Swinburne University of Technology
Faculty of Science, Engineering and Technologies

COS10011 / COS60004 / COS60007
Creating Web Applications [and Databases]

Assignment Part 2, Semester 1, 2017

Develop an interactive Website

Important Dates:

Due Date ESP	**Week 8 - Normal day of you tutorial: 25-28 April,8 am.** (Late submission penalty: 10% of total available marks per day)
Demonstration	Your tutorial: Week 8 *Note: due to the ANZAC day public holiday on Tuesday 25 April, students who normally have Tuesday Labs should demonstrate in Labs on 26-28 April.* *Extra demonstration Labs have been scheduled for: 10:30 Wed 26, 2:30 Thurs 27, and 12:30 Fri 28. Details will be announced on BB*

Contribution to Final Assessment: 14% (Hurdle)

Hurdle: You must meet the Essential Requirements of this assignment to be eligible to submit Part 3.
See the making guide for the *Essential Requirements* for this part of the assignment.

Purpose of the assignment

To gain practical skills and knowledge in coding and using JavaScript, by adding dynamic behavior and extending and enhancing the website you developed in Part 1. In particular, you will:

- Use JavaScript to validate data entered into HTML forms and provide user feedback,
- Understand the ways that data can be stored locally and be transferred between web pages,
- Use client-side data storage to transfer data between pages,
- Understand how, and check that, hidden form data is correctly passed to a server-side script.

As in Part 1, there is an opportunity to enhance your website beyond the basic requirements.

Prerequisite

If you failed to meet the Essential Requirements of **Part 1**, then you must demonstrate that you have fixed these problems ***before*** *this part of the assignment is submitted and marked.*
It is advisable to get these fixes completed and signed off *well before* you hand in this part of the assignment.
How to get your fixes signed off:

1. Arrange a time with your tutor to check your work during your allocated tutorial or during a consultation time.
2. Bring a copy of the ESP assessment printout from Part 1
3. Your tutor will check that your fixes to Part 1 address the issues identified under the Essential Requirements on the mark sheet.
4. If the fixes are successful, your tutor will record this and you will be eligible to have this assignment assessed. If there are issues that have not been fixed, your tutor will inform you of this and you will have *a further chance* to fix the assignment.

Note that any fixes will **not** alter the mark you received for Part 1.

Note: This assignment must be demonstrated in you Lab.
The code that is assessed in your demonstration *must be identical* to the code you submit to ESP. This will be checked before the demonstration.

Note: Do not use JavaScript libraries or frameworks (e.g. jQuery, Angular) in this assignment.

Web Site Description

Extend your Part 1 website by: modifying your existing **`enquire.html`** page; adding an additional **`payment.html`** page to enable a user to enter payment details for a selected product/service; adding JavaScript data validation to your forms; using JavaScript to store data client-side and transfer data between web pages, and correctly passing all the data to a server-side script.

Implement your Web site in standard HTML5; that is also well-formed XML; that is styled with valid CSS3; and that has dynamic behavior coded in 'strict'. JavaScript.

If you wish to make other alterations to the HTML and CSS in your Assignment Part 1 that is OK, but you must keep your assigned product/service topic.

1. Modify your enquire page (enquire.html)

Extend the form to enable a user to purchase a product/service that they have selected.
You will need to:

- Add some **price details** for various items in the product/service range, and add extra options where applicable.
- Add one or more fields so a user can select the **quantity** of the product/service to be purchased. *For a product, the quantity might represent the number of items. For a service, the quantity might represent the dates of hire, or the number of days.*
- Change the 'submit' to '**Pay Now'** with a link to the `payment.html` page, where the user can view their product/service selection and can enter their credit card details.
- The user should ***only*** be able to go to the payment page if all the necessary data in the form has been validated. A user should not be able to 'Pay Now' if they have not chosen a product/service, or the 'quantity' is inappropriate. *(See required validations later.)*

2. Product Payment page (payment.html)

This additional product/service payment page will:

- Display the customer's details, together with the product selection details that a user has entered in the form on the modified `enquire.html` page.
- Display the calculated total price of the purchase.
- Display a form that will allow the user to enter credit card payment details:
    a. Credit card type (e.g. Visa, Mastercard, American Express) – *no valid default selection*
    b. Name on credit card
    c. Credit card number
    d. Credit card expiry date (mm-yy)
    e. Card verification value (CVV)

When a customer decides to proceed with the order, a **Check Out** button should submit the form, and all the customer and product information to the server-side script. (So you check that all name-value data has been correctly submitted.)

