

---

**FEWD**

---

# FINAL PROJECTS

# AGENDA

---



- Review
- Forms and Inputs
- Lab
- Form Validation
- Functional Forms

## **LEARNING OBJECTIVES**

- Be able to differentiate the different types of inputs and why/where we would use each
- Explain how to group elements by name.

---

**FEWD**

---

**REVIEW**

---

**FEWD**

---

# TRANSITIONS

# TRANSITIONS

---

- Provide a way to control animation speed when changing properties
- Instead of having property changes take effect immediately, you can have them take place over a period of time.

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

# TRANSITIONS

---



**PROPERTY**

Which properties  
to animate



**DURATION**

How long the  
transition will last



**TIMING  
FUNCTION**

How the transition  
will run



**DELAY**

When the animation  
will start

# TRANSITIONS - TRANSITION-PROPERTY

---

- ▶ Can specify a specific property to transition or "all" to transition all properties
- ▶ *Default: all*

```
div {  
  transition: opacity 0.5s;  
}
```

```
div {  
  transition: all 0.5s;  
}
```

```
div {  
  transition: height 0.5s;  
}
```

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



---

# TRANSITIONS - TRANSITION-DURATION

---

- ▶ A time value, defined in seconds or milliseconds

```
div {  
  transition: all 0.5s;  
}
```

```
div {  
  transition: all 350ms;  
}
```

```
div {  
  transition: all 3s;  
}
```

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

---

## TRANSITIONS — TIMING

---

- ▶ Describes how a transition will proceed over its duration, allowing a transition to change speed during its course.
- ▶ Timing functions: ease, linear, ease-in, ease-out, ease-in-out

```
div {  
  transition: opacity 0.5s ease;  
}
```

```
div {  
  transition: opacity 0.5s ease-in-out;  
}
```

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

transition timing: [W3 Schools](#)

---

## TRANSITIONS — DELAY

---

- ▶ Length of time before the transition starts

```
div {  
  transition: background-color 0.5s ease 2s;  
}
```

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

---

## MORE FUN WITH TRANSITIONS — CODROPS

---

Fun CSS button styles: [Creative buttons](#)

Icon hover effects: [Icon Hover Effects](#)

Modal dialogue effects (advanced): [Dialogue Effects](#)

---

**FEWD**

---

# TRIGGERING TRANSITIONS

---

# TRIGGERING TRANSITIONS

---

There are two ways to trigger CSS transitions:

1. Using the :hover CSS pseudo-class
2. Adding a class with jQuery

---

**FEWD**

---

# TRANSFORMATIONS

---

# TRANSFORM

---

transform: [W3 Schools](#)  
transform-origin: [W3 Schools](#)



---

**FEWD**

---

**RESPONSIVE — REM/EM**

---

# EM

---

- *Relative* unit
- Sized based on the width of the letter “m”
- 1em = 100% font-size
- .5em = 50% font-size
- Based on parent

```
Parent { font-size:16px; }
```

```
Child {font-size:2em; }
```



**Child's font size is 32px (200% x 16px)**

---

# REM

---

- "Root" em
- Same as em **except** based on the font-size of the html element

---

# PIXELS AND EMS AND REMS, OH MY!!

---

	RELATIVE?	BASED ON
PX	absolute	
EM	relative	parent
REM	relative	html element

# THE BENEFIT OF USING RELATIVE UNITS

```
html { font-size: 16px; }
h1 { font-size: 33px; }
h2 { font-size: 28px; }
h3 { font-size: 23px; }
h4 { font-size: 19px; }
small { font-size: 13px; }
.box { padding: 20px; }

@media screen and (min-width: 1400px) {
  html { font-size: 20px; }
  h1 { font-size: 41px; }
  h2 { font-size: 35px; }
  h3 { font-size: 29px; }
  h4 { font-size: 24px; }
  small { font-size: 17px; }
  .box { padding: 25px; }
}
```

```
html { font-size: 1em; }
h1 { font-size: 2.074em; }
h2 { font-size: 1.728em; }
h3 { font-size: 1.44em; }
h4 { font-size: 1.2em; }
small { font-size: 0.833em; }
.box { padding: 1.25em; }

@media screen and (min-width: 1400px) {
  html { font-size: 1.25em; }
}
```

