Front-End Web Development

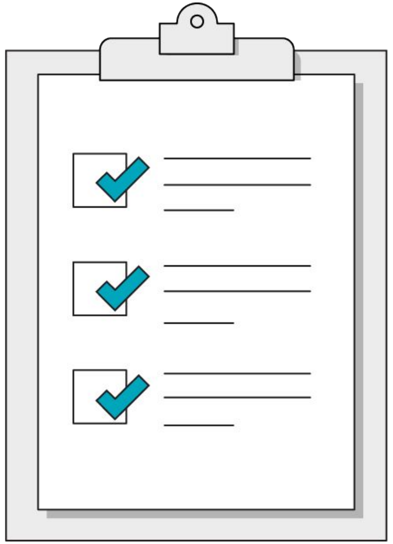
Forms

GENERAL ASSEMBLY

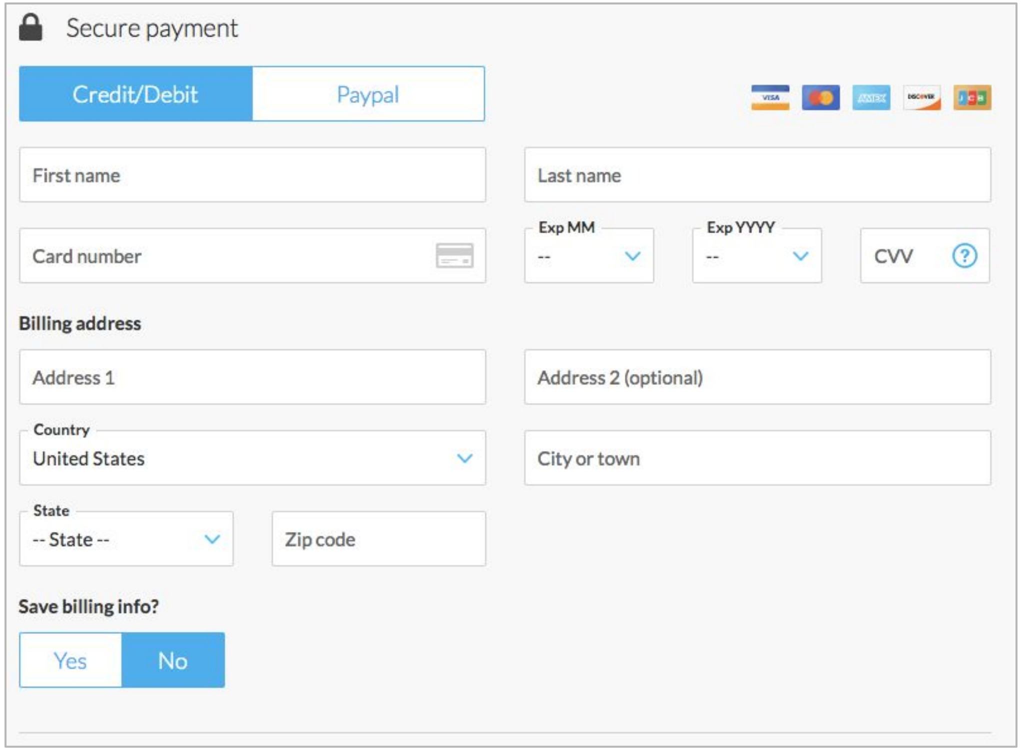
Today’s Learning Objectives

In this lesson, you will:

- Use HTML forms to collect input from users.
- Respond to form submission events to perform validation checks.



Websites Take on Many Forms...



Secure payment

[✓ Credit/Debit] Paypal

First name

Last name

Card number

Exp MM --

Exp YYYY --

CVV

Billing address

Address 1

Address 2 (optional)

Country: United States

City or town

State: -- State --

Zip code

Save billing info?

Yes [✓ No]

Guided Walk-Through:

Forms Reference

Reference the HTML for forms in the following CodePen:

Reference code:

https://codepen.io/GAmarketing/pen/592d34e48473087e5444d4dad31af7d2

What Forms Do

- Forms capture user input from the web and typically send it off to the back-end of a website to be processed in some way.
- Anytime you enter any content into a webpage, you are using a form!

Simplest Form

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

All forms have the `form` tag wrapping around them. Always include all components of your form within these tags.

Forms have a lot of markup and are tedious to build.

