Overview

This homework will provide an overview of how an interactive web form using PHP will work. In this homework, we:

(1) Familiarize ourselves with forms
(2) Learn how to use Javascript to process text boxes
(3) Learn how to use Javascript to manage button behavior

Note the code here might not work with certain versions of Google Chrome. Unfortunately, Chrome has a bug in the onblur event that creates a loop if the onblur returns false. There is a hack around the bug, but I would prefer not to teach that to you as it adds needless complexity.
https://bugs.chromium.org/p/chromium/issues/detail?id=666205

Exercise 1: The Form

Forms are commands in HTML that allow users to input things. Forms have controls like buttons, text boxes, radio buttons, check boxes and pulldowns. You must put these controls in forms. If you do not, the control will display, but will not activate.

Type the following in and save it as hw3.htm. Upload it to webhost.

```
<!doctype html>
<html>
 <head>
   <style>
     .labelsec {
        display: inline-block;
        font-weight: bold;
        text-align: right;
        width: 300px;
     }
     .datasec {
        display: inline-block;
     }
     .errorsec{
        display: inline-block;
        color: red;
     }
   </style>
 </head>
 <body>
   <form>
     <div class="labelsec">
       <label for="numberthing">Some kind of number:</label>
     </div>
     <div class="datasec">
```

```
      <input type="number" id="numberthing" />
    </div><br />

    <div class="labelsec">
      Add 5 to numberthing:
    </div>
    <div class="datasec">
      <span id="add5"></span>
    </div><br />

    <div class="labelsec">
      Times numberthing by 5:
    </div>
    <div class="datasec">
      <button type="button"
            id="multbutt"
            name="multbutt"
            accesskey="m"
            disabled
      >
          <span style="TEXT-DECORATION: underline">M</span>ultiply by 5
      </button>
      <span id="mult5"></span>
    </div><br />

    <div class="labelsec">
      <label for="idthing">Thing with format 999-99-9999:</label>
    </div>
    <div class="datasec">
      <input type="text" id="idthing" name="idthing" />
      <span id="formaterror" class="errorsec"></span>
    </div><br />

    <div class="labelsec">
      <label for="datething">Textbox that does dates:</label>
    </div>
    <div class="datasec">
       <input type="date" id="datething" />
    </div><br />

    <div class="labelsec">
      <label for="emailthing">Textbox that does emails:</label>
    </div>
    <div class="datasec">
      <input type="email" id="emailthing"  />
    </div><br />


  </form>
</body>
```

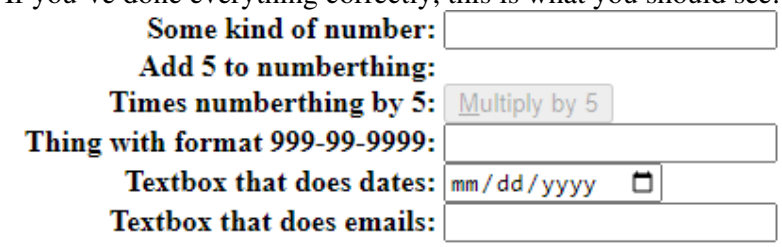
</html>

Within this form, our controls are formatted as a set of divs and spans. The divs and spans are given classes, specifically labelsec is the class for labels which are bolded, and aligned right. Datasec is the class for the data entry element. Errorsec is the class for error messages.

We have also created various simple kinds of form control:

- **Labels:** Each label has a for attribute. When the label is clicked, the for attribute sets the cursor focus to the type of textbox with the id having the same value as the for. Click on the labels to see this.
- **Textbox:** This is created using the command <input type="<kind of textbox"> . Note how we give every textbox both a name and an ID and the two are the same. ID is used by HTML to identify form controls. Name is used by PHP to identify form controls. Javascript can use both. This is an example of the "string and glue" problem with web development- no one agreed on the standard.
  The "text" textbox is a standard textbox. The number textbox only allows numbers and includes a spinner to help you determine the number. The date textbox only accepts dates and includes a built-in date picker. The email textbox only allows correctly formatted emails. Note many of these types of textboxes are only supported by modern browsers.
- **Button**. Buttons are created with <button type="button">. You must say type="button." Otherwise, HTML will consider your button a submit button. Submit buttons are useful in a different style of HTML programming but have no purpose in an AJAX/REST style. There is a critical difference between buttons and submit buttons. When a submit button is pressed, it triggers the hyperlink specified in the form action, which is completely absent in AJAX/REST style. Also note how we linked particular keys to buttons using the accesskey attribute and underlined characters in buttons using inline CSS.

