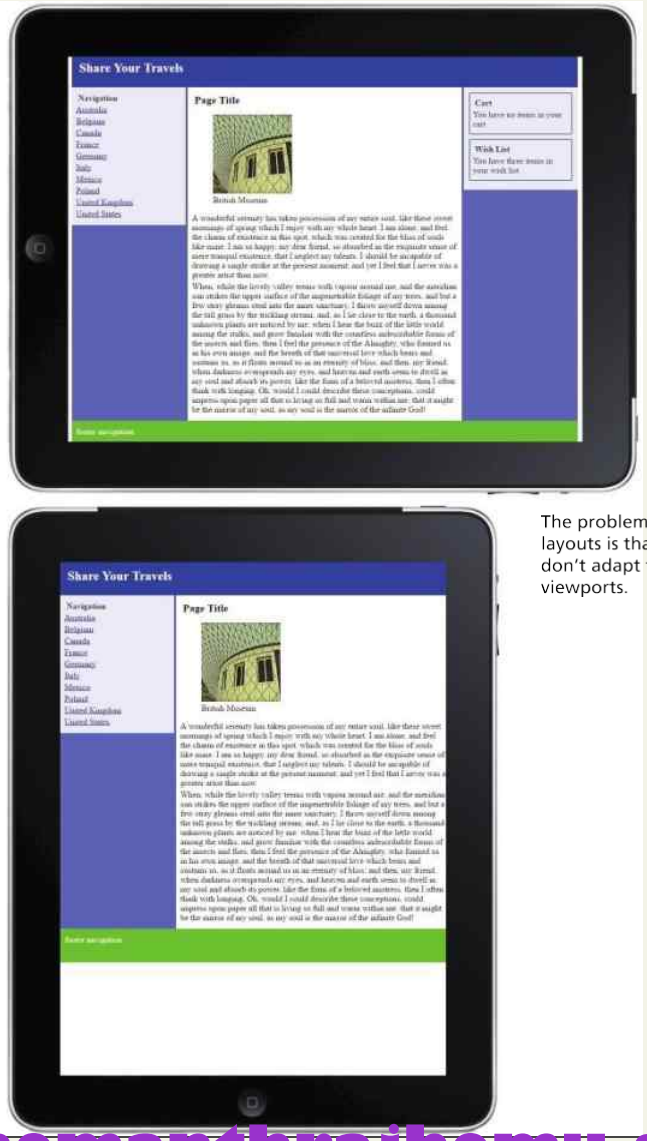
```
<body>
   <div id="wrapper">
      <header>
      ...
      </header>
      <div id="main">
      ...
      </div>
      <footer>
      ...
      </footer>
   </div>
</body>
```


```
div#wrapper {
  width: 960px;
  margin-left: auto;
  margin-right: auto;
  background-color: blue;
}
```


https://hemanthrajhemu.github.io

Approaches to CSS Layout

Problem with Fixed Layout



The problem with fixed layouts is that they don't adapt to smaller viewports.

Approaches to CSS Layout

Liquid Layout

Liquid layout (also called a fluid layout) widths are not specified using pixels, but percentage values

The advantage of a liquid layout is that it adapts to different browser sizes

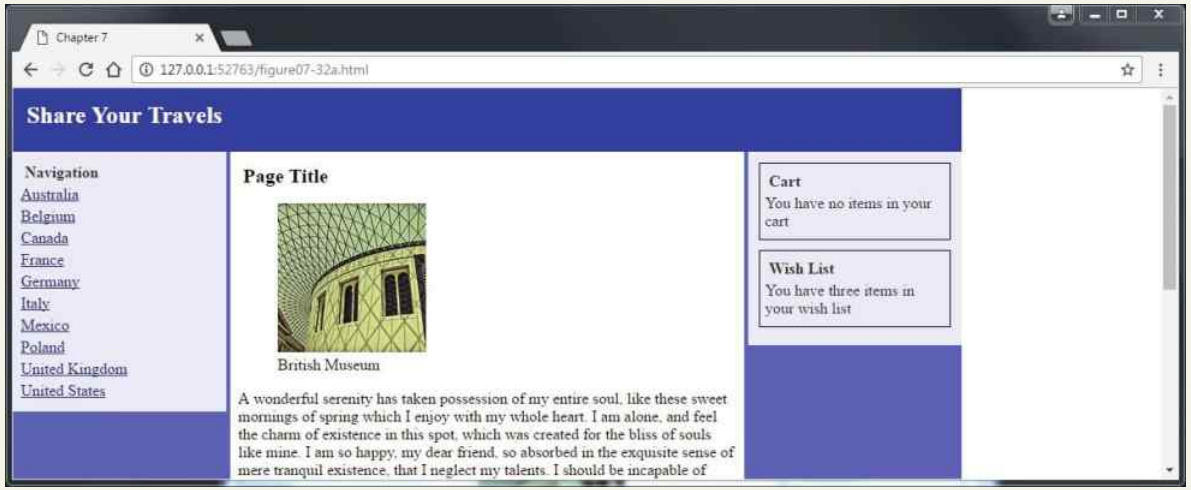
Creating a usable liquid layout is generally more difficult than creating a fixed layout

