GENERAL ASSEMBLY

HTML & CSS

DAY 4

Kim Goulbourne

AGENDA

- Flexbox
- Media
- CSS Transitions
- Forms
- Styling Forms
- Lab

MEDIA: VIDEO

HTML5 VIDEO

With the new <video> tag, we can embed video directly on our site without needing to utilize flash. Huzzah! http://diveintohtml5.info/video.html

EXAMPLE

```
<video>
   <source src="media/video.mp4" type="video/mp4">
</video>
```

HTML5 VIDEO

with controls, looping and autoplay:

```
<video controls=“controls" loop="loop" autoplay ="autoplay">
  <source src="media/video.mp4" type="video/mp4">
</video>
```

can also be written:

```
<video controls loop autoplay>
  <source src="media/video.mp4" type="video/mp4">
</video>
```

HTML5 VIDEO

And with a background (aka "poster") image:

```
<video controls="controls" poster="img/videoimage.png">
  <source src="media/video.mp4" type="video/mp4">
</video>
```

You can also provide a fallback for an older browser:

```
<video controls="controls" poster="img/videoimage.png">
  <source src="media/video.mp4" type=“video/mp4">
  <p>Your stupid browser doesn’t support the video tag</p>
</video>
```

HTML5 VIDEO: IMPORTANT NOTES

- Older browsers don’t support <video> (less than 8% of Global users wont be supported)
- Not all browsers support the same codecs / video file types.
  - MPEG-4/H.264 - Safari, Chrome, and IE9+
  - Ogg - Firefox, Chrome, Opera
  - WebM - Firefox and Chrome
- Creating fallbacks requires multiple video files:

```
<video controls poster="images/fallback.jpg" width="540" height="360">
  <source src="media/video.mp4" type="video/mp4">
  <source src="media/video.webm" type="video/webm">
  <source src="media/video.ogv" type="video/ogg">
</video>
```

HTML5 VIDEO: CROSS BROWSER VIDEO

1 - Use YouTube, Vimeo, or a service like Wistia (wistia.com), as they do all of the detecting and serving for you.

2 - Use a JS library to embed a player: http://www.videojs.com/

3 - Use an off-the-shelf player: https://flowplayer.org/

4 - Create your own video fallbacks and do it yourself!

MEDIA: AUDIO

HTML5 AUDIO

Works very similarly to HTML5 Video:

EXAMPLE

```
<audio controls>
  <source src="media/finalcountdown.mp3" type="audio/mpeg">
</audio>
```

HTML5 AUDIO

As before, certain browsers only support certain types.

MP3 - Chrome, Safari, IE
OGG - (older versions of) Firefox, Opera

```
<audio controls>
  <source src="media/finalcountdown.mp3" type="audio/mpeg">
  <source src="media/finalcountdown.ogg" type="audio/ogg">
</audio>
```

MEDIA: IFRAME

ANOTHER WAY TO INCLUDE MEDIA

If you have your video or hosted somewhere else, you can usually embed it on your page with an iframe.

```
<iframe width="560" height="315" src="https://www.youtube.com/embed/9jK-NcRmVcw" frameborder="0" allowfullscreen></iframe>
```

ANOTHER WAY TO INCLUDE MEDIA

An iframe is basically a way for us to "quote" a website within another website. In practice, we can use this to embed media / pages from various services (which would usually provide the embed code for you) like:

- Youtube
- Vimeo
- Google Maps
- Soundcloud
- Spotify

CSS TRANSITIONS

WHAT ARE CSS TRANSITIONS?

CSS3 transitions allows you to change property values smoothly (from one value to another), over a given duration. It is not fully supported in browsers like IE but supported in most modern browsers.

http://www.w3schools.com/css/css3_transitions.asp

* CSS animations give you even more control but we wont go over that in class so check it out: http://www.w3schools.com/css/css3_animations.asp

CREATING CSS TRANSITIONS

To create a transition effect, you specify:

- the CSS property you want to add an effect to
- the duration of the effect
- the timing function (optional, default is “linear”)

Note: If the duration part is not specified, the transition will have no effect, because the default value is 0.

