Swinburne University of Technology

Faculty of Information and Communication Technologies

HIT1307/HIT6307
Internet Technologies
Assignment 2, Semester 1, 2010

Purpose of Assignment:
An ***individual*** assignment to familiarise students with the techniques and skills involved in developing a multi-page website with dynamic content using XHTML, forms, CSS, JavaScript and SSI.

The main focus of this assignment is for you to:

- develop webpages using valid, well structured **XHTML 1.0 Strict** mark-up;
- style webpages with valid, well structured **CSS**, using 'fluid' page layouts;
- use **JavaScript** and **regular expressions** for **checking form data**;
- use **JavaScript** and the **DOM** to add other page dynamics;
- demonstrate skills in loading and accessing files on a UNIX / Apache server;
- create and use **Server Side Includes** (**SSI**) to simplify common content management;
- demonstrate the use of other technologies (such as: Cookies, XML, XSLT, SVG, XLinks, etc.)
- **report** on the processes and techniques you have used and learnt, in a ***simple*** how to "report" page.

Due Date: **10am Friday 28 May 2010**
Submit as a single zip file via the online submission system (ESP) https://esp.ict.swin.edu.au/

- See also 'Submission Instructions' on later pages.
- Your assignment must work correctly when placed on the Apache server **mercury.it.swin.edu.au**.
- Check the subject website for discussions and for any announcements.

Contribution to Final Assessment: **12%**

Progressive Feedback:
As you progressively develop your assignment (Site Structure, XHTML, SSI, CSS, JS, Options) discuss your work with your Tutor during Labs, and gain useful progressive feedback.
See 'Unit Outline' for suggested timeline.

Assignment Requirements

Web-site Structure:

Create a website of *at least* **five** main webpages:

- Home Page `index.shtml`
- Form Page `form.shtml`
- Show Off Page `showoff.shtml`

} Your topic pages

- About Page `about.shtml`
- Report Page `report.shtml`

} Pages that explain what you have done

The topic / theme of your site can be anything you like, but the content must be appropriate for your form demonstration and form data checking, and any other demonstrations. eg. a fictitious small business, community group, etc.
NOT THE TOPIC used for Assignment 1 ☺

Each webpage should contain *at least* four common div sections, styled by CSS for your page layout design:

- Banner Section (`#banner`)
- Menu Section (`#menu`)
- Footer Section (`#footer`)
- Primary Content (`#main`)

Initially make all these files ***.html*** *then when you create your* ***ssi*** *files, link them as common content, and you load your files to the web server, change them to* ***.shtml***

Create and use three ***server side include*** files:

- `banner.ssi`
- `menu.ssi`
- `footer.ssi`

Note: Server-side 'include' statements need to be processed by a web server. So they will only work when they are placed on the web server, and requested via http from the browser. ☺

Your should also **create** and link the following files to support your website:

- External Stylesheet `style.css`
- External JavaScript `scripts.js`

Standards:

To simplify the assignment process, Firefox has been selected as the standard browser. This is to encourage the full use of XHTML and CSS, and to avoid issues related to different levels of support by other browsers. (Some of the required "features" may not show up in other browsers correctly, or at all. ☹)

- All HTML files, with the SSI content included, should be ***well-structured***, ***well-formed***, ***validated*** and ***'warning free'* XHTML 1.0 Strict.**
- Your pages should be ***well presented*** using ***valid*** and ***'warning free'* CSS 2.1** or **CSS 3**.
- Page Layout should be 'fluid', not 'fixed', and page layout should be done using CSS – ***do not*** use Tables or Frames for page layout.
- Coding quality should follow the 'Swinburne HTML Coding Standards' provided for this unit.
- All pages should address the principles of the W3C Web Content Accessibility Guidelines.
- JavaScript coding should follow the JavaScript coding guidelines – see later.

