

## Week 5 Tasks for Submission

This document contains a number of tasks for you to attempt.

There are three types of task:

- Tasks labelled as "no submission required, not part of portfolio"
  - They are for you to attempt and practice.
  - They provided assistance for you to develop solutions to other tasks that are part of your portfolio
- Tasks labelled as "PASS Submission Task"
  - These tasks are part of your portfolio geared towards all students
  - The solutions to these tasks must be uploaded for marking by your tutor
- Tasks labelled as "CREDIT Submission Task"
  - These tasks are part of your portfolio geared mainly towards students aiming for more than a pass grade. Of course students aiming for a pass grade may attempt and submit these tasks.
  - The solutions to these tasks must be uploaded for marking by your tutor

## JavaScript Tasks

### Task 1. (no submission required, not part of portfolio)

Create a HTML file named **w5a.html** based on **template\_upd.html**

- Change the title to Validating input, description "textbox input tested" and update other meta tags as applicable.
- Add a script element which specified a file named w5a.js within the **src** attribute.
- Inside the <form> section place the following:  
**<label>Your name <input type="text" id="firstname" /></label>**  
**<button type="button" id="send"> Submit </button>**
- After </form>, place the following  
**<p id="msg"></p>**

Create a JavaScript file named **w5a.js** based on **template.js**

Within init():

- Create a local variable **btn** and assign document.getElementById("send") to this variable.
- Assign call to the function acceptName to the onclick event of the btn variable, i.e.  
btn.onclick=...;

Create a function acceptName(). This function does not need any parameters.

Within acceptName():

- Create a local variable **first** and assign the value entered into the textbox to this variable:  
document.getElementById("firstname").value
- Convert the name to all uppercase letters and store the converted name back in the variable **first**.
- Create a local variable **textout** and assign document.getElementById("msg") to this variable.
- Use innerHTML to display a string stating "Greetings, <first>" – replace <first> with the name converted to uppercase.

Test w5a.html

## Task 2. (no submission required, not part of portfolio)

Create a HTML file named **w5b.html** based on **template\_upd.html**

- Change the title to Validating input, description “textbox input tested” and update other meta tags as applicable.
- Add a script element which specified a file named w5b.js within the **src** attribute.
- Inside the `<form>` section place the following:  
`<label>Contact phone <input type="text" id="phone" /></label>`  
`<button type="button" id="send"> Submit </button>`
- After `</form>`, place the following  
`<p id="msg"></p>`

Create a JavaScript file named **w5b.js** based on **template.js**

Within `init()`:

- Create a local variable **btn** and assign `document.getElementById("send")` to this variable.
- Assign call of the function `acceptInput` to the `onclick` event of the `btn` variable, i.e.  
`btn.onclick=...;`

Create a function `acceptInput()`. This function does not need any parameters.

We will get back to this function later. First we need to validate the phone number as per the following rules:

- Length should be exactly 10 characters
- All characters should be digits

Create a function `validatePhone()` which takes one parameter – phone number, e.g. `phone`. This function will return `true` if the value is valid and `false` if invalid.

Within the function `validatePhone()`:

- Create a local variable **valid** and assign **true** to it.
- Use **if** statement to check that length is not 10 characters: `phone.length != 10` and if yes, assign **false** to **valid**.
- Else means that the number of characters in the string is correct so we need to check second validation rule: whether all characters are digits. So inside the **else** branch:
  - Use another **if** statement to check that all characters in the phone string are digits. The easy way of checking is to use `isNaN()` function as shown below:  
**if (isNaN(phone))**  
In this case assign **false** to **valid**.

Note: **isNaN()** returns **true** if there are non-digits in the string.

- Last statement is to return **valid**.

Now back to `acceptInput()`.

Within `acceptInput()`:

- Create a local variable **phoneNo** and assign the value entered into the textbox to this variable.
- Create a local variable **isPhoneValid**.

- To validate **phoneNo**, call the function `validatePhone()` and store the returned value in **isPhoneValid**
- Use `if(isPhoneValid)` to display “Phone number valid” message in the paragraph with the id `msg` and “Phone number invalid” otherwise.

