Web 231 HTML, CSS, and JavaScript Requirements

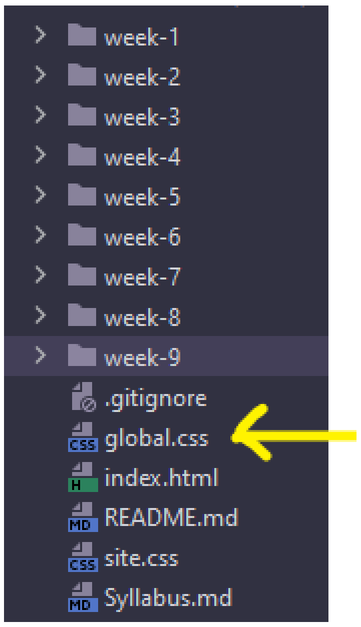
HTML and CSS Requirements

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

> .card, .card-title, .card-content, .card:hover, .btn, .btn-primary, .btn-primary:hover, .input, .full-width, .form, and .form-field.

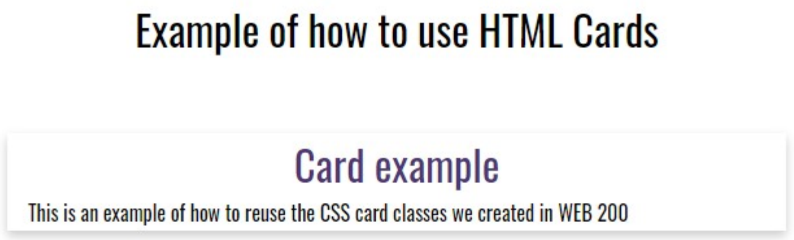
These files will be placed in a CSS file called **global.css**, which will be placed at the root of your web-231 directory. Also, remove the 50% width style from the CSS card class.

Exhibit A. global.css



If you do not remember how to use the classes, please refer back to WEB 200's repository, specifically the last three weeks of assignments. Those assignments provide examples of how to create multiple card's with titles, content, and forms. Having said that, the remaining assignments will not specifically ask you to "create a card div with a title, content, or form-fields," because the assumption is you understand how those CSS classes work and the fundamentals of HTML and CSS programming. Remember, this is a JavaScript course not an HTML and CSS course. For week's where new styling is added, you will need to reuse the existing styles to make the new elements match the Exhibits output.

