Web Technologies

MCA 4254

DR. MANJUNATH HEGDE

Credits, Classes and Objectives

Credits and Classes

- 4 Credits
- 48 Hrs
- 4 Hrs/Week

At the end of the Course you should be able to:

- Describe the semantics of HTML documentation
- Apply the CSS in UI designing
- Create interactive client side webpages
- Apply Server-Side data management using PHP
- Create event driven pages using application framework

Learning

Introduction to HTML and its components

Introduction to CSS, its types and designing web layouts

Basics of Responsive web Design

Client-Side Programming with Javascript

Introduction to Server-Side Programming with PHP

JSON, Database Operations with PHP

Introduction to AJAX

Angular: Fundamental Architecture, Set-Up and Deployment, Components, Templates, Binding, Forms and Web API.

Introduction

Introduction to Web

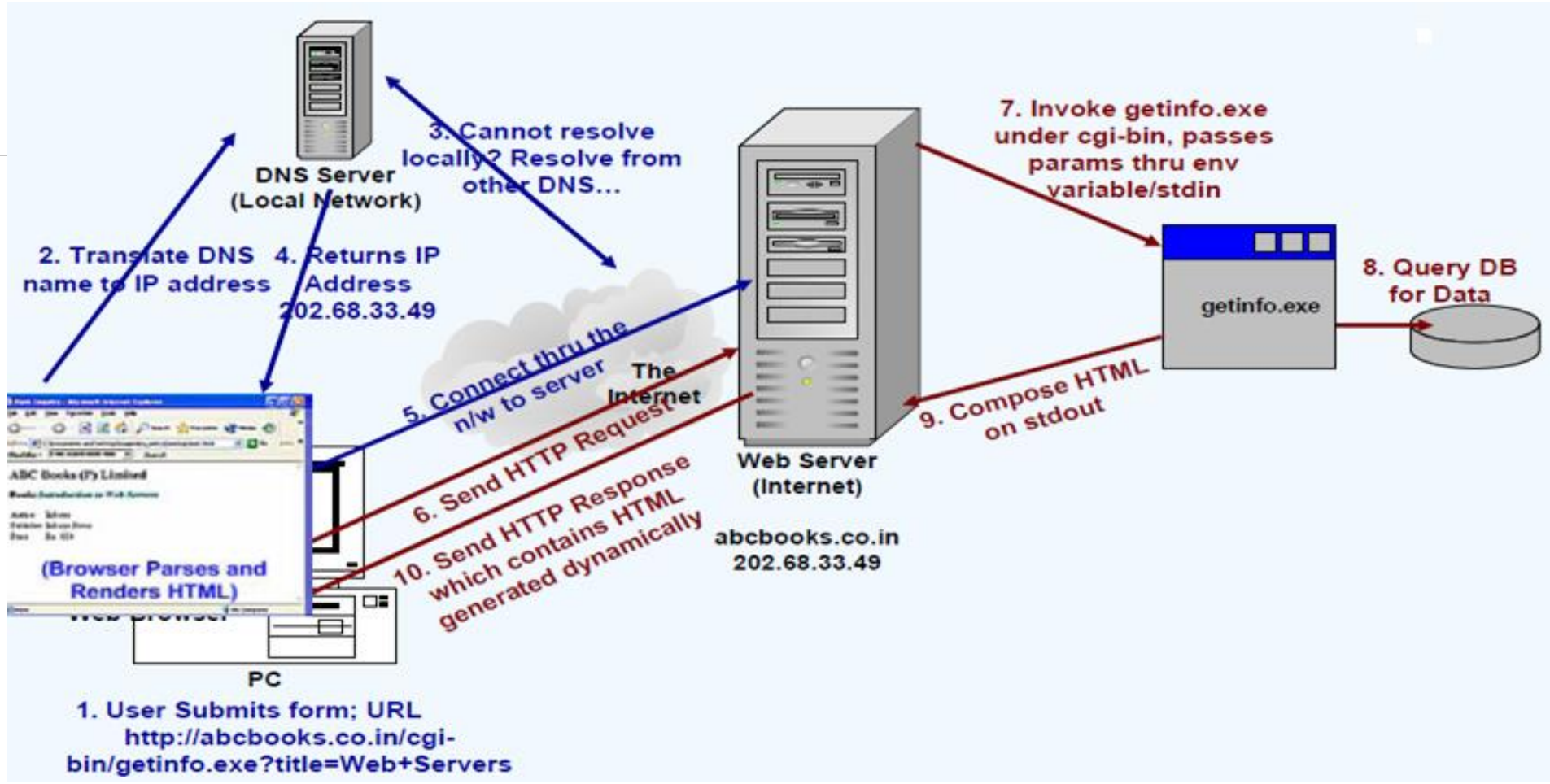
Computers don't communicate with each other the way that people do.

Computers require codes, or directions.

Binary codes and commands allow computers to process needed information.

The methods by which computers communicate with each other through the use of markup languages and multimedia packages is known as web technology.

A markup language is a computer language that uses tags to define elements within a document.


DNS Server
(Local Network)
3. Cannot resolve locally? Resolve from other DNS...
2. Translate DNS name to IP address
4. Returns IP Address 202.68.33.49
7. Invoke getinfo.exe under cgi-bin, passes params thru env variable/stdin
8. Query DB for Data
getinfo.exe
5. Connect thru the n/w to server
The Internet
6. Send HTTP Request
10. Send HTTP Response which contains HTML generated dynamically
9. Compose HTML on stdout
Web Server
(Internet)
abcbooks.co.in
202.68.33.49
(Browser Parses and Renders HTML)
PC
1. User Submits form; URL http://abcbooks.co.in/cgi-bin/getinfo.exe?title=Web+Servers

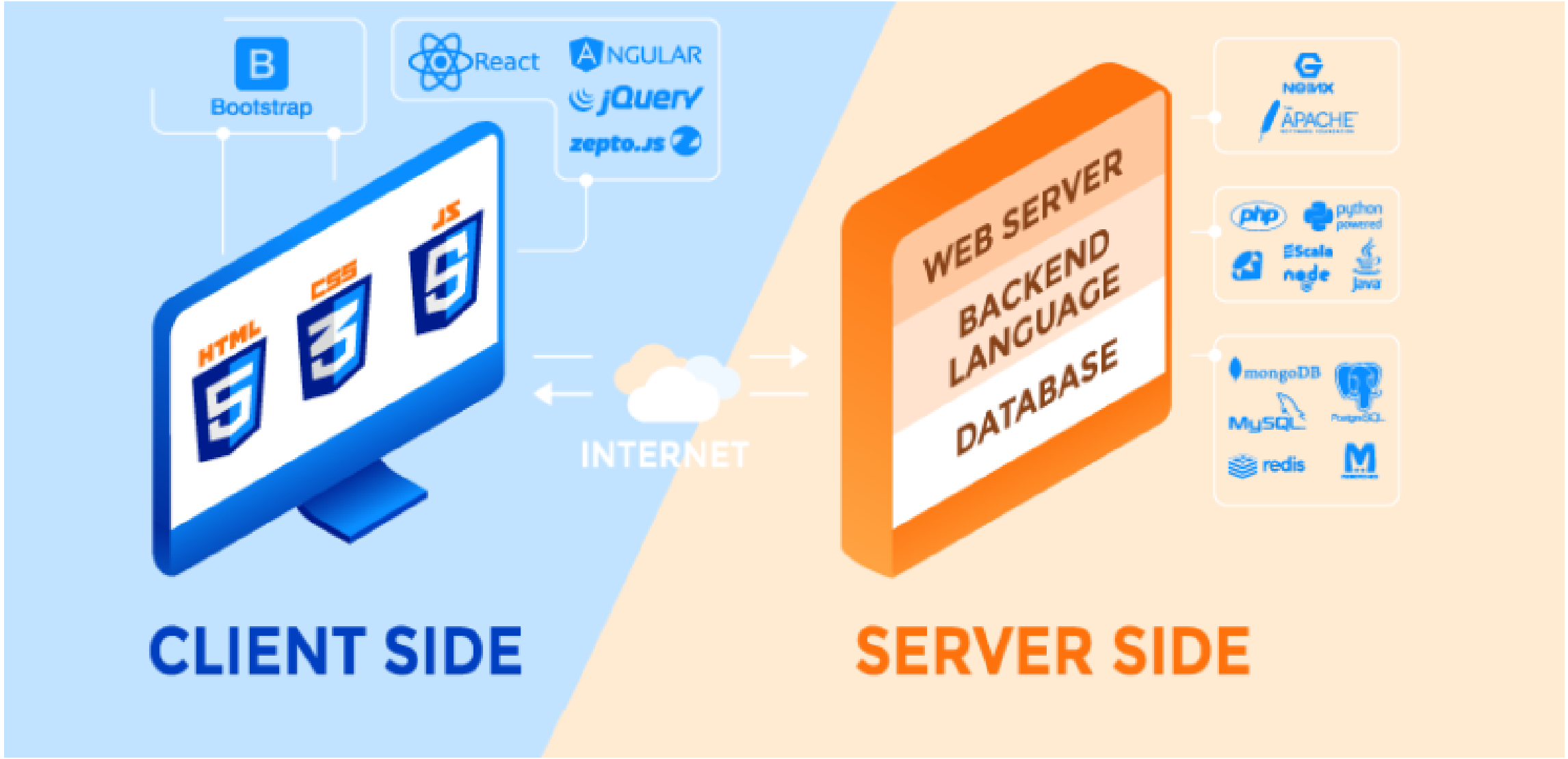
HTTP vs HTTPS

The HTTP protocol stands for Hypertext Transfer Protocol, whereas the HTTPS stands for Hypertext Transfer Protocol Secure.

The HTTP protocol is not secure protocol as it does not contain SSL (Secure Sockets Layer), which means that the data can be stolen when the data is transmitted from the client to the server.

The HTTP transmits the data over port number **80**, whereas the HTTPS transmits the data over **443** port number.

The HTTP protocol works on the application layer while the HTTPS protocol works on the transport layer.



PC: https://www.techwebspace.com/wp-content/uploads/2018/10/Server-Side-vs-Client-Side-Programming-795x385.png

World Wide Web Consortium (W3C)

The World Wide Web Consortium (W3C) is an international community where Member organizations, a full-time staff, and the public work together to develop Web standards.

Introduction to HTML

What is HTML?

