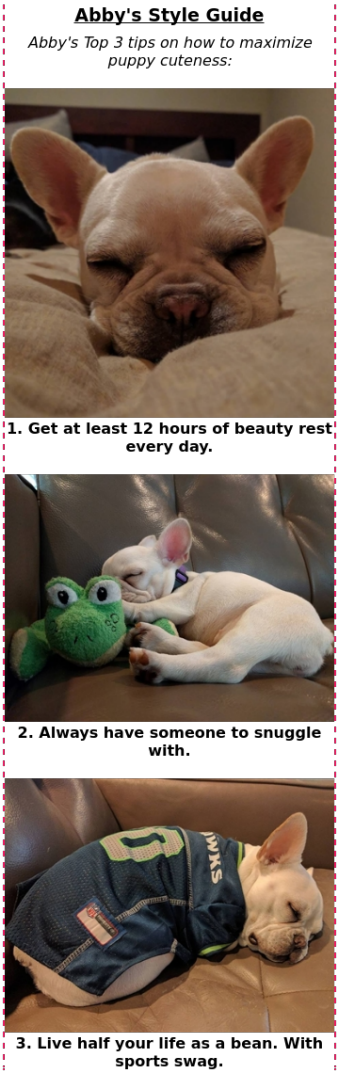
Lab 10: Practice Final

0. A Special Spec for a Special Doggy (HTML/CSS)

Write the HTML and CSS necessary to recreate the given expected appearance:

- All images of Abby are located in the same directory as your HTML and CSS files. The first image is located at sleepy-abby.png, the second image is located at abby-with-toy.png, and the third image is located at seahawks-abby.png
- Each image has 2px of margin on both the top and bottom edges.
- The page uses a font-family of Verdana, or sans-serif if Verdana is not available on a client's computer. All text is center-aligned.
- The body of the page takes up 50% of the overall page's width and is aligned horizontally in the middle of the page. This body section also has left and right borders that are 2px wide with a dashed pattern of the color #e91e63.
- The main heading of the page ("Abby's Style Guide") is underlined.
- Each of the three image and caption groups have a width of 350px, which the associated image should fully span in width. Each of these groups also has 20px of margin on the top and bottom edges.
- The subheading should have 12pt italicized font. The captions also have 12pt font, but are boldly-weighted.

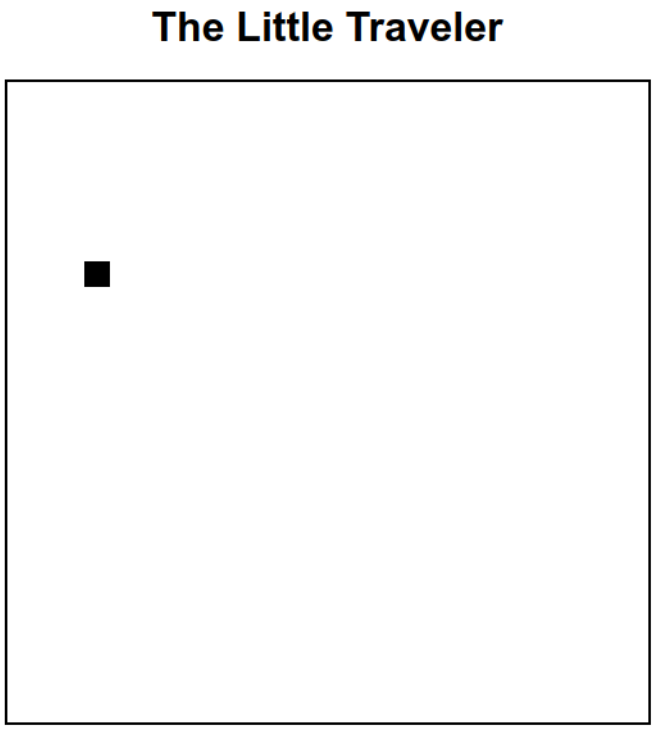


Expected output

1. The Little Traveler (JS)

Given the HTML and CSS on the following page, write the JavaScript code that adds a little-box div to the top left corner of the #box div, and moves the little box inside of the #box 20px up, down, left, or right randomly every 100ms. The litle box may only move to a position that is inside of the parent #boxes boundaries. Note that the #box parent has a width and height of 500px, and the little box has a width and height of 20px (without any border).

Below is a screenshot of the little box during an animation:



The relevant HTML and CSS are provided on the next page.

