HW5 - Components and Beyond

Dennis Lang - Computer Engineering - UCSD Class of 2024

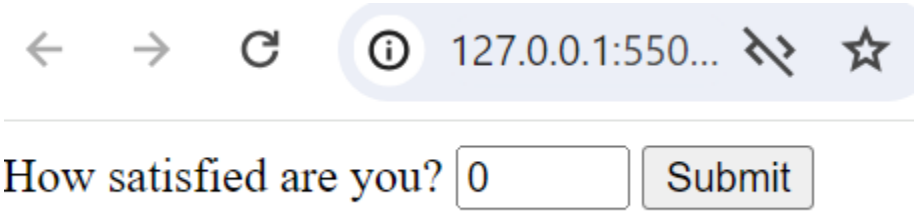
A11397951

CSE 134: Professor Thomas A. Powell

https://github.com/dlang5/cse134-hw5

Rating Widget

No JavaScript



rating.html

demo: https://dlang5.github.io/cse134-hw5/rating.html

code: https://github.com/dlang5/cse134-hw5/blob/main/rating.html

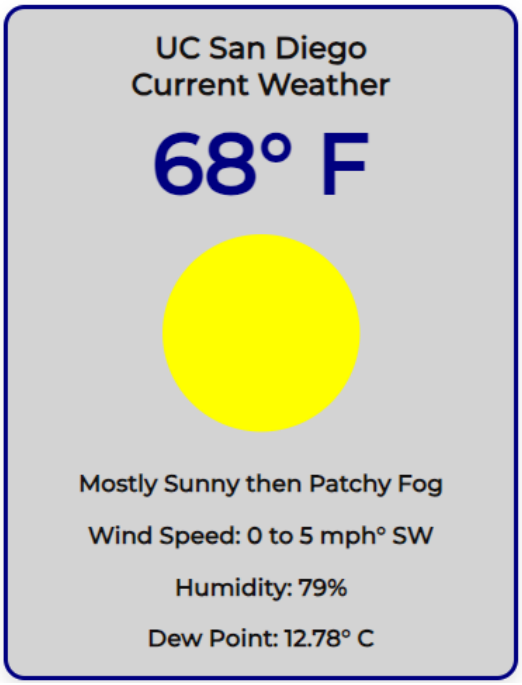
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
    :root {
      --empty-star-color: gray;
      --star-color: blue;
    }

    .star {
      font-size: 2em;
      color: var(--empty-star-color);
      cursor: pointer;
    }

    .star:hover {
      color: var(--star-color);
    }

    #rating-input {
      display: none;
    }

    #message {
      margin-top: 10px;
      font-weight: bold;
    }

    #rating-form {
      display: none;
    }
```

```
    #star-form {
      display: block;
    }
  </style>
</head>
<body>

<rating-widget>
  <noscript>
    <style>
      #rating-form {
        display: block;
      }
      #star-form {
        display: none;
      }
      #color-palette {
        display: none;
      }
    </style>
  </noscript>
  <form action="https://httpbin.org/post" method="POST" id="star-form">
    <label for="rating">How satisfied are you?</label>
    <div>
      <span class="star" data-rating="1">★</span>
      <span class="star" data-rating="2">★</span>
      <span class="star" data-rating="3">★</span>
      <span class="star" data-rating="4">★</span>
      <span class="star" data-rating="5">★</span>
    </div>
    <input type="hidden" id="rating-input" name="rating" value="0"
required>
  </form>

  <form action="https://httpbin.org/post" method="POST" id="rating-form">
    <label for="rating">How satisfied are you?</label>
    <input type="hidden" name="question" value="How satisfied are you?">
    <input type="hidden" name="sentBy" value="HTML">
    <input type="number" id="rating" name="rating" min="1" max="5"
value="0" required>
```

```
    <button type="submit">Submit</button>
  </form>
</rating-widget>

<p id="color-palette">
  <label for="star-color-picker" class="color-picker">Empty
Color: </label>
  <input type="color" id="star-color-picker" onchange="changeStarColor()"
value="#808080">
  <br>
  <label for="star-hover-color-picker" class="color-picker">Star
Color: </label>
  <input type="color" id="star-hover-color-picker"
onchange="changeHoverColor()" value="#0000FF">
</p>