If you've done everything correctly, this is what you should see:



Take a moment to fool around with the form. Note how clicking on labels moves you into the corresponding textbox, and the textboxes behave differently. If the disabled attribute is removed, hitting Alt+m triggers the button.

Exercise 2: Processing Individual Textboxes

Text processing in HTML is triggered using the onblur and onKeyUp attributes. Onblur activates when you leave the textbox. onKeyUp activates every time a key is released. You link these functions to Javascript code. Typically, the Javascript code

is written in a <script></script> tag in the <head></head> tag of HTML. Let's write some Javascript to process the number field.

First, we write the Javascript. The text in bold is to be inserted into the unbolded text.

```
<head>
 <script>
    function addfive(whichcontrol){
     if (whichcontrol.value.length==0){
      add5.innerHTML="";
      multbutt.disabled=true;
      return true;
     }
     else {
      add5.innerHTML=parseFloat(whichcontrol.value)+5;
      multbutt.disabled=false;
      return true;
     }

   }
    </script>
</head>
```

Then, add the following to the numberthing textbox.

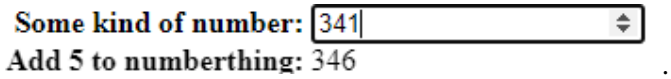
```
<input type="number"
      id="numberthing"
      onblur="addfive(numberthing)"
      onKeyUp="addfive(numberthing)"
/>
```

What we are saying is that when the user attempts to key something into or leave the numberthing textbox (onblur), activate the addfive Javascript code.

Addfive works as follows. If there is nothing in the textbox, blank what is in the span called add5. Also, disable the multiply button. Otherwise, convert the value in numberthing to a floating point number, add 5, display the result in the span called add5 and enable the multiply button.

parseFloat allows us to check all kinds of numbers. If we want to check just integers, we would use parseInt.

If you did everything correctly, you should see this:



Note how in every function we first check for blanks. This is important. While a blank may not be a valid value in an overall form, it is often a valid value at a particular point in time (e.g., when the form first loads). We have to make sure the

individual field allows for blanks. We check that the form is non-blank somewhere else (e.g., in the save button).

Also, notice how once the multiply button is enabled, hitting Alt+M pushes it.

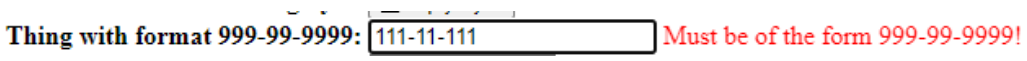
Now, lets validate the textbox with the format of 999-99-9999. There is no built-in HTML way of doing this, so we have to build the code ourselves. Create the following function

```
function checkcomplex(whichcontrol,errormsgfield){
    var passcheck=true;
    var x=0;
    if (whichcontrol.value.length==0){
      errormsgfield.innerHTML="";
      submitter.disabled=true;
      return true;
    }
    else
    {
     var myregex=new RegExp('[0-9]{3}-[0-9]{2}-[0-9]{4}');
     passcheck=myregex.test(whichcontrol.value);
    } //end else
    if (passcheck){
      errormsgfield.innerHTML="";
      submitter.disabled=false;
    }
    else{
      errormsgfield.innerHTML="Must be of the form 999-99-9999!";
      submitter.disabled=true;
    }
    return passcheck;
  }
```

Add it to the idthing textbox as both an onblur and a onKeyUp. Note that you need to pass it two things- the idthing textbox and the formaterror span. The span is where the function will output its errors.

RegExp is a regular expression. This is a way of testing if a text string matches a particular pattern. In this case, the pattern is 3 digits ([0-9]{3}) followed by a dash, followed by 2 digits ([0-9]{2}) followed by a dash, followed by four digits.

If there is an incorrect entry in the field, an error message will appear.



Exercise 3: Buttons and Javascript

In a textbox, you activate Javascript with onblur or onKeyUp. In a button, you write code for an onclick. Create the following Javascript function:

```
function multbyfive(whichcontrol){
   if (whichcontrol.value.length==0)
      mult5.innerHTML="";
   else
      mult5.innerHTML=parseFloat(whichcontrol.value)*5;
}
```

Now, link it to the calculate button:

```
<button type="button"
      id="multbutt"
      name="multbutt"
      onclick="multbyfive(numberthing);"
      accesskey="m"
      disabled
>Try it out:
```

Some kind of number: 6
Add 5 to numberthing: 11
Times numberthing by 5: Multiply by 5 30