

Web 330 HTML, CSS, and JavaScript Requirements

HTML and CSS Requirements

All assignments in this course will be reusing the styles we created in WEB 200 and used in WEB 231. This means you will need to copy the following CSS classes from WEB 200/231's repository

card, card-title, card-content, card: hover, btn, btn-primary, btn-primary: hover, input, drop-down-menu, app-header, return-home, return-home: hover, return-home: visited, form, form-field, full-width

These styles will be split between two files: site.css and theme.css, which will be placed at the root of your web-330 directory.

CSS styles placed in **site.css**

app-header, return-home, return-home: hover, return-home: visited, form, form-field, full-width

CSS styles placed in **theme.css**

card, card-title, card-content, card: hover, btn, btn-primary, btn-primary: hover, input, drop-down-menu

Exhibit A. global.css

If you do not remember how to use the classes, please refer back to the work you completed in WEB 200 and 231. Also, for your reference, I have added example files in Web 330's GitHub repository, under week-0.

Exhibit B. HTML assignment structure (visual)

Example of how to include multiple cards in an assignment.

Card with container example

This is an example of how to reuse the CSS classes: assign-container, assign-content, card, card-title, and card-content

Program Results

The expected results will be displayed in this section

[Return](#)

Exhibit C. HTML assignment structure (code)

```

<body class="light-theme">
  <!-- container wrapper for 1 card -->
  <div class="assign-container">
    <h1 class="app-header">Example of how to include multiple cards in an assignment.</h1>
    <!-- container wrapper for a card -->
    <div class="assign-content">
      <!-- card with a card title and card content -->
      <div class="card">
        <div class="card-title">
          Card with container example
        </div>
        <div class="card-content">
          This is an example of how to reuse the CSS classes: assign-container, assign-content, card, card-title, and card-content
        </div>
      </div>
      <!-- end card -->
    </div>
    <!-- end assign-content -->
  </div>
  <!-- end assign-container -->

  <!-- container wrapper for a card -->
  <div class="assign-container">
    <!-- container wrapper for a card -->
    <div class="assign-content">
      <!-- card with a card title, card content, and assign-results-text -->
      <div class="card">
        <div class="card-title">
          Program Results
        </div>
        <div class="card-content assign-results-text">
          The expected results will be displayed in this section
        </div>
      </div>
      <div class="card-content assign-results-text">
        The expected results will be displayed in this section
      </div>
    </div>
    <!-- end card -->
  </div>
  <!-- end assign-content -->
</div>
<!-- end assign-container -->
</body>

```

HTML assignment structure requirements

1. All HTML pages you create in this course will need the CSS class “light-theme” added to the HTML body element (see Exhibit C, item 1).
2. All HTML pages you create in this course must include a page title (see Exhibit C, item 3). The title should be an h1 header with a CSS class of app-header.
3. All CSS cards you create in this course must be wrapped in two containers: assign-container and assign-content (see Exhibit C, items 2 and 4) and must include a card title and card content (see Exhibit C, items 5, 6, and 7).
4. Most of the output messages you create in this course must use the CSS class assign-results-text. This class will be assigned to the bottom card-content section (see Exhibit C, item 8).

