



# LESSON 15 - FORMS & INPUTS

# AGENDA

- Learning Objectives
- Review - Responsive Layouts
- HTML Forms
- Form Attributes
- Input Elements

# **LEARNING OBJECTIVES:**

# **AFTER TODAY, YOU SHOULD BE ABLE TO...**

- Differentiate between the different types of inputs and why/where we would use each.
- Explain how to group elements by name.
- Apply the method, action, and enctype attributes.

# **REVIEW RESPONSE**

# WHAT DOES "RESPONSIVE" MEAN?

**Responsive Web Design** is about using HTML and CSS to resize, hide, shrink, enlarge, or move the content to make it look good on any screen.

This is more about the design than it is about any coding (though there is a bit of this involved)

Responsive websites came about due to the increase in web-surfing on mobile devices...

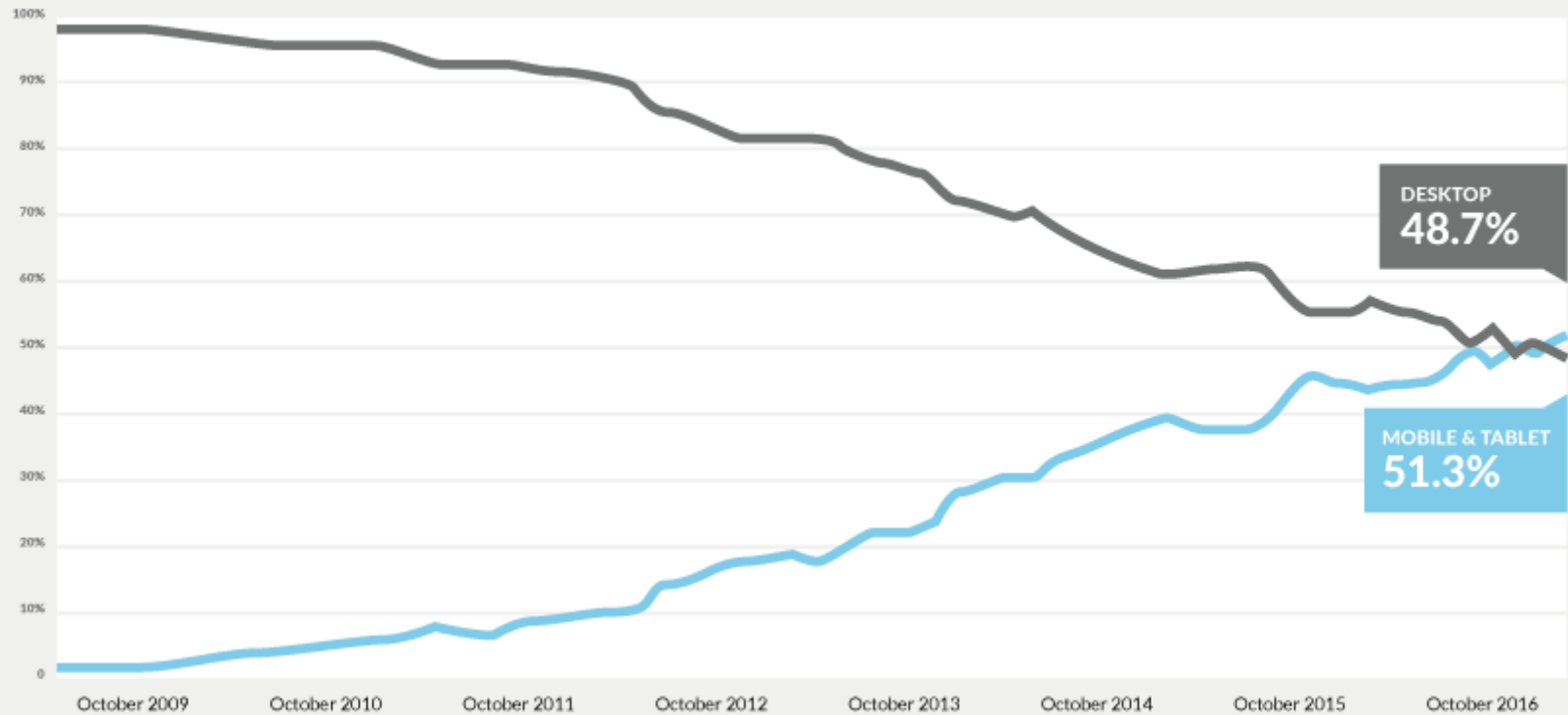


**VS.**





## Internet usage worldwide: October 2009 - October 2016



STATCOUNTER GLOBAL STATS

● DESKTOP

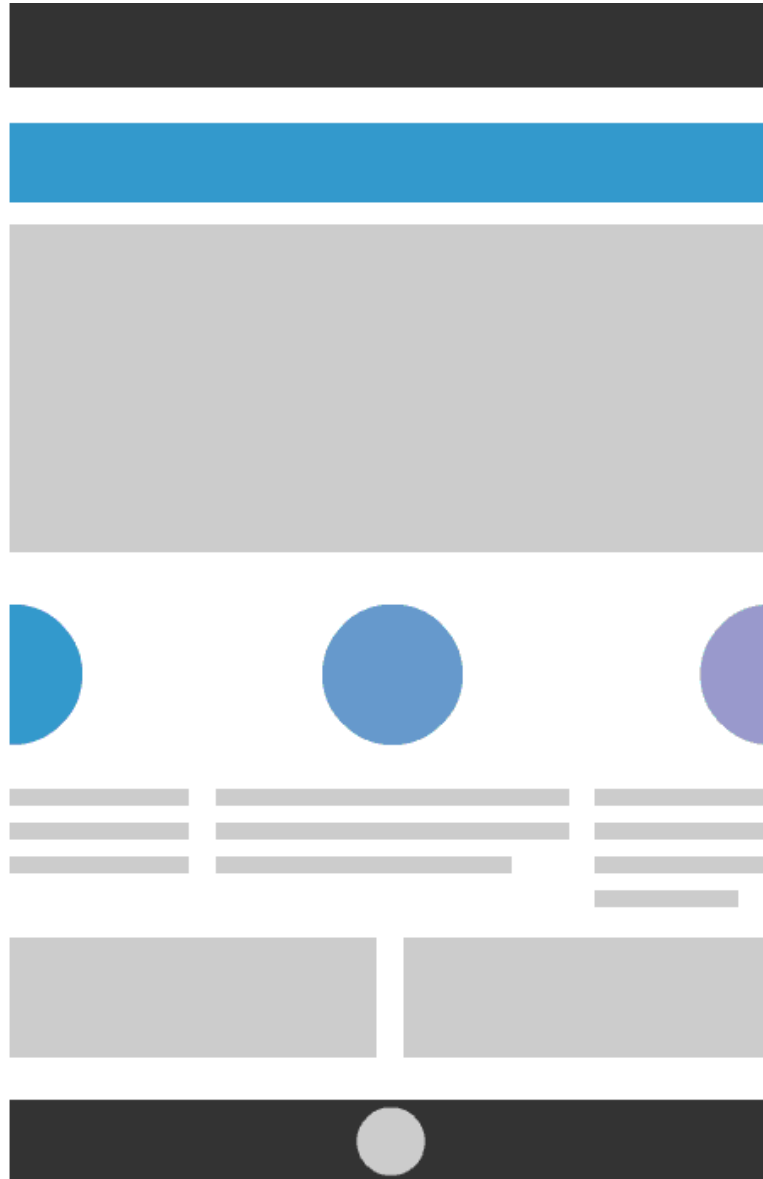
● MOBILE AND TABLET

TUNE

# **FIXED VS FLUID VS RESPONSIVE**

So, let's explore the different types of web layouts...