HTML stands for Hyper Text Markup Language

HTML is the standard markup language for creating Web pages

HTML describes the structure of a Web page

HTML consists of a series of elements

HTML elements tell the browser how to display the content

Basic Terminologies

- Tags: A set of characters constituting a formatted command for a Web page.
  - <html> </html>
- Elements: An HTML element is an individual component of an HTML document or web page. It usually consists of a start tag and end tag, with the content.
  - <title> This is First Page </title>
- Attributes: The attributes are special words used inside the opening tag to control the element's behavior. Attributes provide additional information about HTML elements.
  - <title id="page"> This is First Page </title>

```
<html>

  <head>

    <title>Page title</title>

  </head>

  <body>

    <h1>This is a heading</h1>

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

    <p>This is another paragraph.</p>

  </body>

</html>
```

Basic HTML5 Web Page

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>My First HTML5 Page</title>
    </head>
    <body>
        <p>Hello World</p>
    </body>
</html>
```

DOCTYPE declaration

A DTD indicates the syntax that can be used for the various elements of a language such as HTML.

HTML 4.01 Strict
Does NOT INCLUDE presentational or deprecated elements (like font). Framesets are not allowed.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
   4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
```

HTML 4.01 Transitional
INCLUDES presentational and deprecated elements (like font). Framesets are not allowed.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
   Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
```

HTML 4.01 Frameset
This DTD is equal to HTML 4.01 Transitional, but allows the use of frameset content.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
   Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd"
   >
```

First Look at HTML5

Remember the DOCTYPE declaration from XHTML?

In HTML5, there is just one possible DOCTYPE declaration and it is simpler:

```
<!DOCTYPE html>
```

Doctypes for earlier versions of HTML were longer because the HTML language was SGML-based and therefore required a reference to a DTD.

The <head> Section

```
<head>
    <title>My First HTML5 Page</title>
</head>
```

The <body> Tag

The <body> tag defines the document's body.

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

```
<body>
    <h1>Hello world!</h1>
</body>
```

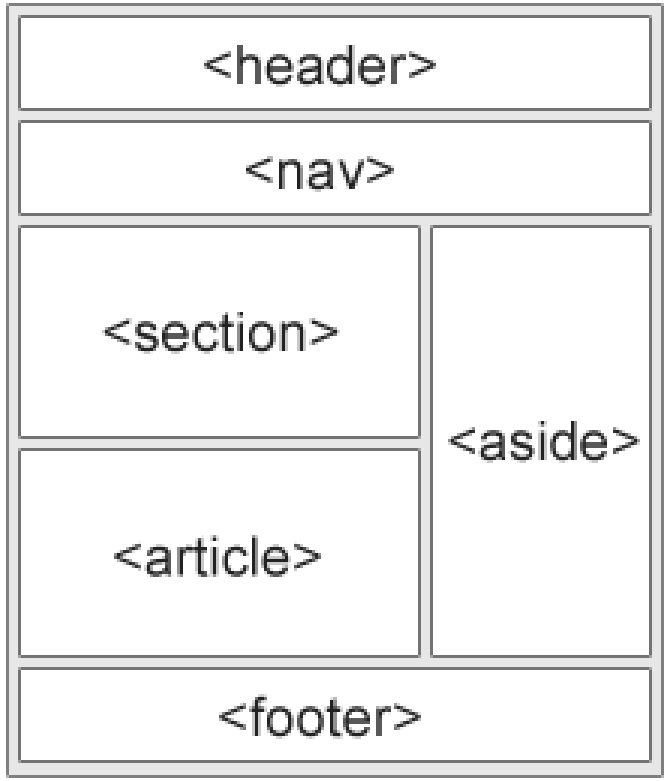
Semantic Elements

What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.

Examples of non-semantic elements: and - Tells nothing about its content.

Examples of semantic elements: <form>, <table>, and <article> - Clearly defines its content.



HTML5 Semantic Elements

<article>
<aside>
<audio>
<canvas>
<datalist>
<figure>

<figcaption>
<footer>
<header>
<hgroup>
<mark>
<nav>

<progress>
<section>
<source>
<svg>
<time>
<video>

Formatting Tags

1. <b> - Bold text
2. - Important text
3. <i> - Italic text
4. - Emphasized text
5. <mark> - Marked text
6. <small> - Small text
7. <del> - Deleted text
8. <ins> - Inserted text
9. <sub> - Subscript text
10. <sup> - Superscript text

https://www.w3schools.com/TAgs/default.asp

Hyper link

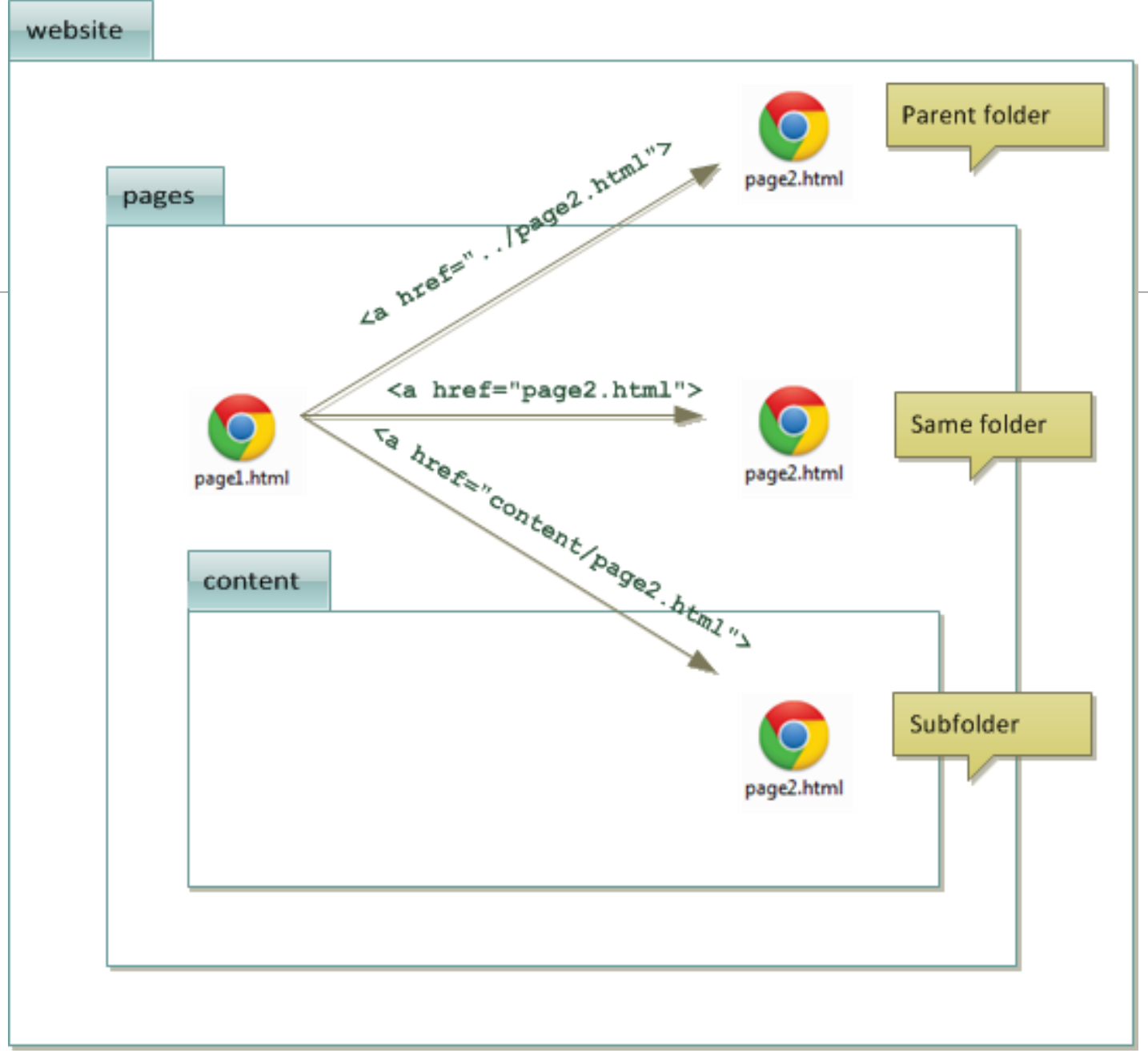
- **To an external Website:**
  - Visit Manipal

- **To an external page of the same website:**
  - Visit Manipal

- **Link Within a page:**
  1. Visit Manipal
  2. MIT

Attributes

download	filename	Specifies that the target will be downloaded when a user clicks on the hyperlink
href	URL	Specifies the URL of the page the link goes to
hreflang	language_code	Specifies the language of the linked document
media	media_query	Specifies what media/device the linked document is optimized for
target	Ex: _blank	Specifies where to open the linked document
type	media_type	Specifies the media type of the linked document


website
pages
page1.html
<a href="../page2.html">
page2.html
Parent folder
<a href="page2.html">
page2.html
Same folder
content
<a href="content/page2.html">
page2.html
Subfolder

Path Convension

- **Relative path**
  - Relative path is defined as the path related to the present working directly(pwd).
  - <img src="sample.jpg" alt="Smiley face" height="400" width="400">
  - <img src="Examples/sample.jpg" alt="Smiley face" height="400" width="400">
  - <img src="../sample.jpg" alt="Smiley face" height="400" width="400">

- **Absolute path**
  - An absolute path is defined as the specifying the location of a file or directory from the root directory
  - <img src="D:\MIT\Subjects for Teaching\II Sem\WP\Examples\sample.jpg" alt="Smiley face" height="400" width="400">

Inserting Images

```
<img src="sample.jpg" alt="Smiley face" height="400" width="400">
```

- The <img> tag has two required attributes:
  - src - Specifies the path to the image
  - alt - Specifies an alternate text for the image
- Other Attributes
  - Width and Height

HTML Lists

Unordered HTML List (UL)

```
<ul style="list-style-type:circle;">
                        <li>ABC</li>
                        <li>DEF</li>
                        <li>IJK</li>
                        <li>LMN</li>
</ul>
```

Ordered HTML List (OL)

```
<ol type="i">
                <li>ABC</li>
                <li>DEF</li>
                <li>IJK</li>
                <li>LMN</li>
</ol>
```

HTML Tables

Tags

The <table> tag defines an HTML table.

Each table row is defined with a <tr> tag. Each table header is defined with a <th> tag. Each table data/cell is defined with a <td> tag.

By default, the text in <th> elements are bold and centered.

By default, the text in <td> elements are regular and left-aligned.

Example

```
<table style="width:100%">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>ABC</td>
    <td>DEF</td>
    <td>50</td>
  </tr>
  <tr>
    <td>XYZ</td>
    <td>LMN</td>
    <td>94</td>
  </tr>
</table>
```

HTML Table Tags

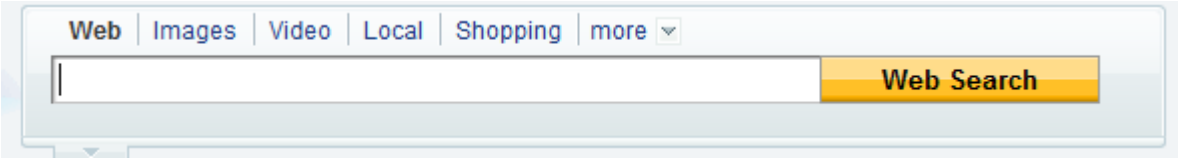
Tag	Description
<table>	Defines a table
<th>	Defines a header cell in a table
<tr>	Defines a row in a table
<td>	Defines a cell in a table
<caption>	Defines a table caption
<colgroup>	Specifies a group of one or more columns in a table for formatting
<col>	Specifies column properties for each column within a <colgroup> element
<thead>	Groups the header content in a table
<tbody>	Groups the body content in a table
<tfoot>	Groups the footer content in a table

Forms

Overview of Forms

Forms are used all over the Web to

- Accept information
- Provide interactivity



Types of forms:

- Search form, Order form, Newsletter sign-up form, Survey form, Add to Cart form, and so on…

Two Components of Using Forms

1. The web page form

   An HTML element that contains and organizes **form controls** such as text boxes, check boxes, and buttons that can accept information from website visitors.

