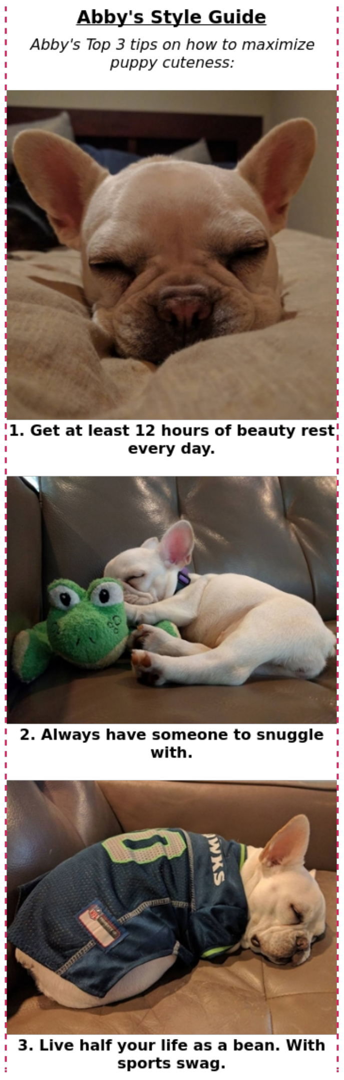
CSE 154: Web Programming Spring 2018

Section 10: Practice Final Solutions

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.



Abby's Style Guide

Abby's Top 3 tips on how to maximize puppy cuteness:

1. Get at least 12 hours of beauty rest every day.

2. Always have someone to snuggle with.

3. Live half your life as a bean. With sports swag.

Expected output

Solution:

```
<body>
  <h1>Abby's Style Guide</h1>
  <p>Abby's Top 3 tips on how to maximize puppy cuteness:</p>
  <div>
    <img src="sleepy-abby.png" />
    1. Get at least 12 hours of beauty rest every day.
  </div>

  <div>
    <img src="abby-with-toy.png" />
    2. Always have someone to snuggle with.
  </div>

  <div>
    <img src="seahawks-abby.png" />
    3. Live half your life as a bean. With sports swag.
  </div>
</body>

/* CSS */
body {
  font-family: Verdana, sans-serif;
  text-align: center;
  width: 50%;
  margin: auto auto;
  border-left: 2px dashed #e91e63;
  border-right: 2px dashed #e91e63;
}

div {
  width: 350px;
  font-size: 12pt;
  font-weight: bold;
  margin: 20px auto;
}

div img {
  display: block;
  margin: 2px auto;
  width: 350px;
}

h1 {
  text-decoration: underline;
}

p {
  font-style: italic;
  font-size: 12pt;
}
```

1. The Three Little Travelers (JS)

Solution:

(Runnable solution)

```
(function() {
  window.onload = function() {
    addBox();
    addBox();
    addBox();
  };

  function updateBox(box) {
    let sides = []; // top, right, down, left
    let topSide = parseInt(window.getComputedStyle(box).top);
    let leftSide = parseInt(window.getComputedStyle(box).left);
    if (topSide >= 20) {
      sides.push("top");
    }
    if (leftSide >= 20) {
      sides.push("left");
    }
    if (topSide <= 480) {
      sides.push("bottom");
    }
    if (leftSide <= 480) {
      sides.push("right");
    }
    let randomSideIndex = Math.floor(Math.random() * sides.length);
    let randomSide = sides[randomSideIndex];
    if (randomSide == "top") {
      box.style.top = topSide - 20 + "px";
    } else if (randomSide == "bottom") {
      box.style.top = topSide + 20 + "px";
    } else if (randomSide == "right") {
      box.style.left = leftSide + 20 + "px";
    } else if (randomSide == "left") { // left
      box.style.left = leftSide - 20 + "px";
    }
  }

  function addBox() {
    let littleBox = document.createElement("div");
    littleBox.classList.add("little-box");
    $("box").appendChild(littleBox);
    let randomInterval = parseInt(Math.random() * 500);
    setInterval(function() { updateBox(littleBox); }, randomInterval);
  }
})();
```