# FIXED LAYOUT



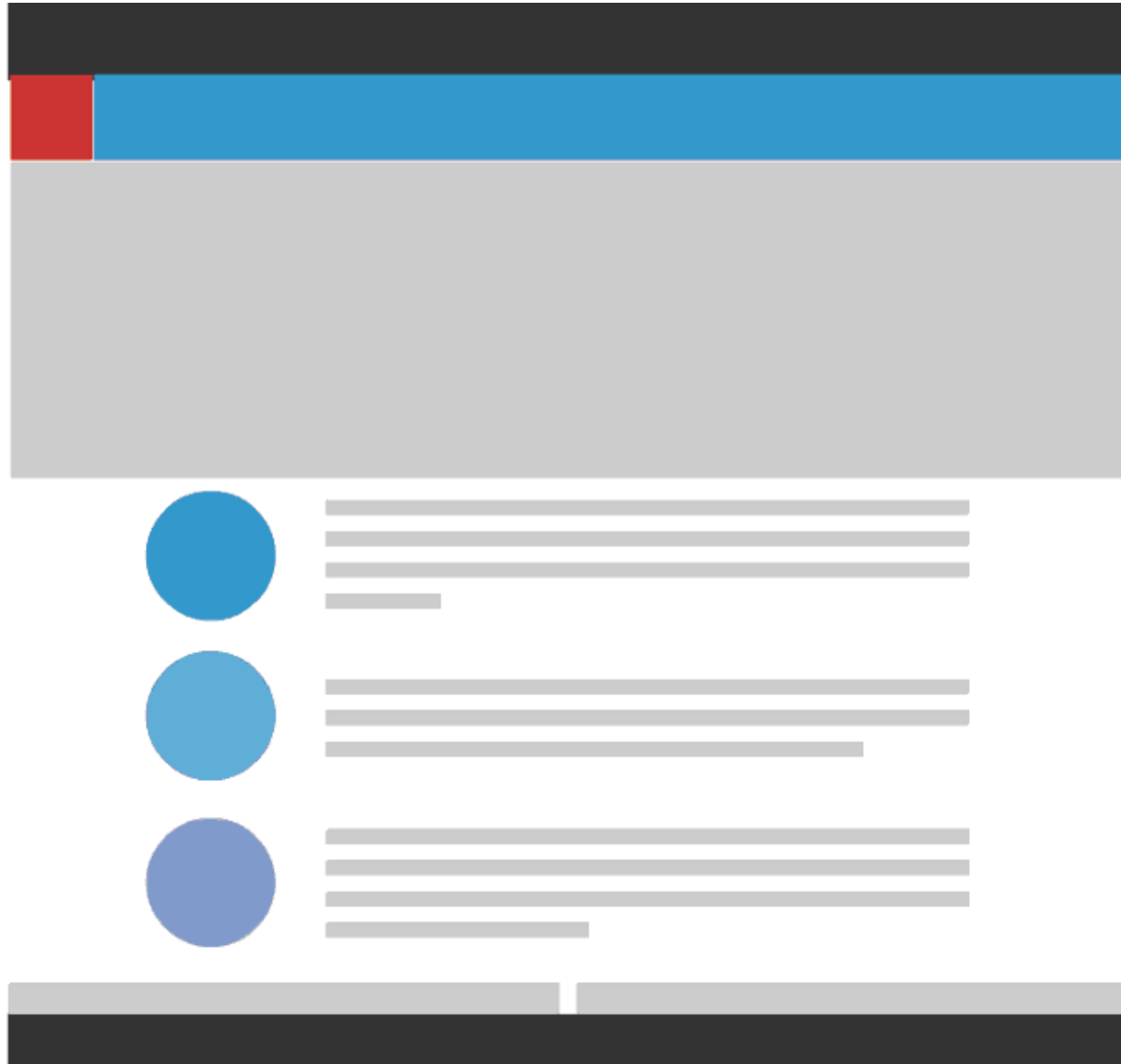


# FLUID LAYOUT





# RESPONSIVE LAYOUT







# **MEDIA QUERIES**

First, make sure the following meta tag is in the  
<head> section of your html file:

```
<meta name="viewport" content="width=device-width, initial-sca
```

Then, use `@media` queries and breakpoints to determine at what size the page layout should shift to better suit the device...