Approaches to CSS Layout

Liquid Layout

Fluid layouts are based on the browser window.

However, elements can get too spread out as the browser expands.



https://hemanthrajhemu.github.io

Responsive Design

Responsive Layouts

Notice how some elements are scaled to shrink as browser window reduces in size.

When browser shrinks below a certain threshold, then layout and navigation elements change as well.

In this case, the `<ul>` list of hyperlinks changes to a `<select>` and the two-column design changes to one column.

https://hemanthrajhemu.github.io

Responsive Design

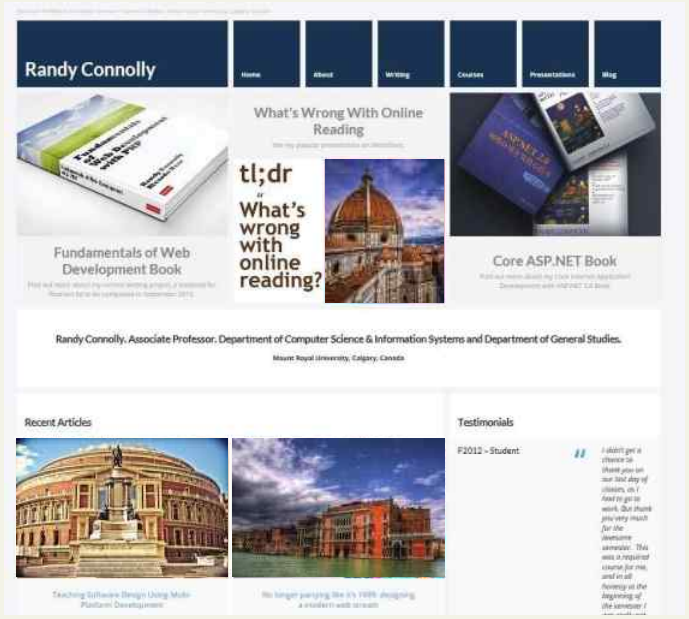
4 elements

1. Liquid layouts
2. Setting viewports via the <meta> tag
3. Customizing the CSS for different viewports using media queries
4. Scaling images to the viewport size

Responsive Design

Setting Viewports

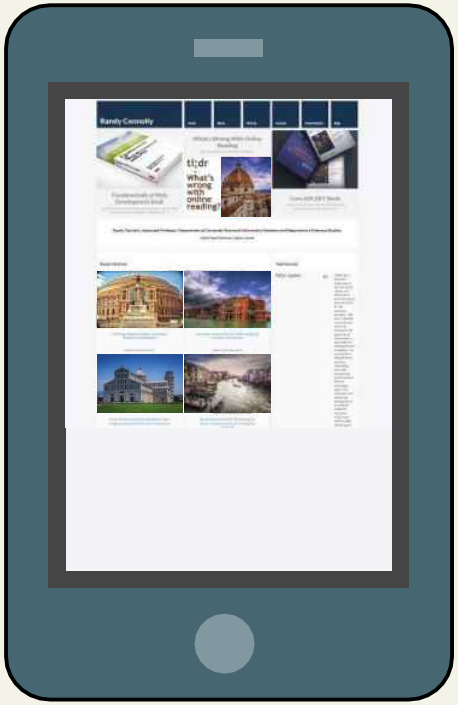
1. Mobile browser renders web page on its viewport



