Lesson 11 Challenges

Challenge 1

Forms 1

Open **form.html** (located in the files folder), the file contain some CSS to help with the form layout.

Add the following elements and attributes:

- <form>
- FirstName
- Input type with a for email
- Input type with a for website (URL)
- Password
- Submit button

To manage your layout use the CSS located in **Unit4/style.css** (paste it into the head).

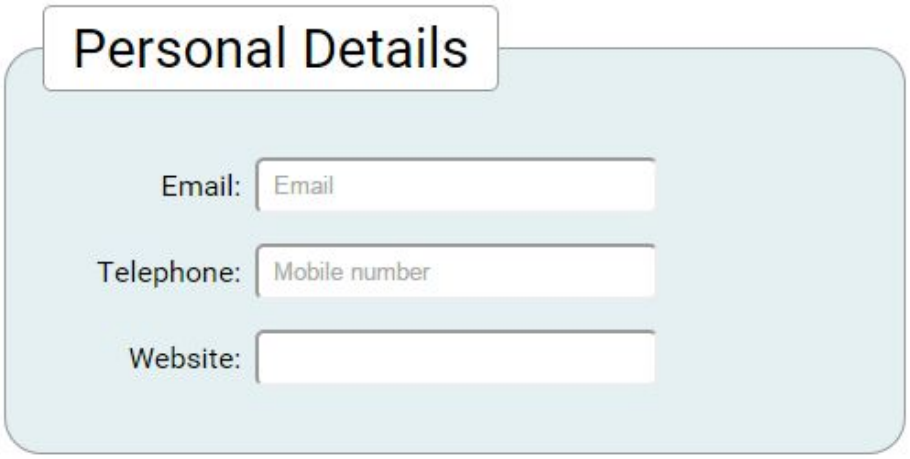
Challenge 2

Forms 2

Continuing from Exercise 1 add a new fieldset with the following elements and attributes:

- Required on Firstname, & email
- Fieldset
- Legends
- Telephone number
- Add placeholder text to all appropriate fields

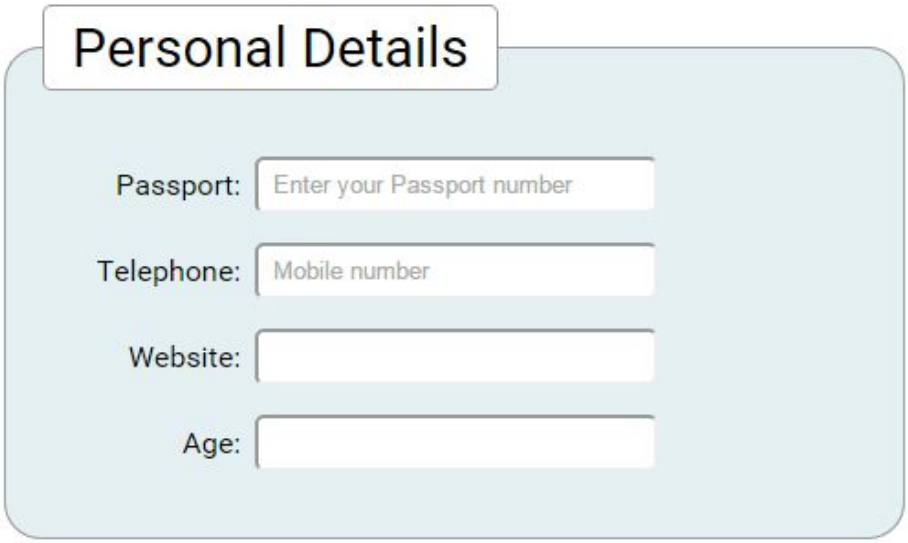
Test your page on the major browsers



Challenge 3

Forms 3

1. Continuing from **Challenge 2,** add the following elements and attributes:
    - Passport number (with correct pattern for Passport)
    - Add pattern to Telephone number (correct pattern for Mobile phone)
    - Age (with Max and Min)
2. Add placeholder text to all appropriate fields.



Challenge 4

Forms 4

1. Continuing from Challenge 3, add the following elements and attributes:
   - Date
   - Select Box with Option Group (i.e., Signs of the Zodiac - sun, earth, fire, water)
   - DataList (i.e Favourite Music Genre)
2. Add placeholder text to all appropriate fields.

Challenge 5

This challenge is about styling our forms and adding validation.

In the lesson folder under challenge 5 there is a folder called forms. Download this folder. There is a html file and a **default.css** file to get you started. There is also an empty css file called **components.css**, This is the file you need to add your css to.

Step 1. Adding styling

After you add new css to the components.css file refresh your browser and observer the changes you have made. If you are unsure of the effect of any css use the browser inspector on an element and switch on/off the css applied to the element.

Essentially we are going to create 3 columns to accommodate our input elements

We will begin by adding css to create the 3 column effect. We are going to use our familiar inline-block technique to achieve the 3 column layout

```
.cbp-mc-column { /*class on 3 column divs */
      width: 33%;
      padding: 10px 30px;
      display: inline-block;
      vertical-align: top
}
```

Note: we needed to add the vertical-align top to the columns as the default for inline-block is to align to the bottom.

Note also as we are inheriting the border-box setting in the default.css, so the padding doesn't cause a problem and mess up our percentage widths. Be careful of adding any margin to the div columns as this will affect the widths.