# COMMON MEDIA QUERIES BREAKPOINTS:

```
/*=====
Mobile First Method
=====*/

... default CSS styles ...

/* Extra Small Devices, Phones */
@media only screen and (min-width : 480px) {

}

/* Small Devices, Tablets */
@media only screen and (min-width : 768px) {

}
```

# EXAMPLE CSS W/ MEDIA QUERIES

```
/* stack boxes */
.box{
    float: none;
}

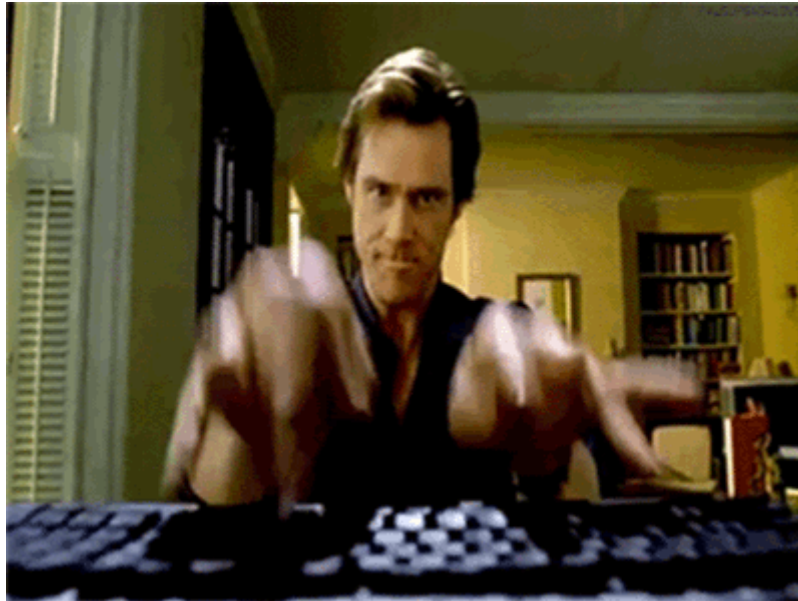
@media screen and (min-width:768px){
    /* insert responsive css here ex: float boxes in columns */
    .box{
        float: left;
    }
}
```



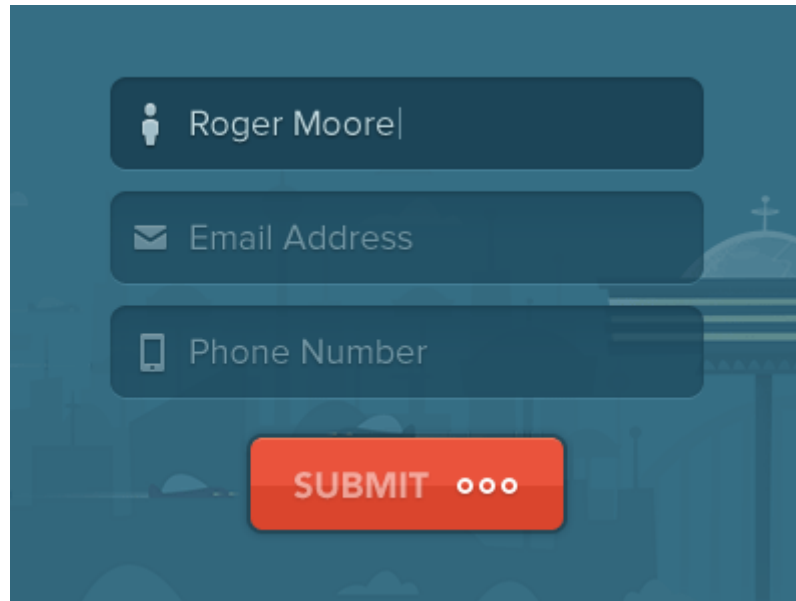
# PULSE CHECK - RESPONSIVE CODEPEN

Basic Media Query Example

# HTML FORMS



In HTML, we use **forms** to collect data from users.



A stylized illustration of a registration form on a dark blue background with a city skyline. The form consists of three input fields stacked vertically, each with a light blue border and a small icon on the left. The first field contains the text 'Roger Moore'. The second field has an envelope icon. The third field has a phone icon. Below the fields is a red button with the text 'SUBMIT' and three dots to its right.