Test `w5b.html`. Make sure you test:

- all digits but less than 10 characters
- exactly 10 characters but not only digits
- less than 10 characters and not only digits
- all digits and exactly 10 characters

### Task 3. (no submission required, not part of portfolio)

Create a HTML file named **w5c.html** based on **template\_upd.html**

- Change the title to Validating input, description “textbox input tested” and update other meta tags as applicable.
- Add a script element which specified a file named `w5c.js` within the **src** attribute.
- Inside the `<form>` section place the following:  
`<label>Subject mark out of 100 <input type="text" id="score" /></label>`  
`<button type="button" id="send"> Submit </button>`
- After `</form>`, place the following  
`<p id="msg"></p>`

Create a JavaScript file named **w5c.js** based on **template.js**

Within `init()`:

- Create a local variable **btn** and assign `document.getElementById("send")` to this variable.
- Assign call of the function `acceptInput` to the onclick event of the `btn` variable, i.e. `btn.onclick=...`;

Create a function `acceptInput()`. This function does not need any parameters.

We will get back to this function later. First we need to validate the subject mark as per the following rules:

- The input should be digits only and if yes,
- The input should be in the range 0-100

Note there is no point checking the range if input contains not only digits.

Create a function `validateMark()` which takes one parameter – subject mark, e.g. `subjMark`. This function will return a string with the relevant error message or an empty string if the mark is valid.

Within `validateMark()`:

- Create a variable **errorMsg** and assign an empty string to it.
- Check if **subjMark** is not a number if `(isNaN(subjMark))` – note 2 closing brackets.
- If true assign a string “Mark must be numeric” to **errorMsg**
- Otherwise
  - Convert `subjMark` to a number using `Number()`
  - Use **if** to check that `subjMark` is NOT within 0-100 range. In this case assign the relevant message to **errorMsg**. This second **if** does not need the **else** branch
- This function should return **errorMsg**

Create a function `markToGrade()` which takes one parameter `subjMark`, checks whether it's <50 and returns fail if true or pass if false.

Within `acceptInput()`:

- Create a local variable **mark** and assign the value entered into the textbox to this variable.
- Create a local variable **message**.
- To validate **mark**, call the function `validateMark()`, pass the necessary argument and store the returned value in **message**.
- Check if `message==""`, meaning there are no errors and the mark is valid, in this case we want to get the corresponding grade so
  - call the `markToGrade()` function with the relevant argument and store the returned value in **message**
  - We do not need *else* branch here.
- Display the message on the HTML page in the paragraph with id **msg**

Test `w5c.html`. Make sure you test:

- Non-digital input
- Input that is negative
- Input above 100
- Valid mark resulting in pass
- Valid mark resulting in fail

#### Task 4. (PASS Submission Task)

Create a **HTML** file named **w5P.html** based on **template\_upd.html**

- Change the title to "Postcode validation", description "Pass level task" and update other meta tags as applicable including **your student name and ID**.
- Add a script element which specified a file named **w5P.js** within the **src** attribute.
- Inside the `<form>` section:
  - Use `<label>` and input tag to display Postcode: text and a textbox with id `postcode`
  - Use `<button>` to create a button with id `check` "check"
- After `</form>`, place an empty paragraph with id `"msg"`.

Create a JavaScript file named **w5P.js** based on **template.js**

Within `init()`:

- Create a local variable **btn** and assign `document.getElementById("check")` to this variable.
- Assign call of the function `validateInput` to the `onclick` event of the `btn` variable, i.e. `btn.onclick=...`;

We will get back to this function later. First we need to validate the postcode as per the following rules:

- Length should be exactly 4 characters
- All characters should be digits

Create a function `validatePCode()` which takes one parameter – postcode. This function will return true if the value is valid and false if invalid.

Within the function `validatePCode()`:

- Create a local variable **valid** and assign **true** to it.
- Use **if** statement to check that length is 4 characters and if not, assign **false** to **valid**.
- Inside the **else** branch use another if statement to check that all characters in the postcode string are digits. Use `isNaN()` as shown earlier in the task for validating a phone number, i.e. if `isNaN(...)` **valid** should be assigned **false**.  
We do not need *else* branch here.
- Last statement is to return **valid**.