XHTML Requirements:

Your webpages should contain the following required HTML features – some features should be present on all your pages, other features need to appear somewhere (maybe once) in your site.

a. Features that should be on ALL of your HTML pages:

- **Each page** should have a **"menu" section**, with a menu of links to the other ***content*** pages in your site. This should be done as an unordered list of links (and appropriately styled).
  - Create a hierarchy of links for your content, with both page and #section links.
  - The "current page" link in the menu should look different. Use a class style and JS to do this?
- **Each page** should have a suitable **"banner" section** - with text, logo, slogan, etc.
- **Each page** should have a **"main" content section** with **headings** and **sub-headings** that are appropriate for the page content.
- **Each page** should have a **"footer" section** that contains "document control information" about who created the page, and when it was last updated.
  - The footer section should contain the **only links to** your **`about.html`** and `report.html` pages.
  - This footer section is often used to state if the site complies with any standards (such as XHTML or CSS). Include ***validation links*** that can be easily selected to enable us to quickly check that your work is valid. Note that "referrer links" will only work when your web pages are loaded to the server ☹
- **All pages** should show a **consistent use** of **heading levels**, paragraphs, sections, etc., and demonstrate good logically structured markup.

b. Features that must be demonstrated at least ONCE in your site:

- **Images for content.** (Not just for presentation – you should also use images for CSS presenatation).
  - Set meaningful 'alt' and 'title' attributes appropriately
  - Set the 'width' and 'height' values to improve layout performance.
- **A table for tabular data.**
  (eg. a table for comparing features, similar products/services, or for important dates / events.)
  - Must show your skills at creating well formed tables: summary attribute / caption element, table headings, scope / header attributes for accessibility, etc.
  - Display a range of table related or block level CSS styles for presentation.
- Lists (ordered or unordered as needed) and nested lists. (Other than the menu links)
- Links, both internal (to other pages and to internal #sections) and external (to other sites).
- Use CSS background images.

You are not limited to ***just*** these elements. You are limited to the elements permitted in XHTML 1.0 Strict.

- This means NO frames, and NO deprecated elements or attributes.
- Only use tables to present tabular data - which you must provide somewhere as a requirement (Table-based layout for a **form** should be avoided. Try styling 'labels' for presentation.)
- Only use `div` tags if there is a specific and sensible reason to do so.
  eg. It is okay to divide / chunk / group a heading and several paragraphs into a div block for presentation or positioning of, say a "banner", but don't use a div instead of just a paragraph or heading element.
- Only use `span` tags if there is a specific reason to do so.
  In particular, don't use span instead of meaningful `strong` or `em` elements.

CSS Requirements:

Individually create your own external stylesheet for your website. This should demonstrate your skills in CSS by including the following features – some features should be present on all your pages, other features need to appear somewhere (maybe once) in your site.

- Each page should use 'fluid' design layout - allowing the layout of the page to flexibly change if the browser window or browser text are resized.
- Your CSS should:
  - ***utilise and demonstrate a good range of selectors.***
    (eg. at least: element, class, ID, grouping, contextual, and link pseudo class selectors)
  - ***utilise and demonstrate a good range of styles***
    (eg. at least: font, text, box model, background, background images, layout styles)
  - apply these styling skills consistently to your headings, main sections, table, form, lists, etc.
- Try to simplify your CSS rules and avoid repetition.
- Use simple context rules. Avoid "bloating" your HTML with excessive ids / classes.
- All CSS rules should be in your external stylesheet file – no embedded or inline style should be used.
- Add some **comments** at the beginning of the CSS file to identify author and purpose
- Add some comments as needed to **briefly** explain the key purpose of any complex style rules.
  eg. /* Menu Style */ *Do not comment each individual rule and property/value pair !*
- Store all images used for your CSS in the **/style** sub-folder
- Add an appropriate 'title' attribute in the HTML 'link' element to your style sheet file, so the browser can display the 'title' through 'View'/'Page Style'.

Originality:
*We expect **your assignment** to contain **your own unique coded content** and demonstrate **your** good solid basic knowledge and skills in using XHTML, CSS and JS.*
*Your work should be **different** from other students work – both current and former students.*
You will be asked to explain similarities between your work and others.

Home Page `index.shtml`

This is the main introduction page for your website, so use it to explain your topic, and create a good first impression for visitors.

- Include in this page: **a simple client-side JavaScript dynamic effect**,
  eg. cycling images, DHTML, etc.