An HTML form is essentially just a wrapper for data collection elements, and it tells the page:

- Where to send the data
- How to send the data
- What data is being sent

# FORM TAG SYNTAX

**<FORM> </FORM>**

Available Attributes:

- Action (url to send data to)
- Method (*POST* or *GET*)
- Enctype (multipart/form-data if uploading files)

# **ACTION ATTRIBUTE**

This attribute defines where the data gets sent. Its value must be a valid URL.

This will typically be a file on your server that parses it and does something with it.

# METHOD ATTRIBUTE

This attribute defines how data is sent from a form.

This will always be either *get* or *post*

- GET - form data is appended to the URL as a series of name/value pairs when submitted
- POST - form data included in the request body instead of the URL

# ENCTYPE ATTRIBUTE

This attribute specifies how the form data should be encoded when submitting it to the server.

- *multipart/form-data* - No characters are encoded. This value is required when you are using forms that have a file upload control
- *application/x-www-form-urlencoded* - (default) All characters are encoded before sent (spaces are converted to "+" symbols, and special characters are converted to ASCII HEX values)

# Example

```
<form action="register.php" method="post" enctype="multipart/form-data">  
  <!--Data collection elements go here-->  
</form>
```



**PULSE CHECK - GET AND POST FORMS**

# INPUT ELEMENTS



HTML inputs are what we place in between the `<form>` `</form>` tags to capture the user's input:

```
<form>  
  <input type="text" name="firstName" placeholder="First Name" />  
</form>
```

## Types of form inputs:

- Text fields
- Checkboxes
- Radio Buttons
- etc.

# INPUT ATTRIBUTES

- Type
- Name
- Placeholder
- Value

## ATTRIBUTE: NAME

The name attribute specifies the name of an `<input>` element.

The name attribute is used to reference elements in JavaScript, or to reference form data after a form is submitted.

```
<input type="text" name="fname">  
<!-- The server-side code will look for "fname" to get the val
```

**IMPORTANT:** Only form elements with a name attribute will have their values passed when submitting a form.

## ATTRIBUTE: PLACEHOLDER

The `placeholder` attribute is a short hint for what to type in the input field.

```
<input type="text" placeholder="Please enter your first name">  
<!-- The placeholder disappears once you click into the field
```

## ATTRIBUTE: VALUE

The `value` attribute specifies the value of an input element.

```
<input type="text" value="Mansoor">  
<!-- This will display "Mansoor" in the field on page load -->
```

**IMPORTANT:** It behaves differently based on the input type:

- For "button", "reset", and "submit" - it defines the text on the button
- For "text" and "password" - it defines the initial (default) value of the input field
- For "checkbox" and "radio" - it defines the value associated with the input (this is also the value that is sent on submit)




Let's take a closer look at some of the more common  
input types...

# TYPE: TEXT

Basic text input field.

**<INPUT TYPE = "TEXT">**

 GENERAL ASSEMBLY

Sign In

FRONT-END WEB DEVELOPMENT

APPLY NOW

Where are you thinking of taking this course?

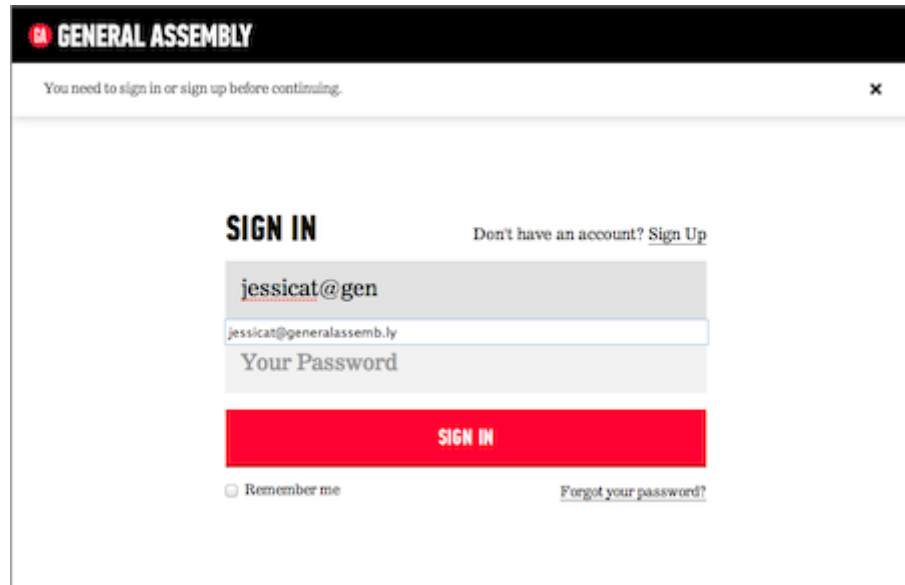
CONTINUE TO APPLICATION

Fill out some basic information and complete the following application to be considered for the course.