<p id="message"></p>

<script>
  const stars = document.querySelectorAll('.star');
  const ratingInput = document.getElementById('rating-input');
  const messageContainer = document.getElementById('message');
  const starColorPicker = document.getElementById('star-color-picker');
  const hoverColorPicker =
document.getElementById('star-hover-color-picker');

  function changeStarColor() {
    document.documentElement.style.setProperty('--empty-star-color',
starColorPicker.value);
  }

  function changeHoverColor() {
    document.documentElement.style.setProperty('--star-color',
hoverColorPicker.value);
  }

  stars.forEach(star => {
    star.addEventListener('mouseover', () => {
      const rating = parseInt(star.getAttribute('data-rating'));
      stars.forEach(s => {
```

```
        if (parseInt(s.getAttribute('data-rating')) <= rating) {
          s.style.color = 'var(--star-color)';
        } else {
          s.style.color = 'var(--empty-star-color)';
        }
      });
    });

    star.addEventListener('mouseout', () => {
      const selectedRating = parseInt(ratingInput.value);
      stars.forEach(s => {
        if (parseInt(s.getAttribute('data-rating')) <= selectedRating) {
          s.style.color = 'var(--star-color)';
        } else {
          s.style.color = 'var(--empty-star-color)';
        }
      });
    });

    star.addEventListener('click', () => {
      const rating = parseInt(star.getAttribute('data-rating'));
      ratingInput.value = rating;
      console.log(`Rating: ${rating}`);

      let message = `Thanks for the ${rating} star rating.`;
      if (rating >= 4) {
        message += ' We appreciate it!';
      }
      if (rating <= 2) {
        message += ' We\'ll try better.';
      }