HOW TO WRITE IT

```
div {
   transition-property: width;
   transition-duration: 2s;
}
```

SHORT HAND

```
div {
   transition: width 2s;
}
```

WITH TIMING FUNCTION

```
div {
   transition-property: width;
   transition-duration: 2s;
   transition-timing-function: linear;
}
```

SHORT HAND

```
div {
   transition: width 2s linear;
}
```

CREATING CSS TRANSITIONS

```
div {
    width: 100px;
    height: 100px;
    background: red;
    transition: width 2s; // property duration
}
```

```
div:hover {
    width: 300px;
}
```

* The transition effect will start when the specified CSS property (width) changes value. In our case, it will happen on hover. https://jsfiddle.net/kimgoulb/ss90f4s9/

CHANGING SEVERAL PROPERTIES

```
div {
    width: 100px;
    height: 100px;
    background: red;
    transition: width 2s, height 4s;
}
```

```
div:hover {
    width: 300px;
    height: 200px;
}
```

* The transition effect will start when the specified CSS property (width) changes value. In our case, it will happen on hover. https://jsfiddle.net/kimgoulb/ss90f4s9/

WE CAN TRANSITION PROPERTIES THAT HAVE A UNIT:

“border-width”
“border-bottom-width”
“border-left-width”
“border-right-width”
“border-top-width”
“margin”
“margin-bottom”
“margin-left”
“margin-right”
“margin-top”

“opacity”
“padding”
“padding-bottom”
“padding-left”
“padding-right”
“padding-top”
“height”
“width”
“font-size”

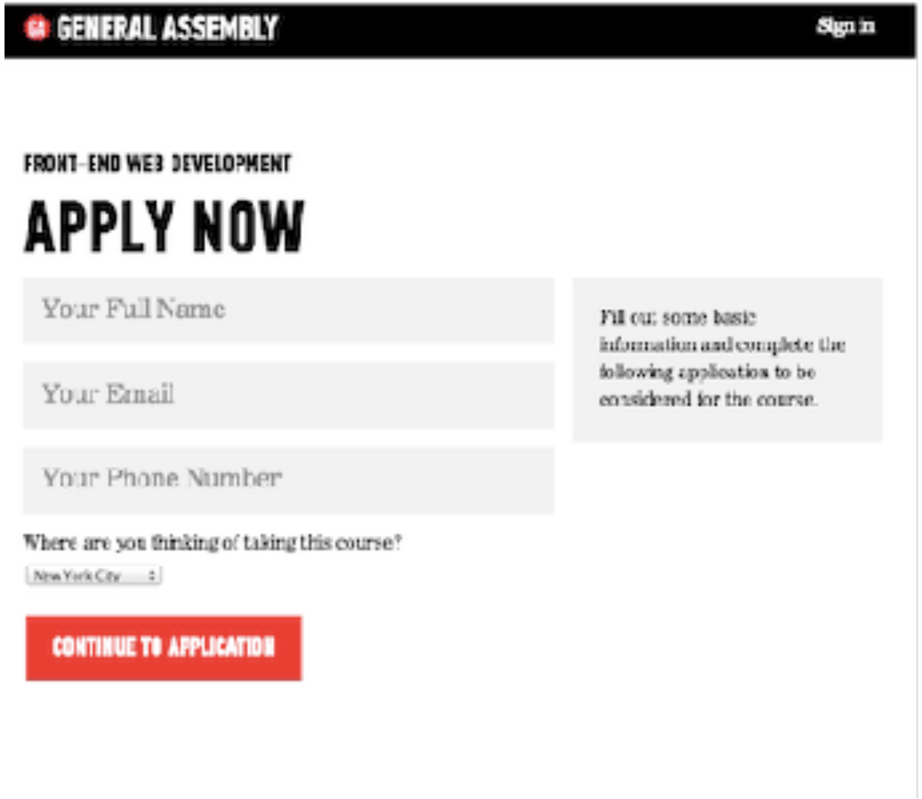
“top”
“left”
“bottom”
“right”

* You cannot animate properties like “display”.

FORMS

FORMS

How we can get data from users.



GENERAL ASSEMBLY

Sign in

FRONT-END WEB DEVELOPMENT

APPLY NOW

Your Full Name

Your Email

Your Phone Number

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

Where are you thinking of taking this course?

New York City

CONTINUE TO APPLICATION

IMPORTANT NOTE

You can’t do anything with the data you collect from your forms unless you are hooked-up to some kind of back-end processing. This is generally done via a Back-End language of some kind (e.g. PHP, Java, Ruby on Rails).