Exhibit B. HTML card with card title and card content

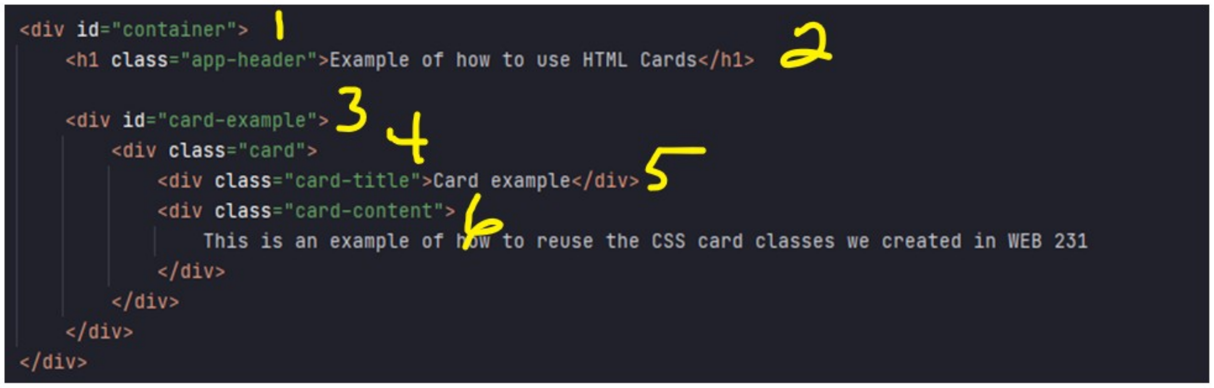


Example of how to use HTML Cards

Card example

This is an example of how to reuse the CSS card classes we created in WEB 200

Exhibit C. HTML page container and wrapper for cards



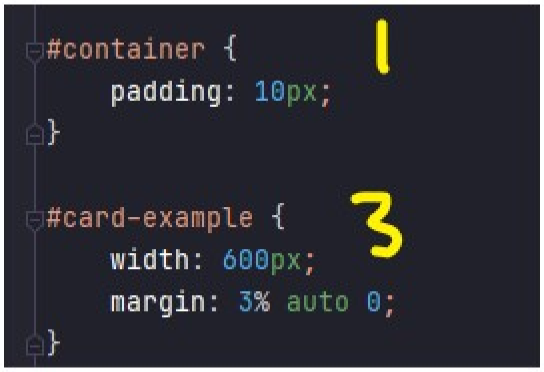
```
<div id="container"> 1
    <h1 class="app-header">Example of how to use HTML Cards</h1> 2

    <div id="card-example">3 4
        <div class="card">
            <div class="card-title">Card example</div>5
            <div class="card-content">6
                This is an example of how to reuse the CSS card classes we created in WEB 231
            </div>
        </div>
    </div>
</div>
```

HTML page and card requirements

1. All HTML pages you create in this course must be wrapped in a container div (see Exhibit C, item 1).

2. All HTML pages must include a h1 page title with a CSS class of app-header (see Exhibit C, item 2).
3. All CSS cards you create in this course must be wrapped in a card container (see Exhibit C, item 3).
4. All CSS cards you create in this course must be mapped to a div (see Exhibit C, item 4).
5. All CSS cards you create in this course must include a div for the card-title and card-content (see Exhibit C, item 5 and 6).

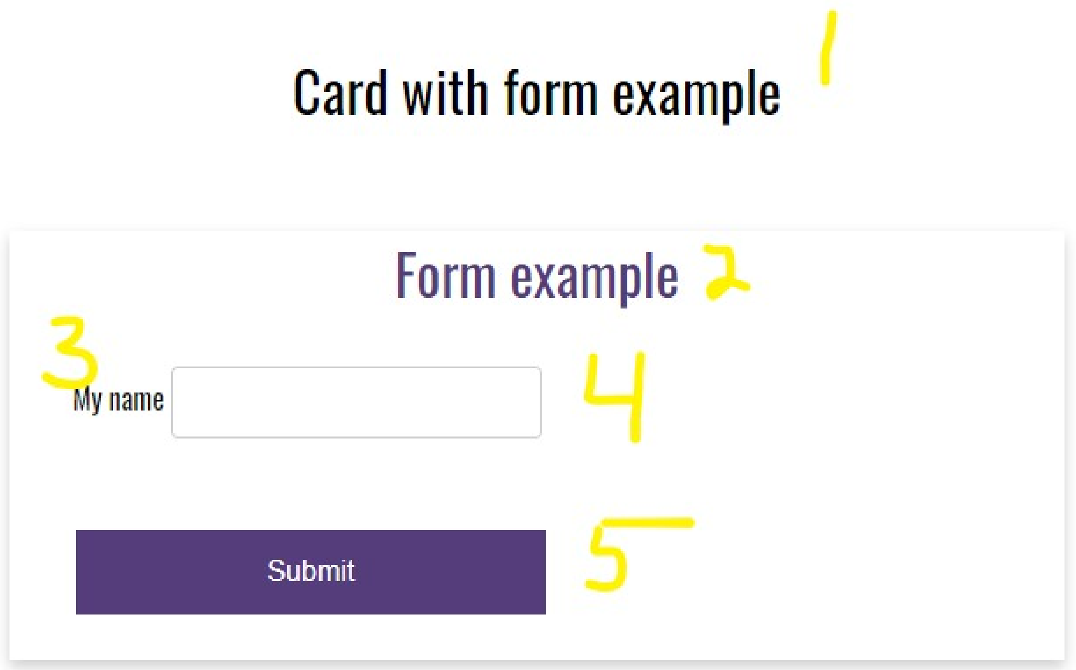
Exhibit D. CSS container and card container styling requirements



```
#container {
    padding: 10px;
}

#card-example {
    width: 600px;
    margin: 3% auto 0;
}
```