2. The server-side processing

   Server-side processing works with the form data and sends e-mail, writes to a text file, updates a database, or performs some other type of processing on the server.

HTML Using Forms

<form> tag
- Contains the form elements on a web page
- Container tag

<input> tag
- Configures a variety of form elements including text boxes, radio buttons, check boxes, and buttons
- Stand alone tag

<textarea> tag
- Configures a scrolling text box
- Container tag

<select> tag
- Configures a select box (drop down list)
- Container tag

<option> tag
- Configures an option in the select box
- Container tag

The Form Element <form>

- Container Tag
- The form element attributes:
  - **action**
    - Specifies the server-side program or script that will process your form data
  - **method**
    - **get** – default value, form data passed in URL
    - **post** – more secure, form data passed in HTTP Entity Body
  - **name**
    - Identifies the form
  - **id**
    - Identifies the form

Sample Form HTML

```
<form>
    E-mail: <input type="text" name="email" id="email" >
    <input type="submit" value="Sign Me Up!">
    <input type="reset">
</form>
```

The Input Element Text Box <input>

- Common Attributes:
  - type=“text”
  - name
  - id
  - size
  - maxlength
  - value
  - required (HTML5)
  - placeholder (HTML5)

<input type="text" name="textfield" size=“value” value="with an initial value">

Text Input (type=“text”)

Example 1: A text field named "text1" that is 30 characters wide.

```
<input type="text" name="text1" size="30">
```

Example 2: A text field named "text2" that is 30 characters wide but will only accept 20 characters.

```
<input type="text" name="text2" size="30" maxlength="20">
```

Example 3: A text field named "text3" that is 40 characters wide with default value.

```
<input type="text" name="text3" size="40" value="We are not alone">
```

The Input Element Password Box <input>

- Accepts text information that needs to be hidden as it is entered
- Common Attributes:
  - type="password"
  - name
  - id
  - size
  - maxlength
  - value
  - required (HTML5)
  - placeholder (HTML5)

Password Input (type=“password”)

Example 4: A password field named "pass1" that is 30 characters wide

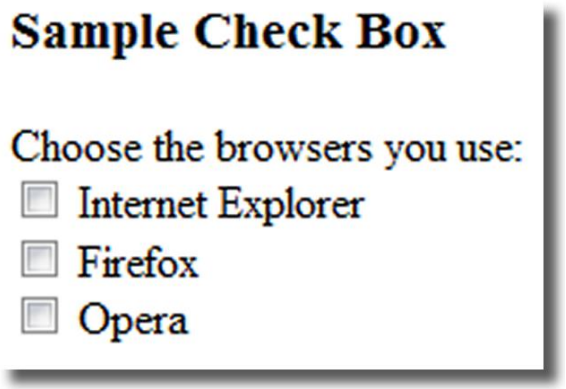
```
<input type="password" name="pass1" size="30">
```

Example 5: A password field named "pass2" that is 30 characters wide but will only accept 20 characters

```
<input type="password" name="pass2" size="30" maxlength="20">
```

The Input Element Check box <input>

- Allows the user to select one or more of a group of predetermined items
- Common Attributes:
  - type="checkbox"
  - name
  - id
  - checked
  - value



The Input Element Radio Button <input>

- Allows the user to select exactly one from a group of predetermined items
- Each radio button in a group is given the same name and a unique value
- Common Attributes:
  - type="radio"
  - name
  - id
  - checked
  - value

```
<input type="radio" name="radiobutton" value="myValue1">
male<br>
<input type="radio" name="radiobutton" value="myValue2" checked>
female
```

```
<input type="checkbox" name="checkbox” value="checkbox" checked>
```

The Textarea Element <textarea>

Configures a scrolling text box

Common Attributes:

- name
- id
- cols
- rows

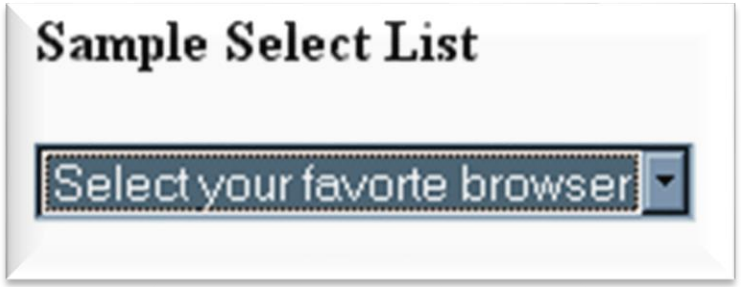
Text Input (type="textarea")

Example 6: A textarea field named "comments" that is 45 characters wide and 6 lines high

```
<textarea name="comments" rows="6" cols="45" wrap="virtual">
The first time I ever saw a web page, I thought.... (continue)
</textarea>
```

The Select Element <select>

- Configures a select list (along with <option> tags)
- Also known as: Select Box, Drop-Down List, Drop-Down Box, and Option Box.
- Common Attributes:
  - name
  - id
  - size
  - multiple

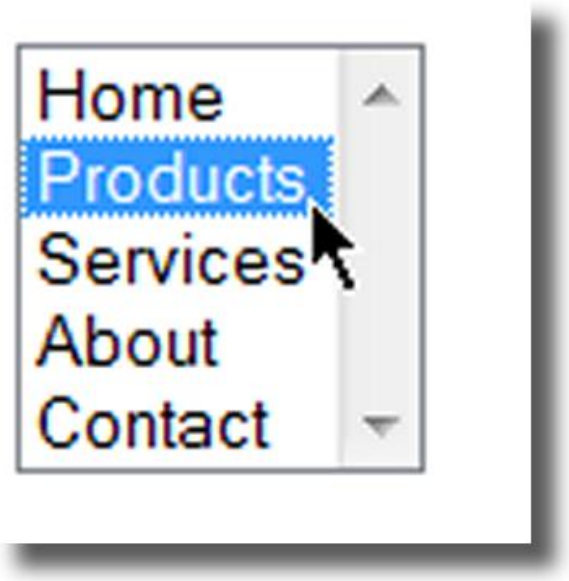


The Option Element <option>

Configures the options in a Select List

Attributes:

- value
- selected



```
<select name="select">
    <option value="red">red</option>
    <option value="green">green</option>
    <option value=“blue">blue</option>
</select>
```

The Input Element Submit Button <input>

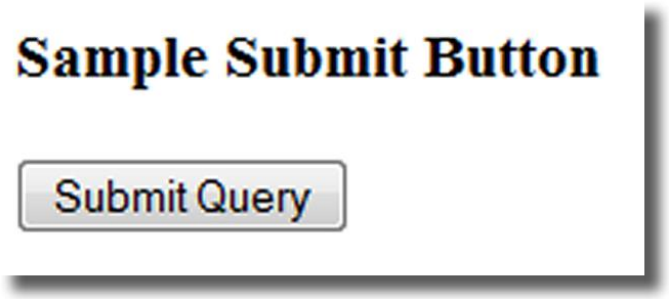
Submits the form information

When clicked:

- Triggers the **action** method on the <**form>** tag
- Sends the form data (the name=value pair for each form element) to the web server.

Attributes:

- type="submit"
- name
- id
- value



The Input Element Reset Button <input>

Resets the form fields to their initial values

Attributes:

- type="reset"
- name
- id
- value

A submit button:

```
<input type="submit" name="Submit" value="Submit">
```

A reset button:

```
<input type="reset" name="Submit2" value="Reset">
```

The Input Element Hidden Field <input>

This form control is *not* displayed on the Web page.

Hidden form fields

- Can be accessed by both client-side and server-side scripting

Common Attributes:

- type="hidden"
- name
- id
- value

The Label Element <label>

Associates a text label with a form control

Two Different Formats:

```
<label>Email: <input type="text" name="CustEmail"  id
="CustEmail"></label>
```

Or

```
<label for="email">Email: </label>
<input type="text" name="CustEmail" id= "email" />
```