3. JavaScript with AJAX

Solution:

```
"use strict";
(function() {
  const URL = "https://webster.cs.washington.edu/staff/medskm/exam/pokedex.php";

  window.onload = function() {
    $("card-it-btn").onclick = newPokemon;
    $("surprise-btn").onclick = newPokemon;
  }

  function newPokemon() {
   let fetchURL = URL;
    if(this.id != "surprise-btn") {
      fetchURL += "?pokemon" + $("pokename-input").value;
    } else {
      fetchURL += "?random=true";
    }
    fetch(url)
      .then(checkStatus)
      .then(JSON.parse)
      .then(updateCard);
   }

   function updateCard(resultJSON) {
     $("pokename").innerHTML = resultJSON.name;
     $("poke-type").src = resultJSON.images.typeIcon;
     $("poke-weakness").src = resultJSON.images.weaknessIcon;
     $("poke-img").src = resultJSON.images.photo;
     $("poke-description").src = resultJSON.info.description;
   }

  function $(id) {
    return document.getElementById(id);
  }
})();
```

4. PHP Web Service (JSON)

Solution:

```
<?php
$all_pokemon = file("pokedex.txt");
if(isset($_GET["random"])) {
    $choice = $all_pokemon[mt_rand(0, count($all_pokemon) - 1)];
  $pokemon = explode(":", $choice);
} else if (isset($_GET["pokemon"])) {
  $pokemon = find_pokemon(strtolower($_GET["pokemon"]), $all_pokemon);
  if ($pokemon == NULL) {
    header("HTTP/1.1 400 Invalid Request");
    die("Pokemon " . $_GET["pokemon"] . " not found.");
  }
} else {
  header("HTTP/1.1 400 Invalid Request");
  die("Please enter name or random parameters");
}
output_json($pokemon);

#searches for passed pokemon in pokedex.txt file
function find_pokemon($pokemon, $all) {
  foreach ($all as $line) {
    $contents = explode(":", $line);
    if (strtolower($contents[0]) == $pokemon) {
      return $contents;
    }
  }
}

# outputs information for passed pokemon as JSON
function output_json($contents) {
  # Format  - Name:ID:Type:Weakness:Stage:Description
  # example - Bulbasaur:1:Grass:Fire:1:Description
  list($name, $id, $type, $weakness, $stage, $desc) = $contents;
  $json = array(
    "name" => $name,
    "info" => array("id" => $id, "type" => $type, "weakness" => $weakness,
            "stage" => $stage, "description" => $desc),
    "images" => array("photo" => "images/" . strtolower($name) . ".jpg",
              "typeIcon" => "icons/" . $type . ".jpg",
              "weaknessIcon" => "icons/" . $weakness . ".jpg")
  );

  header("Content-type: application/json");
  print(json_encode($json));
}
?>
```

5. Regex

a. It's camelCase, not CamelCase!

Write a regular expression to match camelCased strings containing only letters, with at least one letter capitalized.

Solution:

```
/^[a-z]+[A-Z]([a-zA-Z])*\$/
```

b. Caffeeeeeine!

Its finals week, and as true Seattlites, we need caffeine to survive it. Write a regular expression to match all Strings that contain the words "coffee", "mocha", or "green tea" containing no digits or special characters (space characters should be accepted). Both lower-case and upper-case letters are allowed for any letter in the String.

Solution:

```
/^[a-zA-Z\s]*(coffee|mocha|green tea)[a-zA-Z\s]*$/
```

c. To `div`, or not to `div`... that is the question.

Write a regular expression to match an HTML <div> (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 (strings of letters and/or numbers in "" quotes), but you should not accept any other properties (e.g., style). Your expression should not match if there is both a class and id. Any character (whitespace, letters, numbers, <, >, (,), etc.) should be accepted inside of the opening/closing `div` tags.