<input>

```
<form>
  <fieldset>
    <legend>Form Title</legend>
    <input type="text" id="username">
    <input type=”number” id=”age”>
    <input type=”submit” value=”Submit this form!”>
 </fieldset>
</form>
```

<label>

```
<form>
  <fieldset>
    <legend>Form Title</legend>
    <label for="username">Username:</label>
    <input type="text" id="username">
    <button type="submit">
  </fieldset>
</form>
```

Ready, <fieldset>, Go!

```
<form>
  <fieldset>
    <input type="text">
    <button type="submit">
  </fieldset>
</form>
```

Life of a <legend>

```
<form>
  <fieldset>
    <legend>Form Title</legend>
    <input type="text">
    <button type="submit">
  </fieldset>
</form>
```

Front-End Web Development

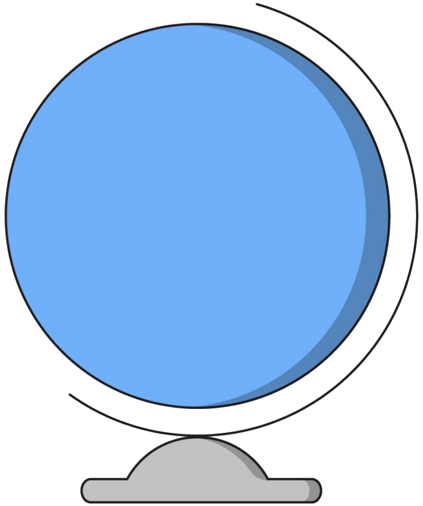
Types of Inputs

The Wide, Wide World of Input Types

There's a huge list of possible input types out there: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input

Depending on how specific you want to get, the same form question could be represented with several different form inputs.

Once again, the world of programming presents us with a large number of possibilities. Like usual, we'll focus in on a few of the most important items that will allow us to master the patterns that will show up for all other types of inputs.



text and number Inputs

```
    <form>
      <fieldset>
        <legend>Form Title</legend>
        <input type="text" id="username">
        <input type="number" id="age">
        <input type="submit" value="Submit this form!">
     </fieldset>
    </form>
```

Data gathered with JavaScript from a `number` input will be in the form of a string! You may have to typecast the value into a `number` type.

<textarea>

```
<textarea placeholder="Write something here"></textarea>
```

This creates a really big area of text that users can fill in. Although you may see this done in examples, don't size it with HTML attributes — please use CSS!

<select>

```
<select id="plays-of-shakespeare">
  <optgroup label="Dramas">
    <option value=”king-lear”>King Lear</option>
    <option value=”hamlet”>Hamlet</option>
  </optgroup>
  <optgroup label="Comedy">
    <option value=”midsummer”>A Midsummer Night’s Dream</option>
    <option value=”twelfth-night”>Twelfth Night</option>
  </optgroup>
</select>
```

- `select` is for lists you click on to select an item.
- `optgroup` is optional but useful for grouping related options.
- `option` is an individual value a user can choose from the select list.

Checkboxes

```
<div class="checkbox">
  <input type="checkbox" id="vip-upgrade">
  <label for="vip-upgrade">VIP upgrade</label>
</div>
```

checkbox inputs can either be “on” or “off,” depending on whether the user has clicked them.

You have to put labels after checkboxes for them to make any sense, otherwise they’re just floating squares in your form.

Radio Buttons

```
<div class="radio">
  <input type="radio" id="radio1" name="quality-feedback">
  <label for="radio1"><span>Good</span></label>
</div>
<div class="radio">
  <input type="radio" id="radio2" name="quality-feedback">
  <label for="radio2"><span>Bad</span></label>
</div>
```

Radio buttons are very similar to checkboxes but function as a group with the **name** attribute. Users can only choose one option among **`radio`** inputs with the same name.

Front-End Web Development

Styling Forms

New CSS Selector: Attributes

```
input[type=text] {
  margin: 25px 0;
}

input[type="text"] {
  margin: 25px 0;
}
```

:focus

```
textarea:focus {
  outline: none;
}
```

There may be cases where you want to style what it looks like when a user is active on a field. To do so, use the **`:focus`** pseudo-selector.

Styling Notes

Don’t

- Style the text boxes, checkboxes, and radio buttons themselves. You can do this, but the techniques will drive you crazy.
- Clutter the form with too many images or animations. Forms are supposed to make interactions with web applications simple and clear.

Do

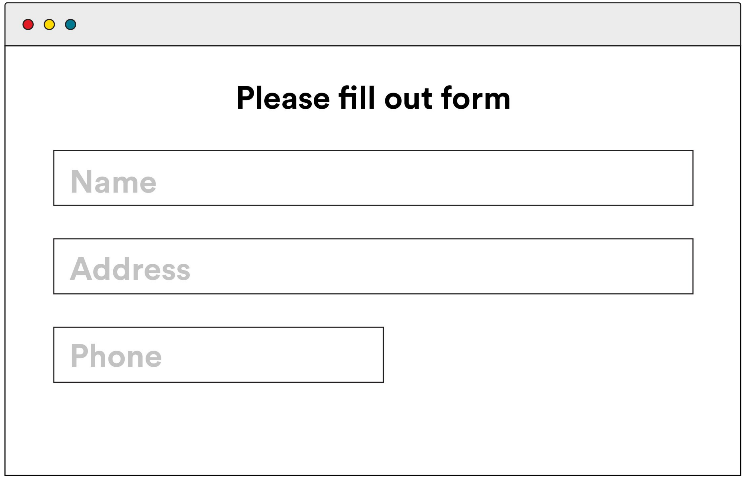
- Put effort into styling valid vs. invalid form submission states. Help your users by clearly communicating when a form is invalid and which fields need to be fixed.