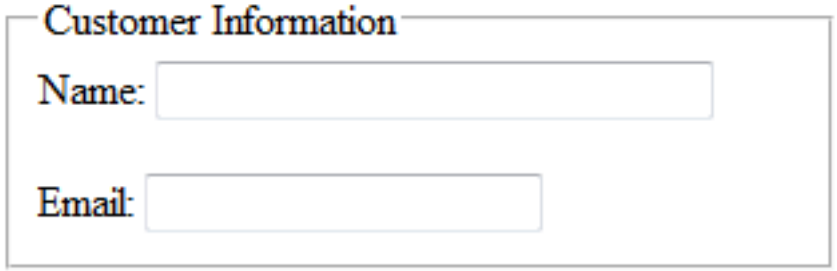
The Fieldset & Legend Elements

The Fieldset Element <fieldset>

- Container tag
- Creates a visual group of form controls on a web page

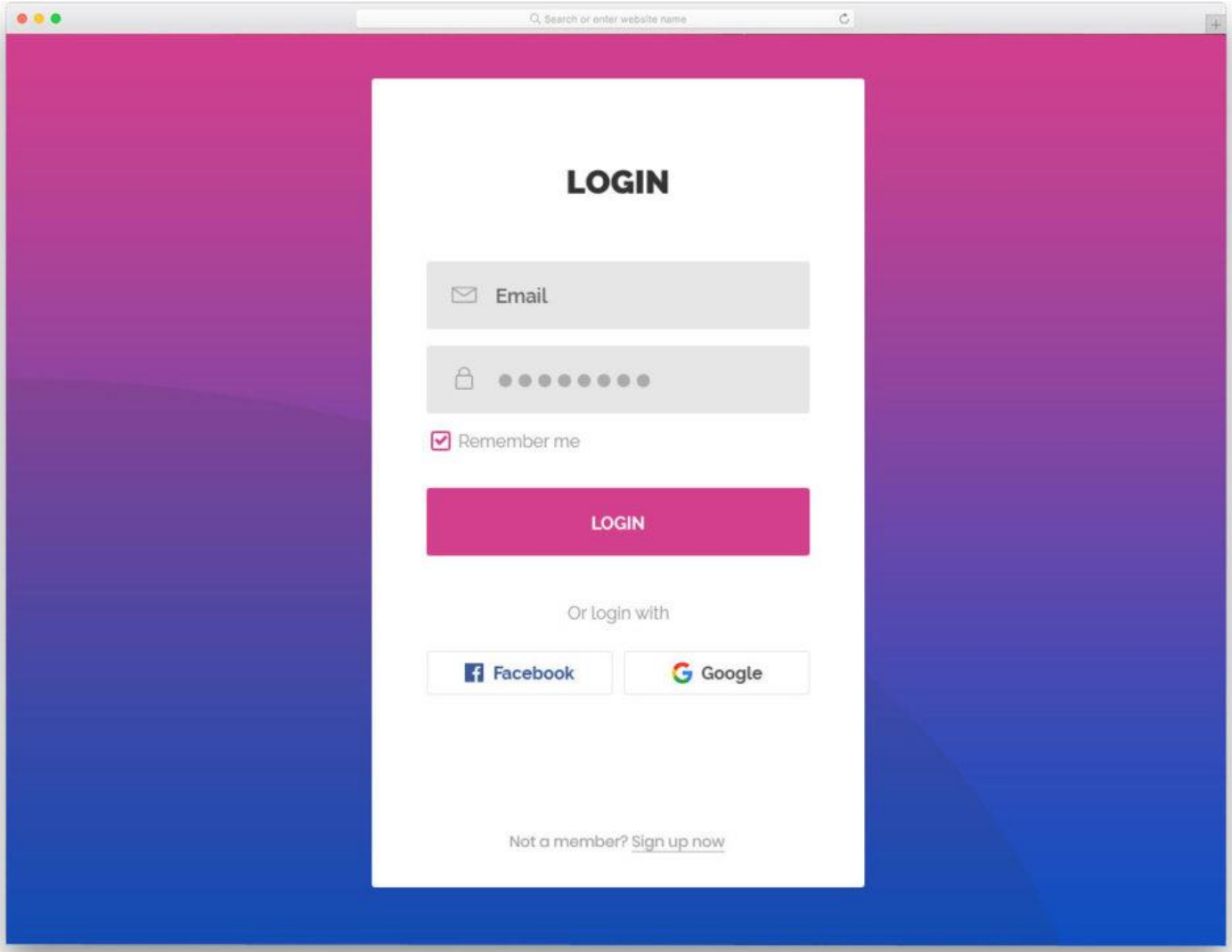
The Legend Element <legend>

- Container tag
- Creates a text label within the fieldset

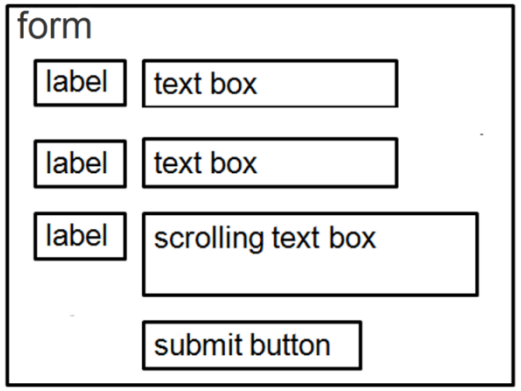


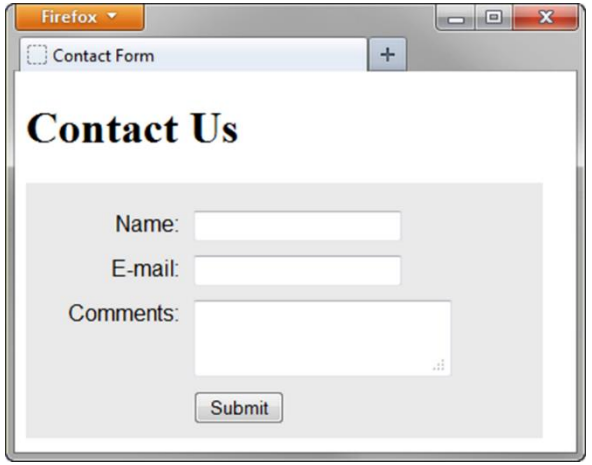
Cont..

```
<fieldset><legend>Customer Information</legend>
  <label>Name:
 <input type="text" name="CName"  id="CName" size="30"></label>       <br><br>
  <label>Email:
 <input type="text" name="CEmail"  id="CEmail"></label>
</fieldset>
```


Search or enter website name
LOGIN
Email
Remember me
LOGIN
Or login with
Facebook
Google
Not a member? Sign up now

Using CSS to Style a Form






```
form { background-color:#eaeaea; font-family: Arial, sans-serif;
        width: 350px; padding: 10px;}
label { float: left; clear: left;  display: block; width: 100px;
          text-align: right;  padding-right: 10px;  margin-top: 10px; }
input, textarea { margin-top: 10px;  display: block;}
#mySubmit { margin-left: 110px; }
```

HTML5: Email Text Box <input>

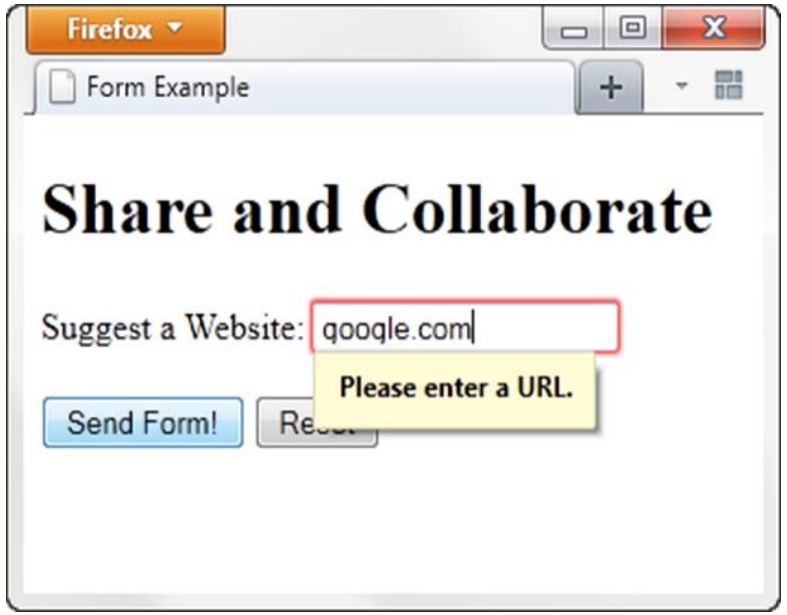
- Accepts text information in e-mail address format

- Common Attributes:
  - type="email"
  - name
  - id
  - size
  - maxlength
  - value
  - placeholder
  - required

HTML5: URL Text Box <input>

- Accepts text information in URL format

- Common Attributes:
  - type="url"
  - name
  - id
  - size
  - maxlength
  - value
  - placeholder
  - required



HTML5: Telephone Number Text Box <input>

- Accepts text information in telephone number format

- Common Attributes:
  - type="tel"
  - name
  - id
  - size
  - maxlength
  - value
  - placeholder
  - required

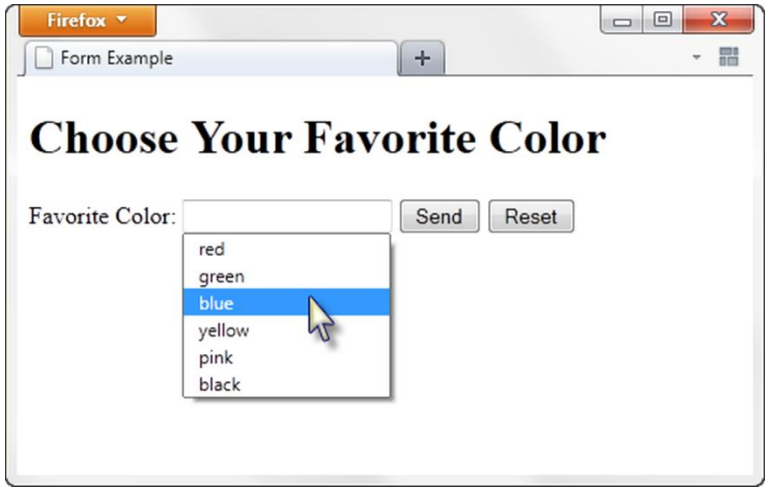
HTML5: Search Text Box <input>

- Accepts search terms
- Common Attributes:
  - type="search"
  - name
  - id
  - size
  - maxlength
  - value
  - placeholder
  - required

HTML5: Datalist Control

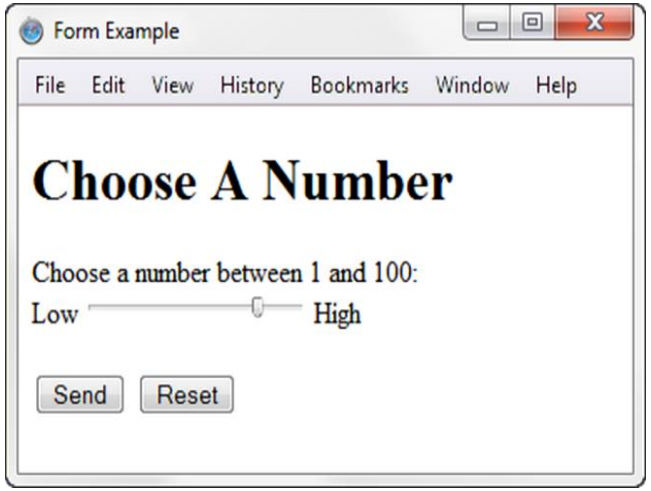
The <datalist> tag is used to provide an "autocomplete" feature on <input> elements.

```
<input type="text" name="color" id="color" list="colors" >
<datalist id="colors">
    <option value="red">
    <option value="green">
    <option value="blue">
    <option value="yellow">
    <option value="pink">
     <option value="black">
</datalist>
```



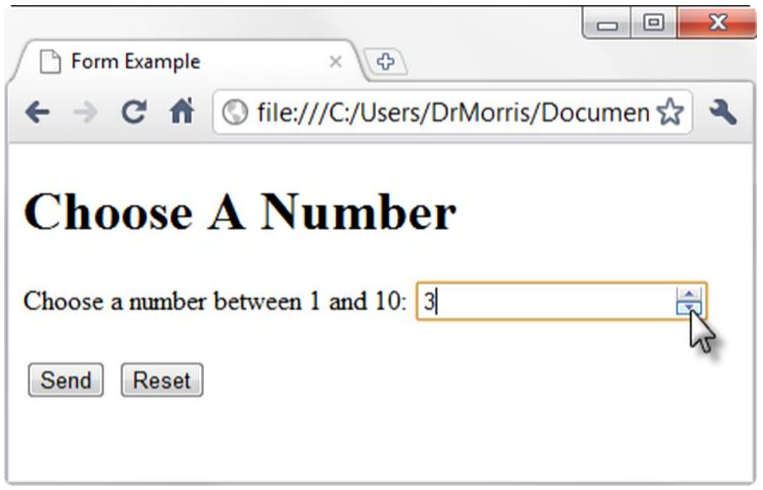
HTML5: Slider Control <input>

```
<label for="myChoice">
Choose a number between 1 and 100:</label><br>
Low <input type="range" name="myChoice" id="myChoice"> High
```