You can validate and manipulate the data on the Front-End, but purely via HTML/CSS/JS alone, you cannot save any data you collect, send an email, create a user account, check a password, etc…

FORM ELEMENTS

THE CONTAINER: <FORM>

Similar to a <ul>, the <form> doesn’t have any inherent use, but it encapsulates the form elements within it.

- Text Fields
- Radio Buttons
- Dropdowns
- Check Boxes
- Submit Buttons and more!

http://www.w3schools.com/html/html_form_elements.asp

THE CONTAINER: <FORM>

```
<form>
   <!--Data collection elements go here-->
</form>
```

FORM ATTRIBUTES: ACTION

The action attribute specifies where to send the form-data when a form is submitted (to be processed). In the following example: On submit, send the form-data to a file named "demo_form.asp" (to process the input):

```
<form action="demo_form.php" method="get">
   <!--Data collection elements go here-->
</form>
```

* if you want the form in your final project to actually work, I'll have to provide something for you so let me know!

FORM ATTRIBUTES: METHOD

The method attribute specifies how to send form-data (the form-data is sent to the page specified in the action attribute). The form-data can be sent as URL variables (with method=“get") typically expecting a return value or as HTTP post transaction (with method=“post”) typically to save data for later use.

```
<form action="demo_form.php" method="post">
   <!--Data collection elements go here-->
</form>
```

FORM ATTRIBUTES: METHOD

Notes on GET:

- Appends form-data into the URL in name/value pairs
- The length of a URL is limited (about 3000 characters)
- Never use GET to send sensitive data! (will be visible in the URL)
- GET is better for non-secure data, like query strings in Google

Notes on POST:

- Appends form-data inside the body of the HTTP request (data is not shown is in URL)
- Has no size limitations

ATTRIBUTE: NAME

Your form elements will look very nice presented on the page but if you don't "name" your elements, they will all be ignored. This is because the form fields need the "name" attribute so the backend(action file) knows what the data represents.

So to all of the fields we're going to cover, the attribute "name" needs to be added, for example <input type="text" name="firstname">.

ATTRIBUTE: VALUE

The value attribute specifies the value of an <input> element. It is used is used differently for different input types: https://jsfiddle.net/kimgoulb/auheh030/

- For "text", "password", and "hidden" - it defines the initial (default) value of the input field (this changes when a user enters something)
- For "button", "reset", and "submit" - it defines the text on the button
- For "checkbox", "radio", "image" - it defines the value associated with the input (this is also the value that is sent on submit)

<INPUT> WITH THE “TYPE” ATTRIBUTE

The <input> tag specifies an input field where a user can enter or select data. An input field can vary in many ways, depending on the “type” attribute.

We’re going to cover: text, number, email, checkbox, radio and submit. More: http://www.w3schools.com/tags/tag_input.asp

<INPUT> : TEXT

An user can enter plain text in a text input field. Examples usage would be for a name or an address.

```
<input type="text" name="firstname" class="name" />
```

- *Get the value of this input field:* `$('.name').val();`
- *Set the value of this input field:* `$('.name').val("Kim");`

<INPUT> : NUMBER

An user can only enter a number in a number input field. Examples usage would be for age or number of tickets.

```
<input type="number" name="age" class="age" />
```

- *Get the value of this input field:* $('.age').val();
- *Set the value of this input field:* $('.age').val(16);

<INPUT> : EMAIL

An user has to enter a *valid* email address in an email input field or the browser will throw errors. Examples usage would be for collecting email addresses in a subscription form.

<input type=“email” name=“email” class=“email” />

- *Get the value of this input field:* $(‘.email’).val();
- *Set the value of this input field:* $(‘.email’).val(“test@gmail.com”);

<INPUT> : SUBMIT

The submit input field renders as a button to submit the form. You can control the text that appears on the submit button with the value attribute. It is used interchangeably with the <button> tag.

```
<input type="submit" name="submit" class="subscribe-btn"
value="Subscribe" />
```

- *Get the value of this input field:* `$('.subscribe-btn').val();`
- *Set the value of this input field:* `$('.subscribe-btn').val("Submitting…");`

EXERCISE: BUILD A SIMPLE NEWSLETTER FORM

* We only need to capture the user’s email
* It should POST to newsletter.php

<INPUT> : CHECKBOX

A checkbox input field is used to toggle on/off multiple responses by the user. Typically you would have more than one in a form (but with the same “name”). Example usage could be categories a user is interested in.

```
<div class=“categories-group”>
  <input type=“checkbox” name=“categories” value=“pop” class=“category pop” />
  <input type=“checkbox” name=“categories” value=“r&b” class=“category rb” />
</div>
```

CHECKBOX ATTRIBUTE: CHECKED

* A boolean attribute (returns true or false)

The “checked” attribute is used to pre/auto-check a field and determine whether or not it’s checked in js.

