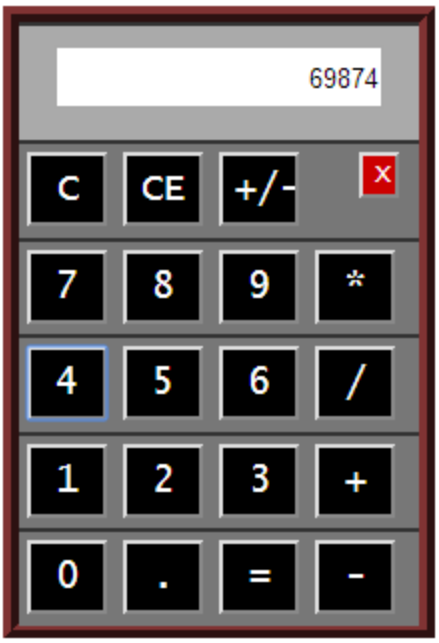
HTML & CSS



Calculator Layout Assignment

In this assignment, we are going to build the layout for a calculator. This is a layout only assignment – we won't be adding the functionality at this time.

The calculator that we will be building is illustrated in the image on the right.

Where to begin

Start by creating a new folder (directory) inside your local git repository. Call it calculator.

Next, open Brackets. In the Brackets file menu, select Open Folder. Navigate to and select the calculator folder that you just created. Now select File>New. This will create a new file which will begin as unnamed. Select File>Save As and save the file as calculator.html in the calculator folder. Create another new file using the same process and name it calculator.css.

Now you can start writing the HTML. All HTML begins with the same structure, so this is a good place to begin:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Calculator</title>
        <link rel='stylesheet' href='calculator.css' type='text/css'>
    </head>
    <body>
    </body>
</html>
```

When working on layouts, I typically start from the outside and work my way inwards. So the first thing I would do is create a container to hold the calculator and name it (assign it an id) so that I can style it later in css. The container would be placed between the body tags in the html:

```
<div id='calculatorBody'>
</div>
```

Next I would observe that the calculator is divided into six horizontal sections. The top section contains the display and the bottom five each contain a row of buttons. The button rows all have the same styling (background color, size, margins, etc.) so I will create a class for those so they can all be styled in one rule. The html for the horizontal sections would be placed inside of the calculator body:

```
<div id='displaySection'>
</div>
<div id='buttonRow1' class='buttonRow'>
```

```
</div>
<div id='buttonRow2' class='buttonRow'>
</div>
<div id='buttonRow3' class='buttonRow'>
</div>
<div id='buttonRow4' class='buttonRow'>
</div>
<div id='buttonRow5' class='buttonRow'>
</div>
```

The display section contains the numerical display. It is an active element, meaning that we want the display to be changeable from the Javascript that we will write later. I am going to use an input element for the display because that will work well for coding. I don't want to allow the user to click into the element and enter text, however, so I will set the readonly attribute. This element is placed inside the displaySection:

```
<input type='text' id='display' name='display' readonly='true' value='0.'>
```

For the buttons I will be using button type input elements. We can style them however we want in CSS to get the calculator to look more like a calculator and less like an HTML generic form. Each button will be assigned a name and a value. All of the buttons will share a class since they all have the same appearance and we can just use one set of rules in the CSS to apply to all buttons of the same class. The value determines the character(s) that will be displayed in the button. Here is a sample button:

```
<input type='button' id='num6' name='num6' value='6' class='numberButton'>
```

The buttons should be placed within the buttonRows that we defined above, in left to right order (we will be using float left for the button layout).

And that's it. We have built our HTML calculator. Of course it won't look like a calculator yet because we haven't styled it, but all of the content should be there. You can check your completed HTML by viewing it in a browser. The next page shows the completed html (you can compare your completed document against this one when you're done writing it).

Check that your document is HTML5 compliant by testing it in the W3 HTML validator (http://validator.w3.org).