Within `validateInput()`:

- Create a local variable **pcode** and assign the value entered into the textbox to this variable.
- Create a local variable **isPCodeValid**.
- To validate **pCode**, call the function `validatePCode()` and store the returned value in **isPCodeValid**
- Use `if(isPCodeValid)` to display “Postcode valid” message in the paragraph with the id `msg` and “Postcode invalid” otherwise.

Test `w5P.html`. Make sure you test:

- all digits but less than 4 characters
- exactly 4 characters but not only digits
- less or more than 4 characters and not only digits
- all digits and exactly 4 characters

**Screen Capture** the **HTML page** displayed in your browser and paste into **W05P.DOCX**.

**Copy and Paste** the **HTML code** (or screenshot of your code) from your code editor into **W05P.DOCX**

**Copy and Paste** the **JavaScript code** (or screenshot of your code) from your code editor into **W05P.DOCX**

### Task 5. (CREDIT Submission Task)

Make a copy of `w5P.html` and rename it as `w5C1.html`.

- Change description to Credit level task

Make a copy of `w5P.js` and rename it as `w5C1.js`

Within `w5C1.js` create a function `pcodeToState()` that takes one parameter – postcode and determines and returns the Australian state or territory based on the first digit of the postcode. Note the list below is not comprehensive.

First digit	State or territory
2	NSW
3 or 8	Victoria
4 or 9	Queensland
5	South Australia
6	Western Australia
7	Tasmania
0	Northern Territory

Inside pcodeToState():

- Create a local variable **state**.
- Create a local variable **firstDigit** and assign the first character of the postcode string to it. The most suitable function to use here is `charAt(0)` – the first character is at position 0.
- Use **switch** to assign the value to **state** based on **firstDigit**.
- Return **state**

Within validateInput():

- Create a local variable **ausState**.
- In the statement `if(isPCCodeValid)` call the function `pcodeToState()` passing the relevant argument to it and store the returned value in **ausState**
- Instead of displaying "Postcode valid", display the state or territory corresponding to the postcode.

Test w5C1.html. Make sure you test:

- At least one value resulting in the error message
- Every first digit option resulting in every state or territory (total 9 screenshots of valid inputs).

**Screen Capture** the **HTML page** displayed in your browser and paste into **W05C.DOCX**.

**Copy and Paste** the **HTML code** (or screenshot of your code) from your code editor into **W05C.DOCX**

**Copy and Paste** the **JavaScript code** (or screenshot of your code) from your code editor into **W05C.DOCX**

### Task 6. (CREDIT Submission Task)

Create a **HTML** file named **w5C2.html** based on **template\_upd.html**

- Change the title to "Expiry date validation", description "Credit level task" and update other meta tags as applicable including **your student name and ID**.
- Add a script element which specified a file named **w5C2.js** within the **src** attribute.
- Inside the `<form>` section:
  - Between `<p>` tags put text "Credit card expiry date"
  - Use `<label>` and input tag to display "Month:" text and a textbox with id `monthNo`
  - Use `<label>` and input tag to display "Year:" text and a textbox with id `year`
  - Use `<button>` to create a button with id `check` "check"
- After `</form>`, place an empty paragraph with id `"msg"`.

Create a JavaScript file named **w5C2.js** based on **template.js**

Within `init()`:

- Create a local variable **btnCheck** and assign `document.getElementById("check")` to this variable.
- Assign call of the function `validateExpiry` to the `onclick` event of the **btnCheck** variable.

Create a function `validateMonth()` that takes one parameter **month** and returns true if month is valid and false if invalid.

Within `validateMonth()`:

- Create a local variable **valid** and assign **true** to it
- Check if **month** contains all digits (use `isNaN()`). If yes, assign **false** to **valid**.
- Within the else branch:
  - use `Number()` or `parseInt()` to convert **month** to a number

- Check whether month is outside 1-12 range, if yes assign **false** to **valid**.  
You do not need *else* branch.
- The function should return **valid**.

Create a function `validateYear()` that takes one parameter `yr` and returns true if year is valid and false if invalid.

