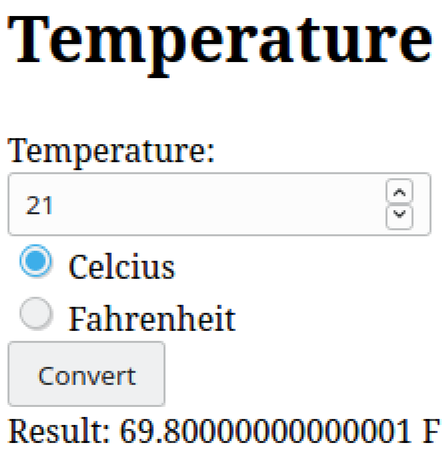
# CMPT 281: Website Design and Development
# JAVASCRIPT EXAMPLE: TEMPERATURE CONVERTER

In this example, we are going to be making a simple web page that can convert between units of temperature. We want the user to enter in a temperature measurement and specify whether that measurement is in Celsius or Fahrenheit. Then they will click a button to convert the measurement to the the other unit.

This is what the final web page will look like:


Temperature Converter

Result: 69.80000000000001 F

## 1 The HTML Code

For the web page's HTML code, we have a few goals:

1. The user needs to know what purpose the page serves.
2. The user needs to have a place to enter the temperature measurement.
3. The user needs to be able to select the unit of their measurement.
4. The user needs to be able confirm their input.
5. The result of the conversion needs to be displayed somewhere.

Let's start with a basic HTML starter template, for example, the one from SitePoint. This way we don't forget to include any of the meta information.

```
1  <!doctype html>
2  <html lang="en">
3  <head>
4    <meta charset="utf-8">
5    <title>The HTML5 Herald</title>
6    <meta name="description" content="The HTML5 Herald">
7    <meta name="author" content="SitePoint">
8    <link rel="stylesheet" href="css/styles.css?v=1.0">
9  </head>
10 <body>
11   <script src="js/scripts.js"></script>
12 </body>
13 </html>
```

For this example, there are some aspects of this starting template that we don't need. I'll remove line 7 (it's optional and serves no purpose when working locally), 8 (our web page will lack style, but that's okay for this example), and 11 (our script will be very small, and I don't expect to re-use the code elsewhere, so there is little benefit to having it as a separate file).

We can satisfy goal 1 by setting the values within the `<title>` and description `<meta>` tags, but these aren't always displayed the user in an obvious way. We should add a header to the web page to make the page's purpose obvious to a user. We can do this with a `<h1>` tag.

For the remaining goals, we need to define a `<form>` in HTML. W3 Schools provides a tutorial that covers the basics. What we need is to define a number of `<input>`s that live inside of a `<form>`, and each of these inputs should have a `<label>` that describes it.

e.g.,

```html
<form>

<label>First Input</label><br>
<input> <br>

<label>Second Input</label><br>
<input>

<!--... etc.-->

</form>
```

Each `<input>` requires a `type`, `name`, and `id`. This page describes the different input types. The most basic type is a single line text entry (`"text"`), and this is what you will get if you do not specify any type. `name` and `id` serve a similar purpose, and in many cases will have the same value — the difference between the two is only relevant for certain input types, but it is useful to set both. `name` gets used whenever the form gets submitted (though we won't see this in this example). `id` is referenced from the label, and we will make use of it within our JavaScript code.

We need inputs for three things: the temperature measurement, the unit selection, and the button that initiates the unit conversion.

For the temperature measurement, there is a `"number"` type that allows the user to enter numbers and only numbers, which is perfect for our needs.

For the unit selection, there is a `"radio"` type to display radio buttons. These are buttons that are grouped in that only one button can be selected at a time. Each button needs its own `<input>` with its own unique `id`, but its `name` is shared with other buttons in the group. (Note: there is also a `<select>` element that would work, but radio buttons are a good choice for when there are few options — they require fewer clicks and in my opinion are more visually appealing.)