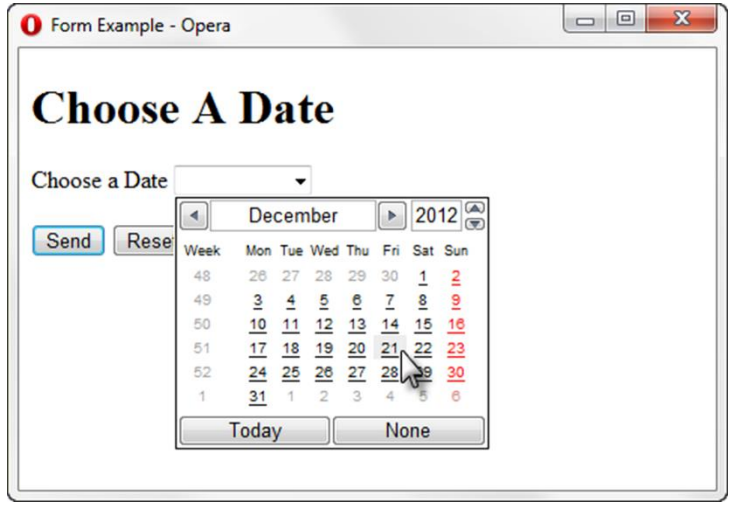
HTML5: Spinner Control <input>

```
<label for="myChoice">Choose a number between 1 and 10:</label>
<input type="number"  name="myChoice"  id="myChoice"
    min="1" max="10">
```



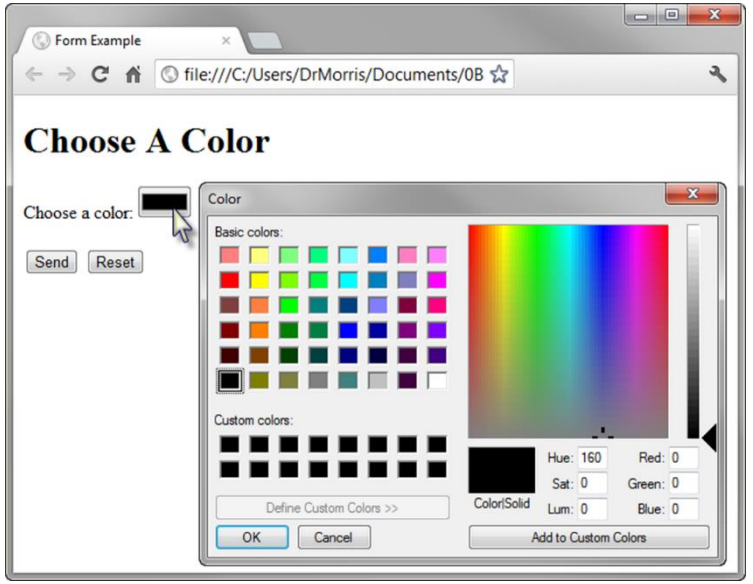
HTML5: Calendar Control <input>

```
<label for="myDate">Choose a Date</label>
<input type="date" name="myDate" id="myDate">
```



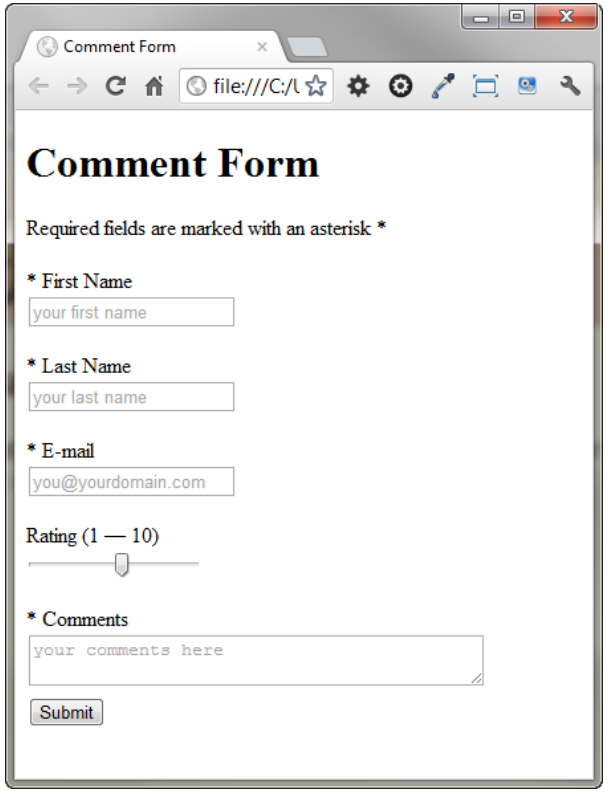
HTML5 Color-well Control

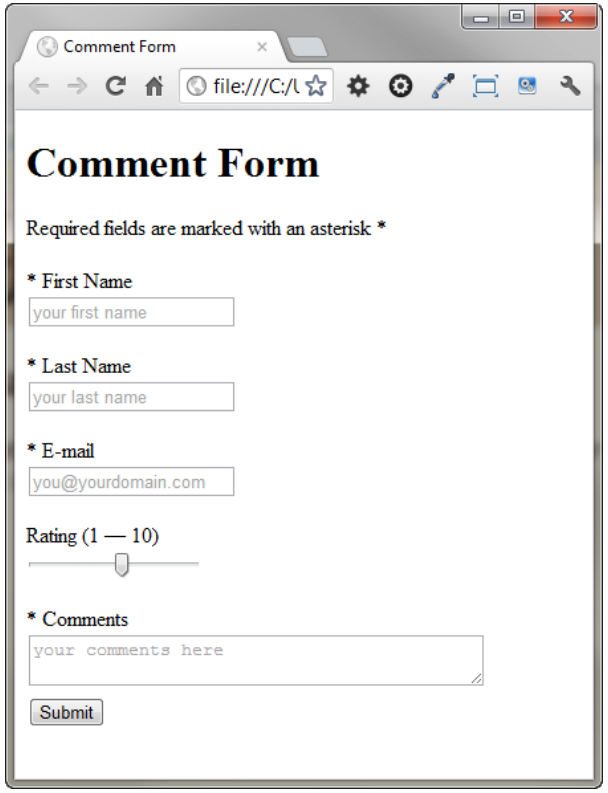
```
<label for="myColor">Choose a color:</label>
<input type="color" name="myColor" id="myColor">
```



Practice with an HTML5 Form

The form display and functioning varies with browser support.





Methods

- HTTP Methods: Tells the browser how to send form data to a web server.
- **GET:** In the GET method, after the submission of the form, the form values will be visible in the address bar of the new browser tab. It has a limited size of about 3000 characters. It is only useful for non-secure data not for sensitive information.
- **POST:** In the post method, after the submission of the form, the form values will not be visible in the address bar of the new browser tab as it was visible in the GET method. It appends form data inside the body of the HTTP request. It has no size limitation. This method does not support bookmark the result.

Cont..

- **The GET Method:** GET is used to request data from a specified resource.
  - GET requests can be cached
  - GET requests remain in the browser history
  - GET requests can be bookmarked
  - GET requests should never be used when dealing with sensitive data
  - GET requests have length restrictions
  - GET requests is only used to request data (not modify)

Cont..

The POST Method: POST is used to send data to a server to create/update a resource.

POST requests are never cached

POST requests do not remain in the browser history

POST requests cannot be bookmarked

POST requests have no restrictions on data length

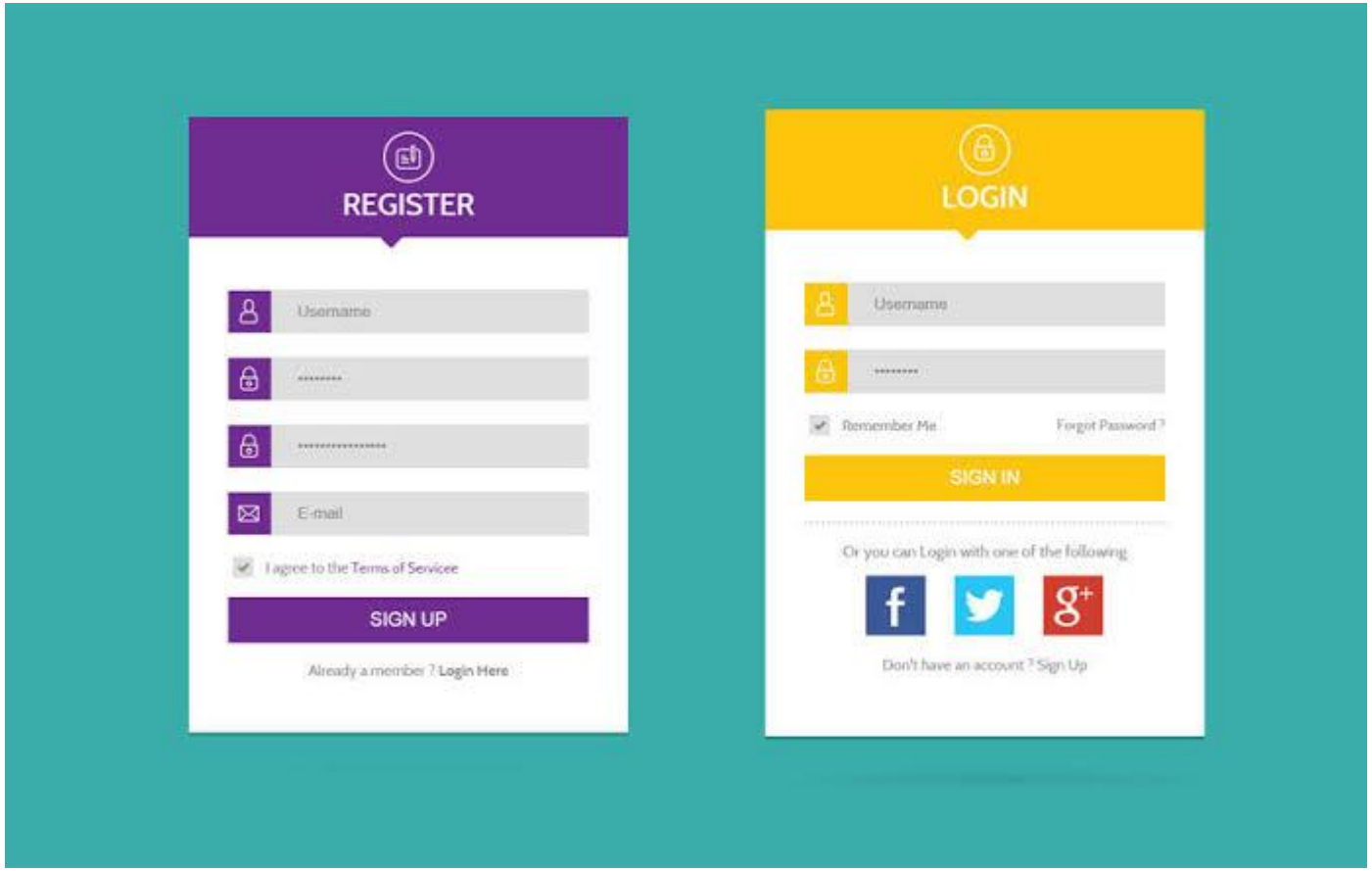
Efficient Web Forms Designing

ARTICLE CURTESY:

HTTPS://UXPLANET.ORG/DESIGNING-MORE-EFFICIENT-FORMS-STRUCTURE-INPUTS-LABELS-AND-ACTIONS-E3A47007114F

Web Forms

- Someone who uses your app or website has a particular goal.
- Often, the one thing standing between the user and their goal is a form.
- Forms remain one of the most important types of interactions for users on the web and in apps.



