CPSC1520 Client Side Scripting With JavaScript

In Class Lab 1

Form Basics

Shipping Calculator

In this exercise you are going to add an event to the form shipping rate form and use the form elements' submit event to make a calculation and output the result to the DOM.

Language Syntax

addEventListener()
element.attributes
element.value
node.textContent
typeof()

Task: Create a reference variable to the form element.

Give the form element a unique id attribute and use the querySelector() method to create a reference to the form.

```
const shippingForm = document.querySelector("#shipping-form");
console.log(shippingForm);
```

Best Practice

Check that you have the reference to the form element by using the console.log() method. To do this use the browser dev tools and make sure that you see the reference to the form before you move onto the next task.

Task: Add the addEventListener() method

Use the reference variable you created and add the addEventListener() method. This method takes two parameters, the event type “submit”, and a custom function with the actions you wish to perform when the form is submitted.

```
const shippingForm = document.querySelector("#shipping-form");
 shippingForm.addEventListener("submit", onCalculateRate);

 function onCalculateRate(e) {
    e.preventDefault();
    console.log("submit event fired");
 }
```

Best Practice

If using a button element in the form make sure that the type is set to submit.

```
<button type="submit" class="...">
  Calculate Shipping
</button>
```

Best Practice

Always create the custom event handler function right away or you get an error in the console. The error will be similar to the one shown below. Most of the time ReferenceErrors are caused by either spelling mistakes or something missing. In this case the “onCalculateRate” function.

```
▶ Uncaught ReferenceError: onCalculateRate is not defined
    at index.js:8:41
```

Task: Prevent the default behavior of the form element.

Inside the “{...}” curly braces of the event handler function you must stop the default behavior of the form element. To do this use the “e.preventDefault()” method. This method will prevent the form from trying to submit the name value pairs of the form inputs.

```
const shippingForm = document.querySelector("#shipping-form");
shippingForm.addEventListener("submit", onCalculateRate);

function onCalculateRate(e) {
    e.preventDefault();
}
```

Best Practice

Make sure that e.preventDefault() is added to the event handler function. If you do not add this method then you will not see an error but you will see that when you add a value into the input element and click on submit the input value will disappear. Try it by removing the preventDefault() method and adding a console.log(e). Check the browser dev tools and note what happens.

When you are clear on what is happening remove the console.log() and add the e.preventDefault().

```
function onCalculateRate(e) {
  console.log(e);
 }
```

CPSC1520 Client Side Scripting With JavaScript

In Class Lab 1

Task: Get a reference to the Input Element

Inside the “{...}” curly braces of the event handler function after the e.preventDefault(), create a reference variable to the input element with the id attribute of weight.

Use the console.log() method to log both the reference variable and the value attribute of the input element. What do you notice?. To use an element's attributes in your code it will alway be element.attribute .

```
function onCalculateRate(e) {
 e.preventDefault();
 const shippingWeight = document.querySelector("#weight");
 console.log(shippingWeight);
 console.log(shippingWeight.value);
}
```

Best Practice

Always check a form input elements value attribute to make sure that the data type and value are what you expect to see

Use a console.log() and inside use typeof() method which returns the data type of the operand (the thing between the brackets). Did the data type returned from this method surprise you? What were you expecting?

```
function onCalculateRate(e) {
   e.preventDefault();
   const shippingWeight = document.querySelector("#weight").value;
   console.log(typeof(shippingWeight));
   }
```

Task: Making the calculation.

First get a reference to the output element before making the calculation. To make the calculation we need to convert the string to a number. In JavaScript there are various ways to perform this task.

1. Number(stringValue)
2. parseFloat(stringValue)

Once you have made the calculation check the value in the browser console.

```
function onCalculateRate(e) {
    e.preventDefault();
    const shippingWeight = document.querySelector("#weight").value;
    const rateOutput = document.querySelector("#rate-display");
    const rate = parseFloat(shippingWeight) * 5;
    console.log(rate);
}
```

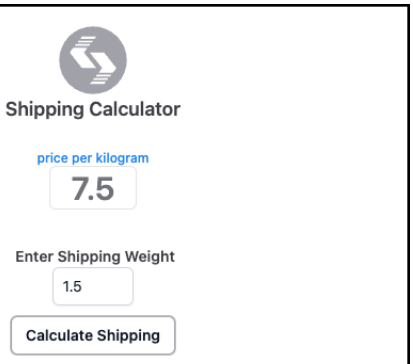
Best Practice

Always check a form input element's value attribute to make sure that the data type and value are what you expect to see.

Task: Updating the DOM

The last thing you have to do is to update the DOM and enter the rate into the output element. To do this you will use the element reference variable along with the textContent property.

```
function onCalculateRate(e) {
e.preventDefault();
const shippingWeight = document.querySelector("#weight").value;
const rateOutput = document.querySelector("#rate-display");
const rate = parseFloat(shippingWeight) * 5;
rateOutput.textContent = rate;
}
```

Best Practice Notice that depending on the number being calculated the output is not quite in the form of a price. To format the output we need to convert the calculated number to a floating point number with two decimal places and add the dollar sign.	

CPSC1520 Client Side Scripting With JavaScript

In Class Lab 1

Task: Formatting the Rate

To format the rate with the dollar sign and show the price to two decimal places you have to complete a couple of steps.

To get the floating point number use the toFloat() method which sets the number of values after the decimal point. In the case of a price we need two.

To add the dollar sign we can use a special type of string called a template literal. In the example below the formatted rate uses a template literal. **Template literals** always use the back ticks (above the tab key) and you can insert variables or other javascript using the dollar sign along with the curly braces "${}". This is called string interpolation

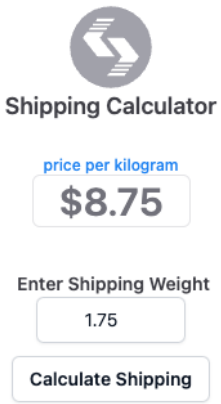
```
function onCalculateRate(e) {
e.preventDefault();
const shippingWeight = document.querySelector("#weight").value;
const rateOutput = document.querySelector("#rate-display");
const rate = (parseFloat(shippingWeight) * 5).toFixed(2);
const formattedRate = `${rate}`;
rateOutput.textContent = formattedRate;
}
```

Best Practice

When using the toFixed() method make sure to output the value to the console just in case you get a Not-A-Number property **NaN**. We will deal with **NaN** values later in the week.

Best Practice

Test your login by entering several different values into the text input field and calculating the rate. You should see if you get any unexpected values.



CPSC1520 Client Side Scripting With JavaScript

In Class Lab 1

Exercise 1

Replacing Text

Inside the index.html document there is an h2 element right above the **output** element. Currently the text reads “price per kilogram”. Once a new price is calculated and displayed change the text to buy now

Exercise 2

Replacing Number Field with text field

Replace the text field with a number field and have the user type in the name of the city where the parcel will start its journey. Place a default city into the output element and remove the price. When the user types in the name of the city and clicks on the submit button, update the output element with the name of the city.