---

**FEWD**

---

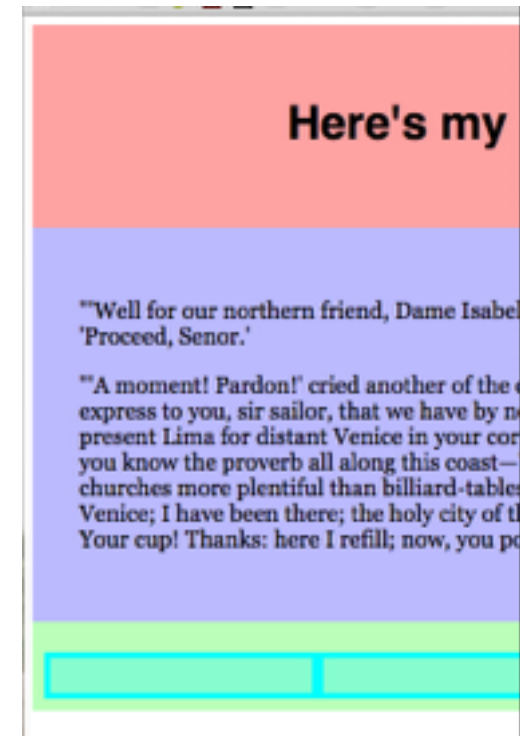
# RESPONSIVE — TYPES OF LAYOUTS

---

# FIXED LAYOUT

---

- ▶ Relies on a container of a fixed width (uses static units)
- ▶ Resizing the browser/viewing it on a different device won't have an effect on the page