The Components Of Forms

The typical form has following five components:

- **Structure**. It includes ordering for fields, appearance on the page and logical connections between multiple fields.
- **Input fields**. They include text fields, password fields, check boxes, radio buttons, sliders and any other field designed for user input.
- **Field labels.** They tell users what the corresponding input fields mean.
- **Action buttons.** When user presses the button, the action is performed (such as submitting the data).
- **Feedback**. User understands the result of the input by a feedback. Most apps or sites use messages as a form of feedback. Messages notify the user about result, they can be positive (indicating that the form was submitted successfully) or negative (“The number you’ve provided is incorrect”).

Cont..

Forms could also have following components:

- **Assistance**. Any help that explains how to fill out the form.
- **Validation.** Automatic check that ensures that user's data is valid.

Form Structure

Only Ask What's Required

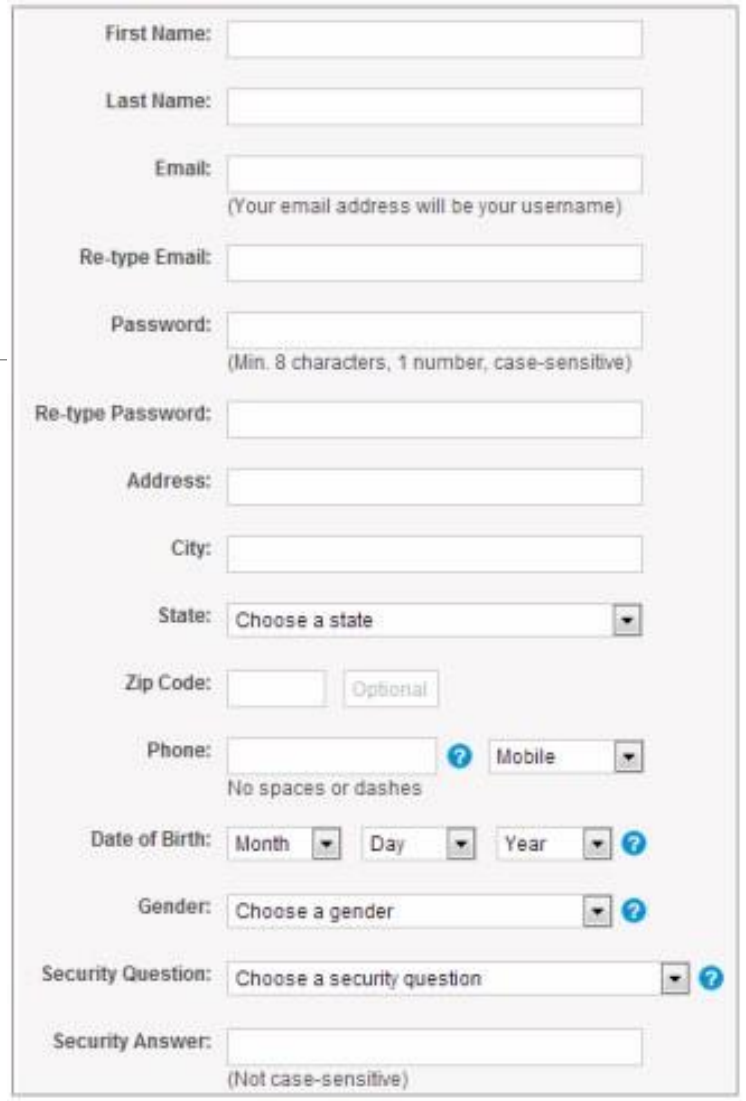
Make sure you only ask what you really need. Every extra field you add to a form will affect its conversion rate. That's why you should always question why and how the information you request from your users is being used.

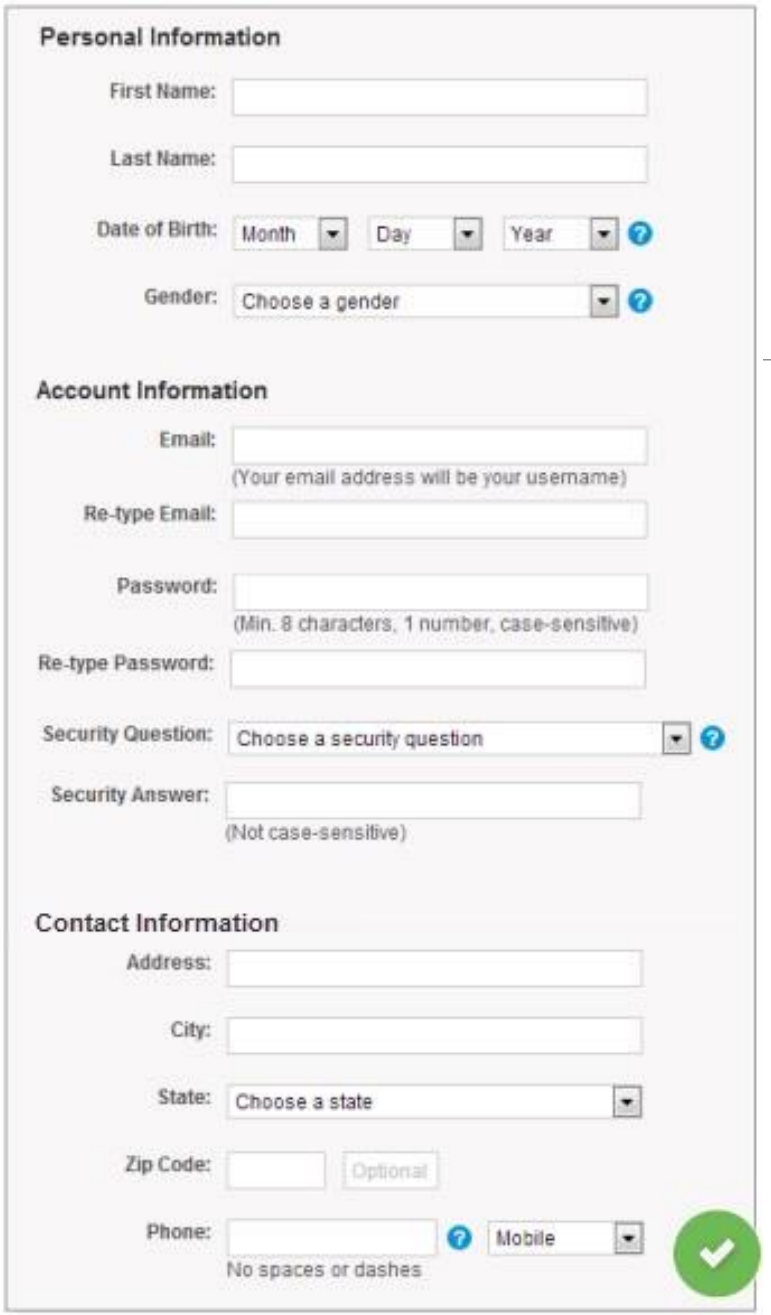
Order the Form Logically

Details should be asked logically from a user's perspective, not the application or database logic. Typically, it's unusual to ask for someone's address before their name.

Group Related Information

You should group related information in logical blocks or sets. The flow from one set of questions to the next will better resemble a conversation. Grouping related fields together also helps users make sense of the information that they must fill in. Below is an example for *Contact Information*.

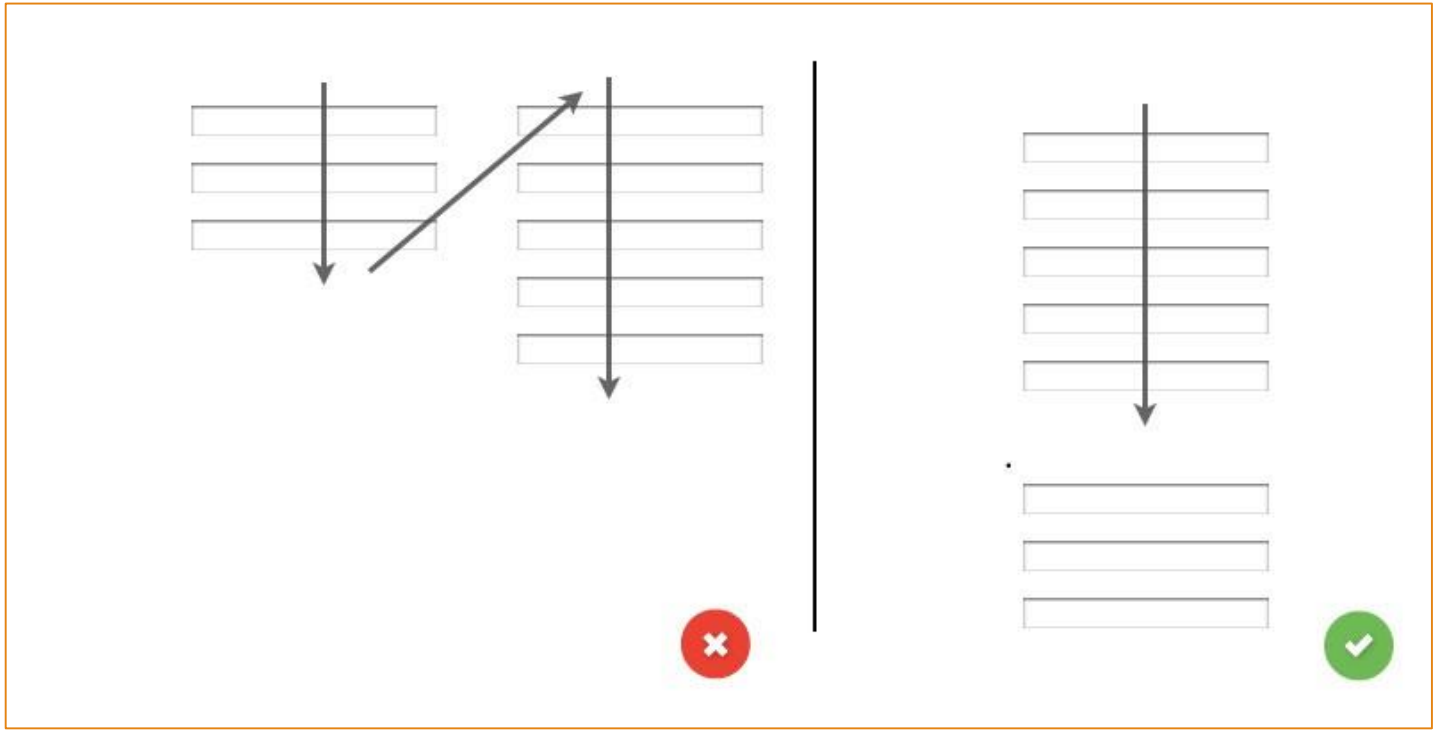

First Name:
Last Name:
Email:
(Your email address will be your username)
Re-type Email:
Password:
(Min. 8 characters, 1 number, case-sensitive)
Re-type Password:
Address:
City:
State:
Choose a state
Zip Code:
Optional
Phone:
Mobile
No spaces or dashes
Date of Birth:
Month
Day
Year
Gender:
Choose a gender
Security Question:
Choose a security question
Security Answer:
(Not case-sensitive)


Personal Information
First Name:
Last Name:
Date of Birth:
Month
Day
Year
Gender:
Choose a gender
Account Information
Email:
(Your email address will be your username)
Re-type Email:
Password:
(Min. 8 characters, 1 number, case-sensitive)
Re-type Password:
Security Question:
Choose a security question
Security Answer:
(Not case-sensitive)
Contact Information
Address:
City:
State:
Choose a state
Zip Code:
Optional
Phone:
Mobile
No spaces or dashes

Cont..

