




FEWD - FORMS & INPUTS

NICO CASTRO

Web Developer at Red Badger

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

FORMS

- Wrapper for data collection elements
 - Text fields
 - Dropdowns (selects)
 - Radio Buttons
 - etc

FORMS

Tells the page:

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

FORM TAG

`<FORM> </FORM>`

Available Attributes

- `method`: `post`, `get`, `put`, `delete`
- `action`: url to send data to
- `enctype`: `multipart/form-data` if uploading files, otherwise no need to specify

FORM TAG

In Action

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

INPUTS

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


INPUTS - ATTRIBUTES

- **type:**
text, submit, password, email, checkbox, button, radio, file, etc
- **name:**
the name of the input, which is submitted with the form data to the server
- **placeholder**
- **value:**
the initial value for that input. This attribute is optional except when the input has a `type` of `radio` or `checkbox`.

TEXT

Use `value` to set initial text value or `placeholder` to set instructional text

<INPUT TYPE = "TEXT">

 GENERAL ASSEMBLY

Sign In

FRONT-END WEB DEVELOPMENT

APPLY NOW

Your Full Name

Your Email

Your Phone Number

Where are you thinking of taking this course?
New York City

CONTINUE TO APPLICATION

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

EMAIL

Allows browser to autofill field

<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 light gray message bar states 'You need to sign in or sign up before continuing.' with a close button. The main content area features a 'SIGN IN' heading and a link for users without an account. The email input field is highlighted with a gray background and contains 'jessicat@gen'. A blue autofill dropdown menu is open, showing 'jessicat@generalassemb.ly' as a suggestion. Below the email field is a password field labeled 'Your Password'. A prominent 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

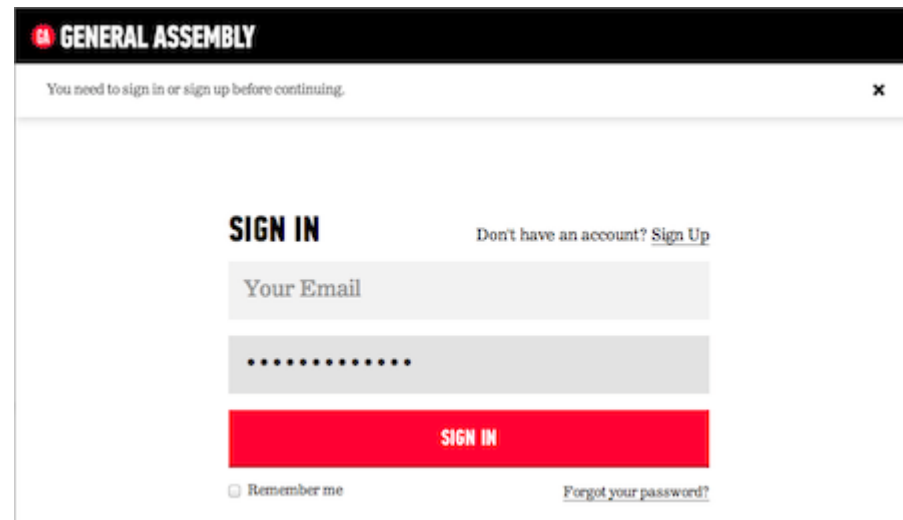
SIGN IN

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

PASSWORD

Hides characters as typed

<INPUT TYPE = "PASSWORD">



GA GENERAL ASSEMBLY

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

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

Your Email

.....

SIGN IN

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

INPUT TYPES ARE IMPORTANT!

<http://mobileinputtypes.com/>

ALL THE BUTTONS

- Button that submits its parent form:

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

- Another button that submits its parent form (default type for a `<button>` is "submit"):

```
<button>Submit</button>
```

ALL THE BUTTONS

- A button with no default functionality:

```
<button type="button">Click Me</button>
```