Exhibit D. CSS container and card container styling requirements

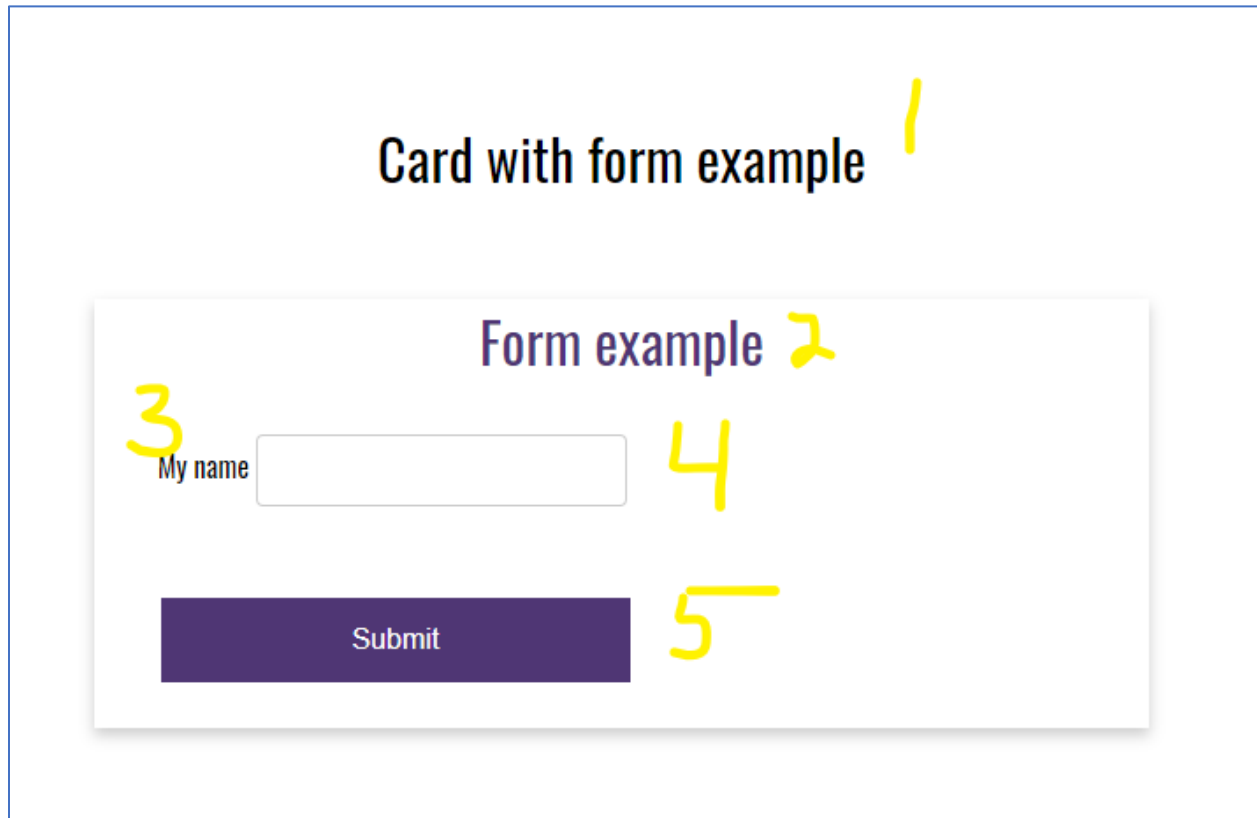
By now you should feel comfortable with creating custom CSS styles and the power of reusing CSS classes. In Web 231 we manually created containers for each of the weekly assignments. For this course, we will create three global classes to promote code reusability. The first class you will create is named **assign-container**. Give this class a padding of 10 pixels. The second class you will create is called **assign-content**. Give this class a width of 600 pixels and a margin of **3% auto 0**. The third class you will create is called **assign-results-text**. Center the text and give this class a font size of 28 pixels. Add these styles to the site.css file.

Exhibit D. site.css Assignment Layout Styling

```
/*
 * Page container. This should be placed in the body of each HTML page you create.
 */
.assign-container {
  padding: 10px;
}

/*
 * Box to hold the content for the assignment's in this course.
 */
.assign-content {
  width: 600px;
  margin: 3% auto 0;
}

/*
 * Box title for the results program output section.
 */
.assign-results-text {
  text-align: center;
  font-size: 28px;
}
```

Exhibit E. HTML cards with a form and form-fields

Card with form example 1

Form example 2

3 My name 4

5 Submit

1. app-header
2. card-title
3. HTML label. All labels must include a “for” directive that references the HTML input element
4. HTML input element. All input fields must include an id attribute that starts with the prefix of **txt**, a CSS class of **.input**, and the name directive that matches the id attribute.
5. HTML button. All buttons must include an id attribution that starts with the prefix **btn** and CSS classes for “**btn**, **btn-primary** full-width”

Exhibit F. HTML code for cards with nested form and form-fields

```
<div class="assign-container">
  <h1 class="app-header">Card with form example</h1>

  <div class="assign-content">
    <div class="card">
      <div class="card-title">Form example</div>
      <div class="card-content">
        <div class="form">
          <div class="form-field">
            <label for="txtMyName">My name:</label>
            <input type="text" class="input" id="txtMyName" name="myName" />
          </div>

          <div class="form-field">
            <button class="btn btn-primary full-width">Submit</button>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

1. CSS form div.
2. First form-field, which represents the label and input field from Exhibit E.
3. HTML label with the accepted naming conventions for the “for” attribute.
4. HTML input with the accepted naming conventions for the id and name attribute. Notice how the id and name values match the “for” in the label field on item 3.
5. Second form-field, which represents the container for the submit button.
6. HTML button with CSS classes for btn, btn-primary, and full-width. If this button were interactive (i.e., we incorporated JavaScript), there must be a defined id attribute of btn<name>). For example, <button class=”btn btn-primary full-width” id=”btnSubmit”>Submit</button>

Exhibit H. HTML formatting for a form with form results

Form with form results example

Form example

My name:

Submit

Form Results

Results from the form submission will go here...

Exhibit I. HTML code for a form with form results