Form Page `form.shtml`

Include in the main content section of this page: a demonstration of your ***form creation*** and your ***form data checking*** skills using good XHTML, client-side ***JavaScript*** and ***Regular Expressions***.

The topic of the information that the form is seeking can be anything you wish, as long as it is appropriate for your form validation demonstration (eg. an order form, a purchasing form, a form providing feedback, a form to request brochures, free samples, etc)

- The form must contain, as a ***minimum***, the following form control elements:
  text input**, radio** input**, checkbox** input**, select** list**, textarea, submit, reset.**
- *Optionally,* include a **password** input, and **subgroups** for the select list.
- Use `fieldset` and `legend` elements to provide structure for the form
- Use `labels` with `for` attribute values, to label each of your form input elements.
- Use CSS to style and present the form appropriately.
- Do ***not*** use tables for form layout, use CSS.
- Set the form **method** to **post** and the **form action** to
  http://dev.ict.swin.edu.au/hit1307/demos/showpost.php
- *Optionally,* include a **hidden** input to set the "debug" value="true".
- Use **JavaScript** to check the form data input, and display ***feedback*** to the user.
  - Data checking can occur progressively during data entry, or on field change, and/or just on submit, but ***the form must not be able to be submitted unless the data is OK.***
  - Use **regular expressions**, if possible, to validate the data.

- You will be marked on the **quality** of your **data checking** and ***client side feedback*** to users.
    - Just showing an alert box is a start, but not good enough for a top marks. Give the user enough information to fill in the form correctly – don't just say "It's wrong" – explain what needs to be done!.
    - If the tutor marking the assignment cannot work out how to fill in the form, because it is not clear what is required, then you will lose marks because of poor design. ☹
    - The use of separate alerts for each data error can be very annoying, so consider building feedback messages into single alerts.
    - Perhaps alter the CSS presentation of the form elements to show the user what needs to be done – eg. use JS to change class names or change style (eg. hidden or visible display?).
- Demonstrate the principle of "**unobtrusive JavaScript**" – if the JavaScript is disabled, your form should still correctly submit data to the server-side script.

Show Off Page `showoff.shtml`

This is a free page related to your topic, where you could place some of the required features (tables, image content, lists, CSS, etc) and demonstrate some of your chosen "options

About Page `about.shtml`

The purpose of this page is for you to tell us about, and link to, the features of the website that you have created it. It should have seven sections:

- ***Student Details:*** *Who are you?* Your student ID's, student name, student email address with a **mailto** link - to your Swinburne student email account, not to gmail etc accounts!
- ***Introduction:*** *What is your webpage about?* A short statement about ***why*** you chose this content.
- ***Overall website and Page Structure:*** *How have you structured and styled your website?* **Very Briefly t**ell us ***how*** you have structured your website and common page sections. (Include a simple diagram?)
  Briefly tell us about ***how*** you have styled your website. (Range of selectors? Common classes? Common Ids? Grouping? etc.) Tell us about ***anything special*** that you have done, or attempted, in creating the site that we should know about.
- ***HTML Features:*** *How can we see all your features quickly?* Provide a list of links to an example of each of the ***required features*** in **a**, and **b** above, in the order as above, so we can quickly find them. Please also use the CSS3 selector `:target` with appropriate style, so we quickly see these features on the page.
- ***CSS Features:*** *How can we see all your CSS features quickly?*
  Explain / list where you have applied your CSS demonstrations - provide links if possible.
- ***JavaScript Features:*** How can we see all your JS features quickly?
  Explain / list where you have applied your JS demonstrations - provide links in possible.
- ***Optional Features:*** How can we see all the extra things you have done quickly?
  Explain/list and provide link to these features.

Report Page `report.shtml`

The purpose of this page is for you to **explain** and **reflect** on **what you have done** to get your **JavaScript** working; what you have done to test your pages for **Accessibility compliance, how you have implemented any "Options"**.

Your **"report"** should ***very briefly explain*** to others the **'how to'** - the ***processes*** and ***techniques*** you have used, so that others (and you in the future ☺) might be able to repeat what you have done in this assignment. Think of the page as an brief training resource for others, and a report you can come back to in the future.
ie. Provide a concise **"report"**, that ***very briefly*** and ***succinctly*** describes, how you have created, tested, debugged your JS, validated your work against WAI / WCAG, implemented the "options".