Solution:

```
/^<div(( class| id)=["]\S+["])?>.*<\/div>$/
```

6. SQL Joins

1) Write a query to find the full names of all countries where English is spoken as an official language. Do this as a single query.

Solution:

```
SELECT c.name
FROM countries c
JOIN languages l on l.country_code = c.code
WHERE l.language = 'English' AND l.official = 'T';
```

2) Write a query to find the full names of all countries that contain at least 2 cities of at least 5,000,000 people, as well as the country's gnp. Order your results by gnp (descending).

Solution:

```
-- Solution without JOIN keyword
SELECT DISTINCT c.name, c.gnp
FROM countries c, cities ci1, cities ci2
WHERE ci1.country_code = c.code AND ci2.country_code = c.code AND ci1.id <> ci2.id
AND ci1.population >= 5000000  AND ci2.population >= 5000000
ORDER BY c.gnp DESC;

-- Solution with JOIN keyword
SELECT DISTINCT c.name, c.gnp
FROM countries c
JOIN cities ci1 ON ci1.country_code = c.code
JOIN cities ci2 ON ci2.country_code = c.code
WHERE ci1.id <> ci2.id  AND ci1.population >= 5000000 AND ci2.population >= 5000000
ORDER BY c.gnp DESC;
```

3) Write a query to find all city names that correspond to at least two different cities in the same country. For example, there are at least two cities named "Portland" in the United States, so one row in result should correspond to the city name "Portland" and the country name "United States". Show both the country name and the city name. Eliminate duplicate results and order by the country name alphabetically, breaking ties by the city name.

Solution:

```
-- Solution without JOIN keyword
SELECT DISTINCT c.name, ci1.name
FROM countries c, cities ci1, cities ci2
WHERE c.code = ci1.country_code AND c.code = ci2.country_code
AND ci1.name = ci2.name AND ci1.id <> ci2.id
ORDER BY c.name, ci1.name;

-- Solution with JOIN keyword
SELECT DISTINCT c.name, ci1.name
FROM cities ci1
JOIN countries c ON c.code = ci1.country_code
JOIN cities ci2 ON ci2.country_code = ci1.country_code
WHERE ci1.name = ci2.name AND ci1.id <> ci2.id
ORDER BY c.name, ci1.name;
```

7. Short Answer

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

   ### Solution:

   `onchange`, `onload`, `onmouseover`, `onmouseout`, `ondblclick`

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;`)

   ### Solution:

   It's possible that the user does not have a font on their computer, so the next specified font can be used instead, defaulting to the last default font.

3. Where are cookies stored?

   ### Solution:

   Browser

4. Where is session data stored?

   ### Solution:

   Server

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

   ### Solution:

   SQL injection may involve malicious queries which manipulate or delete tables from the database.
   HTML/JS injection may insert HTML/JS where otherwise not desired (forms, inserting scripts on the page, etc.)

6. Provide one example of user input validation that would be best handled on the client-side (HTML or JS) and one example that would be best handled on the server-side (PHP).

   ### Solution:

   An example of validation that would be best handled on the client-side would be checking for a valid username format when creating a new username (e.g checking for a sufficient complexity). An example of validation that would be best handled on the serve would be checking for sensitive data such as a social security number or password sent through a POST request.

7. Consider the following JSON object:

```
let miniJSON = {
    "foo" : ["b", 1, 2],
    "bar" : 0,
    "FOO" : "Foo?"
};
```

For each of the following statements, write the value that would be returned (include `""` around any string values; if any expression would result in an error, write `error`. If any expression would return the value `undefined` or `null`, specify this as your answer.

Solution:

```
a. miniJSON.foo           : [b, 1, 2]
b. miniJSON["FOO"]        : "Foo?"
c. miniJSON["FOO"][1]     : "o"
d. miniJSON[foo]          : error
e. miniJSON["foo"].length : 3
```