- *Return all checked fields*:	$(‘.category:checked’);
- *Get the value of a field:*	$(‘.category.pop’).val();
- *Is this field checked (true/false)?*	$(‘.category.pop’).prop(‘checked’);
- *Auto check this input field:*	$(‘.category.pop’).prop(‘checked’,true);

AUTO CHECKED IN HTML

```
<input type=“checkbox” name=“categories” value=“pop” class=“category pop” checked />
```

<INPUT> : RADIO

A radio input field is similar to a checkbox except the user is only allowed one response. Typically you would have more than one in a form (but with the same "name") to group them. Example usage could be a package option.

```
<div class="packages-group">
  <input type="radio" name="packages" value="silver" class="package silver" />
  <input type="radio" name="packages" value="gold" class="package gold" />
</div>
```

RADIO ATTRIBUTE: CHECKED

* A boolean attribute (returns true or false)

The “checked” attribute is used to pre/auto-check a field and determine whether or not it’s checked in js.

- *Return all checked fields*:	$(‘.package:checked’);
- *Get the value of a field:*	$(‘.package.silver’).val();
- *Is this field checked (true/false)?*	$(‘.package.silver’).prop(‘checked’);
- *Auto check this input field:*	$(‘.package.silver’).prop(‘checked’,true);

AUTO CHECKED IN HTML

```
<input type=“radio” name=“packages” value=“silver” class=“package silver” checked” />
```

<TEXTAREA>

Textarea is, basically, a large, multi-line textbox. The anticipated number of rows and columns can be defined with rows and cols attributes, although you can manipulate the height to your heart's content using CSS. It has opening and closing tags.

```
<textarea name="bio" rows="5" cols="20">A big load of text</textarea>
```

- *Get the value of this field:* $('.bio').val();
- *Set the value of this field:* $('.bio').val("Submitting…");

<SELECT>

The select tag works with the option tag to make drop-down of options for the user to select from. When the form is submitted, the “value” of the selected option will be sent..

```
<select name=“dropdown>
  <option value=“Option 1”>Option 1</option>
  <option value=“Option Two” class=“test”>Option 2</option>
  <option value=“Option 3">Option 3</option>
</select>
```

<SELECT>

The “selected” attribute is used to pre/auto-select an option in the dropdown and determine whether or not it’s selected in js.

- *Get the value of this field*:	$(‘select’).val();
- *Get the value of the selected field*:	$(‘select option:selected’).val();
- Auto select an option:	$(‘select .test’).prop(‘selected’,true);

AUTO SELECTED IN HTML

<option value=“Option 1” selected>Option 1</option>

<LABEL>

The <label> tag defines a label for an form element. It does not render as anything special for the user. However, it provides a usability improvement for mouse users, because if the user clicks on the text within the <label> element, it toggles the control.

* The “for” attribute of the <label> tag should be equal to the id attribute of the related element to bind them together.

```
<label for=“firstname">First Name</label>
<input type="text" name="firstname" id="firstname" />
```

EXERCISE: BEEF UP OUR NEWSLETTER FORM

ADDITIONS

* Add labels for all fields
* First Name
* Last Name
* Age
* Checkboxes to select one or more interests (Music, Dance, Tech, Fashion, Art, Politics, Education)
* Radio buttons to select how I want the newsletter (daily, weekly, monthly)
* Select dropdown to select my gender (animal, person, object).