# TYPE: EMAIL

Automatically validates the field to ensure its a properly-formatted email address

**<INPUT TYPE = "EMAIL">**



The screenshot shows a login interface for 'GENERAL ASSEMBLY'. At the top, a black header contains the logo and name. Below it, a message states 'You need to sign in or sign up before continuing.' with a close button. The main section is titled 'SIGN IN' and includes a link for users without an account. The form contains three input fields: the first contains 'jessicat@gen', the second contains 'jessicat@generalassemb.ly', and the third is labeled 'Your Password'. A red 'SIGN IN' button is positioned below the password field. At the bottom, there is a 'Remember me' checkbox and a 'Forgot your password?' link.

**GENERAL ASSEMBLY**

You need to sign in or sign up before continuing. ✕

**SIGN IN** Don't have an account? [Sign Up](#)

jessicat@gen

jessicat@generalassemb.ly

Your Password

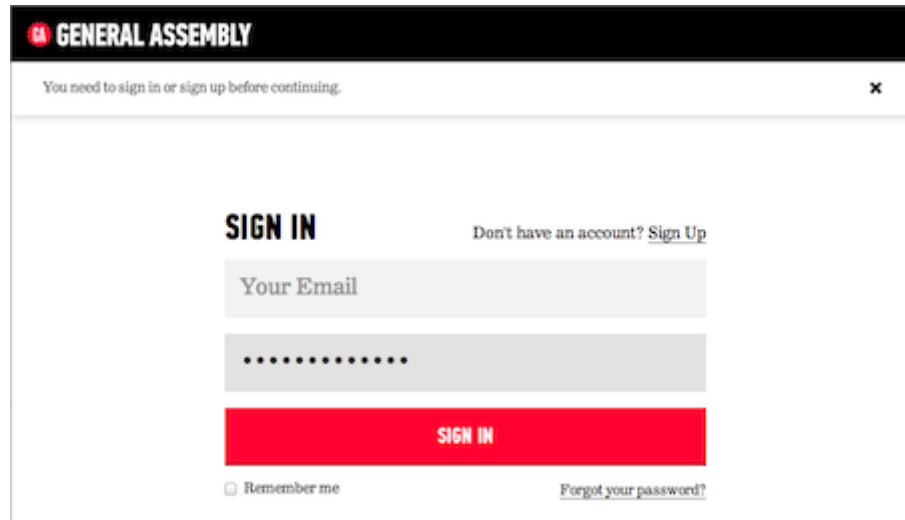
**SIGN IN**

☐ Remember me [Forgot your password?](#)

# TYPE: PASSWORD

Hides characters as the user types them

**<INPUT TYPE = "PASSWORD">**



The screenshot shows a web interface for 'GENERAL ASSEMBLY'. At the top, a black header contains the logo and name. Below it, a white banner with a close button (X) states: 'You need to sign in or sign up before continuing.' The main content area is titled 'SIGN IN' and includes a link for users without an account: 'Don't have an account? [Sign Up](#)'. The form consists of two input fields: 'Your Email' and a password field represented by a series of dots. Below these is a prominent red 'SIGN IN' button. At the bottom, there is a checkbox labeled 'Remember me' and a link for users who forgot their password: '[Forgot your password?](#)'.

# TYPE: SUBMIT VS FILE VS BUTTON

- `type="submit"` creates a clickable button that submits the form when clicked on
- `type="file"` creates a file upload element
- `type="button"` creates clickable button



# PULSE CHECK - INPUT TYPES

Input Types Codepen Example

# 5 MINUTE BREAK



**INPUT TYPE (CONTINUED)**



## **TYPE: RADIO**

Radio buttons are generally used in radio groups. These groups allow the user to select any 1 of a limited number of choices.