960px

Mobile browser viewport

2. It then scales the viewport to fit within its actual physical screen

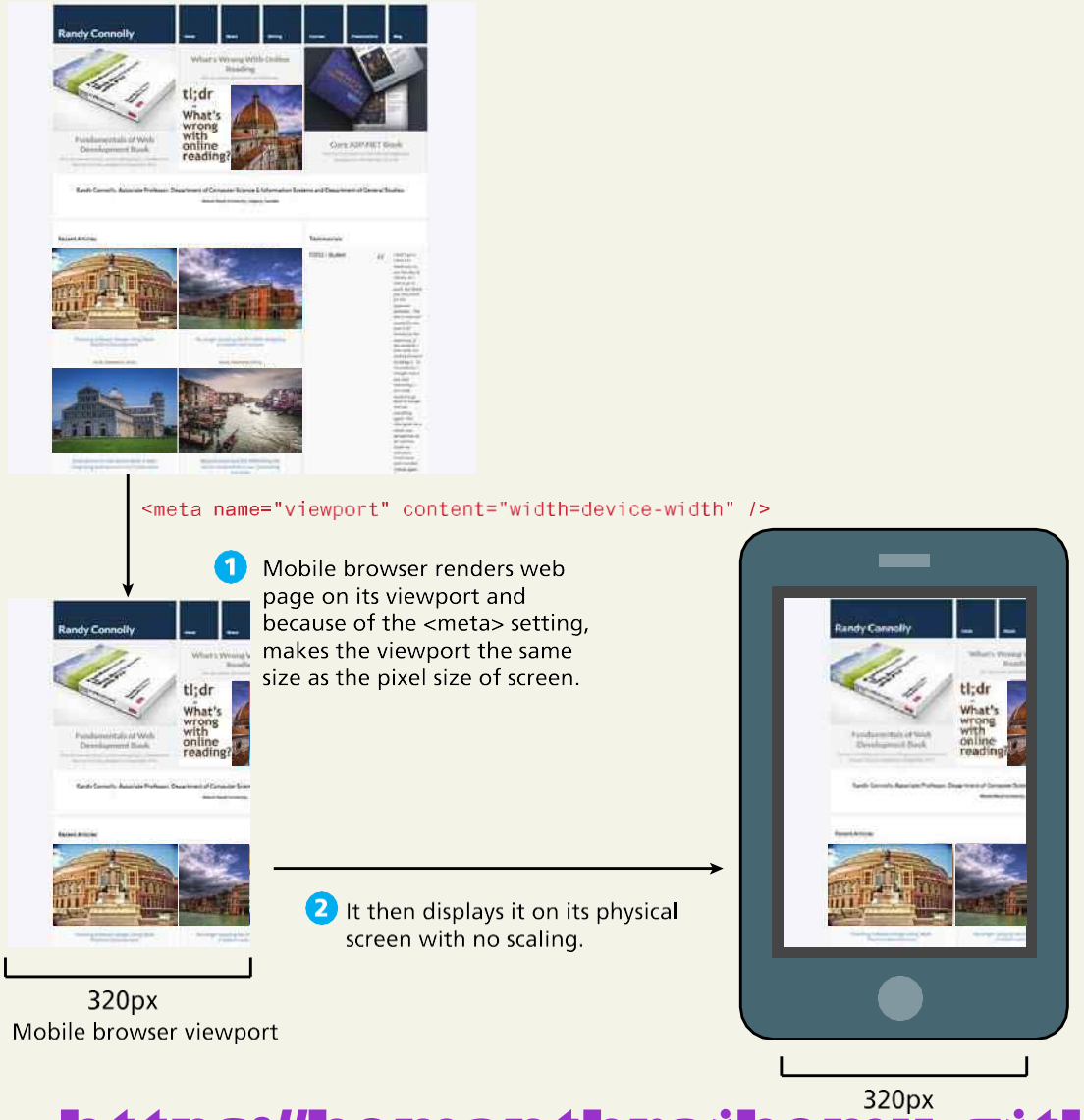


320px

Mobile browser screen

Responsive Design

Setting Viewports

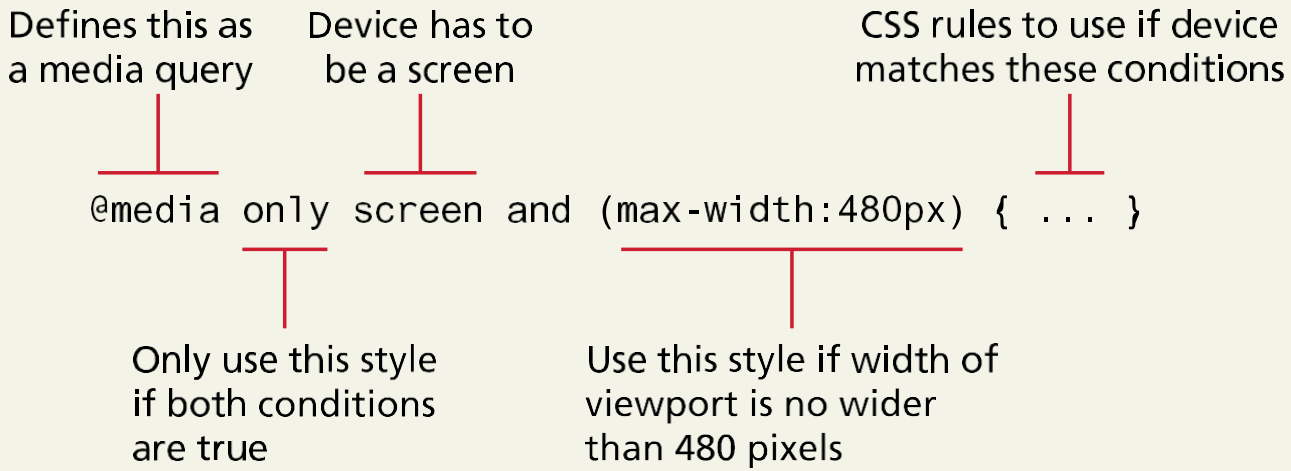


https://hemanthrajhemu.github.io

Responsive Design

Media Queries

A media query is a way to apply style rules based on the medium that is displaying the file



https://hemanthrajhemu.github.io

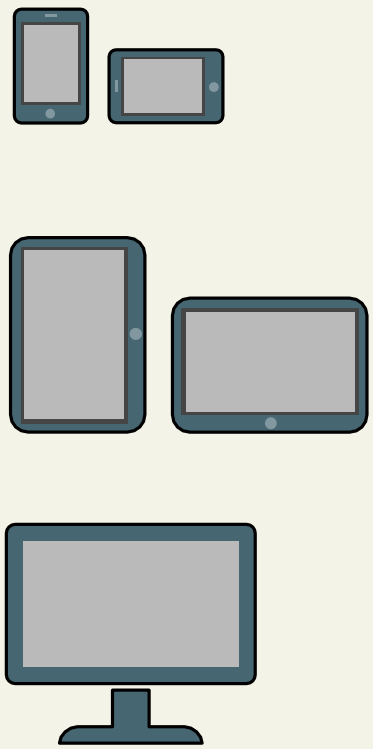
Responsive Design

Media Queries

- **width:** Width of the viewport
- **height:** Height of the viewport