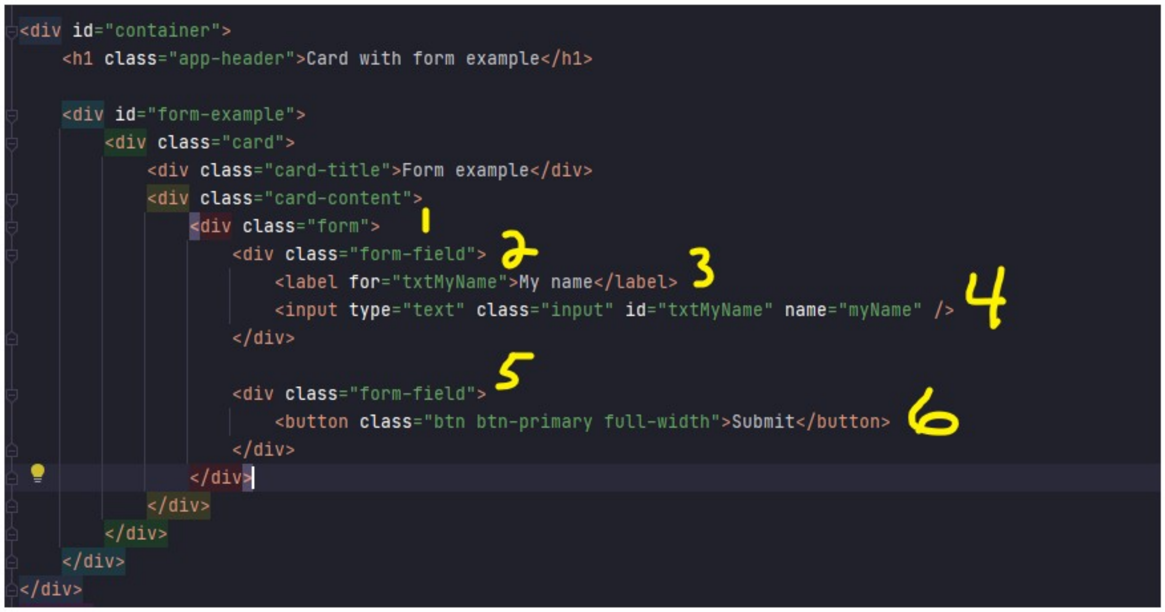
The numbers identified in this image correspond to the numbers identified in Exhibit C.

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



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”

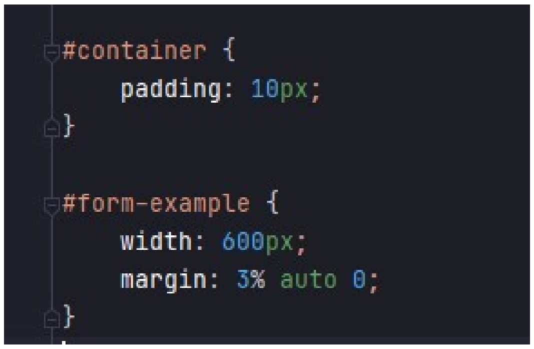
Exhibit F. HTML code for cards with nested form and form-fields



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

    <div id="form-example">
        <div class="card">
            <div class="card-title">Form example</div>
            <div class="card-content">
                <div class="form">  1
                    <div class="form-field">  2
                        <label for="txtMyName">My name</label>  3
                        <input type="text" class="input" id="txtMyName" name="myName" />  4
                    </div>