To accomplish this, all of the related radio buttons must have the same name attribute and different value attributes:

```
<input type="radio" name="car" value="Toyota">  
<input type="radio" name="car" value="Honda">  
<input type="radio" name="car" value="Ford">
```

## **TYPE: CHECKBOX**

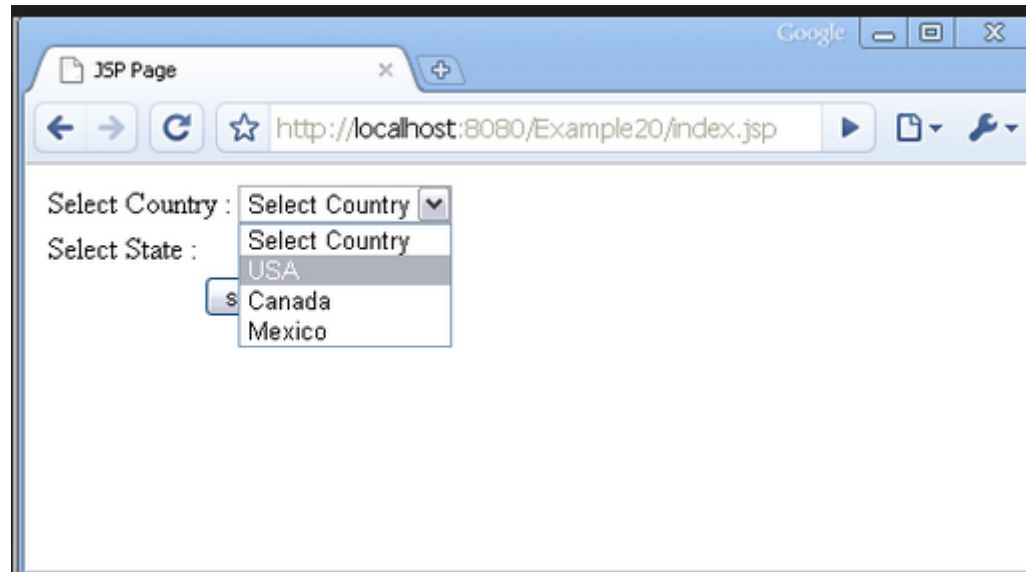
Similar to radio buttons, checkboxes can be used in groups. However, the user can select multiple options.

Again, all of the related checkboxes must have the same name attribute and different value attributes:

```
<input type="checkbox" name="car" value="Toyota">  
<input type="checkbox" name="car" value="Honda">  
<input type="checkbox" name="car" value="Ford">
```

# SELECT AND OPTION

The `<select>` element creates a dropdown list. And the `<options>` tag is used to populate the dropdown list



**<SELECT>**

**<OPTION VALUE =“FIRSTOPTION”> </OPTION>**

**<OPTION VALUE =“ANOTHEROPTION”> </OPTION>**

**</SELECT>**

# TEXTAREA

The `<textarea>` element allows you to type multi-line text data.

# LABELS

Information about the input field should be put in a  
<label> tag

To tie the two together choose one of these methods:

```
<label>Name <input type="text" name="yourName"></label>  
<label for="yourName">Name</label><input type="text" name="yourName">
```



# STYLING

Some things to be aware of when styling inputs:

- The font-family for an input is not inherited!!!
- This can lead to funny sizing issues on Macs vs. PCs where the default font is not the same
- Some form elements can't be styled directly [MDN](#)

[Styling HTML Forms](#)

# FORM VALIDATION

When accepting **any** user input, you always want to validate it

There are several methods for validating user input:

- Simple if/else statements
- Javascript libraries (Parsley.js is a good example)
- Server-side scripts



# APPLICATION FORM

# LEARNING OBJECTIVES REVIEW

- We differentiated between the different types of inputs and why/where we would use each.
- We explained how to group elements by name.
- We applied the method, action, and enctype attributes.

**EXIT TICKETS!**