Within `validateYear()`:

- Create a local variable **valid** and assign **true** to it. Remember to return this variable at the end of the function.
- Check if `yr` does not contain all digits (use `isNaN()`). If yes, assign **false** to **valid**.
- Within the *else* branch:
  - use `Number()` or `parseInt()` to convert `yr` to a number
  - Get current date using `Date()`, e.g. `var d= new Date()`.
  - Business rule: a year is valid if it is between current year and no more than 5 years into the future (i.e. current year + 5). Use `d.getFullYear()` to determine current year. Check if `yr` is NOT within the valid range and update variable **valid** accordingly.

Within `validateExpiry()`:

- Create local variables and store user input from textboxes with IDs `monthNo` and `year` in those variables.
- Create a local variable **output** and assign an empty string to it.
- Create a local variable **validMonth** and assign the call to the function **validateMonth()** to it passing the correct argument.
- Create a local variable **validYear** and assign the call to the function **validateYear()** to it passing the correct argument.
- Check if **validMonth** and **validYear** are true simultaneously. If yes, assign "Expiry date is valid" to the variable **output**, otherwise assign "Expiry date is invalid".
- Display the output.

Place the **Screen Captures** of the **web page** in your browser and paste into **W05C.DOCX**  
**Copy and Paste** the **HTML code** and **JavaScript code** from your code editor into **W05C.DOCX**

# HTML Tasks

## Task 7.(no submission required, not part of portfolio)

- Create a HTML file named **w5b.html** based on template\_upd.htm
- Change the title to Testing flexbox, description “Containers, flexboxes” and update other meta tags as applicable.
- Add the following to the .html file <head> section:  
`<link rel="stylesheet" href="w5b.css">`
- Add the following to the .html file <header> section:  
`<h1> A flexbox example</h1>`
- Add the following to the .html file <article> section:  
`<section class="flex-container">`  
`<p> Replace paragraph text with your own text. Replace paragraph text with your own text. </p>`  
`</section>`
- Create a CSS file named **w5b.css** in the same folder.
- Add the following to the .css file:  
`.flex-container {`  
`display: flex;`  
`display: -webkit-flex; /* for safari browser */`  
`background-color: pink;`  
`}`
- View the html file in your Web Browser.
- Resize the web page.  
The shape and size of the flexbox will alter.

### A flexbox example

Replace paragraph text with your own text. Replace paragraph text with your own text.

### A flexbox example

Replace paragraph text with your own text. Replace paragraph text with your own text.



**Task 8.(no submission required, not part of portfolio)**

- Modify the file named **w5b.html**
- Duplicate the <section> in the html file.

```
<section class="flex-container">
```

```
  <p>Replace paragraph text with your own text. Replace paragraph text with your own text. </p>  
</section>
```

```
<section class="flex-container">
```

```
  <p>Replace paragraph text with your own text. Replace paragraph text with your own text. </p>  
</section>
```

- Refresh the web page:  
Two flexboxes are displayed.

**A flexbox example**

Replace paragraph text with your own text. Replace paragraph text with your own text.

Replace paragraph text with your own text. Replace paragraph text with your own text.

- Add a **margin** between the flexboxes by altering the .css and adding a margin attribute to the flex-container:

`margin: 10px;`

**A flexbox example**

Replace paragraph text with your own text. Replace paragraph text with your own text.

Replace paragraph text with your own text. Replace paragraph text with your own text.

- Resize the page:

**A flexbox example**

Replace paragraph text with your own text. Replace paragraph text with your own text.

Replace paragraph text with your own text. Replace paragraph text with your own text.

**Task 9. (no submission required, not part of portfolio)**

We are about to nest flexboxes (i.e. place flexboxes within another flexbox).

Nested flexboxes are a very powerful way to layout web pages.

We will refer to the outer flexbox as the **flex container**. This will be the *parent*.

The inner flexboxes are referred to as **flex items**. They are considered to be *children*.