Next lets add some styling to the Labels

```
.cbp-mc-form label {
      display: block;
      padding: 40px 5px 5px 2px;
```

```
      font-size: 1.1em;
      text-transform: uppercase;
      letter-spacing: 1px;
      cursor: pointer;
}
```

Next we will add some styling to the input area

```
.cbp-mc-form input,
.cbp-mc-form textarea,
.cbp-mc-form select {
      font-family: 'Lato', Calibri, Arial, sans-serif;
      line-height: 1.5;
      font-size: 1.4em;
      padding: 5px 10px;
      color: #fff;
      display: block;
      width: 100%;
      background: #47a3da;
}
```

Now let's add add a border around our input elements and set a minimum height on the textareas

```
.cbp-mc-form input,
.cbp-mc-form textarea {
      border: 3px solid #fff;
}

.cbp-mc-form textarea {
      min-height: 200px;
}
```

Let's change the border color when an input element gets focus i.e. when user clicks or tabs into and input element.

```
.cbp-mc-form input:focus,
.cbp-mc-form textarea:focus,
.cbp-mc-form label:active + input,
.cbp-mc-form label:active + textarea {
      outline: none;
      border: 3px solid #10689a;
}
```

Next let's change the look of the placeholder text

```
::-webkit-input-placeholder { /* WebKit browsers */
    color: #10689a;
    font-style: italic;
}

:-moz-placeholder { /* Mozilla Firefox 4 to 18 */
    color: #10689a;
    font-style: italic;
}

::-moz-placeholder { /* Mozilla Firefox 19+ */
    color: #10689a;
    font-style: italic;
}

:-ms-input-placeholder { /* Internet Explorer 10+ */
    color: #10689a;
    font-style: italic;
}
```

Now let's style the submit button

```
.cbp-mc-submit-wrap {
      text-align: center;
      padding-top: 40px;
}
```

```
.cbp-mc-form input.cbp-mc-submit {
	background: #10689a;
	border: none;
	color: #fff;
	width: auto;
	cursor: pointer;
	text-transform: uppercase;
	display: inline-block;
	padding: 15px 30px;
	font-size: 1.1em;
	border-radius: 2px;
	letter-spacing: 1px;
}
```

Finally let's add a hover effect to the submit button

```
.cbp-mc-form input.cbp-mc-submit:hover {
	background: #1478b1;
}
```

Once you have completed the styling consider applying your own styling where you think appropriate.

Step 2. Making Responsive

Resize your browser and see the effect. It doesn't look too good at smaller browser sizes. Let's add some media queries to make it look more acceptable at smaller widths.

Let's change to a 2 column layout for row 1 and i column layout for row two at browser width of less than 900px.

```
@media screen and (max-width: 900px) {
	.cbp-mc-column {
		width: 49%;
	}
	.cbp-mc-column:nth-child(3) {
		width: 100%;
```

```
      }
}
```

Remember that :nth-child(n) finds for every element that is the nth child of its parent.

Finally add another media query at width less than 600px. Where all elements will become stacked.

```
@media screen and (max-width: 600px) {
      .cbp-mc-column {
            width: 100%;
            padding: 10px;
      }
}
```

Step 3. Adding Validation

Add appropriate validation to each of the input elements. Use the lesson notes to determine the most appropriate setting.

Using CSS to highlight required fields and invalid data

In tandem with the new input types and attributes provided by HTML5, CSS3 gives us some new pseudo-classes we can use to provide visual clues to the user as to which form fields are required, which are optional, and which contain validation errors.

Required fields can use the :required pseudo-class:

```
input:required {
    background:hsl(180, 50%, 90%);
    border:1px solid #999;
}
```

The success or failure of form validation can be signified to the user through the use of the :valid,:invalid, :in-range, and :out-of-range pseudo-classes:

N.B. You may be familiar with the hsl format for colors. Change to regular hex or rgb if you wish. HSL stands for Hue, Saturation and lightness. See a generator at http://hslpicker.com/#7aa4ff .

```
input:valid,
input:in-range {
    background:hsl(120, 50%, 90%);
    border-color:hsl(120, 50%, 50%);
}

input:invalid,
input:out-of-range {
    border-color:hsl(0, 50%, 50%);
    background:hsl(0, 50%, 90%);
}
```

More on validation at https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Forms/Data_form_validation

Challenge 6

Find one of the forms you worked which has a tel input. Add another input field and apply a pattern for mobile and landline numbers. Patterns are a form of regular expressions.

We can apply more complex pattrerns using regular expressions see example at http://www.sitepoint.com/client-side-form-validation-html5/ .

Regular expressions can get very complex and require a lot of effort to get right. See some tutorials at http://regexone.com/

Challenge 7

Taking the form we built in challenge 5. Change the CSS so that the form is now using flexbox instead of the floating concept we used in challenge 5.

Can we achieve the same effect with Flex.

Refer back to your lesson notes on flex box. Also you may find the site useful

http://the-echoplex.net/flexyboxes/

http://bennettfeely.com/flexplorer/

Challenge 8

Revert to your wireframes and concentrate on any forms you may have incorporated into your design.

Think about the data you will require to be processed

Reflect those data items in your wireframes

What sort of validation will be required.

Use post it notes (control in balsamiq) if you require to indicate validation.

Try and complete the wireframes for these pages

Challenge 9

If time permits start coding up the forms you will require based on your completed wireframes.