CSS3 EXERCISES:

CSS3 EXERCISE 1: Create locations.css file for the rotating images in locations.html

For the image tag, please add:

- **`border-radius: 10px;`**: Rounds the corners of the image to a 10-pixel radius.
- **`box-shadow: 10px 10px 10px gray;`**: Adds a shadow effect to the image, offset by 10 pixels in all directions and colored gray.
- **`width: 400px;`**: Sets the initial width of the image to 400 pixels.
- **`max-width: 100%;`**: Ensures the image doesn't exceed its container's width, making it responsive.
- **`height: 300px;`**: Sets the height of the image to 300 pixels.
- **`display: block;`**: Makes the image a block-level element, allowing it to take up the full width of its container.
- **`margin: 0 auto;`**: Centers the image horizontally within its container.

CODE:
```
img {
  border-radius: 10px;
  box-shadow: 10px 10px 10px gray;
  width: 400px;
  max-width: 100%;
  height: 300px;
  display: block;
  margin: 0 auto;
}
```

Styling for links *directly* within nav tags:

- **`display: block;`**: Makes each navigation link a block-level element, allowing it to take up the full width of its container.
- **`margin: 0 auto;`**: Centers each navigation link horizontally within its container.

CODE:
```
nav > a {
  display: block;
  margin: 0 auto;
}
```

Styling main content:

- **`text-align: center;`**: Aligns the text within the main content area to the center.

```
main {
  text-align: center;
}
```

Styling caption text:

- **`color: #f2f2f2;`**: Sets the text color to a light gray.
- **`font-size: 1em;`**: Sets the font size to the default font size.
- **`padding: 8px 12px;`**: Adds padding to the top, bottom, left, and right of the text.
- **`position: absolute;`**: Positions the text absolutely within its container.
- **`bottom: 8px;`**: Positions the text 8 pixels from the bottom of its container.
- **`width: 100%;`**: Makes the text span the full width of its container.
- **`text-align: center;`**: Centers the text within its container.

```
.text {
  color: #f2f2f2;
  font-size: 1em;
  padding: 8px 12px;
  position: absolute;
  bottom: 8px;
  width: 100%;
  text-align: center;
}
```

Making empty span tags dots:

- **`height: 13px;`**: Sets the height of the dot to 13 pixels.
- **`width: 13px;`**: Sets the width of the dot to 13 pixels.
- **`margin: 0 2px;`**: Adds a 2-pixel margin to the left and right of each dot.
- **`background-color: #bbb;`**: Sets the background color of the dot to a light gray.
- **`border-radius: 50%;`**: Makes the dot circular.
- **`display: inline-block;`**: Makes each dot an inline-block element, allowing them to be displayed side-by-side.
- **`transition: background-color 0.6s ease;`**: Adds a smooth transition effect to the background color of the dots, which can be used for active/inactive states (we will see the effect of this later).

CODE:
```
.dot {
  height: 13px;
  width: 13px;
  margin: 0 2px;
  background-color: #bbb;
  border-radius: 50%;
  display: inline-block;
  transition: background-color 0.6s ease;
}
```

Add a zoom feature (in the solution file it is in style tags in locations.html):

CODE:
```
 .zoom-image {
            transition: transform 0.5s ease-in-out;
          }

 .zoom-image:hover {
            transform: scale(1.5);
          }
```

In locations.html you must add classes for `image-gallery`, `zoom-image` and `dot`.

CSS3 EXERCISE 2: For the contact.html page's CSS (can be in separate file or style tags in html page), we will add form styling for incorrect fields, and we will animate the submit button:

Basic styling for the form:

- **`margin: 20px;`**: Adds a 20-pixel margin around the form.
- **`padding: 20px;`**: Adds a 20-pixel padding inside the form.
- **`border: 1px solid #ccc;`**: Adds a 1-pixel solid border around the form.
- **`border-radius: 5px;`**: Rounds the corners of the form.

CODE:
```
form {
  margin: 20px;
  padding: 20px;
  border: 1px solid #ccc;
  border-radius: 5px;
}
```

Style the label and input tags:

For label:

- **`display: block;`**: Makes each label a block-level element, displaying it on a new line.
- **`margin-bottom: 5px;`**: Adds a 5-pixel margin below each label.

For input:
:

- **`width: 100%;`**: Makes the input field take up the full width of its container.
- **`padding: 10px;`**: Adds 10-pixel padding inside the input field.
- **`margin-bottom: 10px;`**: Adds a 10-pixel margin below each input field.
- **`border: 1px solid #ccc;`**: Adds a 1-pixel solid border around the input field.
- **`border-radius: 5px;`**: Rounds the corners of the input field.