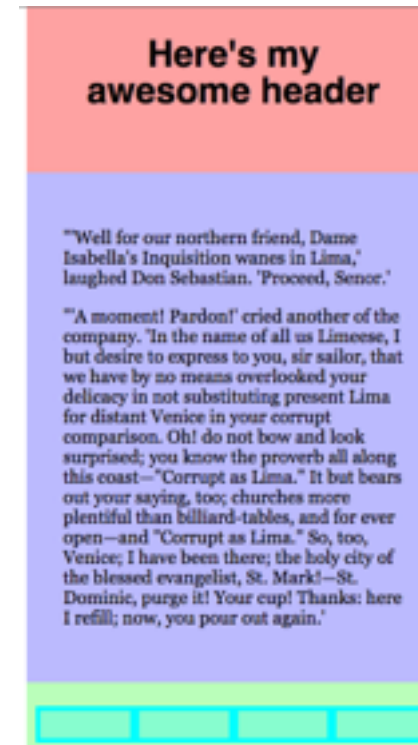


---

# FLUID LAYOUT

---

- Uses relative widths (percentages)
- No media queries



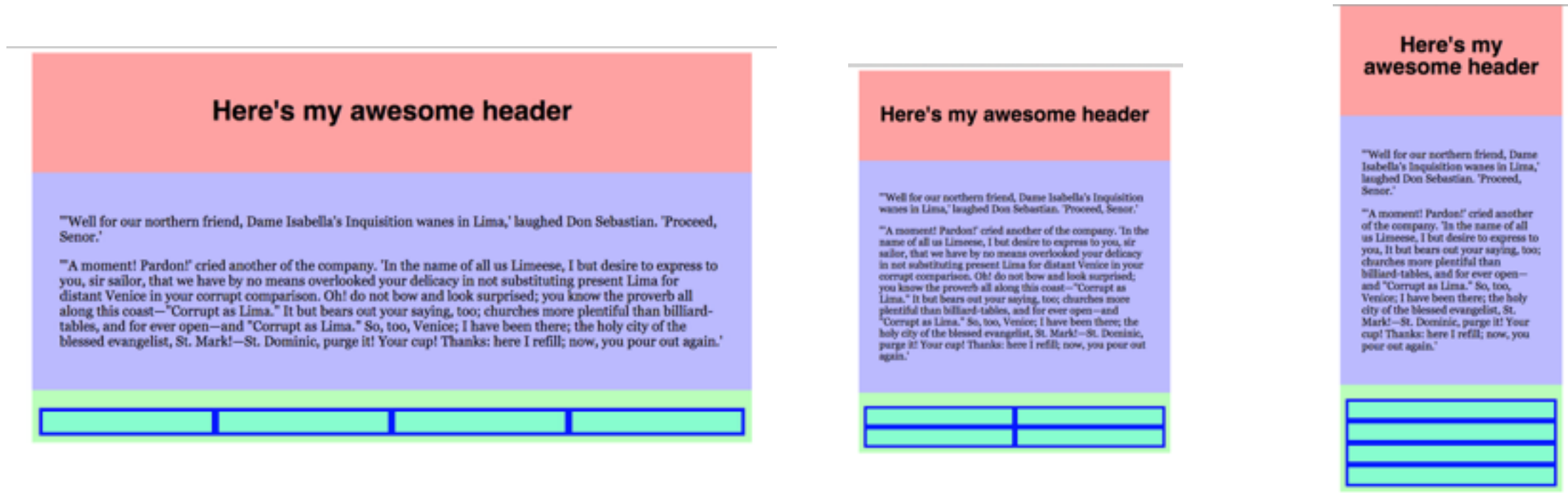


---

# RESPONSIVE LAYOUT

---

- Uses relative widths (built on a fluid grid)
- Use media queries to control design and content as it scales down or up with the browser or device



# MAX-WIDTH — A HELPFUL TOOL FOR LAYOUT

---

Max width



No max width



---

**FEWD**

---

# RESPONSIVE — MEDIA QUERIES

# RESPONSIVE DESIGN

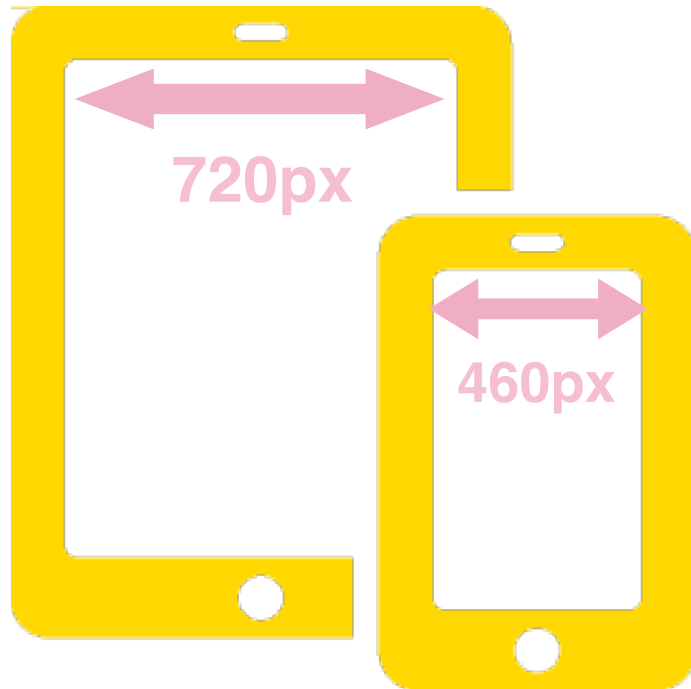
---



# MEDIA QUERIES

---

- ▶ Media queries allow us to target CSS rules based on screen size, device orientation, display density, etc.



---

# MEDIA QUERIES — FIRST METHOD

---

*Create separate stylesheets for different devices*

For example:

- Have one main stylesheet as the default stylesheet
- If the screen becomes too narrow, short, tall, wide, etc. we can detect that and load in an additional stylesheet

```
<link rel="stylesheet" media="screen and (max-width: 460px)" href="css/iphone.css" />
```

---

## MEDIA QUERIES — SECOND METHOD

---

*Use media queries directly in your CSS*

```
@media screen and (max-width: 600px) {  
  .box {  
    width: 100%;  
  }  
}
```