A user should also have the ability to **Cancel Order**, and if they cancel, the stored data should be cleared, and they should be returned to the ***home page*** of the website.

Implement your website in standard HTML5, that is also well-formed XML.

Implementation of JavaScript `(part2.js)`

There should be ***no embedded*** and ***no inline*** JavaScript in your HTML files.
In other words, no event registration *(e.g.* `<form onsubmit="return validate()" ...` *)* and no function definitions within the HTML.

The `"use strict";` directive should be included at the start of ***all*** your JavaScript files.

Ideally, use a ***single*** JavaScript file named **`part2.js`** for all your scripts.

Data Transfer between Forms

In the `enquire.html` page, JavaScript and local or session storage will be used to store the entered data. This data will then be retrieved and displayed on the `payment.html` page in a *read-only* manner. These data values, together with the payment information the user enters into the payment form will then be sent to the server.

> **Note:** Change the **`enquire.html`** form you created in Part 1 of the Assignment and set the **`action`** attribute to **`payment.html`**. When the form is submitted, JavaScript will be used to store the data on the client. When the payment page is requested and returned from the server to the browser, the JavaScript will initialize it with the client-side local or session storage information, and display it in the page. None of the web pages will do any server-side processing of the form data, we will add server-side processing in Part 3.

Data Validation

In general, all necessary text input fields should be validated to ensure they are not empty, and other form controls should be checked to ensure at least one selection per group has been made by the user, unless that selection is in a group of optional features.

HTML5 validation was used for the `enquire.html` form in Assignment Part 1, and most of the new fields will be able to be validated with HTML5. However ***both forms*** in `enquire.html` and `payment.html` require some additional data validation using JavaScript.

If a field does not validate according to the data validation rules, meaningful feedback should be given to the user. Where JavaScript validation is used, combine multiple entry error messages into a single JavaScript alert message, or alternatively use JavaScript to display the error message in the page.

Specific data validation rules ***in addition*** to those defined in Assignment Part 1 are:

Enquiry and Selection page (enquire.html)

- Quantity of product must be a positive integer. (and a product must be selected)
- The selected state must match the first digit of the postcode
  VIC = 3 OR 8, NSW = 1 OR 2 ,QLD = 4 OR 9 ,NT = 0 ,WA = 6 ,SA=5 ,TAS=7 ,ACT= 0
  (e.g. the postcode **3122** should match the state **VIC**)

Product Payment page (payment.html)

- Credit card type must be one of Visa, Mastercard, or American Express (no vaild default)
- Name on credit card: maximum of 40 characters, alphabetical and space only
- Credit card number: exactly 15 or 16 digits
- Credit card expiry date: exactly 2-2 digits in the format mm-yy (where 01 ≤ mm ≤ 12).
- The card expiry date must be after the current date.
- Credit card numbers must be checked against the selected card type according to the following rules:
  - Visa cards have 16 digits and start with a **4**
  - MasterCard have 16 digits and start with digits **51** *through to* **55**
  - American Express has 15 digits and starts with **34** *or* **37.**
- Card verification value (CVV)
  - 3 digits, for Visa and Mastercard, 4 digits for American Express.

Data Submission to Server

For this assignment all forms should have a Submit button: `enquire.html` submits to `payment.html`, but ***only*** `payment.html` submits data to the server.

When submit is actioned on `payment.html` all the name-values, including customer and purchased product data, and payment details from the associated form should be sent to the server using the **post** http **method**. The server **action** address is https://mercury.swin.edu.au/it000000/formtest.php. This server-side script will allow you to test and check that the form data has been submitted correctly.

CSS

All pages should be styled appropriately using CSS as in Part 1, and should be valid CSS3.

Reflection *(optional)*

You might like to add a sub-heading to your 'Reflection' section on your `about.html` page, and include notes to yourself, about anything that might help you reflect on what you have learnt in Part 2, both now, and later when you revise your work. We will not be assessing this.

Enhancements using JavaScript

Please complete all the ***essential requirements*** before attempting any enhancements.

As with Part 1 you have an opportunity to extend your learning by adding extensions/ enhancements to the main pages of your Web site, using techniques not covered in the tutorials.
Each enhancement must be described on a page called **`enhancements2.html`** and the entries on this page should:

- briefly describe the interaction required to trigger the event ***and*** what a programmer has to do to implement the feature. ***(A reminder to your future self of what you have done.)***
- provide a hyperlink to the page where the enhancement is implemented in your Web site.
- reference any third party sources used to create the extension/enhancement

It is a good idea to discuss any proposed enhancements with your tutor before you implement them.