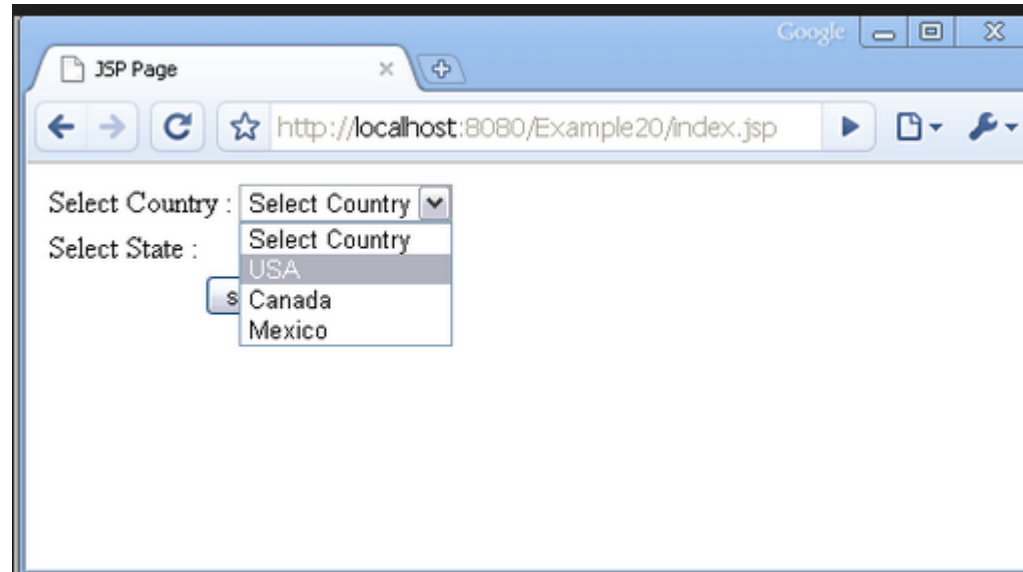
- Creates a file upload button:

```
<input type="file" />
```

Choose File

No file chosen

SELECT AND OPTION



SELECT AND OPTION

<SELECT>

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

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

</SELECT>

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" />  
</label>
```

OR

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

STYLING

It's SO SO HARD! (probably not worth it unless you're getting paid to do it)

The general gist: you have to visually hide the real form element and style a random `<div>` around it to LOOK like the form element



INSULT GENERATOR - PART 1

JS OBJECTS

How to create an object:

```
var x = new Object();  
var y = {};
```

JS OBJECTS

Much like with arrays:

```
var arr = [];
```

Better to declare object directly:

```
var obj = {};
```

JS OBJECTS

Much like you can declare and assign values to variables in one go:

```
var age = 21;
```

You can do the same for objects:

```
var ages = {  
  "james": 26,  
  "anna": 10,  
  "toby": 40  
}
```

JS OBJECTS

You can also add in properties later:

```
var ages = {};  
ages.james = 26; //though it may not look like it "james" is a string  
ages.toby = 40; //the dot notation (obj.property syntax) lets you not  
  
ages //returns {james: 26, toby: 40}
```

JS OBJECTS

Objects have **keys** (aka: properties) and **values**:

```
var ages = {  
  "james": 26,  
  "anna": 10,  
  "toby": 40  
}
```

keys are "james", "anna", and "toby"

values are 26, 10, and 40

JS OBJECTS

Accessing the values:

```
var ages = {  
  "james": 26,  
  "anna": 10,  
  "toby": 40  
}  
  
ages.james //returns 26  
ages["anna"] //returns 10
```

JS OBJECTS

Just like you can store many data types on a variable:

```
var ages = ["21", "22", "23"];  
var favFruit = "kiwi";  
var check = ages[0] > 18; //check's value is true  
var youngestPlusOldest = age[0] + age[2]; //youngestPlusOldest's value
```

JS OBJECTS

...you can store many data types on an object key:

```
var cars = {  
  "nico": {  
    "make": "Toyota",  
    "model": "Corolla",  
    "year": 2000  
  },  
  "dan": false,  
  "anna": ["Ferrari", "BMW", "Audi"]  
}
```

JS OBJECTS

Changing the existing value on a key:

```
var ages = {  
  "james": 26,  
  "anna": 10,  
  "toby": 40  
}  
  
ages.james //returns 26  
  
ages.james = 56;  
ages.james //returns 56  
  
ages //returns {james: 56, anna: 10, toby: 40}
```

JS OBJECTS

By the way, just like object keys can store arrays, the reverse is true (arrays can contain objects):

```
var photo1 = {  
  "src": "images/landscape-mountain.jpg",  
  "tags": ["black and white", "landscape", "mountains"]  
}  
  
var photo2 = {  
  "src": "images/lion.jpg",  
  "tags": ["africa", "animals", "lion"]  
}  
  
var photo3 = {  
  "src": "images/jennifer.jpg",  
  "tags": ["portrait", "people", "black and white"]  
}  
  
var photos = [photo1, photo2, photo3];
```



INSULT GENERATOR - PART 2