- **device-width:** Width of the device
- **device-height:** Height of the device
- **orientation:** Whether the device is portrait or landscape
- **color:** The number of bits per color

Responsive Design

Media Queries



styles.css

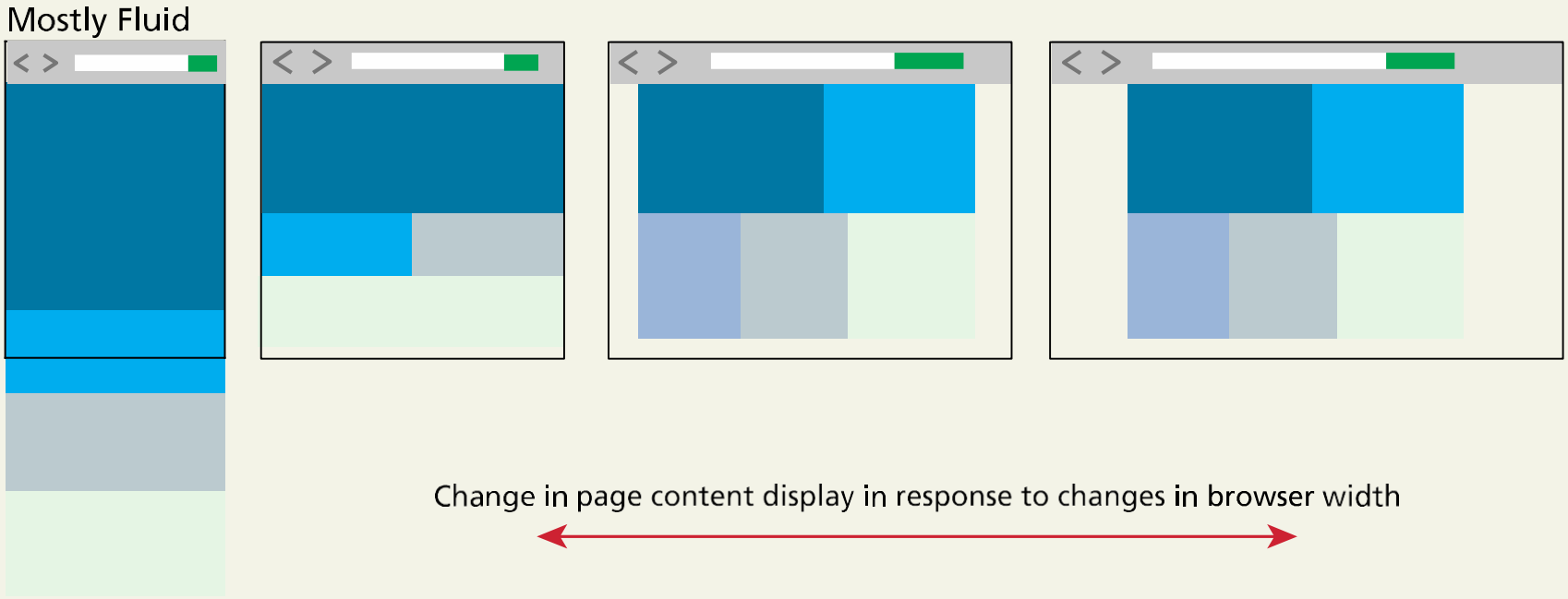
```
/* rules for phones */
@media only screen and (max-width:480px)
{
  #slider-image { max-width: 100%; }
  #flash-ad { display: none; }
    ...
}


/* CSS rules for tablets */
@media only screen and (min-width: 481px)
    and (max-width: 768px)
{
    ...
}



/* CSS rules for desktops */
@media only screen and (min-width: 769px)
{
    ...
}
```