The page should have three sections:

- **JavaScript:** This very briefly explains: which functions do what; how you have linked your JavaScript to your HTML objects; how your JavaScripts are triggered; and the resources used.
- **WAI / WCAG:** Very briefly explains what you needed to do to get your webpages to address the WAI / WCAG Guidelines, and the resources used.
- **Options:** Briefly explain how you have implemented any features, solved problems, and the resources used.

Linked External Stylesheet `style.css`

- All CSS rules should be in this external file – no embedded or inline style should be used.
- Add some **comments** at the beginning of the CSS file to identify author and purpose
- Add some comments as needed to **briefly** explain the key purpose of any complex style rules.
- Demonstrate a full range of style rules: element, id, class, box model style.
- Try to simplify your CSS rules and avoid repetition.
- Avoid "bloating" your HTML with excessive id / classes, use simple context rules.
- **Option:** If you **optionally use any valid CSS 3** properties / values that are supported by Firefox, then list and link this in your `about.shtml` page, telling us what you have done, so the marker can validate your CSS against CSS 3, and reward you for your extra effort.

Linked External JavaScript `scripts.js`

- You must have at least one JavaScript file, **`scripts.js`**, for your client side JavaScript functions - no embedded or inline JavaScript should be used.
- **Except** for form **'onsubmit'** and page **'onload'** attributes, **do NOT use other attribute based 'onevent' handlers to call your JavaScript functions.**
- If possible, aim for good separation of markup and event handling:
  - Use the DOM to connect events with your functions;
  - Use event listeners and event handlers in your external JavaScript.

JavaScript Coding Standards

- Place all the functions you create and use in the common "`scripts.js`" file.
- Only use script in your HTML files to call the functions you have stored in the external file.
- When checking form data, try to show a single dialog box that summarises all the incorrect form data.
  The use of separate alerts for each error can be very annoying to the user. ☹
- If your scripts handle ***mouse events***, also ensure they handle the similar ***keyboard events***.
- Formatting and Syntax: Adopt a syntax and style and stick to it consistently for all your code.
- Comments: If you are familiar with a commenting standard, try to apply it.
  Help us understand your code!
  Ask your tutor for suggestions if you want some pointers on commenting and formatting.

File Header Comments: Include a header comment block such as … (exact format can vary)

```
/**
 * Author: (Your name and student ID)
 * Purpose: This file is to…
 * Created: ??
 * Last updated: ??
 * Credits: (Any guidance/help/code? Credit it here.)
 */
```

Function Comments: Before each function, add a method comment block to tell us what it does. Include single line comments in functions to explain what's happening.

```
/**
 * Perform "world domination". Requires a large amount of money
 * as a parameter. Assumes a small world. Returns a lunchbox.
 */
function DominateWorld(money)
{
  // Spend money on lunchbox etc ...
}
```

Suggested List of Possible "Options"

Before attempting any options. complete the detailed requirements of this assignment first.
In other words, make sure that all the basic html, css, js, ssi, reporting requirements are met - with no negatives - before adding in 'options'. (ie. aim for the full 9/9 marks on basics first, before attempting 'options'.)

As you add options, list and link them in the 'about' page, and explain them in the 'report' page..
Simple options get 1 or 2 marks each. Maximum of 4 marks for implementing optional features.
The maximum possible mark for the assignment cannot exceed 12 marks total.

Options are not restricted to the suggestions in the list below, and not all suggestions are worth the same amount of marks. The marks awarded will depend on the quality and difficulty of the tasks your try.

Just adding a single feature does not get you full marks - you need to display the feature; use good techniques/practices; and list and explain in your "report" page how we could do it too.