                    <div class="form-field">  5
                        <button class="btn btn-primary full-width">Submit</button>  6
                    </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 G. CSS styling requirements for a card’s with forms



```
#container {
    padding: 10px;
}

#form-example {
    width: 600px;
    margin: 3% auto 0;
}
```

Exhibit H. HTML formatting for a form with form results

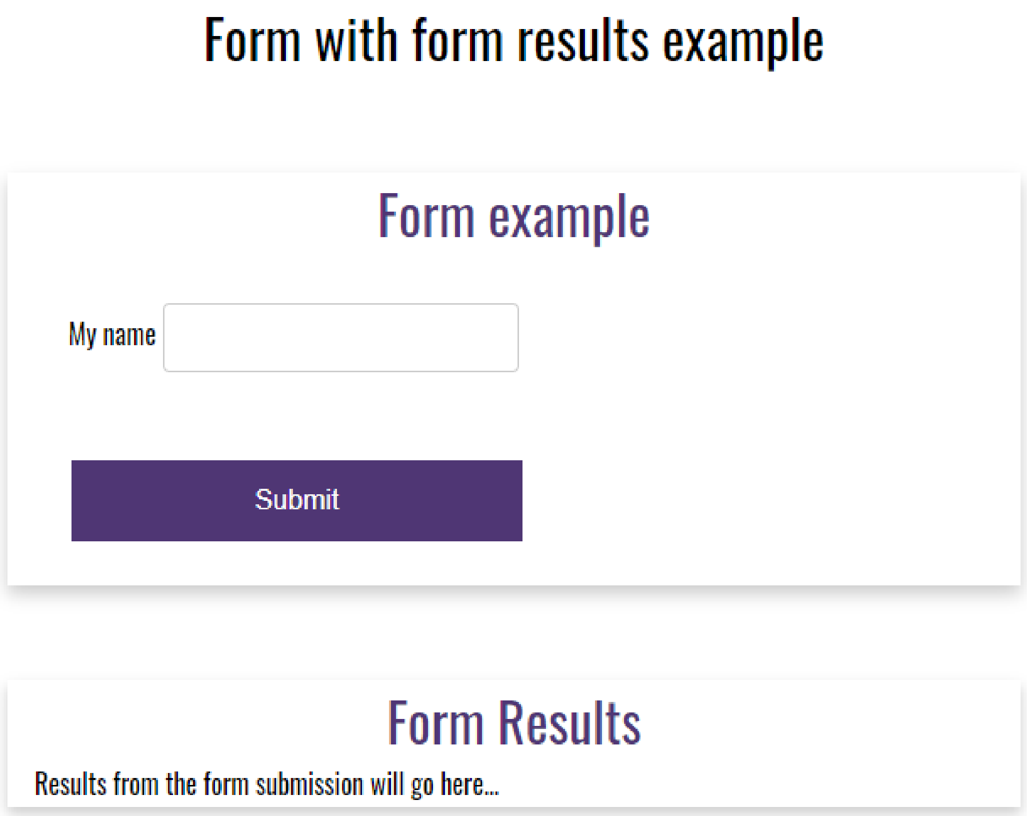
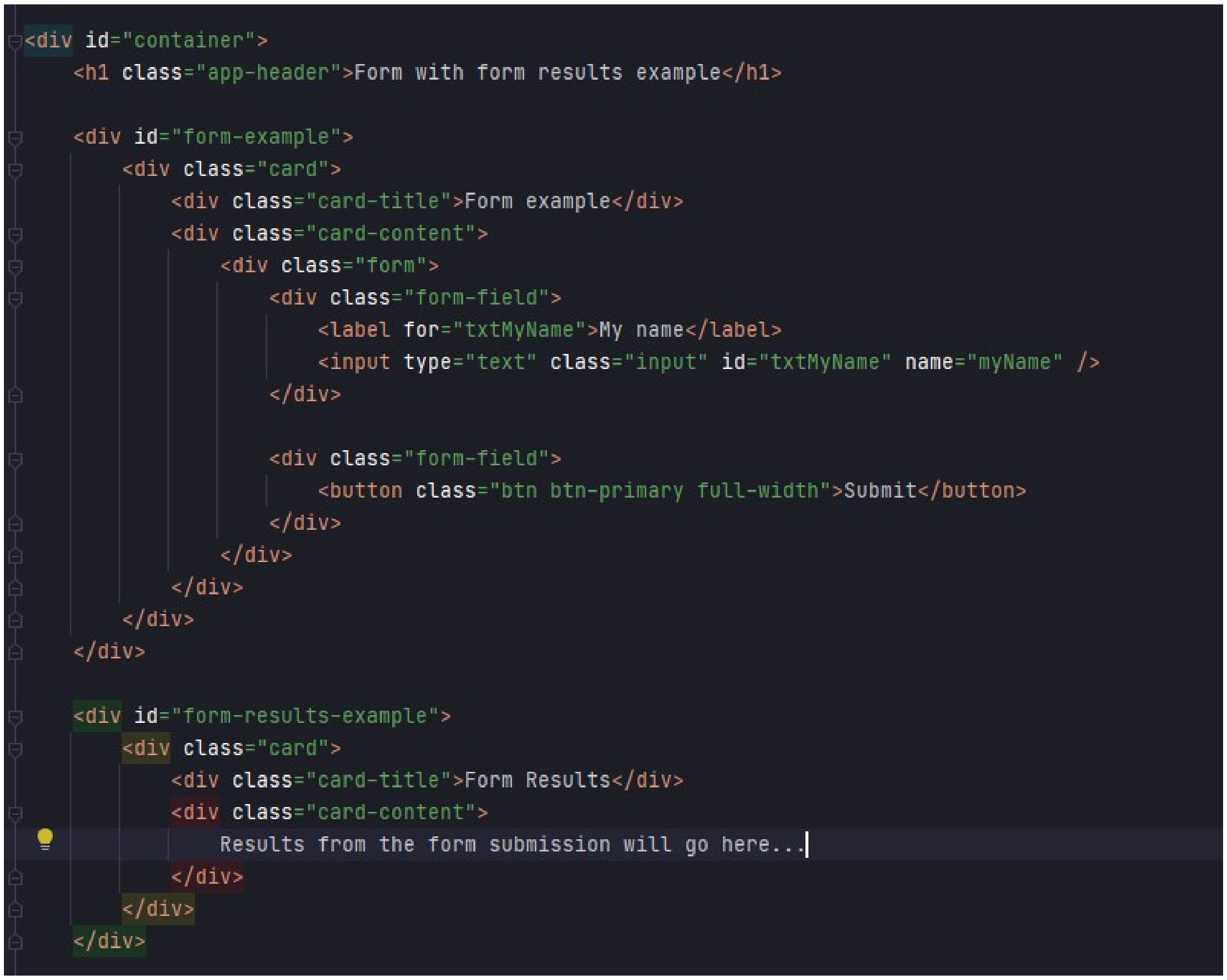
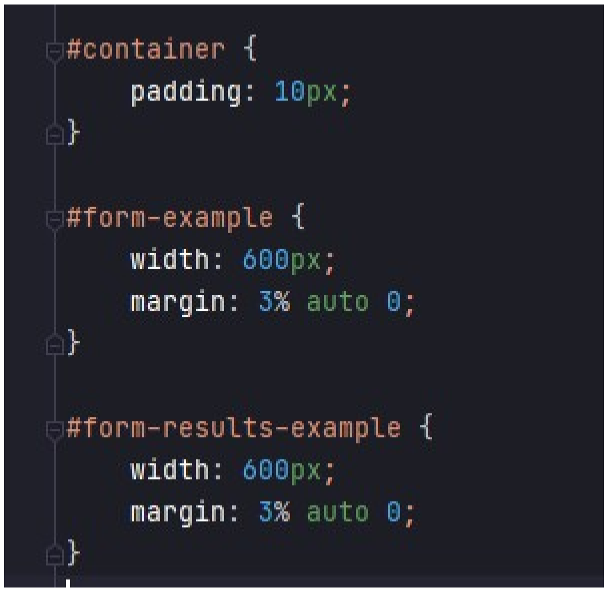

Form with form results example
Form example
My name
Submit
Form Results
Results from the form submission will go here...