*\*Usually goes at the end of stylesheet*

---

# MEDIA QUERIES — SYNTAX

---

## MEDIA TYPES

- **screen:** color computer screen
- **print:** print preview mode
- **all:** suitable for all devices

```
@media screen {  
  /* Styles for color computer screen */  
}
```

```
@media print {  
  /* All your print styles go here */  
  #header, #footer, #nav { display: none !important; }  
}
```

## MEDIA FEATURES

- **width:** viewport width
- **height:** viewport height
- **orientation:** orientation of the viewport

```
@media screen and (max-width: 600px){  
  /* Styles for screens with a maximum width of 600px */  
}
```

```
@media screen and (min-width: 600px){  
  /* Styles for screens with a minimum width of 600px */  
}
```



# MEDIA QUERIES — SYNTAX

---

## LOGICAL OPERATORS

- › **and:** can be used to combine multiple media features together, as well as combining media features with media types.

```
@media (min-width: 700px) and (orientation: landscape) { ... }
```

- › **comma-separated lists:** behave like the logical operator *or*

```
@media (min-width: 700px), handheld and (orientation: landscape) { ... }
```

- › **not:** applies to the whole media query and returns true if the media query would otherwise return false

```
@media not print { ... }
```

- › **only:** prevents older browsers that do not support media queries with media features from applying the given styles

```
@media only screen and (min-width: 400px) { ... }
```



---

## VIEWPORT META TAG — AN IMPORTANT NOTE!!

---

- The viewport meta tag controls how a webpage is displayed on a mobile device.
- Without the tag, mobile devices will assume you want the full desktop experience and will set the viewport width at 980px (iOS)

### DEVICE-WIDTH

- This tells the browser “My Website adapts to your width”

### INITIAL-SCALE

- Sets the initial zoom level and prevents default zooming

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

# **FORM DESIGN & INTERACTION**

---

**FEWD**

---

# FORMS AND INPUTS

---


**FEWD**

---

# FORM BASICS

# FORMS

How we get data from users

 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.

---

# FORMS

---

1. The user fills out the form and presses the submit button



FRONT-END WEB DEVELOPMENT

## APPLY NOW

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

Where are you thinking of taking this course?

CONTINUE TO APPLICATION

# FORMS

2. The name of each form field is sent to the server along with the value the user entered or selected





# FORMS

3. The server processes the data using a language such as PHP, C# or Java. It may also store the information in a database



# FORMS

4. The server creates a new page to send back to the browser based on the information received.



# FORMS

---

Form controls live inside the <form element>

```
<form action="http://www.example.com/login.php" method="post">  
  <!--Data collection elements go here-->  
</form>
```

# FORMS

---

Form attributes:

```
<form action="http://www.example.com/login.php" method="post">  
  <!--Data collection elements go here-->  
</form>
```

# FORMS

---

Form attributes:

## **ACTION (REQUIRED)**

Where to send the data (URL)



```
<form action="http://www.example.com/login.php" method="post">  
  <!--Data collection elements go here-->  
</form>
```

# FORMS

---


Form attributes:

**ACTION (REQUIRED)**

Where to send the data (URL)

**METHOD (WILL USUALLY HAVE)**

How to send it (post or get)



```
<form action="http://www.example.com/login.php" method="post">  
  <!--Data collection elements go here-->  
</form>
```

# FORMS — METHODS

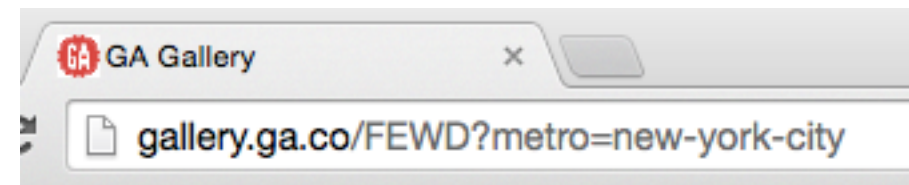
---

## POST

- Data is not shown in URL
- Can contain sensitive data
- No size limitations
- Adds information to, or deletes info from a database

## GET

- Short forms (such as search fields)
- Appended to URL in name/value pairs
- Never use for sensitive info!!!
- Useful for form submissions when user wants to bookmark results



```
<form action="http://www.example.com/login.php" method="post">  
  <!--Data collection elements go here-->  
</form>
```