```
<!DOCTYPE html>
<html>
    <head>
        <title>Calculator</title>
        <link rel='stylesheet' href='calculator.css' type='text/css'>
    </head>
    <body>
        <div id='calculatorBody'>
            <div id='displaySection'>
                <input type='text' name='display' value='0.' id='display' readonly='true'>
            </div>
            <div id='buttonRow1' class='buttonRow'>
                <input type='button' id='clear' name='clear' value='C' class='numberButton'>
                <input type='button' id='clearErr' name='clearErr' value='CE'
class='numberButton'>
                <input type='button' id='sign' name='sign' value='+/-' class='numberButton'>
                <input type='button' id='close' name='close' value='X'>
            </div>
            <div id='buttonRow2' class='buttonRow'>
                <input type='button' id='num7' name='num7' value='7' class='numberButton'>
                <input type='button' id='num8' name='num8' value='8' class='numberButton'>
                <input type='button' id='num9' name='num9' value='9' class='numberButton'>
                <input type='button' id='opMult' name='opMult' value='*' class='numberButton'>
            </div>
            <div id='buttonRow3' class='buttonRow'>
                <input type='button' id='num4' name='num4' value='4' class='numberButton'>
                <input type='button' id='num5' name='num5' value='5' class='numberButton'>
                <input type='button' id='num6' name='num6' value='6' class='numberButton'>
                <input type='button' id='opDiv' name='opDiv' value='/' class='numberButton'>
            </div>
            <div id='buttonRow4' class='buttonRow'>
                <input type='button' id='num1' name='num1' value='1' class='numberButton'>
                <input type='button' id='num2' name='num2' value='2' class='numberButton'>
                <input type='button' id='num3' name='num3' value='3' class='numberButton'>
                <input type='button' id='opPlus' name='opPlus' value='+' class='numberButton'>
            </div>
            <div id='buttonRow5' class='buttonRow'>
                <input type='button' id='num0' name='num0' value='0' class='numberButton'>
                <input type='button' id='opPoint' name='opPoint' value='.'
class='numberButton'>
                <input type='button' id='opEqual' name='opEqual' value='='
class='numberButton'>
                <input type='button' id='opMinus' name='opMinus' value='-'
class='numberButton'>
            </div>
        </div>
    </body>
</html>
```

Calculator CSS

Now it's time to make the calculator look like a calculator. We will need to add rules to the CSS file which will control the layout and appearance of each of the elements that we put into the HTML. Like in HTML, I prefer to start at the outermost element and work my way inwards. You can use quick-edit in brackets to add rules to your CSS file (or you can just edit the CSS file). To use quick-edit, you select the element tag, id or class that you want to add/change a rule for and then hit ctrl-e (cmd-e for Mac). Any existing rule for the element will be displayed and can be edited. There is also a button to create a new rule.

It is a good idea to first style the body to occupy the full browser area. If you don't do this, the size of the body will be determined by the size of its contents because that is the rule for block type elements. If we allow the body to be automatically sized then we cannot create a responsive layout. So start with the rule:

```
block {
    width: 100%;
    height: 100%;
}
```

The next element that we come to is the calculatorBody. This one will determine the overall size of the calculator and also provide us with the border. By setting the text-align to center here, the property will be inherited by all children which will cause our output display and button texts to be centered with very little effort. All of the other blocks will fit inside this one.

```
#calculatorBody {
    width: 200px;
    height: 300px;
    border: 8px ridge #803333;
    background-color: #333333;
    text-align: center;
}
```

The horizontal row that holds the display is slightly larger than the other five horizontal rows and has a different color. It occupies the full width of the container that it is placed in (the calculatorBody):

```
#displaySection {
    width : 100%;
    height : 19%;
    background-color : #aaaaaa;
    margin : 2px;
}
```

The buttonRows are very similar, but slightly less tall:

```
.buttonRow {
    width : 100%;
    height: 15.5%;
    background-color : #777777;
    margin : 2px;
}
```

The output display has a white background and its contents are right aligned. It looks like:

```
#displaySection input {
    width : 80%;
    height : 25px;
    background-color : #ffffff;
    margin : 10px;
    text-align : right;
}
```

We can style all of the calculator buttons with one rule since they all belong to the same class:

```
.numberButton {
    width : 20%;
    height : 80%;
    background-color : #000000;
    float : left;
    margin: 2%;
    color: white;
    font-size: 1.2em;
    font-family:"Lucida Console", Verdana, Arial;
}
```

And finally, one button is not like all of the others, so we will style it separately:

```
#close {
    float : right;
    margin-right: 5%;
    background-color: #cc0000;
    width: 10%;
    height: 50%;
    font-size: 0.8em;
    font-family:"Lucida Console", Verdana, Arial;
}
```

That should be everything. Your calculator should look like the example.

Now go back and try some different styles. Can you change the colors? Use border-radius to add rounded corners to the buttons. Try changing the border styles on the buttons.

What happens if you change the size of the calculatorBody? What causes it to behave that way?

Try adding the following two attributes to the button elements in the HTML file:

```
onmouseout ='this.style.backgroundColor="#000000"'
onmouseover='this.style.backgroundColor="#773333"'
```

In the HTML, we can specify what action should be taken when an event is detected. The code above defines actions for the mouseover and mouseout events. Mouseover is when the mouse is moved over an element. Mouseout is when the mouse is moved away from the element it was just over. The values assigned to these attributes should be Javascript code which will be executed when the event occurs. In this case the Javascript that is executed changes the background color of the displayed element. Other events you might want to experiment with are onmousedown, onmouseup and onclick.