Exhibit I. HTML code for a form with form results



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

    <div id="form-example">
        <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 id="form-results-example">
        <div class="card">
            <div class="card-title">Form Results</div>
            <div class="card-content">
                Results from the form submission will go here...
            </div>
        </div>
    </div>
```

Exhibit J. CSS styling for a form with form results

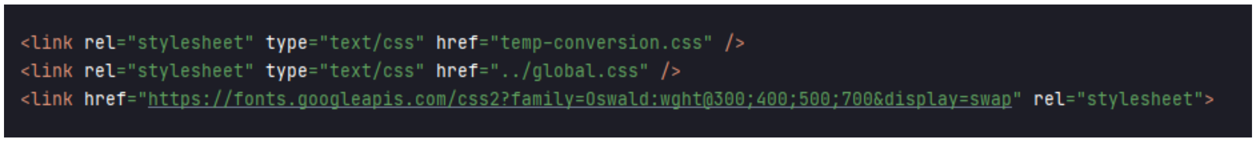


```css
#container {
    padding: 10px;
}

#form-example {
    width: 600px;
    margin: 3% auto 0;
}

#form-results-example {
    width: 600px;
    margin: 3% auto 0;
}
```

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 global.css, active week's CSS file, and Google font kit will need to be linked in every week's assignment. For example, let's say I am working on Assignment 3.2 and the assignment asks for me to create an HTML file named <yourLastName>-temp-conversion.html, I will need to add links in the head of the HTML file for the global.css, current week's CSS file, and the courses Google font kit.

Exhibit K. HTML JavaScript references

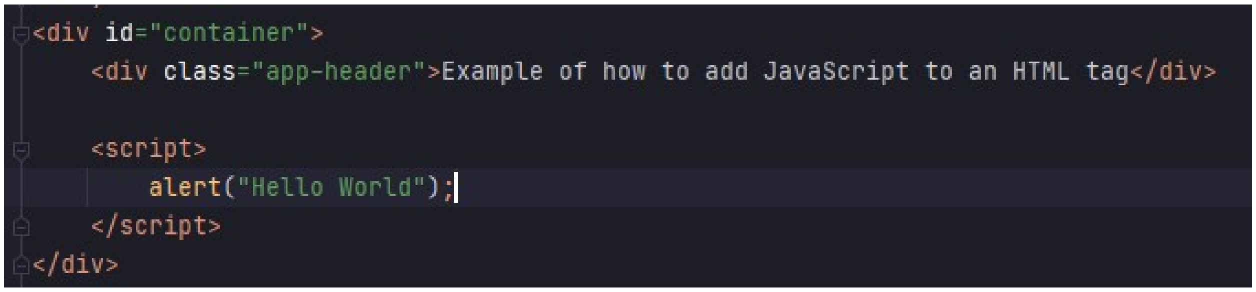