HTML and CSS for problem 1:

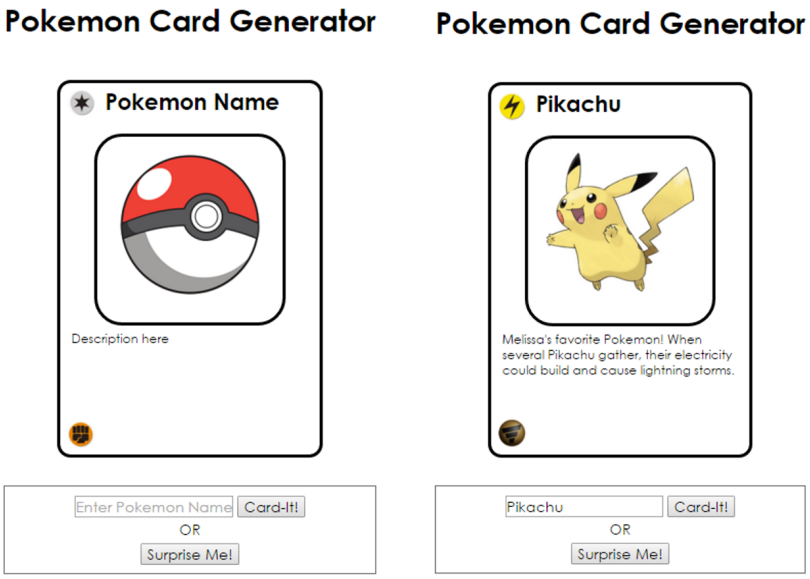
```
<!-- HTML -->
<!DOCTYPE html>
<html lang="en">
<head>
  <link href="traveler.css" type="text/css" rel="stylesheet" />
  <script src="traveler.js" type="text/javascript"></script>
</head>
<body>
  <h1>The Little Traveler</h1>
  <div id="box"></div>
</body>
</html>
```

```
/* CSS */
#box {
  border: 2px solid black;
  height: 500px;
  margin: auto auto;
  position: relative;
  width: 500px;
}

.little-box {
  background-color: #000;
  height: 20px;
  position: absolute;
  width: 20px;
}

h1 {
  font-family: Helvetica, Arial, sans-serif;
  text-align: center;
}
```

3. JavaScript with AJAX

Write a program called pokedex.js that adds behavior to an existing page called pokedex.html. pokedex.html is a page that lets a user generate a Pokemon card of any Pokemon they choose of the first 151 Pokemon, with the alternative option to get a random Pokemon card.

The image below on the left is the default state of the page, before a user has searched for a Pokemon. On the right is the state after a user has entered the Pokemon name Pikachu and clicked Card-It. This state may also be reached from Pikachu being a randomly-chosen card after the user clicks Surprise Me.



Left: default page state, Right: after a user has searched for Pikachu.

The center image (by default a Pokeball) is where a searched Pokemons photo will appear. The description below gives a description about that Pokemon, as you would normally find on Pokemon cards. The icon on the top left is the Pokemons type Pokemon may have types such as electric, fire, water, etc. The default type is normal, represented by a white star icon. The similar icon on the bottom left is the Pokemons weakness every Pokemon has a type they are weak to, and these are all types of other Pokemon.

You should implement functions in your JavaScript file to handle the following two possible actions using AJAX with a pokedex.php web service already created. The base url for both requests will be https://webster.cs.washington.edu/staff/medskm/exam/pokedex.php.

First, if the user enters a Pokemon name in the input box, you should make an AJAX GET request to the base url, adding the query parameter ?pokemon= followed by the Pokemon name input by the user.

Otherwise, if the Surprise Me button is clicked, you should make an AJAX request to the base url, adding the query parameter ?random=true.

Once the php web service handles the randomized selection, the JSON response format is the same for both requests, returning JSON for the selected Pokemon. For example, if a user input Pikachu in the input box, or if the Surprise Me action returned Pikachu as the random Pokemon, the JSON would be the following:

```
{
"name":"Pikachu",
"info": {
    "id":"25",
    "type":"electric",
    "weakness":"ground",
    "stage":"1",
    "description":"Melissa's favorite Pokemon! When several Pikachu gather, their electricity
                   could build and cause lightning storms."
    },
"images": {
    "photo":"img/pikachu.jpg",
    "typeIcon":"icons/electric.jpg",
    "weaknessIcon":"icons/ground.jpg"
    }
}
```

With this response, you should replace the card name with the name in the response (e.g., Pikachu), the type icon src with the returned typeIcon, the weakness icon src with the return weaknessIcon, the card image src with the returned Pokemon photo, and the card description with the returned description.

You may assume that users only enter valid pokemon to search for. You may also assume that each Pokemon has an associated photo in the /images folder and each possible icon type (fire, water, electric, etc.) is found in the /icons folder.