```

<div class="assign-container">
  <h1 class="app-header">Form with form results example</h1>

  <div class="assign-content">
    <div class="card">
      <div class="card-title">Form example</div>
      <div class="card-content">
        <div class="form">
          <div class="form-field">
            <label for="txtMyName">My name:</label>
            <input type="text" class="input" id="txtMyName" name="myName" />
          </div>

          <div class="form-field">
            <button class="btn btn-primary full-width">Submit</button>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>

<div class="assign-container">
  <div class="assign-content">
    <div class="card">
      <div class="card-title">Form Results</div>
      <div class="card-content assign-results-text">
        Results from the form submission will go here...
      </div>
    </div>
  </div>
</div>

```

Google font kit and linking CSS files

Unless otherwise specified, all assignments in this course will use the Google font kit we added in Assignment 1.3 – Environment Setup. The theme.css, site.css, and Google font kit will need to be added to each weekly assignment. For example, let's say I am working on Assignment 2.2 and the assignment asks for me to create an HTML file named <yourLastName>-palindrome.html I will need to create a least two files (<yourLastName>-palindrome.html and <yourLastName>-palindrome.css) and add links in the head of the HTML file for the theme.css, site.css, current week's CSS file, and the courses Google font kit. Pay close attention to the order in Exhibit K. Stylesheets should be placed in the order of how they are applied to the pages. Otherwise, you may run into styling issues (i.e., certain styles not being applied correctly).

Exhibit K. HTML CSS references

```
<link rel="stylesheet" type="text/css" href="../../theme.css" />
<link rel="stylesheet" type="text/css" href="../../site.css" />
<link rel="stylesheet" type="text/css" href="palindrome.css" />

<link href="https://fonts.googleapis.com/css2?family=Oswald:wght@300;400;500;700&display=swap" rel="stylesheet">
```

JavaScript requirements

All JavaScript code will be placed inside a <script> tag in the weekly HTML document. For example, let's say I am working on Assignment 2.2 and I am asked to add JavaScript functionality to the HTML code. The JavaScript portion of the code will be placed above the closing </body> tag of the HTML page.

Exhibit L. JavaScript <script> tag

```
<body class="light-theme">
  <div class="assign-container">
    <h1 class="app-header">Example of how to add JavaScript to an HTML tag.</h1>
    <div class="assign-content">
      <div class="card">
        <div class="card-title">
          Card Header
        </div>
        <div class="card-content assign-results-text" id="results">
        </div>
      </div>
    </div>
  </div>
  <script>
    alert("Hello World!");
  </script>
</body>
```

Additional JavaScript coding instructions

In order to bind data to an HTML element in JavaScript, we use the `document.getElementById` function to locate the HTML element. As the name suggests, this function scans the document for matching elements with the referenced ID. Once the element is located, we assign values to the div. To this we leverage the `innerHTML` property of the `getElementById` function. For example, `document.getElementById("myName").innerHTML = "Professor Krasso";` This basically says, find an HTML div with an id of "myName" and assign the `innerHTML` a value of "Professor Krasso." See Exhibit M. for a live example.

Exhibit M. JavaScript data binding