- Provide a second CSS file, that displays a very different page presentation layout for the whole website. Provide and use JavaScript to ***randomly*** decide which stylesheet to display when the page is loaded.
- Use CSS to transform a list of links into an 'image map'
- Create a page(s) in XML, and transform it client-side with XSLT.
- Use SSI to show in the footer, "last modified" date of *each* page. *(Not as easy as it might first appear)*
- Use valid W3C CSS 3 properties / values that are supported by Firefox.
  - Just adding CSS 3 that doesn't work will not score any marks ☹
- Create a valid Compound Document, using another xmlns such as SVG, XLinks, MathML, etc.
- Create or include an ***extra*** client side **JavaScript dynamic effect**. eg. random image displayed onload, cycling images, etc. The code and structure of this is open, but must be documented and explained as clearly as possible. (***Not*** just a repeat of the required effect on index.html ☹)
- Create or include an ***extra*** simple **client side JavaScript game, calculator, currency converter, etc.** The code and structure of this is open, but must be documented and explained as clearly as possible.
- Use **cookies and JavaScript** to store and display user information.
  (Set cookie on one page, get cookie on another)
- Create a client-side multiple choice quiz, using JavaScript.
  Provide users with feedback on how many questions they got right (or wrong).
  Hide and show answers afterwards using DHTML ?
  If you wish, you could use some of the w3schools.com multiple choice questions as content.
- Use XML and XSTL to transform data into a structured valid XHTML page.
  Possibly use to structure / display a quiz ?
- Use AJAX (or JSON based) to help ***assist*** the completion of some form input fields.
- Add a Web Service to a webpage.
- Write an alternate PHP script to *receive* and display the data content of your form.
  (Name this `receive.php`)
- Write an alternate PHP script to *receive* and *validate* the data content of your form.
  (Name this `receive.php`) Suggest using the same regular expressions you use in your client-side validation script.
- Set up a session between the user and the server using PHP.

Remember to list and link in the "about" page, what options you have included.

Remember to provide in the "report" page details about how we could do it too.

Standards and Submission

1. **Coding Standards:**
   - **Use the Swinburne HTML Coding Standards.**
     The standards can be downloaded from the subject web site. Importantly, all pages must have:
     - an appropriate **title** in the head element.
     - the required HTML header **comments**,
     - **meta** elements for author, the page description, and for keywords about the page content,
       ... see the standards for full details!

   - **Layout and Presentation:**
     - Use CSS and logical markup.
       Do not use physical markup elements or physical attributes.
     - *Do not use* the `<font>`, `<b>`, `<u>` or `<i>` elements
     - Only use Tables for tabular data and nowhere else.
     - Only use the external file `style.css` for your CSS information.
       Do not use inline or embedded style.

   - **HTML Validation:** All your HTML files must be **Valid XHTML 1.0 Strict**.
     - The W3C HTML online validation software is used to check your HTML files.
     - Update the comments in your HTML page with the **date** your page was last validated.
     - We use the W3C HTML validator located at http://validator.w3.org/ to test your pages.

   Note: Tidy is good to help you "Tidy" your HTML, but Tidy is not a full validator, and so should not be used as your final HTML validation check as it may miss some important errors!

   - **CSS Validation:** Your **`style.css`** file must be **Valid CSS**.
     - We use the W3C CSS validator located at http://jigsaw.w3.org/css-validator/

   - **Browsers:** The assignment will be marked using the latest version of Firefox.
     This is to encourage and reward XHTML and CSS standards compliant work.

   Note: *Check that your work is **usable** in Internet Explorer (IE). If you fix issues, explain it in the 'report' page, however do not be too concerned about display issues (differences) for this assignment. Unfortunately, IE is not fully CSS 2 compliant.*
   Be aware of the issues in adding an xml declaration, as IE may go into 'quirks' mode.

2. **Submission Instructions:**
   - ***You will need to zip up a copy of your complete assignment files and upload it to ESP***
   - ***We will unzip your files onto mercury for marking to make sure that you have the correct folder and file structure.***
   - ***You will need to test your assignment on mercury***
     (ie. only use relative links, file names case sensitive, etc).
   - Create a folder called "**`assign2`**" for all your assignment files.
   - Place all you .shtml and .ssi files in this folder
   - Create a sub-folder called "**`images`**" and place all images used for content in this folder.
   - Create a sub-folder called "**`style`**" and place all style related files in this folder.
   - You may create a sub-folder called "**`scripts`**" and place all js file(s) in this folder.
     - *Please do not* create any other folders.
   - Compress the folder "assign2" into a **single zip or rar file**.
   - Do not include any file that is not part of your assignment. eg. please remove "thumbs.db" files!
   - Rename the zip file with the **7 digits of your student ID number**.
     - eg. Rename to "1234567.zip" or "1234567.rar".
     - This is the single file you need to upload through the ESP system.