---

**FEWD**

---

# GETTING INFORMATION FROM USER



---

## GETTING INFO — INPUTS

---

Place any inputs between `<form></form>` tags

Attributes:

- **type** — text, submit, password, email, checkbox, button, radio, file, etc.
- **name, value** — The name attribute is sent to the user along with the value the user selects.
- **placeholder** — For text inputs - hint for what user should enter in field

*Note: For a complete spec see [MDN](#)*

# FORM

---

**ADD TEXT**

Your Email

**MAKE CHOICES**

Where are you thinking of taking this course?

Chicago

**SUBMIT**

**Continue**

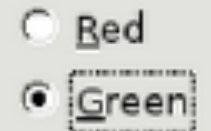
# NAME/VALUE PAIRS

- Information is sent from the browser to the server using name/value pairs.

```
<input type="text" name="username">
```

NAME	VALUE
username	eboyer@gmail.com

```
<input type="radio" name="color" value="red" label="Red">  
<input type="radio" name="color" value="green" label="Green" checked="checked">
```



☐ Red  
☒ Green

NAME	VALUE
color	green

# INPUTS — TEXT

---

```
<input type="text" name="username">
```

Your Full Name

```
<input type="email" name="email">
```

Your Email

```
<input type="password" name="password">
```

.....

\*Can also carry a maxlength attribute to limit the number of characters the user may enter

# LABELS

---

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

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

To tie the two together:

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

*Note: Clicking the label text places the focus in the input field (great for radio buttons)*

# ACTIVITY

---



## EXERCISE

### **KEY OBJECTIVE**

---

- Identify input types, add styles to a form

### **TYPE OF EXERCISE**

---

- Individual/partner

### **TIMING**

---

45 min

1. Review contact\_form.png
2. Write html for contact form
3. Style the form
4. Optional: add responsive styles (see contact\_form\_responsive.png)
5. Super optional: Add validation (see later slides)

*\* You will need to look up the textarea element*

# SELECT AND OPTION

```
<select name="referral">
  <option value="friend">Friend</option>
  <option value="instructor">Instructor</option>
  <option value="online">Online</option>
</select>
```

Where are you thinking of taking this course?

Chicago

- Atlanta
- Austin
- Boston
- ✓ Chicago
- Hong Kong
- London
- Los Angeles
- Melbourne
- New York City
- San Francisco
- Seattle
- Sydney
- Washington D.C.

---

# CHECKBOXES AND RADIO BUTTONS

---

```
<input type="checkbox" name="store_credentials">
```

☐ Remember me

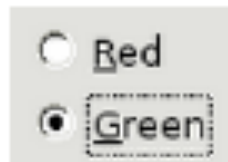


# RADIO BUTTONS

---

Radio buttons are grouped together by their name attribute

```
<input type="radio" name="color" value="red" label="Red">  
<input type="radio" name="color" value="green" label="Green" checked="checked">
```



Red  
Green

---

# SUBMIT FORM

---

```
<input type="submit" value="Continue">
```

Continue

---

**FEWD**

---

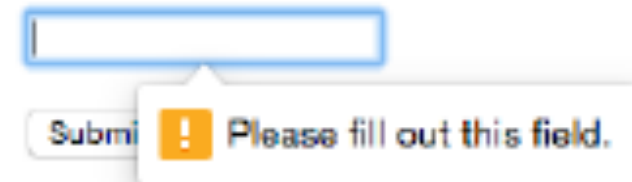
**VALIDATION**

# VALIDATION

---

- ▶ You've probably seen forms on the web that give users messages if the form control has not been filled out correctly.
- ▶ Traditionally, validation has been performed using Javascript.
- ▶ HTML5 also introduced browser-based form validation.

```
<input type="text" name="fullname" required />
```



---

# VALIDATION

---

- ▶ For more substantial validation, it is highly recommended that you use a validation library, such as [Parsley](#).
- ▶ To add parsley validation:
  1. Add jQuery to your project
  2. Add the parsley.js file to your project after you've included jQuery

```
<script src="js/jquery-2.1.3.min.js"></script>  
<script src="js/parsley.js"></script>
```

