Due Sep 23 by 11:59pm **Points** 100
Available Sep 3 at 12am - Sep 23 at 11:59pm 21 days

This assignment was locked Sep 23 at 11:59pm.

Module 2 Assignment

Purpose

Students will create an HTML page and utilize JavaScript to control a form and validate the data the user enters into the form.

Related Module Objectives

This assignment satisfies Module Objectives 1, 2, 3, 4, 5 and 6.

Possible Points

This assignment is worth a maximum of 100 points.

Important Notes

Students may refer to the following pages in case they forget how to perform the following tasks:

- Access the **Course Web Server**
- Viewing your **ePortfolio in a web browser**

Required Tools

Students will be required to use one or more of the following tools to earn a passing grade on the module assignment. Each of the tools listed below can be downloaded for free or already exist in the indicated

operating system.

- Web browser (Chrome or Firefox recommended)
- Basic text editor
  - Notepad++ (Windows)
  - TextEdit in plain-text mode (Mac OS)
  - pico or vi (Linux)
- Secure Shell (SSH) client
  - PuTTy (Windows)
  - ssh (Mac OS and Linux)
- File transfer tool (must support SFTP via SSH - **DO NOT USE FTP**)
  - WinSCP (Windows)
  - CyberDuck (Mac OS)
  - sftp (Linux)

Warnings

Students must complete this assignment without the assistance of third-party development tools or frameworks such as jQuery or Bootstrap. Assignments that appear to be the product of third-party development tools or frameworks (professor's discretion) will receive **0** points.

Directions

Review the requirements listed in the Assignment Requirements section
Create a website that meets all of the stated requirements
Complete the assignment before the due date (refer to the Course Schedule)
  - Note: Students will not submit anything to Canvas.

Assignment Requirements

Assignment Description

Students will create a payment processing web page using a combination of HTML, CSS and JavaScript. Like many websites that allow users to make payments online, this web page will offer users two (2) options for making a payment: credit card or PayPal. The fields and controls presented to the user depends on which payment option is selected. For example, if the credit card payment option is selected, only the fields and controls related to a credit card are displayed and the fields and controls related to PayPal are removed.

The following screenshots provide examples of the form with either the **credit card option** or the **PayPal option** selected.

Preliminary Tasks

- Several pictures have been preselected for use with this project
  - **Click here to download the pictures**
- Log onto the **Course Web Server**
- Create a folder called images in the **module2** folder
  - Store all of the pictures used in your web pages for this assignment in the **images** folder
    - Use relative URLs to access the pictures
- For all the web pages created for this assignment
  - Hyperlinks to pages outside the web server (i.e., Wikipedia.org) should open the linked pages in a new browser **tab** or **window**
    - All other hyperlinks should open the linked pages in the same browser **tab** or **window**
- An automatic 10-point (10%) **penalty** will be assessed for any **disorganized** pages.

Payment Processing Page (30 points)

Create a **Payment Processing** web page using the filename index.html in the **module2** folder.

- Give the page a descriptive title
- Display Enter Payment Information along the top of the page using an **<h2>** tag
- Create a **form** that includes the following **controls** (at a minimum):
  - Note 1: Use the form tag <form onSubmit="return validateForm();"> to create the form.
  - Note 2: The form is comprised of two (2) sections - **Payment Selection** and **Payment Information**.
  - Note 3: The credit card controls should be displayed in the Payment Information section by default.
  - Payment Selection section
    - Two (2) **radio buttons**
      - Configure the **onClick** event of each radio button with the statement "updateForm(this);"
      - Credit Card Payment (American Express, Discover, MasterCard and Visa)
        - Note 1: This option should be selected by default.
        - Note 2: Use the provided images rather than text to indicate the purpose of the radio button.
      - PayPal Payment
        - Note: Use the provided image rather than text to indicate the purpose of the radio button.
    - One (1) button
      - Submit
        - Note 1: The **submit button** should be placed at the bottom of the form (below the Payment Information section).
        - Note 2: Use the provided image rather than text to represent the submit button.
  - Payment Information section
    - Store the fields and controls within a single tag
      - Note: The Payment Processing form should contain only one (1) ** ** tag.

- **Credit Card Payment Controls**
  - Nine (9) **text fields**
  - Note: **All** text fields should be configured to require user input.
    - First Name
    - Last Name
    - Address
    - City
    - Zip
    - Email Address
    - Name on Card
    - Card Number
    - CVV2/CVC
      - Include a **hyperlink** to an article about CVV2/CVC on **Wikipedia.org**
  - One (1) **select field** (drop-down list)
    - State
      - The first <option> tag should contain the text Select State
        - The option should be selected by default
      - Populate the select field with the names of all 50 states
  - One (1) **date field**
    - Expiration Date
      - Limit the field to display only the month and year values of the selected date
      - Restrict the date range to **January 1, 2017** and **December 31, 2020**
      - Set the default value to **December, 2017**
- **PayPal Payment Controls**
  - One (1) **text field**
    - Email Address
      - This is a different text field; therefore, give it a unique name
  - One (1) **password field**
    - Password

- At the bottom of the page, include one (1) additional hyperlink to your **ePortfolio**
  - Use a relative URL for the **ePortfolio** link

External CSS File (10 points)

Modify the site.css file located in the **public_html/css/** folder to control the presentation of the **Payment Processing** page.

Note: You can use existing functionality when possible.