Further submission instructions may be placed on the BlackBoard Subject Web site *if required*.

- Late penalties are automatically applied (10% per day).
- More than five days late results in a zero (0%) mark.
- Please submit well before the due time to avoid any problems.
- Results published through ESP are always provisional and subject to moderation.

Marking Scheme:

The following is an ***indication*** of the marking scheme used.

Marks will be awarded for completing the required tasks. ☺
Marks will be deducted for poor quality, not meeting requirements, or not following guidelines. ☹

The assignment is out of 12 marks and contributes 12% to your final subject result (subject to moderation).

Marks awarded for completing the required tasks. ☺

12 marks maximum

3 marks **Good XHTML Strict, Good CSS rules, SSI working, overall website**
- Overall site structure (files/folder names)
- Overall page structure (banner, menu, footer, SSIs, main)
- Valid XHTML, Valid CSS, Swinburne Coding Standards. (all pages)
- All required features provided. (table; nested lists; form with: all form controls,fieldset legend, labels; form links to script)
- All required CSS features provided and consistently applied (selectors: element, class, id,grouping,contextual, pseudoclass. styles: font,text,box model,background, images, layout)

3 marks **Good JavaScript**
- Javascript dynamic effect provided. (index page)
- JavaScript form data validation / feedback provided. (form page)
- Regular expressions used for form data verification
- Unobtrusive JavaScript used? (Test with JS off. Form submits without JS)
- All code in external JS file, good formatting and comments; avoids repetition, etc
- Only uses 'onload' 'onsubmit' event attributes inline. (DOM and JS for others)

3 marks **Good "about" page and "report" page**
- All details provided
- 'about' – 7 sections: lists and links working, tells us 'what' has been done.
- 'report' – 3 sections: explains and reflects what has been done.
- Describes enough detail to explain how the author created, tested, debugged and QA'd work.
- Good information, written in a clear and accessible manner.
- Overall Mark based on quality.

4 possible **"options"** marks (depending on difficulty and quality)
- In general, simple options get 1 or 2 marks.
- Involved features may get more, but not 4 marks just for one feature.
- Would need a combination of features to get the full 4 marks.
- Must be listed / linked in 'about' page, and described in 'report' page

Marks deducted for failure to use the required standards. ☹

We don't wish to focus on negatives, but this list may help you realise what is important and help you avoid common mistakes:

XHTML and CSS and Site Structure:

-2 marks	*invalid XHTML 1.0 Strict, invalid CSS, poor code quality*
-2 marks	*for not having the minimum pages/files (and named correctly).* `index.shtml, form.shtml, showoff.shtml, about.shtml report.shtml, banner.ssi, menu.ssi, footer.ssi, style.css, scripts.js.` *Note: the file extensions used for the web pages will depend on the technology you have used eg .html .shtml, .php etc.*
-2 marks	*poor form coding (form missing required form control elements, or not passing data to the server-side script);*
-1 mark each	missing any required features; bloated CSS / excessive use of id's / classes in your XHTML; not providing requested divs with id's; poor usability / accessibility (unexplained CSS warnings / WCAG warnings.); not following the Swinburne HTML Coding Standards

Yes - there are more negatives than positives if you have poor XHTML or CSS

Form and JavaScript:

-1 mark each	using inline JS code sections; poor JS coding standards (poorly formatted, not commented); not using regular expressions as part of data validation; not avoiding repetition in code; using event attributes within XHTML (except onsubmit / onload);

"about" and "report" pages

-2 marks	features not listed and linked; poor explanations; not including important details needed by another person to be able to repeat the JS and Options you have done.
-1 mark	not including references links to resources, help sites you have used.
-1 mark	poor spelling, grammar, punctuation