3. Add the data-parsley-validate attribute to your form tag

```
<input data-parsley-validate/>
```

4. Add the required attribute to any fields you want to be required.

```
<input type="text" name="fullname" required />
```

---

**FEWD**

---

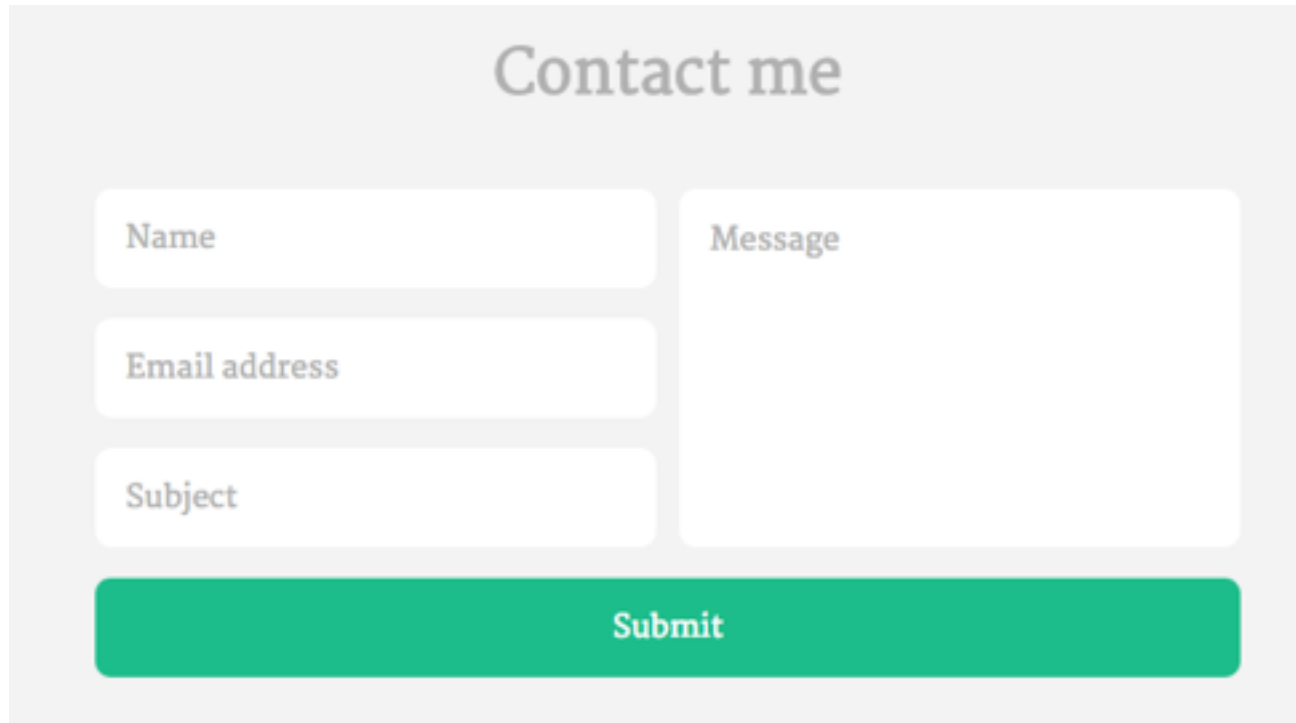
# CONTACT FORM

---

# BUILDING A FUNCTIONAL CONTACT FORM

---

1. I recommend [FormSpree](#) for contact forms that send you an email without having to use PHP.
2. Your site will need to be hosted on a server in order to test emails.



Contact me

Name

Email address

Subject

Message

Submit

## **LEARNING OBJECTIVES**

- Be able to differentiate the different types of inputs and why/where we would use each
- Explain how to group elements by name.



---

**FEWD**

---

**LAB**

---

# LAB

---



---

**FEWD**

---

# **HOMEWORK**

**HAVE SOME OF YOUR RESPONSIVE CSS  
READY THIS SUNDAY**

---

**FEWD**

---

# EXIT TICKETS