- Modify the file named **w5b.html**
- Add this code to the html file:  
Please take note of the **class names**, as getting these correct is crucial.

```
<header><h1> A flexbox example</h1></header>
<article class="flex-container">
  <section class="flex-item">
    <p>Replace paragraph text with your own text. Replace paragraph text with your own text. </p>
  </section>
  <section class="flex-item">
    <p>Replace paragraph text with your own text. Replace paragraph text with your own text. </p>
  </section>
</article>
```

- Add this code to the **w5b.css** file:  
.flex-container {  
 display: flex;  
 display: -webkit-flex; /\* for safari browser \*/  
 background-color: pink;  
}  
  
.flex-item {  
 /\* apply styles as needed \*/  
 background-color: lightblue;  
 margin: 10px;  
}

- Load the web page in a browser

As you can see, the flex-items with the blue background are displayed within the parent flex-container with the pink background.

Note, default display is horizontal, i.e. row.

---

### A flexbox example

Replace paragraph text with your own text. Replace paragraph text with your own text.

Replace paragraph text with your own text. Replace paragraph text with your own text.

**Task 10. (no submission required, not part of portfolio)**

- Modify the file named w5b.css  
Add the flex-property to the flex-container rule in the CSS file

```
.flex-container {
    display: flex;
    display: -webkit-flex; /* for safari browser */
    background-color: pink;
    flex-direction: column;
}
```

- Refresh the webpage in the web browser.

**A flexbox example**

Replace paragraph text with your own text. Replace paragraph text with your own text.

Replace paragraph text with your own text. Replace paragraph text with your own text.

**Task 11. (no submission required, not part of portfolio)**

- Modify the file named w5b.css  
Change the flex-direction property in the CSS file to have the value **row**  
We are back to the default flex-direction, although this time we explicitly specified it.

**A flexbox example**

Replace paragraph text with your own text.  
Replace paragraph text with your own text.

Replace paragraph text with your own text.  
Replace paragraph text with your own text.

**Task 12. (no submission required, not part of portfolio)**

Each <section> in the html file can have its own headings, paragraphs, images, etc.

- Alter the HTML file.  
Within each <section> with the class name "flex-item", add a <h2> element.  
The heading can be whatever you wish.
- Refresh the webpage in the browser.

**A flexbox example****Heading A**

Replace paragraph text with your own text.  
Replace paragraph text with your own text.

**Heading B**

Replace paragraph text with your own text.  
Replace paragraph text with your own text.

**Task 13. (PASS Submission Task)**

- Create the file named **w5P2.html** and **w5P2.css**
- Using <article>, <sections>s, flex container and flex items create a page looking something like this:  
You **MUST** do the following:
  - **Change** some of the text (please keep the theme of Australia Cities)
  - **Change** some aspect of the layout
  - **Change** some aspect of the formatting

## Australian Cities



Produced by Tanya Linden, Student ID 123456

**Screen Capture** the **HTML page** displayed in your browser and paste into **W05P.DOCX**  
**Copy and Paste** the **HTML code** and **CSS code** from your code editor into **W05P.DOCX**

**Task 14. (CREDIT Submission Task)**

- Create the file named **w5C2.html** and **w5C2.css**
- Using <header>, <footer>, <article>, <section>s, flex container and flex items to create a page something like this:  
You **must**:
  - **Change** any of the text that says 'blah'
  - **Change** at least one of the images
  - **Change / improve** the formatting (but keep the 6 flexbox items - 4 pets and header and footer)
  - **Replace email address** in the footer with your student email
  - **Your choice** whether to put your name in the header or keep it in the footer

Student Name: Sally Lee Id: 100200300 <<<<<<<< USE YOUR NAME & ID

## Pet Games

### Games to play with your pet

#### Dog

- Fetch
- Hide and Seek
- Shake hands



#### Cat

- blah
- blah
- blah



#### Goldfish

- blah
- blah
- blah



#### Parrot

- blah
- blah
- blah



## Follow-up

If you have suggestions for other games to play with your pets, then please contact me at the following email address [thepetguywithideas@hotmail.com.au](mailto:thepetguywithideas@hotmail.com.au)

**Screen Capture** the **HTML page** displayed in your browser and paste into **W05C.DOCX**  
**Copy and Paste** the **HTML code** and **CSS code** from your code editor into **W05C.DOCX**