HTML for the page:

```
<html>
<head>
  <link type="text/css" href="pokedex.css" rel="stylesheet" />
  <script src="https://courses.cs.washington.edu/courses/cse154/17au/lecture/AjaxCheckStatusHelper.j
  <script  src="pokedex.js" type="text/javascript"></script>
</head>
<body>
  <div id="card">
   <div id="cardTop">
     <img id="pokeType" src="img/normal.jpg" />
        <span id="PokeName">Pokemon Name</span>
    </div>
    <img id="pokeImg" src="icon/pokeball.jpg" />
    <p id="pokeDescription">Description here</p>
    <img id="pokeWeakness"  src="icon/noweakness.jpg"/>
  </div>
  <div id="userInput">
    <input id="pokeNameInput" text="text" placeholder="Enter Pokemon Name" />
    <button id="cardItBtn">Card-It!</button>
    <br>
    OR
    <br>
    <button id="surpriseBtn">Surprise Me!</button>
  </div>
</body>
</html>
```

Write your js code here:

4. PHP Webservice (JSON)

You will now write the PHP webservice that the Pokedex problem above called.

Your PHP file will rely on a file called pokedex.txt. Each line of the file has information about a single pokemon (You may assume that all first 151 Pokemon are in the file and have complete information). The information is separated by colons in the following order:

Name:ID:Type:Weakness:Stage:Description

So, for example, the line for Bulbasaur looks like this:

Bulbasaur:1:Grass:Fire:1:Bulbasaur is an awesome Pokemon!

Your PHP should output pokemon information as shown in the previous problem (note that the image file names are lowercase versions of the information in the file, with the appropriate directory and a .jpg extension).

If a user has the "pokemon" GET field set (regardless of what else is set), you should find the information for that pokemon and output the JSON from the previous problem. If you can't find that pokemon, you should respond with a "400 Invalid Request" and the text "Pokemon X not found" where the X is what the user requested.

If a user has the "random" GET field set (and not the "pokemon" field), you should use the php function call "rand(1,151)" to pick a random line to choose in in the pokedex.txt file and output the JSON for that Pokemon.
If a user has neither the "pokemon" or "random" GET fields set, you should respond with a "400 Invalid Request" and the text "Please enter a pokemon or random parameter".
Write your PHP code here:

5. Regex

a. To div, or not to div that is the question. Write a regular expression to match an HTML (not <DIV>) element, including the opening div, the inner HTML, and the closing div. This expression should accept cases where the opening div has an id or class properties, but you should not accept any other properties (e.g., style). Any character (whitespace, letters, numbers, <, >, (,), etc.) should be accepted.

```
match:
<div></div>
<div>Hello world</div>
<div id=foo>bar baz</div>
<div id=divvy class=example>Such divness. Much style.</p></div>
```

b. a. Spelunking in Regex. Write a regular expression that matches words that start with 's' and end with 'ion' or 'ing', case-insensitive.

```
match:
station
situation
String
spelunking

dont match:
stations
alliteration
Spe1unking
```

6. SQL Joins

You are working with a database for the Q-Store ("We sell quilts, quartz and quinoa!"). The Q-Store has a database with the following tables:

Table 1: products

product_id	name	description	image	price
1	Quilt #1	This is a lovely quilt!	quilt1.jpg	$50
...	...	...	...	...

Table 2: customers

customer_id	name	address	email
7	Hip Stirr #1	small loft #7	TooCoolForEmail@email.com
...	...	...	...

Table 3: reviews

review_id	customer_id	product_id	review
12	7	1	This quilt came to life and ate my dog!
...	...	...	...

Table 4: orders

order_id	customer_id	product_id	order_date
57	7	3	October 3, 2008
...	...	...	...

We suspect the customer "Hip Stirr" is making up reviews for products he has not tried. We want to see which products he has reviewed and which ones he reviewed that he also purchased so we can compare those lists.

1) Write a SQL statement to list the names of all the products that this customer reviewed. Do not list the same product twice, and list the products in ascending alphabetical order.

1) Write a SQL statement to list the names of all the products that this customer has both ordered and reviewed. Do not list the same product twice, and list the products in ascending alphabetical order.

7. Short Answer

1. `onclick` is an example of an event handler in JavaScript. List two other event handlers.

   _____

2. Why is it important to specify multiple font styles for the same element in your CSS? (e.g., `font-family: Helvetica, Arial, sans-serif;`)

   _____

3. Where are cookies stored?

   _____

4. Where is session data stored?

   _____

5. Explain the basic idea of a SQL injection attack or an HtML/JavaScript injection attack.

   _____

6. Choose one type of disability and give on example for how a website can be made more accessible for users with that disability.

   _____