For the button, there is a `"submit"` type which displays a button that can be used to submit the form. (Note: you might notice that there is also a `<button>` element. This is similar to an `<input>` with `type="submit"`, but can be used outside of a `<form>`. You can use whichever element you like more.)

For the output, we can do whatever we want (a `<div>` would be a good choice), but there is a handy `<output>` element that will work.

Based on all this, let's update our web page to include our form, heading, and updated title.

```html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Temperature Converter</title>
  <meta name="description" content="Temperature Converter">
</head>
<body>

<h1>Temperature Converter</h1>

<form>
        <label for="temperature">Temperature:</label><br>
        <input type="number" id="temperature" name="temperature"><br>

        <input type="radio" id="celsius" name="unit" checked>
        <label for="celsius">Celsius<label><br>

        <input type="radio" id="fahrenheit" name="unit">
        <label for="fahrenheit">Fahrenheit<label><br>

        <input type="button" id="convert" value="Convert">

        Result: <output name="result" id="result"></output>
</form>

<script>
// todo
</script>

</body>
</html>
```

Also included is a placeholder for the JavaScript code, which we will now create.

## 2 The JavaScript Code

For the JavaScript code, the user should click a button, and if the temperature measurement entered is in Fahrenheit, it should be converted to Celsius and display; if the measurement is in Celsius, it should be converted to Fahrenheit and displayed.

Every `<input>` element (and `<button>`) has an “onclick" event. This allows us to call a function when the user clicks on something. If you follow the link to W3 Schools, you will see that there are several methods of accomplishing this. The `addEventListener()` variant has some minor benefits[1], so that's the approach we'll use. We just need to reference the relevant `<input>` by using `document.getElementById()`.

```javascript
document.getElementById("convert").addEventListener('click', function() {
    // todo
});
```

Next, let's create variables for each of the other inputs, as well as the output.

[1]See this Stack Overflow answer for some more detail, if you're curious.

```
document.getElementById("convert").addEventListener('click', function() {
    let temperature_input = document.getElementById("temperature");
        let celsius_input = document.getElementById("celsius");
        let fahrenheit_input = document.getElementById("fahrenheit");
        let result_output = document.getElementById("result");

        // todo
});
```

I like to include the element name in the variable's name, so that I am always reminded of what data type I am working with.

Next, we need to check what unit type is being used. Radio buttons (and `<input>` elements in general) have a "checked" attribute that indicates whether the radio button has been selected. For example, `celsius_input.checked` will return a boolean indicating its state.

Once we've figured out what unit we are working with, we need to implement the code to convert to the other unit. We can look up the formula for this. To convert to Fahrenheit, it is Celsius * 1.8 + 32. To convert to Celsius it is (Farenheit - 32) / 1.8.

Finally, to display the temperature to the user, we can update the `innerText` of `result_output`.

```
document.getElementById("convert").addEventListener('click', function() {
    let temperature_input = document.getElementById("temperature");
        let celsius_input = document.getElementById("celsius");
        let fahrenheit_input = document.getElementById("fahrenheit");
        let result_output = document.getElementById("result");

        if (celsius_input.checked) {
                let fahrenheit = temperature_input.value * 1.8 + 32;
                result_output.innerText = fahrenheit + " F";
        } else {
                let celsius = (temperature_input.value - 32) / 1.8;
                result_output.innerText = celsius + " C";
        }
});
```

That's it! You now have your very own temperature converter.

## 3 Things to try on your own

1. Check that the user actually entered in a number for temperature and not text or left it blank (old browsers will not respect the "number" input type and instead revert to allowing text).
2. Add the option to convert from Kelvin. The output text should then be updated to show conversions to Fahrenheit and Celsius, and Kelvin should also be output if the input is either Fahrenheit or Celsius.
3. Make it look pretty! What I made is very minimal and it could easily be made more attractive.
4. Use a `<select>` element instead of radio buttons.