      messageContainer.textContent = message;
    });
  });
</script>

</body>
</html>
```

Weather Widget



weather.gov fetching

Name	Status	Type	Initiator	Size	T...	Waterfall
weather.html	200	document		7.0 kB	3...	
css2?family=Lato:wght@700&family=Mont...	200	stylesheet	weather.html:59	780 B	7...	
JTUHjIg1_i6t8kCHKm4532VJOt5-QNFgpCtZ...	200	font	css2	15.1 kB	5...	
ws	101	websocket	weather.html:177	0 B	P...	
32.8801,-117.2340	301	fetch / Redirect	weather.html:98	879 B	3...	
32.8801,-117.234	200	fetch	32.8801,-117.2340	1.2 kB	3...	
forecast	200	fetch	weather.html:103	1.9 kB	4...	

I used the National Weather Service's API with UCSD's latitude and longitude in order to display the forecast for the current time of day by accessing a second JSON and I also included the shortDescription, windSpeed and windDirection, humidity.value, and dewpoint.value in order to display more detailed information.

Two simple .svg images of clouds and rain I found online (credit freesvg.org) plus two .svg formulas, one yellow circle for sunny and one blue circle for a clear night were my icons.

weather.html

demo: https://dlang5.github.io/cse134-hw5/weather.html

code: https://github.com/dlang5/cse134-hw5/blob/main/weather.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>

@import
url('https://fonts.googleapis.com/css2?family=Lato:wght@700&family=Montser
rat:wght@500&family=Roboto+Slab&display=swap');

h1 {
  font-size: 0.9em;
  margin-bottom: 0.3em;
  text-align: center;
  font-weight: 900;
}

h2 {
  font-size: 2.5em;
  margin-bottom: 0em;
  margin-top: 0em;
  color: #000080;
  text-decoration-color: #000080;
}

#container {
  display: flex;
  width: 15em;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  background-color: lightgray;
  border-radius: 10px;
```

```
  border: 2px solid #000080;
  font-family: 'Montserrat', sans-serif;
  color: #121010;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}


#pic {
  width: 80%;
  max-width: 6em;
  height: auto;
}

h3 {
  margin-top: 0em;
  margin-bottom: .2em;
  text-align: center;
}

h4 {
  font-size: 0.7em;
  margin-top: 0em;
  margin-bottom: 1em;
  align-items: center;
  justify-content: center;
}

    </style>
    <title>UCSD Weather</title>
</head>
<body>

    <div id="container">
      <h1>UC San Diego<br> Current Weather</h1>
      <h2 id="weather-info"></h2>
      <h3 id="pic"></h3>
      <h4 id="description"></h4>
      <h4 id="wind"></h4>
      <h4 id="humidity"></h4>
      <h4 id="dewpoint"></h4>
```

```
      <noscript>
        <h4>Current Weather Conditions Unavailable.</h4>
      </noscript>

    </div>

    <script>
        const apiUrl = "https://api.weather.gov/points/32.8801,-117.2340";

        const weatherIcons = [
  // Yellow circle
  '<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24"
fill="yellow"><circle cx="12" cy="12" r="10"/></svg>',

  // Light blue circle
  '<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24"
fill="lightBlue"><circle cx="12" cy="12" r="10"/></svg>',

  '<object type="image/svg+xml" data="cloud.svg" width="100%"
height="100%"></object>',

  '<object type="image/svg+xml" data="rain.svg" width="100%"
height="100%"></object>',

  '<object type="image/svg+xml" data="UCSD.svg" width="100%"
height="100%"></object>'
];

        async function getWeatherData() {
            try {
                const response = await fetch(apiUrl);
                const data = await response.json();

                const forecastUrl = data.properties.forecast;

                const forecastResponse = await fetch(forecastUrl);
                const forecastData = await forecastResponse.json();

                const period1 =
forecastData.properties.periods.find(period => period.number === 1);
```

```
                if (period1) {
                    const temperature = period1.temperature;
                    const temperatureUnit = period1.temperatureUnit;
                    const description = period1.shortForecast;
                    const icon = period1.icon;
                    const windDirection = period1.windDirection;
                    const windSpeed = period1.windSpeed;
                    const humidity = period1.relativeHumidity.value;
                    var dewPoint = period1.dewpoint.value.toFixed(2);

                    document.getElementById("weather-info").textContent =
`${temperature}° ${temperatureUnit}`;
                    document.getElementById("description").textContent =
`${description}`;
                    document.getElementById("wind").textContent = `Wind
Speed: ${windSpeed}° ${windDirection}`;
                    document.getElementById("humidity").textContent =
`Humidity: ${humidity}%`;
                    document.getElementById("dewpoint").textContent = `Dew
Point: ${dewPoint}° C`;

                    const weatherIconContainer =
document.getElementById("pic");
                    const description2 = description.toLowerCase();

                    switch (true) {
                      case description2.includes("sunny"):
                        weatherIconContainer.innerHTML = weatherIcons[0];
                        break;
                      case description2.includes("clear"):
                        weatherIconContainer.innerHTML = weatherIcons[1];
                        break;
                      case description2.includes("cloudy"):
                        weatherIconContainer.innerHTML = weatherIcons[2];
                        break;
                      case description2.includes("rain"):
                        weatherIconContainer.innerHTML = weatherIcons[3];
                        break;
                      default:
```

```
                        weatherIconContainer.innerHTML = weatherIcons[4];
                        break;
                    }

                } else {
                    console.error("Period 1 not found in the forecast
data.");
                }
            } catch (error) {
                console.error("Error fetching weather data:", error);
            }
        }

        window.onload = getWeatherData;
    </script>
</body>
</html>
```

Extra Credit

rating-react.html

https://dlang5.github.io/cse134-hw5/rating-react.html

weather-react.html

https://dlang5.github.io/cse134-hw5/weather-react.html

Converting the pages to React gave me an insight on how to use a component-based approach to frontend development. As I never used a JavaScript library before, I can see how people can get used to not relying on old-school JS methods to standardize scripting languages. The code is a little confusing at first, but after designing the second component, it appears to be a lot more intuitive over the long run.

Astro Site Walkthrough

https://youtu.be/c9GQuAHaBro