All JavaScript enhancements code should be in a separate **`enhancements.js`** file.
Include adequate comments in the code to explain the enhancement, including references/sources if applicable.

Examples of JavaScript and other enhancements you might create include, but are not limited to:

- On every page, dynamically style the menu item for the current page, based on the page filename.
- Use JavaScript methods that dynamically change style in response to user actions. *For example, using querySelector().*
- On enquire.html, have a checkbox that says "Billing address different from Delivery Address". When the user checks this, a separate set of Delivery Address inputs appears, and this is also added to the data stored and passed to payments.
- On payment.html, implement a timer so that the user only has a limited time to complete the payment after which a warning is displayed / then the browser redirects to the home page.
- On payment.html, pre-load the 'Name on Credit Card' as a concatenation of the firstname and lastname, into the input field, to enable a user to change the value.
- On product.html, have all your products/services as variables in JavaScript and dynamically display them on the product page.
- On enquire.html, set the selection of certain product/service options to disable the ability to select another sets of options. *For example of a motor car product, the selection of an "automatic transmission" might disable the ability to select between "4 speed" and "5 speed" transmissions.*
- On enquire and/or payments pages, utilise the JavaScript Form API, to enhance the HTML5 validation interaction, and feedback.

- Change the flow of the website, to a 3 stage process: by modifying product.html, so that is has a "Buy Now" under each product, then when actioned it stores and passed data to the enquire page, where a user can input their details and select quantity and any other product/service options, then pass the data to payment.

All enhancements must be listed and linked on the page **`enhancements2.html`,** coded in **`enhancements.js`** and implemented in the main webpages– otherwise they will not be assessed. Up to 6 marks will be allocated to each enhancement. A maximum of 3 enhancements will be assessed.

Web Site Folder Structure and Deployment Requirements

Your website folder structure should follow a similar structure as in Part1.
All files should be under a folder `/assign2`. JavaScript should sit in an `assign2/scripts` folder.

```
assign2/            You must have this folder – case sensitive!
  index.html
  product.html
  enquire.html
  about.html
  enhancements.html
  enhancements2.html
  ...other html pages
  scripts/          Folder for JavaScripts
     part2.js
     enhancements.js
  images/           Folder for images for your page content
  styles/
     style.css      other css files
  styles/images/    Folder for images referred to by your css files e.g. background
```

Notes:

- HTML files should only be in the base "assign2/" folder – not anywhere else.
- All links to your files (JavaScript, CSS or images) should be ***relative***.
  Do not use absolute links, as these links will be broken when files are transferred for marking. No marks will be allocated if links are broken.

Assignment Submission

An electronic copy of your assignment should be submitted through ESP at https://esp.ict.swin.edu.au on or before the deadline.

- Make sure all your files are in the correct folders and compress your root folder with all your sub-folders with HTML, CSS, JavaScript, and image files into a zip file named "**assign2.zip**". Submit this to ESP. When the zip file is decompressed, the entire Web site should be able to be run from index.html without needing to move any files.
  You can submit more than once through ESP.
- Note that all deliverables must be submitted electronically. There is no need to submit an assignment cover sheet as ESP generates a receipt upon successful submission.

Make sure you complete your ESP submission process.
You should get a PDF receipt if you have submitted successfully.

MAKE SURE YOU ENTER THE CORRECT UNIT CODE WHEN YOU SUBMIT YOUR ASSIGNMENT TO ESP
Submitting to the wrong unit will not be accepted as grounds for granting an extension.

Mark Sheet – Assessed by demonstration in your tutorial ***Marker: ……***

Declaration: **Fill this in before you start**

I hereby confirm that the assignment to be demonstrated is identical to that I submitted to

Student number ……. Student name ……..

Signature ……. Date ……

Product ……..

Tutorial Day …… Tutorial Time …….. Tutor Name ……...

A2

Marker checks:

Prerequisite – Essential Requirement in Assignment Part 1 - OK? ☐

ESP with receipt sighted ☐ *File date:* ☐

During the demonstration the marker will ask you about various aspects of your Web site, and ask you to explain the code you have used in the assignment.