Instead of having all the rules in a single file, we can put them in separate files and add media queries to <link> elements.

```
<link rel="stylesheet" href="mobile.css"  media="screen and (max-width:480px)" />
<link rel="stylesheet" href="tablet.css"  media="screen and (min-width:481px)
    and (max-width:768px)" />
<link rel="stylesheet" href="desktop.css" media="screen and (min-width:769px)" />
```

Responsive Design

Responsive Design Patterns

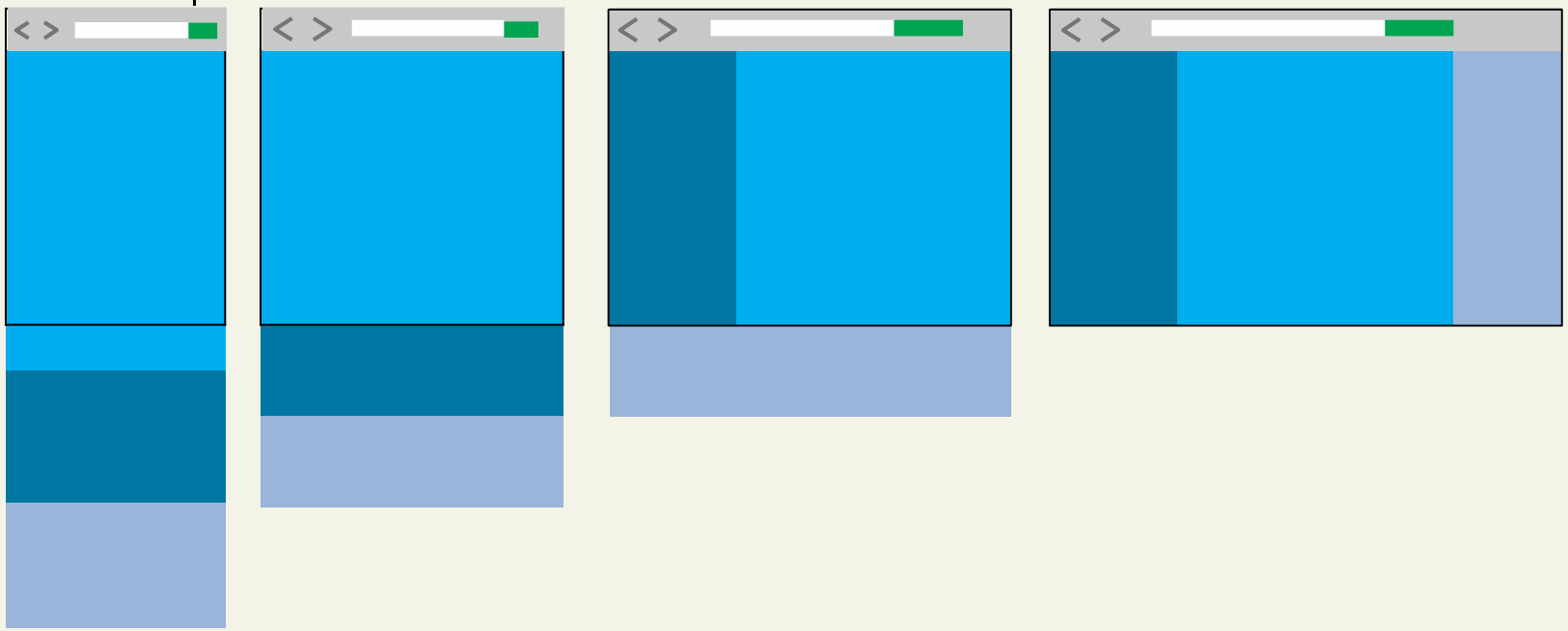


https://hemanthrajhemu.github.io