```
<link rel="stylesheet" type="text/css" href="temp-conversion.css" />
<link rel="stylesheet" type="text/css" href="../global.css" />
<link href="https://fonts.googleapis.com/css2?family=Oswald:wght@300;400;500;700&display=swap" rel="stylesheet">
```

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 3.1 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

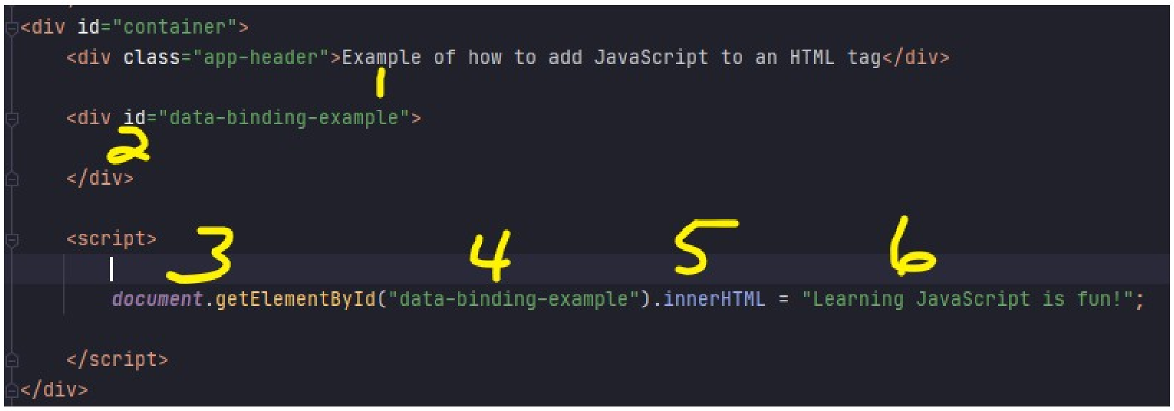


```
<div id="container">
    <div class="app-header">Example of how to add JavaScript to an HTML tag</div>

    <script>
        alert("Hello World");
    </script>
</div>
```

document.getElementById("id").innerHTML = "Hello World"

In order to bind data to an HTML element in JavaScript, we must first use the document.getElementById function to locate the HTML element. As the name suggests, for this work we have to assign the HTML element and id attribute. Once we have the element, we assign the values to the elements innerHTML section. This basically says, write the values inside the div with the id reference.

Exhibit M. JavaScript data binding



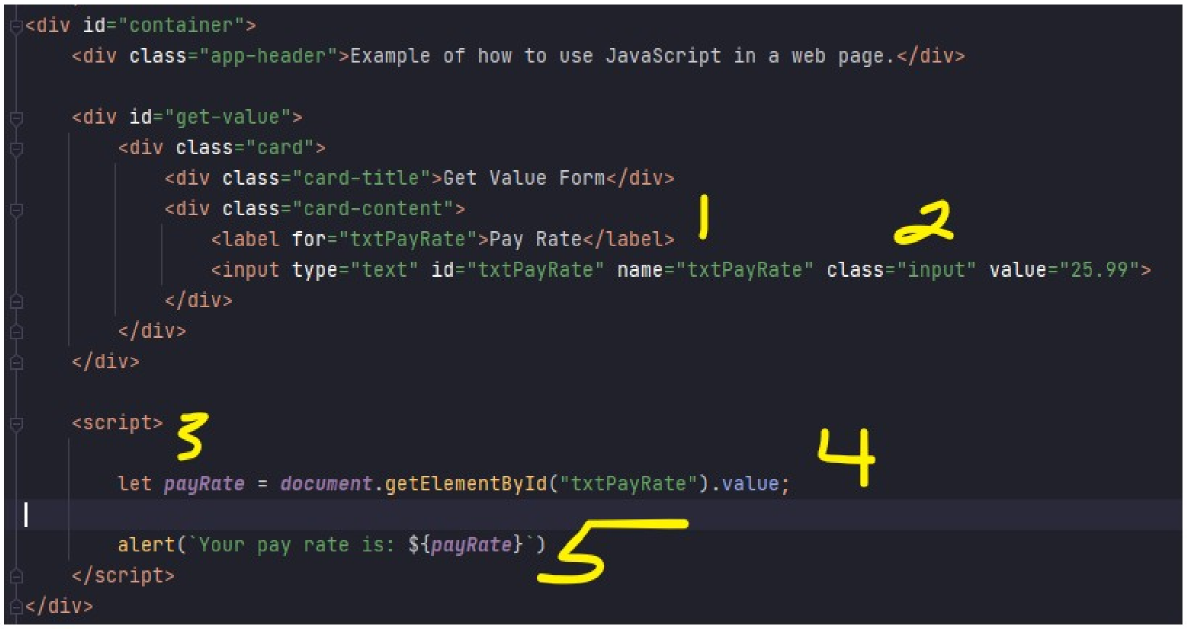
```
<div id="container">
    <div class="app-header">Example of how to add JavaScript to an HTML tag</div>