Essential Requirements Tick box ☑ if requirement met	**Y/N**
enquire.html / JavaScript - validates to HTML5 ☐ and XML ☐ - price displayed ☐ - quantity input added ☐ - 'Pay Now' form submission links to payment ☐ - cannot submit form if errors ☐ - some JavaScript validation / error messages working ☐ - at least some data written to client storage ☐	
payment.html / JavaScript - validates to HTML5 ☐ and XML ☐ - at least some data read from client storage and displayed on web page, read only☐ - total price calculated / displayed ☐ - form credit card details provided ☐ - some JavaScript validation / error messages working ☐ - 'Check Out' submits to server-side script, values echoed back ☐ - 'Cancel Order' provided, destroys stored data. ☐	
CSS - CSS validates with no errors ☐ - External CSS applied to all HTML pages ☐	
Deployed to Mercury ☐	
(all Y)	

Specified Requirements Place ☑ or ☒ in box - 2 / 4 marks each tick	**Comment**	**Mark**
enquire.html / JavaScript *HTML: (2 marks each)* *(HTML5 validation errors - deduct 2 marks each, -12 max)* [ ] - price displayed ☐ - quantity input added – *must be positive integer* ☐ - 'Pay Now' form submission links to payment ☐ *JavaScript: (4 marks each)* *Embedded or inline JS (deduct 4 marks each)* [ ] - State / Postcode checks implemented ☐ - JavaScript validation / error messages working ☐ - all data in form stored in client-side storage ☐ - error messages in single alert, or displayed in page ☐ - cannot onsubmit form if errors ☐		/6 /20
product.html / JavaScript *HTML: (2 marks each)* *(HTML5 validation errors - deduct 2 marks each, -12 max)* [ ] - well coded display of read only data ☐ - calculated data displayed ☐		

- form hidden data well coded ☐ - form credit card details provided with HTML5 validation: *Credit card type (Visa, Matercard, AmEx)* ☐, *Name on Card (max40 alpha+space)* ☐, *Credit card Number (15-16 digits)* ☐, *Credit Card Expiry (2-2 digits)* ☐, *Card Verification Value* ☐, - Check Out ☐, Cancel ☐		/20
JavaScript: (4 marks each) *Embedded or inline JS (deduct 4 marks each)* [ ] - all client/product data read from client storage ☐ - all client/product data displayed on web page, read only ☐ - total price calculated / displayed ☐ - card type / card number / formats checked ☐ - card expiry checked (date greater than current date) ☐ - error messages in single alert, or displayed in page ☐ - cannot onsubmit form if errors ☐ - 'Check Out' submits to server-side script, **all** values echoed back ☐ - 'Cancel Order' destroys stored data, redirects. ☐		/36
Subtotal		**/82**

Enhancements to Specified Requirements *listed and linked* from ***enhancements2.html***

Maximum of 3 Enhancements will be assessed (put your best ones at the top of the list).
Up to 18 marks are available overall. Poorly implemented or trivial enhancements may receive zero marks.

Feature Name	Described	Linked (to place in website)	Sourced (if applicable)	Mark
	Y/N	Y/N	Y/N/na	
	Y/N	Y/N	Y/N/na	
	Y/N	Y/N	Y/N/na	
HTML: (deductions if invalid HTML5) [ ]				
CSS: (deductions if invalid CSS3) [ ]				
JS: (deductions if not 'strict') [ ]				
Total Additions				**/18**

Other Deductions *based on demonstration, documentation, code and file inspection*

Requirement	Max Deduction if requirements not met	Deduct
HTML *(deduct up to 3 marks each)* - Invalid XML - (-3 / page) **up to -6** - Deprecated elements/attributes have been used - Inappropriate use of HTML semantics (e.g. use of when <section> <article> should be used) - HTML usability does not follow standards (e.g. alt on images, label in forms, tables) - HTML Image height, width attributes missing or incorrect - HTML has embedded Style markup. CSS is not fully separated from HTML - Code comments inadequate to inform later code understanding/maintenance	-20	
JavaScript *(deduct up to 3 marks each)* - JavaScript is not in an external file - The required JavaScript is not in one single file - Inappropriate header comments - do not match in-house standard. - "use strict" directive not present - Code comments inadequate to inform later code understanding/maintenance	-12	
Web site - Directory and file structure not as specified - Third party content inadequately acknowledged *- Failure to acknowledge third party code or content at all is plagiarism and may result in zero marks for this assessment, or other penalties in accord with Swinburne policy.*	-4 -4 - All marks	
Total Deductions		

A final assignment mark is *not* provided during the demonstration.

All code is inspected after the demonstration by your tutor, before a final mark is allocated.

Comments: .....