```
<div class="assign-container">
  <h1 class="app-header">Example of how to add JavaScript to an HTML tag.</h1>
  <div class="assign-content">
    <div class="card">
      <div class="card-title">
        Card title
      </div>
      <div class="card-content assign-results-text" id="data-binding-example">
      </div>
    </div>
  </div>
</div>
<script>
  document.getElementById("data-binding-example").innerHTML = "Learning JavaScript is fun!";
</script>
```

Handwritten yellow numbers 1 through 6 are present in the image, marking specific parts of the code: 1 is next to the `id="data-binding-example"` attribute, 2 is next to the `<div class="card-content"` opening tag, 3 is next to the `</div>` closing tag for the `assign-content` div, 4 is next to the `document.getElementById` function, 5 is next to the `innerHTML` property, and 6 is next to the `"Learning JavaScript is fun!"` string value.

1. Assign an id to the div we want to find/bind data to.
2. Section of HTML where we want the data to appear.
3. JavaScript document.getElementById reference.
4. Tell the JavaScript function where to look.
5. Tell the JavaScript function to write the values to section 2.
6. The message to display in section 2.

document.getElementById("id").value

Similar to the example in Exhibit M, document.getElementById("id").value searches the HTML code for the appropriate HTML, but this time it retrieves the value.

Exhibit N. JavaScript get the value from an HTML element

```
<div class="assign-container">
  <h1 class="app-header">Example of how to use JavaScript in a web page.</h1>
  <div class="assign-content">
    <div class="card">
      <div class="card-title">Get Value Form</div>
      <div class="card-content">
        <label for="txtPayRate">Pay Rate:</label>
        <input type="text" id="txtPayRate" name="txtPayRate" class="input" value="25.99">
      </div>
    </div>
  </div>
</div>

<script>
  let payRate = document.getElementById("txtPayRate").value;
  alert('Your pay rate is: ${payRate}')
</script>
```

1. Define a label for the pay rate input field.
2. An HTML input field with a value of 25.99.
3. A variable to hold the pay rate value.
4. Ask JavaScript to get the value of the txtPayRate input field.
5. Alert the value.

document.getElementById("id").onclick = function() {}

Similar to examples in Exhibit M and N, if you want JavaScript to respond to click events, you must tell JavaScript which button to create an event listener for.

Exhibit O. onclick example

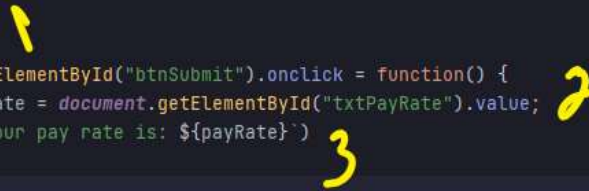
```
<div class="assign-container">
  <h1 class="app-header">Example of how to use JavaScript in a web page.</h1>

  <div class="assign-content">
    <div class="card">
      <div class="card-title">Get Value Form</div>
      <div class="card-content">

        <div class="form">
          <div class="form-field">
            <label for="txtPayRate">Pay Rate</label>
            <input type="text" id="txtPayRate" name="txtPayRate" class="input" value="25.99">
          </div>

          <div class="form-field">
            <button type="button" class="btn btn-primary full-width" id="btnSubmit">Submit</button>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>

<script>
  document.getElementById("btnSubmit").onclick = function() {
    let payRate = document.getElementById("txtPayRate").value;
    alert('Your pay rate is: ${payRate}')
  }
</script>
```



1. Use the `document.getElementById` function to register an `onclick` event for the HTML button.
2. Get the value of the pay rate input field.
3. Output the pay rate using JavaScript's alert box.

Theme propagation

Once the user has selected their desired theme and our code has set their preferences in the browser's local storage, we will need to access this information to propagate it to the weekly assignments. To do this, you will need to call the `localStorage.getItem()` function and assign the results to a variable. Next, you will need to call `document.body.classList.value` and assign it the

variable. To make our lives easier and to follow DRY (Don't Repeat Yourself) principles, we will add a JavaScript file to the courses root directory named `theme.js`. Inside this file you will create two functions: `setDefaultTheme()` and `setSelectedTheme()`. The code for `setDefaultTheme()` will be created in Assignment 1.3. The code to set the selected theme is shown in Exhibit P.

Exhibit P. Code to set the users selected theme

```
function setSelectedTheme()  
{  
    document.body.classList.value = localStorage.getItem( key: "mode") || "light-theme";  
}
```

Once the function is added, you will need to reference this file in all of the assignment's you create. Make sure the reference is placed at the `<head>` of the HTML document. See Exhibit Q.

Exhibit Q. `theme.js` HTML reference

```
<script type="application/javascript" src="../../theme.js"></script>
```

To use this function, simply call `setSelectedTheme()`. Be sure to place this above all other JavaScript code. See Exhibit R.

Exhibit R. Call to `setSelectedTheme()` function

```
<script>  
    setSelectedTheme();  
</script>
```