    <div id="data-binding-example">

    </div>

    <script>

        document.getElementById("data-binding-example").innerHTML = "Learning JavaScript is fun!";

    </script>
</div>
```

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 search the HTML code for the appropriate HTML, but this time it retrieves the value.

Exhibit N. JavaScript get the value from an HTML element



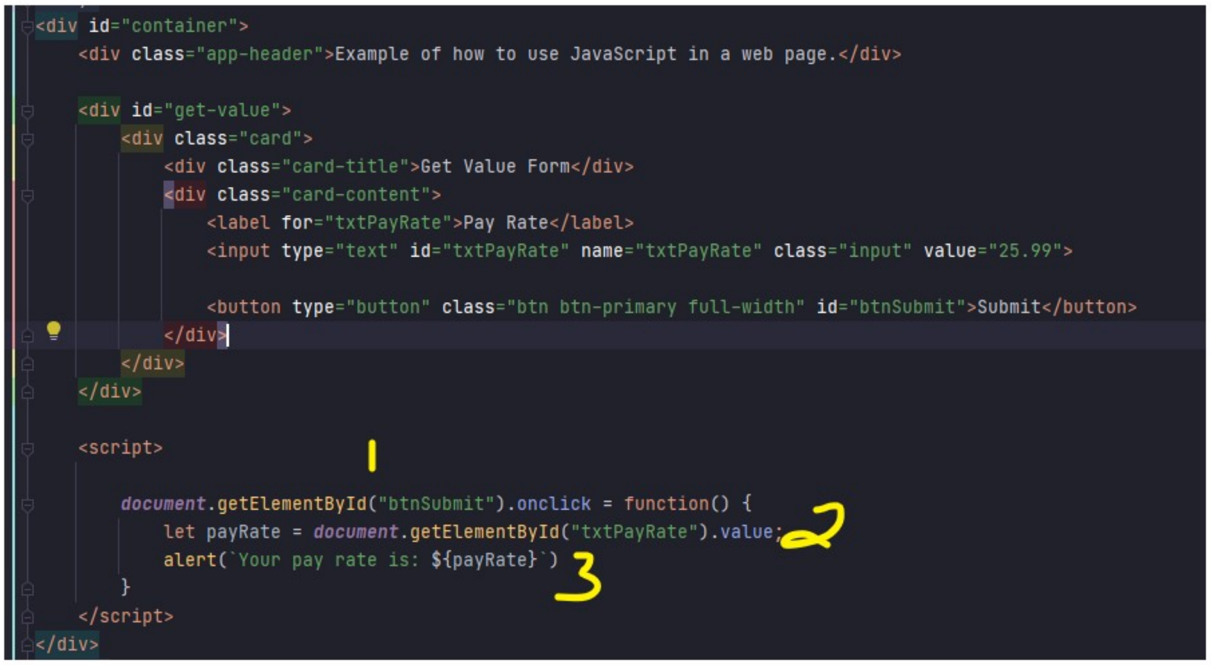
```
<div id="container">
    <div class="app-header">Example of how to use JavaScript in a web page.</div>

    <div id="get-value">
        <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>

    <script>
        let payRate = document.getElementById("txtPayRate").value;

        alert(`Your pay rate is: ${payRate}`)
    </script>
</div>
```

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

    <div id="get-value">
        <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">

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

    <script>

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

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.