Group Exercise:

Registration Form

30 minutes

Choose one of your favorite websites and take a look at its registration form for new users. In groups, try recreating the HTML and style of the form. The more types of inputs, the better!



Please fill out form

Name

Address

Phone

Front-End Web Development

Validating Form Submissions

Guided Walk-Through:

Responding to Form Submissions

Let’s use this form as an example for responding to a form submission:

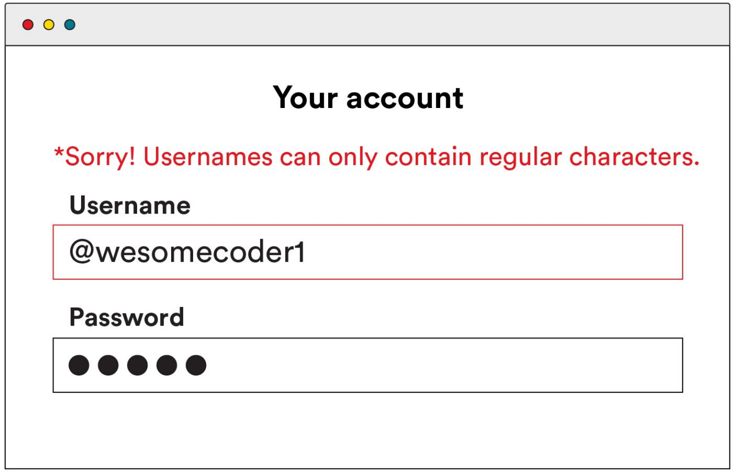
Reference code:

https://codepen.io/GAmarketing/pen/GRgvqOM

Validating Form Submissions

We're going to break down how this happens in the next few slides by:

1. Responding to the "submit" event from a form.
2. Using the DOM to access the value a user has typed into an input.
3. Using DOM manipulation to add error messages to the page.



Your account

*Sorry! Usernames can only contain regular characters.

Username

@wesomecoder1

Password

●●●●●

Responding to the `submit` Event

Remember, nearly any user action triggers an event from the DOM. It’s up to us to attach a listener to the specific DOM element that fires off the event. Let’s grab that element from the DOM and use the **`addEventListener()`** method:

```
document.getElementById(“#registrationForm”).addEventListener(“submit”,
    function(event){
        // event.preventDefault() prevents the browser from refreshing
        event.preventDefault();
    }
);
```

Reading the Form Content

We can use the `.value` getter method to access the value of an input field...

```
    document.getElementById(“#registrationUserName”).value
```

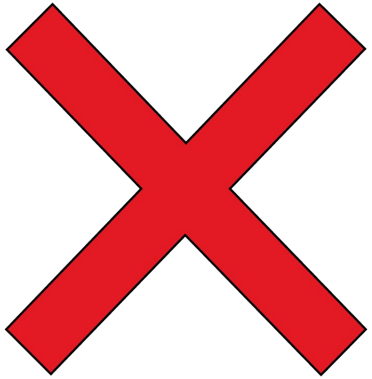
...then, we can test that value for anything we want!

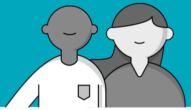
```
    if(username.length < 3){
        // Show an error message on the DOM
    }
```

Displaying Error Messages

Think about most error messages you've seen from forms. Chances are they were displayed close to the input with the invalid value or beneath the form itself, if the problem was more general. You can do this several ways:

- You could have an empty, hidden element for errors that gets shown or filled in once the error occurs.
- You could append a new element to the DOM then remove it upon resubmission of the form.
- You could use an alert box — though this is a bit old fashioned.





Partner Exercise:

Form Submissions

60 minutes

Add validations and error messages for fields in this example form:

Starter code:

https://codepen.io/GAmarketing/pen/GRgvqOM

Key Takeaways

Forms Power User Interaction

- Forms have many types of inputs.
- CSS attribute selectors can target specific types of inputs.
- We can respond to form submissions using event listeners.

For Next Time

APIs

- We'll use forms to collect data from users and use that data to make requests to other servers.
- Our apps will finally interact with other members of the internet community: APIs!