JAVASCRIPT

* Pre-check one of the categories

LUNCHTIME

FORM ELEMENT ATTRIBUTES

PLACEHOLDER

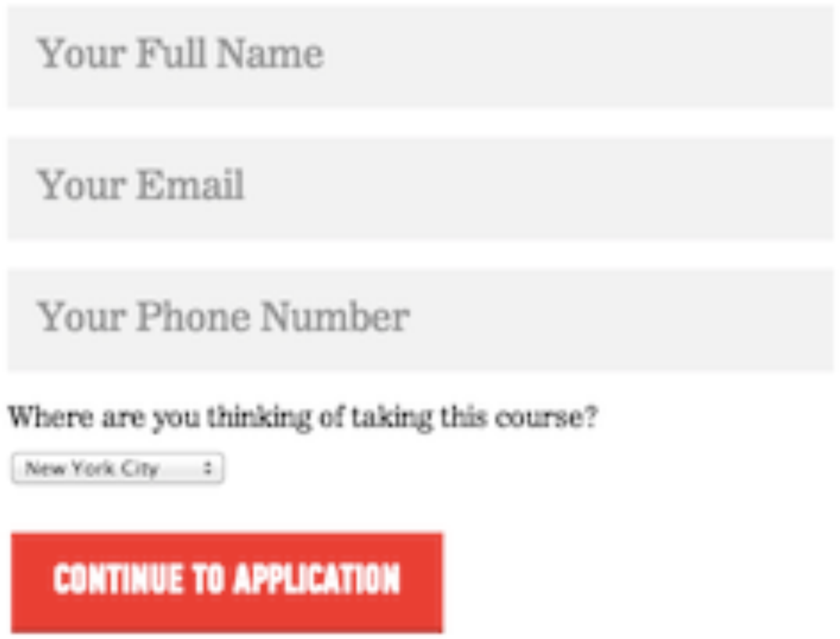
The “placeholder” attribute specifies a short hint that describes the expected value of an input field. It is displayed in the input field before the user enters a value.

USAGE

```
<input type=“text” name=“first_name” placeholder=“First Name” />
```

USING THE PLACEHOLDER ATTRIBUTE

The placeholder attribute works with the following input types: text, number, email, password, search, url and tel.



DISABLED

* A boolean attribute (returns true or false)

The "disabled" attribute specifies that the <input> element should be disabled which means it is unusable and un-clickable. It can be set to keep a user from using the <input> element until some other condition has been met (eg. selecting a checkbox). Then, JS could remove the disabled value, and re-enable the <input>.

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

* Disabled <input> elements in a form will not be submitted.

REQUIRED

* A boolean attribute (returns true or false)

The “required” attribute specifies that an input field must be filled out before submitting the form. It works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

```
<input type=“text” name=“username” required>
```

AUTOCOMPLETE

The autocomplete attribute specifies whether a form should have autocomplete on or off. When autocomplete is on, the browser automatically complete values based on values that the user has entered before.

* It is possible to have autocomplete "on" for the form, and "off" for specific input fields, or vice versa.

```
<input type=“text” name=“username”  autocomplete="off">
```

AVOIDING THE BACKEND

ONLINE SERVICES

There are services that will let you create forms that you can drop onto your site via iframe. http://mashable.com/2012/02/16/web-form-builders/

- Use a Google Form that you embed with an iframe to collect data.
- Use a service like https://formspree.io/ or http://www.jotform.com/
- Use a email subscription service like MailChimp / Constant Contact.
- Look up how to submit data to a Google Spreadsheet using your custom HTML built form.

* There are also plenty of pre-existing contact forms used on popular CMSes, such as Squarespace or Wordpress.

STYLING FORMS

STYLING <INPUT> & <TEXTAREA>

Most css properties can be used to style <input> and <textarea> fields. To style a specific input field you can use attribute selectors.

```
input {
  width: 100%;
  height: 50px;
}
```

```
input[type="text"] {
  width: 80%;
  border: solid 1px #333;
}
```

```
input[type="email"] {
  width: 300px;
  padding:15px;
}
```

Check it out: http://www.w3schools.com/css/css_form.asp

STYLING FOCUSED <INPUT>

By default, some browsers will add a blue outline around the input when it gets focus (clicked on). You can remove this behavior by adding outline: 0; to the input and provide an alternate style for accessibility.

```
input[type=text]:focus {
    outline: 0;
    background-color: lightblue;
}
```

STYLING CHECKBOXES OR RADIO BUTTONS – A TRICK

Checkboxes and radio buttons can’t be styled but there is a work around. Using the <label> element we can simulate a click on a checkbox or radio button and therefore style the label instead.

Example: https://jsfiddle.net/kimgoulb/akwz4f9g/4/

STYLING <SELECT>

<select> elements aren't so easy to style. You can use basic properties such as width, height, padding, background-color and some font styles but these will not affect the style of the inner <option> elements. These cannot be directly styled with css and will always hold the browser defaults.

* You can create custom dropdowns using <ul> and use that in your forms but you would have submit the data yourself using JS.
* Here's a plugin you can use that uses this method: https://select2.org/

EXERCISE: STYLE YOUR NEWSLETTER FORM