One Column vs. Multiple Columns

Forms should never consist of more than one column. One of the problems with form fields in multiple columns is *that your users are likely to interpret the fields inconsistently*.



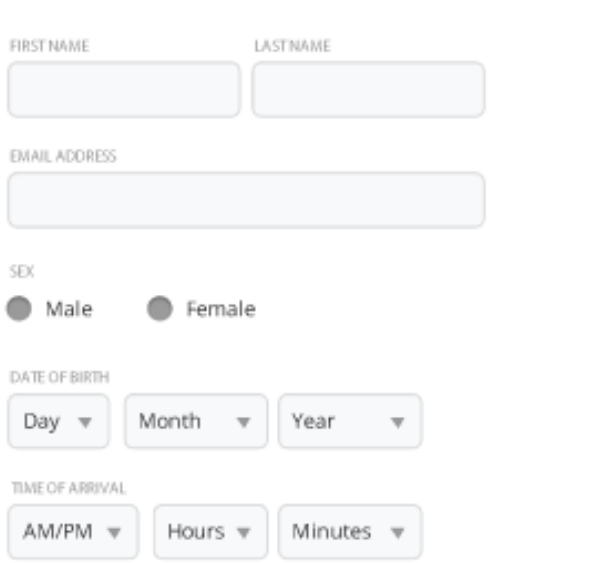
Cont..

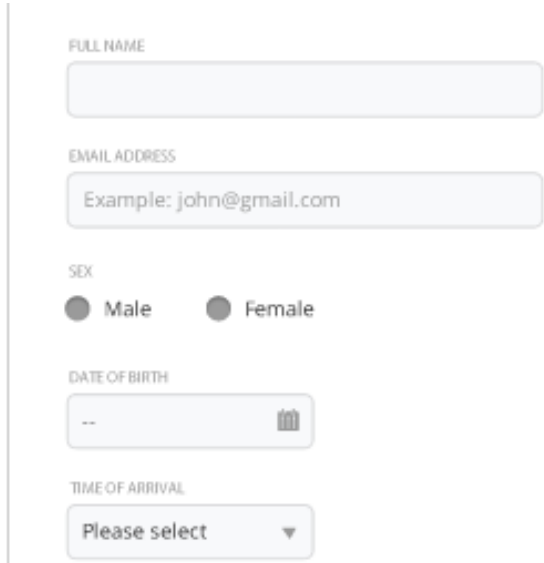
Input Fields

Input fields are what allow your users to fill in your form. Depending on what information you ask, there are various types of fields — text fields, password fields, dropdowns, check boxes, radio buttons, date pickers and others.

Number of Fields

Try to minimize the number of fields as much as possible. This makes your form less loaded, especially when you request a lot of information from your users. However don't over do it, no one likes a three field form that turns into a 30 field interrogation.





Cont..

Mandatory vs Optional

Try to avoid optional fields in forms. But if you use them, you should at least clearly distinguish which input fields cannot be left blank by the user. The convention is to use an asterisk (*) or ‘optional’ (which is a preferable for long forms with multiple required fields).

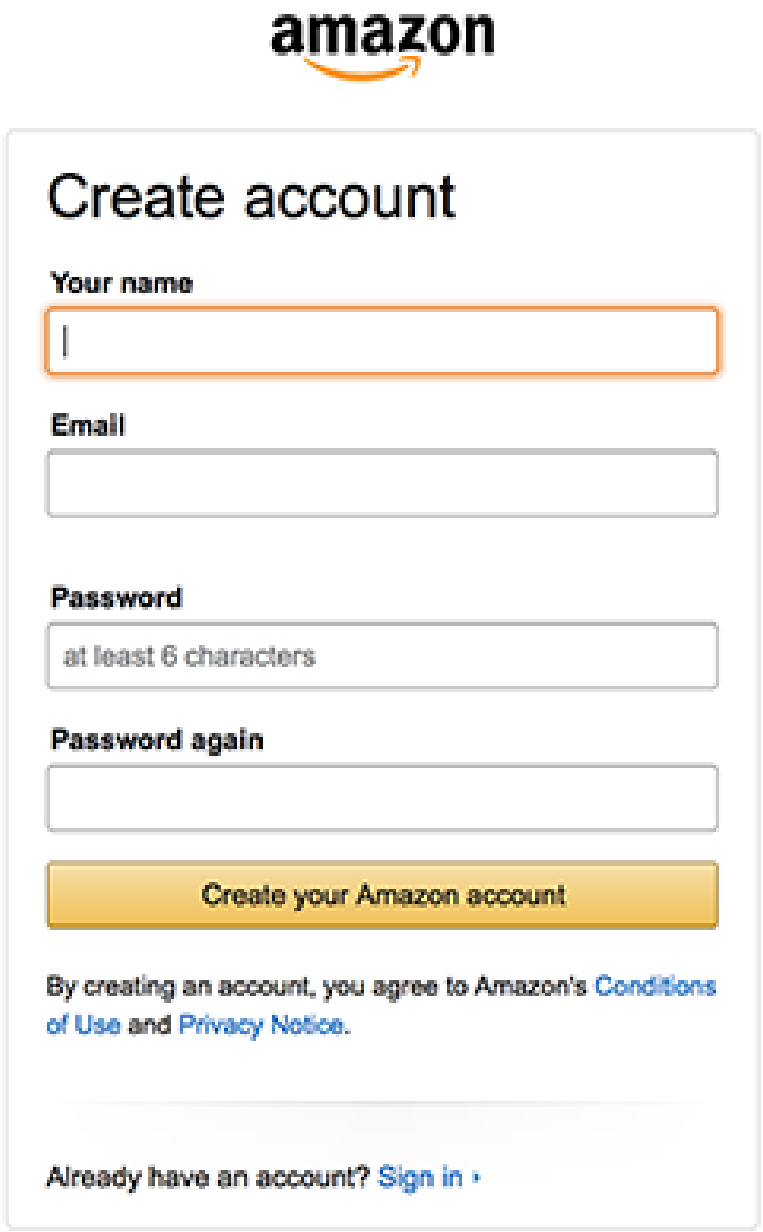
Cont..

Desktop-only: Make Form Keyboard-friendly

Users should be able to trigger and edit every field using only the keyboard. Power users, who tend to use keyboards heavily, should be able to easily **tab** through the fields and make necessary edits, all without lifting their fingers off the keyboard.

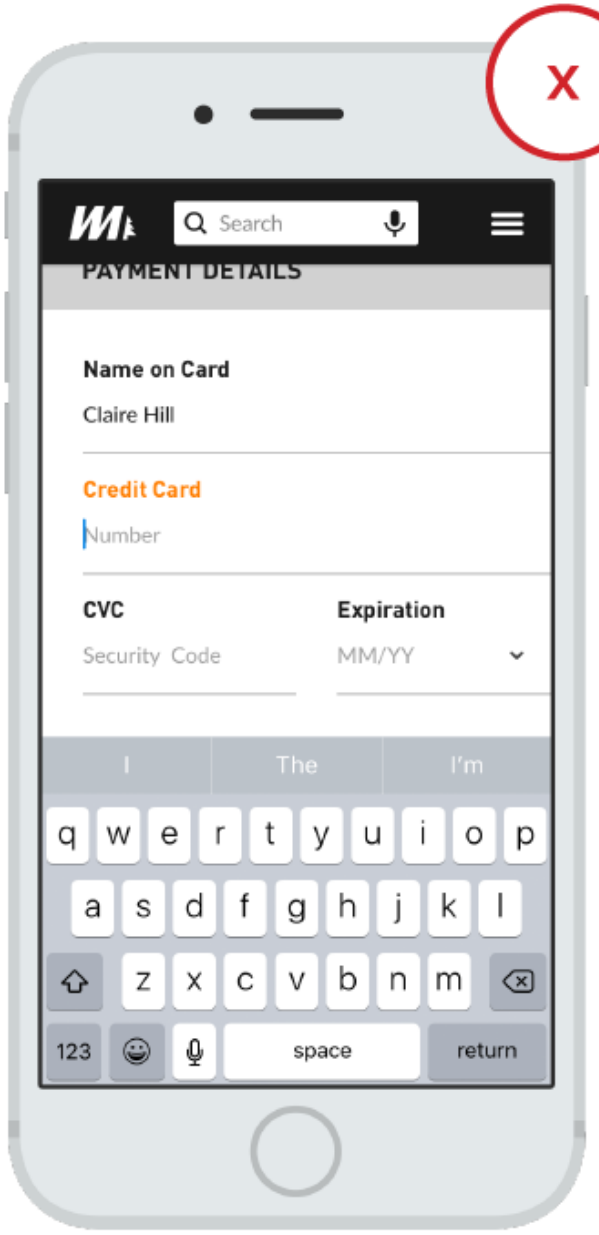
Desktop-only: Autofocus for Input Field

Autofocusing a field gives the users an indication and a starting point to quickly begin to fill out the form. You should provide a clear visual 'notification' that the focus has moved there — change color, fade in a box, flash in an arrow, whatever. Amazon registration form has both autofocus and visual notification for the user.