Responsive Design

Responsive Design Patterns

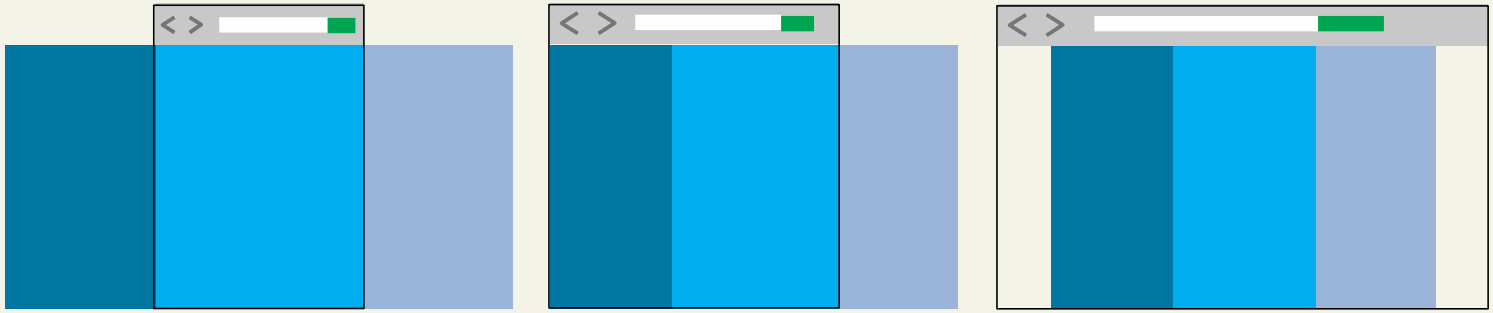
Column Drop



Responsive Design

Responsive Design Patterns

Off Canvas



Flexbox Layout

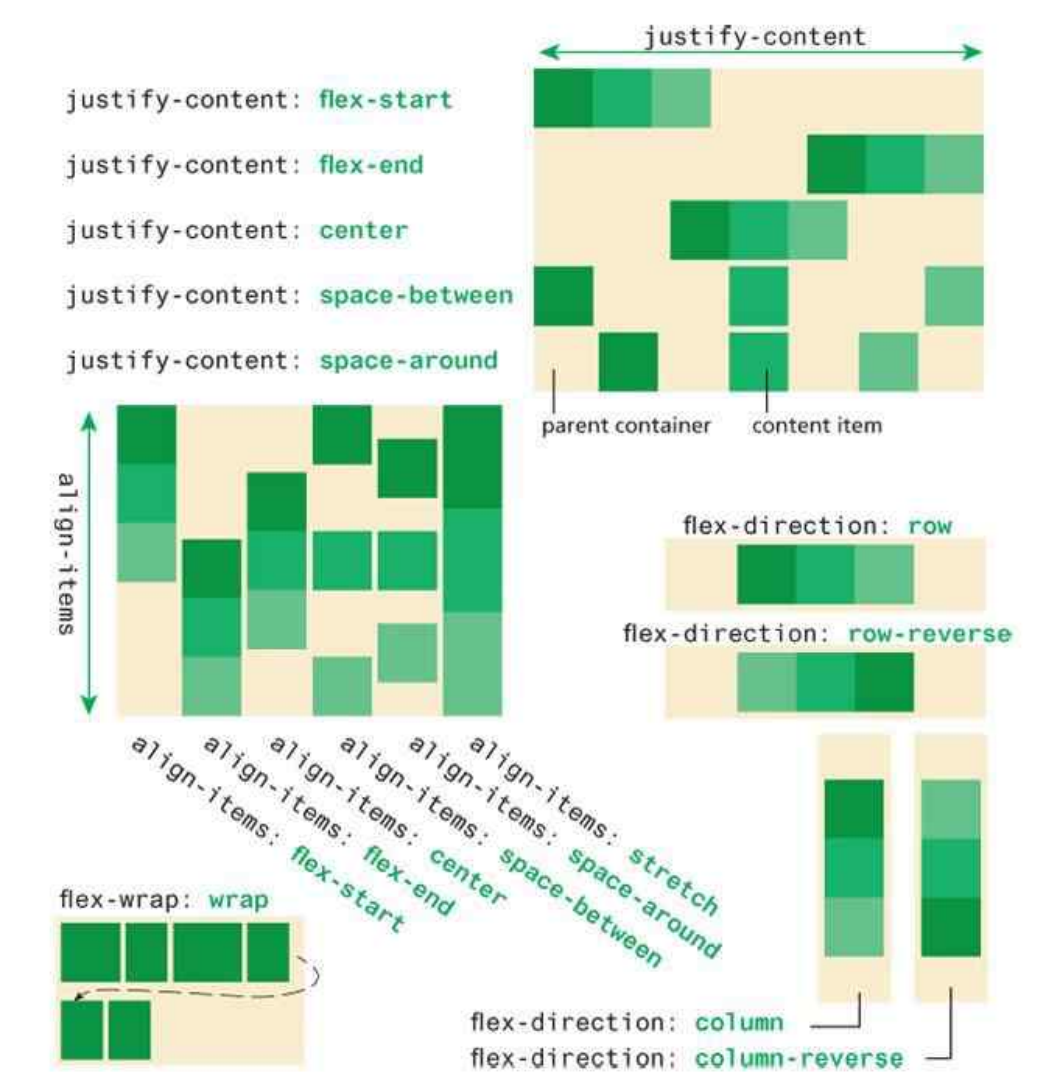
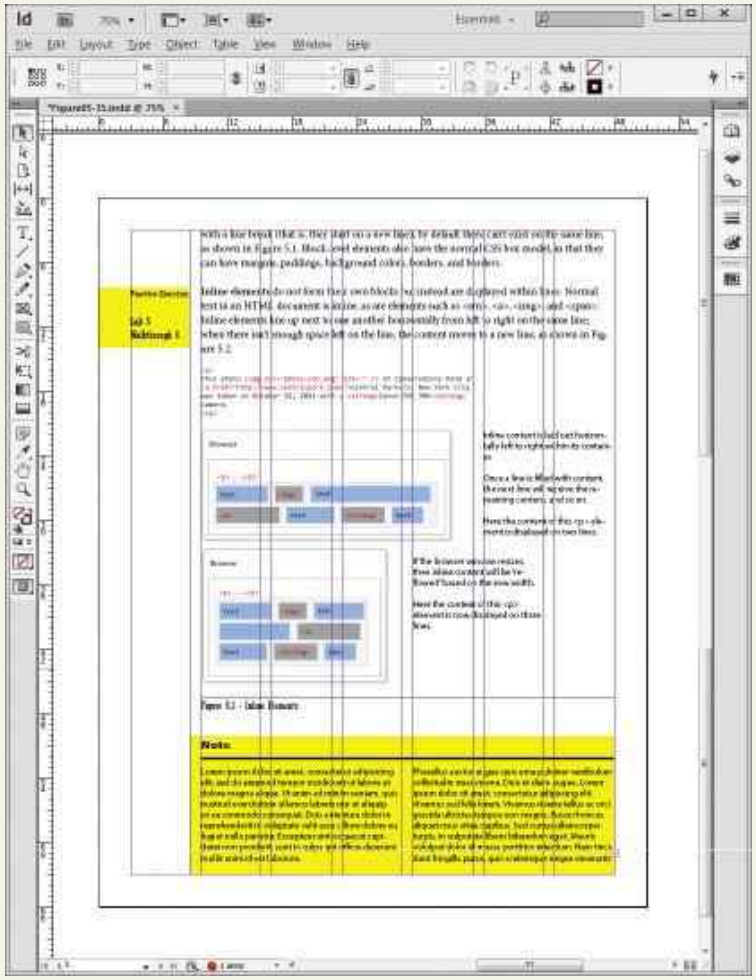