CODE:

```
label {
  display: block;
  margin-bottom: 5px;
}

input {
  width: 100%;
  padding: 10px;
  margin-bottom: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
}
```

Style for input validation:

- **`input:invalid`**: Styles invalid input fields with a red border.
- **`input:invalid:focus`**: Removes the default outline and sets the border color to blue when an invalid input field is focused.

CODE:
```
input:invalid {
  border-color: red;
}

input:invalid:focus {
  outline: none;
  border-color: blue;
}
```

Let's animate the button, first with a transition and then with a full animation. This includes:

Base button styles:

- **Basic Styling:** Sets the button's color, background, padding, border-radius, font weight, and text transform.
- **Transition:** Enables smooth transitions for properties that change on hover.
- **Position and Overflow:** These properties are crucial for the glow effect. `position: relative` allows the glow to be positioned absolutely within the button, while `overflow: hidden` ensures the glow is clipped to the button's boundaries.

Hover effects:

- **Color and Shadow:** Darkens the text color and adds a subtle box shadow to create a 3D effect.
- **Glow Effect:**
  - **Pseudo-element:** A pseudo-element `::after` is added to the button, positioned absolutely over it.
  - **Gradient Background:** A radial gradient is used to create a circular glow effect.
  - **Opacity and Transition:** The glow is initially hidden and then gradually fades in on hover.
  - **Animation:** The `glow` animation is applied to the pseudo-element, making it pulsate and change opacity over time.

Keyframes animation:

The `@keyframes glow` rule defines the animation's keyframes:

- **0%:** The glow is partially transparent and at its original size.
- **50%:** The glow becomes more opaque and slightly larger.
- **100%:** The glow returns to its initial state.

CODE:

```
 /* Button animation */

button {

  text-decoration: none;

  color: rgba(255, 255, 255, 0.8);

  background: rgb(145, 92, 182);

  padding: 15px 40px;

  border-radius: 4px;

  font-weight: normal;

  text-transform: uppercase;

  transition: all 0.2s ease-in-out;

  position: relative; /* Necessary for absolute positioning
of the glow effect */

  overflow: hidden; /* Necessary for clipping the glow
effect */

}


button:hover {

  color: rgba(255, 255, 255, 1);

  box-shadow: 0 5px 15px rgba(145, 92, 182, .4);

}


button:hover::after {

  content: "";
  position: absolute;

  top: 0;

  left: 0;

  width: 100%;

  height: 100%;

  background: radial-gradient(circle at 50% 50%, rgba(255,
255, 255, 0.3) 0%, rgba(255, 255, 255, 0) 70%);

  opacity: 0;

  transition: opacity 0.5s ease-in-out;

}


button:hover::after {

  opacity: 1;

  animation: glow 1s infinite alternate;

}


@keyframes glow {

  0% {

    opacity: 0.2;

    transform: scale(1);

  }

  50% {

    opacity: 0.8;

    transform: scale(1.1);
  }
  100% {
    opacity: 0.2;
    transform: scale(1);
  }
}
```

Save and test the contact.html page by hovering over the button and checking invalid data.

CSS3 EXERCISE 3: RESPONSIVE WEB DESIGN EXERCISE:

Add this to the style.css page. Start with LABstyle.css and change the name. Make sure it is being used by index.html. Out goal is to apply specific styles based on different screen sizes. In this case, it targets screens with a maximum width of 768 pixels, which is often associated with smaller devices like tablets and smartphones. Test by making screen size smaller.

Adjust padding. This reduces the padding on the `header`, `main`, and `footer` elements to 1rem. This can help to reduce the amount of space taken up by these elements on smaller screens, making the layout more compact.

```
header, main, footer {
  padding: 1rem;
}
```

Adjust the aside. Remove the `float` property, which is often used for sidebars. This prevents the `aside` element from floating next to the main content on smaller screens. Set the `width` to 100%, making the `aside` element take up the full width of the screen. Add a bottom margin of 2rem to separate the `aside` from the following content.

```
aside {
  float: none;
  width: 100%;
  margin-bottom: 2rem;
}
```

Adjust navigation. Change the `flex-direction` of the navigation list to `column`, which stacks the navigation items vertically. Remove the right margin from the navigation list items. Add a bottom margin of 1rem to separate the navigation items.

```
nav ul {

  flex-direction: column;
}

nav li {
  margin-bottom: 1rem;
  margin-right: 0;
}
```

CODE:

```
/* Media Queries */

@media (max-width: 768px) {

/* Adjust layout for smaller screens */

header, main, footer {

padding: 1rem;

}


aside {

float: none;

width: 100%;

margin-bottom: 2rem;

}
nav ul {
flex-direction: column;
}

nav li {
margin-bottom: 1rem;
margin-right: 0;
}
}
```

CSS3 EXERCISE 4: Convert styles.css to SASS/SCSS as much as possible.