- Make the text of the <h2> header a color other than the default color
- Do not modify the **text size** of hyperlinks on this page
- Use custom **radio buttons** for the Payment Selection section,
  - Set the **background** of the selected radio button to green
  - Set the **background** of the unselected radio buttons to grey

External JavaScript File (50 points)

Create an external **JavaScript** file with the filename script.js in the **module2** folder.

- Import the **script.js** file into the HTML page
- Add the following functions to the **script.js** file. Use the information provided with the function to assist with creating the function.

Control	Data Type	# Digits	Related Function
Address	String		
Card Number	String	Various	validateCreditCard
City	String		
CVV2/CVC	Number	3	validateControl
Email Address	String		validateEmail
Expiration Date	Date		validateDate
First Name	String		
Last Name	String		
Name on Card	String		
Password	String		validatePassword
State			validateState
Zip	Number	5	validateControl

  - **testLength**(value, length, exactLength)
    - Inputs
      - value - the value to test
      - length - the required length of the value
      - exactLength - (boolean) true means value.length = length; false means value.length >= length
    - Process
      - Test whether the value is the correct length
        - If the test passes,
          - Return true
        - If the test fails,
          - Return false
    - Output
      - Return true **only if the test passes**; otherwise, return false

  - **testNumber(**value)
    - Inputs
      - value - the value to test
    - Process

        - Test whether the value represents a number
            - If the test passes,
                - Return true
            - If the test fails,
                - Return false
    - Output
        - Return true **only if the test passes**; otherwise, return false

- **updateForm**(control)
    - Inputs
        - control - the radio button control clicked
    - Process
        - Update the Payment Information section with the appropriate controls based on which radio button control was clicked
            - Hint: Update the innerHTML of the ** ** tag
            - View **Credit Card Payment controls** example
            - View **PayPal Payment controls** example
    - Output
        - none

validateControl(control, name, length)

- Inputs
    - control - the control containing the string value to test
    - name - the proper name of the control (i.e., First Name, Zip, etc.)
    - length - the required length of the control value
- Process
    - Test whether the control's value is the correct length
        - Call the testLength function
        - If the test fails,
            - Display an appropriate error message
            - Return false
    - Test whether the control's value represents a number
        - Call the testNumber function
        - If the test fails,
            - Display an appropriate error message
            - Return false
- Output
    - Return true **only if both tests pass**; otherwise, return false

- **validateCreditCard**(value)
    - Note: Refer to the credit card table below for important details about credit card numbers.

Card Type	1st Digit	Length
AmEx	3	15
Discover	6	16
MasterCard	5	16
Visa	4	16

- Inputs
  - value - the credit card string value to test
- Process
  - Remove any **spaces** from the value
    - 1234 5678 9012 3456 becomes 1234567890123456
  - Test whether the **credit card value** is the correct length (see table)
    - Call the testLength function
    - If the test fails,
      - Display an appropriate error message
      - Return false
  - Test whether the **credit card value** represents a number
    - Call the testNumber function
    - If the test fails,
      - Display an appropriate error message
      - Return false
  - Test whether the first digit of the **credit card value** represents a valid credit card type (see table)
    - If the test fails,
      - Display an appropriate error message
      - Return false
- Output
  - Return true **only if all three (3) tests pass**; otherwise, return false

- **validateDate**(value)
  - Inputs
    - value - the date value to test
  - Process
    - Test if value is greater than **today's date**
      - **Do NOT test** using the **current month**
      - If the test fails,
        - Display an appropriate error message
        - Return false
  - Output
    - Return true if the date value is **at least one (1) month greater** than today's date; otherwise, return false

- **validateEmail**(value)
  - Inputs
    - value - the email string to test
  - Process
    - Use a Regular Expression (RegEx) to determine if the string value conforms to a typical email address (i.e., **username@domain.com**)
      - If the test fails,
        - Display an appropriate error message
        - Return false
  - Output
    - Return true if the string value represents a **typical email address**; otherwise, return false

- **validateForm**()
  - Inputs
    - none
  - Process
    - Call each of the necessary functions **in some order** to validate the form's fields
      - Note 1: Only call the functions needed to test the fields related to the selected payment option.
        - Credit Card option: Test all of the fields except PayPal fields (Email Address and Password)
        - PayPal option: Test only the PayPal fields (Email Address and Password)
      - Note 2: Some functions may need to be called more than once.
      - validateControl
      - validateCreditCard
      - validateDate
      - validateEmail
      - validatePassword
      - validateState
    - **If all the tested functions return true**, display a message such as Payment Submitted
      - Credit Card option: Test five (5) functions
        - Some functions may need to be called multiple times
      - PayPal option: Test two (2) functions
  - Output
    - **Always** return **false**
      - Otherwise, the web browser may display an error message

- **validatePassword**(value, minLength)
  - Inputs
    - value - the password string to test
    - minLength - the minimum length of the string value

        - Process
            - Test if the string value is greater than or equal to the minLength value
                - Call the testLength function
                - If the test fails,
                    - Display an appropriate error message
                    - Return false
        - Output
            - Return true if the string value is **equal to or greater than** minLength; otherwise, return false

    - **validateState**()
        - Inputs
            - none
        - Process
            - Test whether the **Select State** option is currently selected
                - If the test fails,
                    - Display an appropriate error message
                    - Return false
        - Output
            - Return true if a valid state option is selected; otherwise, return false

ePortfolio Page (10 points)

Update your **ePortfolio** web page (index.html) in the **public_html** folder.

- Replace the text [student name] with your **first name**
- Create a hyperlink using the existing text Module 1 to open your **Personal** web page
    - Use a relative URL for the **Personal** web page link