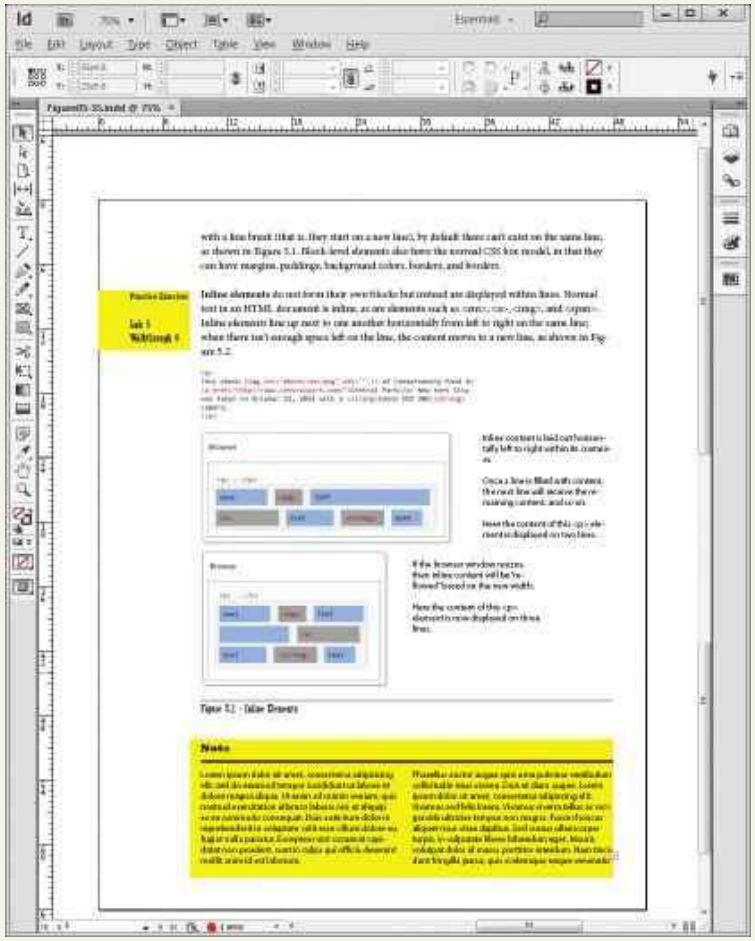
Figure 7.30 The flexbox parent (container) properties

CSS Frameworks and Preprocessors

Grid in print design



Most page design begins with a grid. In this case, a seven-column grid is being used to layout page elements in Adobe InDesign.



Without the gridlines visible, the elements on the page do not look random, but planned and harmonious.

CSS Frameworks and Preprocessors

Using Bootstrap

```
<head>
   <link href="bootstrap.css" rel="stylesheet">
</head>
<body>
   <div class="container">
      <div class="row">
         <div class="col-md-2">
            left column
         </div>
         <div class="col-md-7">
             main content
         </div>
          <div class="col-md-3">
             right column
          </div>
      </div>
   </div>
</body>
```


https://hemanthrajhemu.github.io

CSS Frameworks and Preprocessors

CSS Preprocessors

```
$colorSchemeA: #796d6d;
$colorSchemeB: #9c9c9c;
$paddingCommon: 0.25em;


footer {
  background-color: $colorSchemeA;
  padding: $paddingCommon * 2;
}


@mixin rectangle($colorBack, $colorBorder) {
  border: solid 1pt $colorBorder;
  margin: 3px;
  background-color: $colorBack;
}

fieldset {
  @include rectangle($colorSchemeB, $colorSchemeA);
}

.box {
  @include rectangle($colorSchemeA, $colorSchemeB);
  padding: $paddingCommon;
}
```

Sass source file, e.g., `source.scss`

This example uses Sass (Syntactically Awesome Stylesheets). Here three variables are defined.

You can reference variables elsewhere. Sass also supports math operators on its variables.

A mixin is like a function and can take parameters. You can use mixins to encapsulate common styling.

A mixin can be referenced/called and passed parameters.

The processor is some type of tool that the developer would run.

The output from the processor is a normal CSS file that would then be referenced in the HTML source file.

```
footer {
  padding: 0.50em;
  background-color: #796d6d;
}

fieldset {
  border: solid 1pt #796d6d;
  margin: 3px;
  background-color: #9c9c9c;
}

.box {
  border: solid 1pt #9c9c9c;
  margin: 3px;
  background-color: #796d6d;
  padding: 0.25em;
}
```

Generated CSS file, e.g., `styles.css`