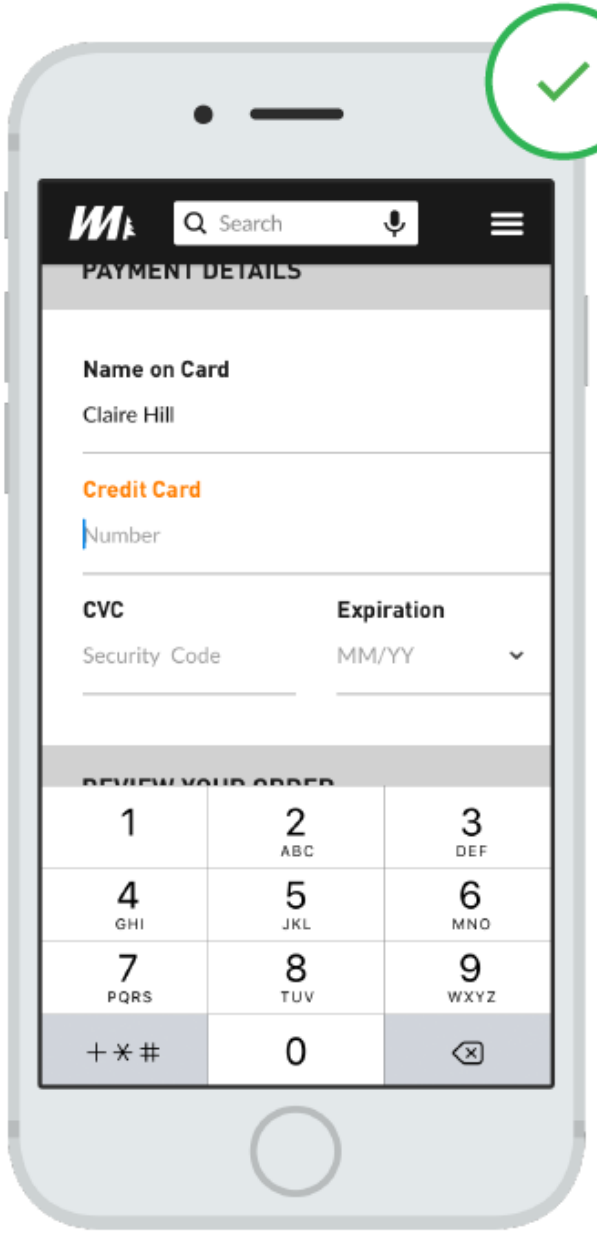


Mobile-only: Match the Keyboard With the Required Text Inputs

App users appreciate apps that provide an appropriate keyboard for text entry. Ensure that this is implemented consistently throughout the app rather than only for certain tasks but not others.



X The user is required to tap the number key in the keyboard to enable number entry.



✓ An appropriate numeric keyboard is automatically provided for fields that require numeric entry.

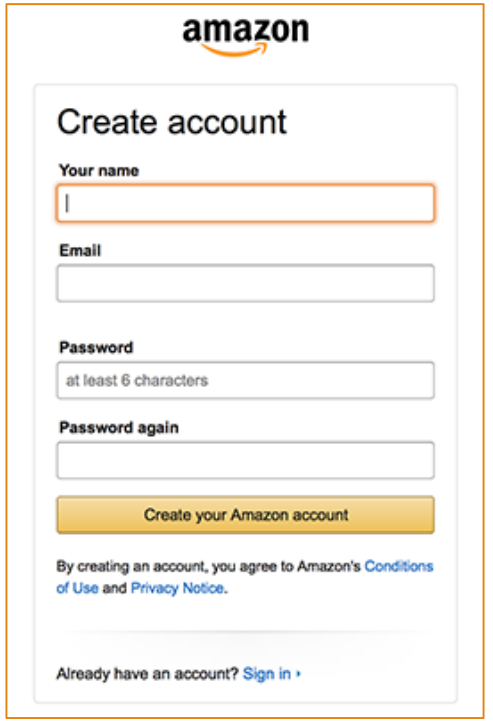
Cont..

Labels

Clear label text is one of the primary ways to make UIs more accessible. Labels tell the user the purpose of the field, maintain their usefulness when focus is placed inside of the field and should remain even after completing the field.

Number of Words

Labels are not help texts. You should use succinct, short and descriptive labels (a word or two) so users can quickly scan your form. Previous version of the Amazon registration form contained a lot of extra words which resulted in slow completion rates.

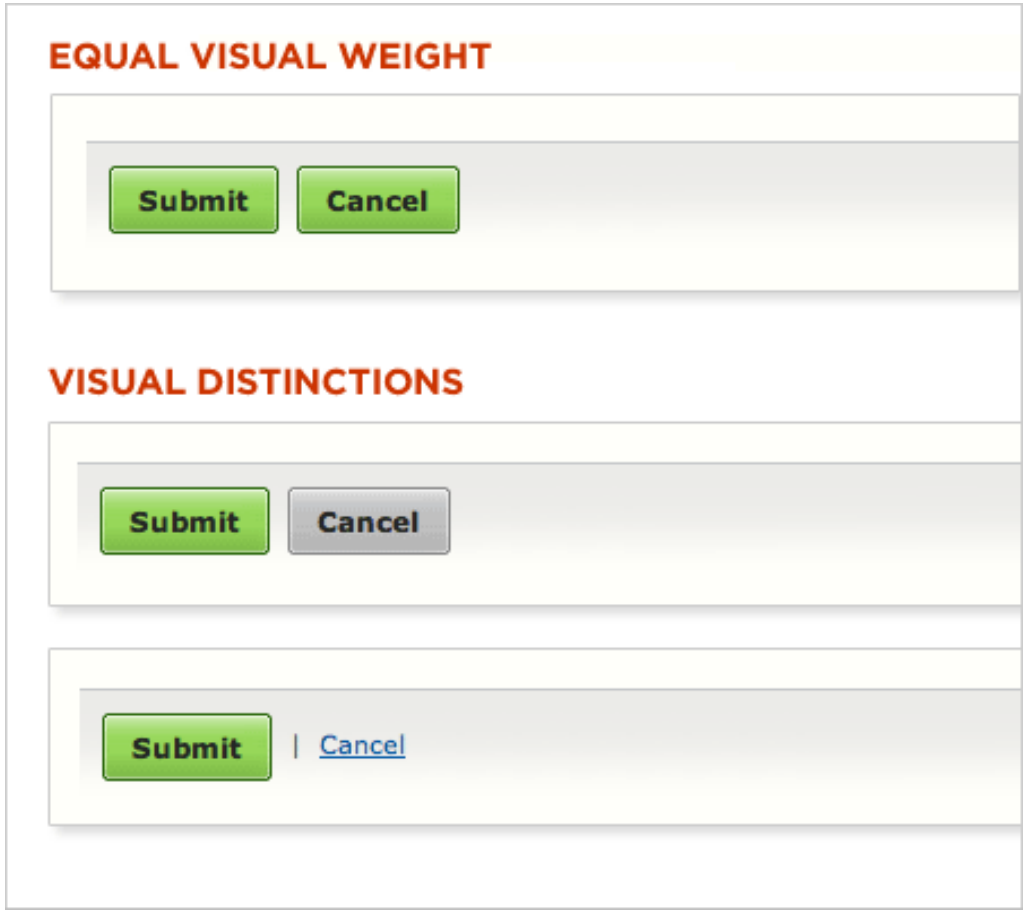


Action Buttons

When clicked, these buttons trigger an action such as submitting the form.

Primary vs Secondary Actions

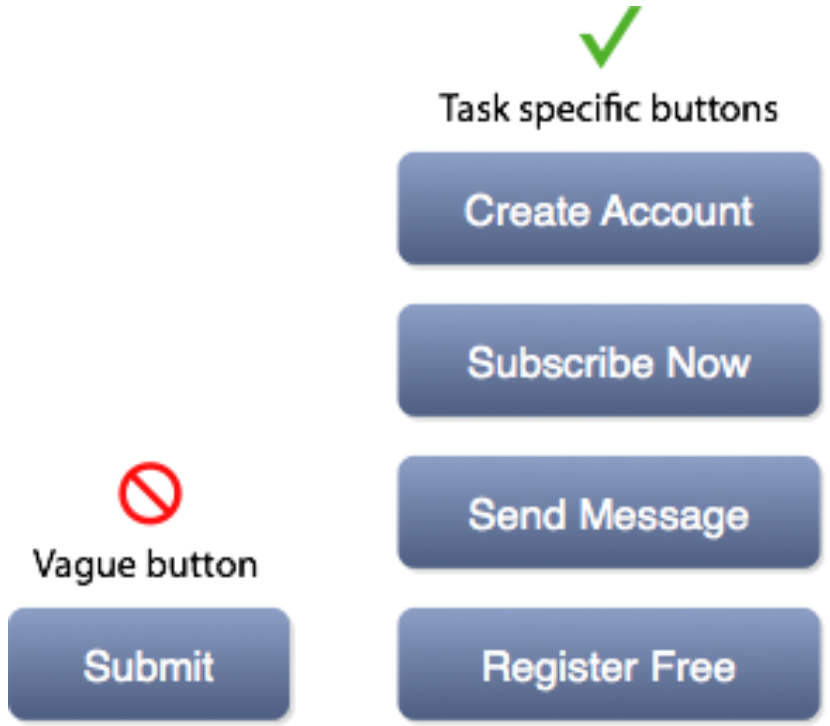
Lack of visual distinction between primary and secondary actions can easily lead to failure. Reducing the visual prominence of secondary actions minimizes the risk for potential errors and further directs people toward a successful outcome.



Cont..

Naming Conventions

Avoid generic words such as “Submit” for actions, because they give the impression that the form itself is generic. Instead state what actions the buttons do when clicked, such as ‘Create my FREE account’ or ‘Send me weekly offers’.



Cont..

Multiple Action Buttons

Avoid multiple action buttons since it might distract users from the goal (submitting the form).

Don’t use ‘Reset’ Button

Don’t use a ‘reset’ button. The web would be a happier place if virtually all *Reset* buttons were removed. This button almost never helps users, but often hurts them.

Visual Appearance

Make sure action buttons look like buttons. Style them in such a way that they should indicate that it is possible to click.

HTML form accessibility guidelines

https://www.w3.org/WAI/tutorials/forms/

HTML Comments

<!-- Write your comments here -->

Example

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

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

<!-- Remember to add more information here -->
```

HTML Block and Inline Elements

Every HTML element has a default display value, depending on what type of element it is.

There are two display values:

block

Inline

Block-level Elements

A block-level element always starts on a new line.

A block-level element always takes up the full width available (stretches out to the left and right as far as it can).

A block level element has a top and a bottom margin, whereas an inline element does not.

<address>	<article>	<aside>	<blockquote>	<canvas>	<dd>	
<dl>	<dt>	<fieldset>	<figcaption>	<figure>	<footer>	<form>
<h1>-<h6>	<header>	<hr>		<main>	<nav>	<noscript>
	<p>	<pre>	<section>	<table>	<tfoot>	
<video>						

Inline Elements

An inline element does not start on a new line.

An inline element only takes up as much width as necessary.

<a>	<abbr>	<acronym>	<b>	<bdo>	<big>	
<button>	<cite>	<code>	<dfn>		<i>	<img>
<input>	<kbd>	<label>	<map>	<object>	<output>	<q>
<samp>	<script>	<select>	<small>			<sub>
<sup>	<textarea>	<time>	<tt>	<var>		

HTML Media

HTML Video

- The HTML <video> element is used to show a video on a web page.

HTML Audio

- The HTML <audio> element is used to play an audio file on a web page.

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

```
<video controls src="sample_ocean.mp4" width="320">
        Your browser does not support the HTML5 Video element.
        </video>

        <video controls="controls">
            <source src="sample_ocean.mp4" type="video/mp4">
            <source src="sample_ocean.avi" type="video/avi">
            <source src="sample_ocean.3gp" type="video/3gp">

            Your browser does not support the HTML5 Video element.
        </video>
```