UI Design & Evaluation

CO2104

Week 9 – Lab/Workshop 3

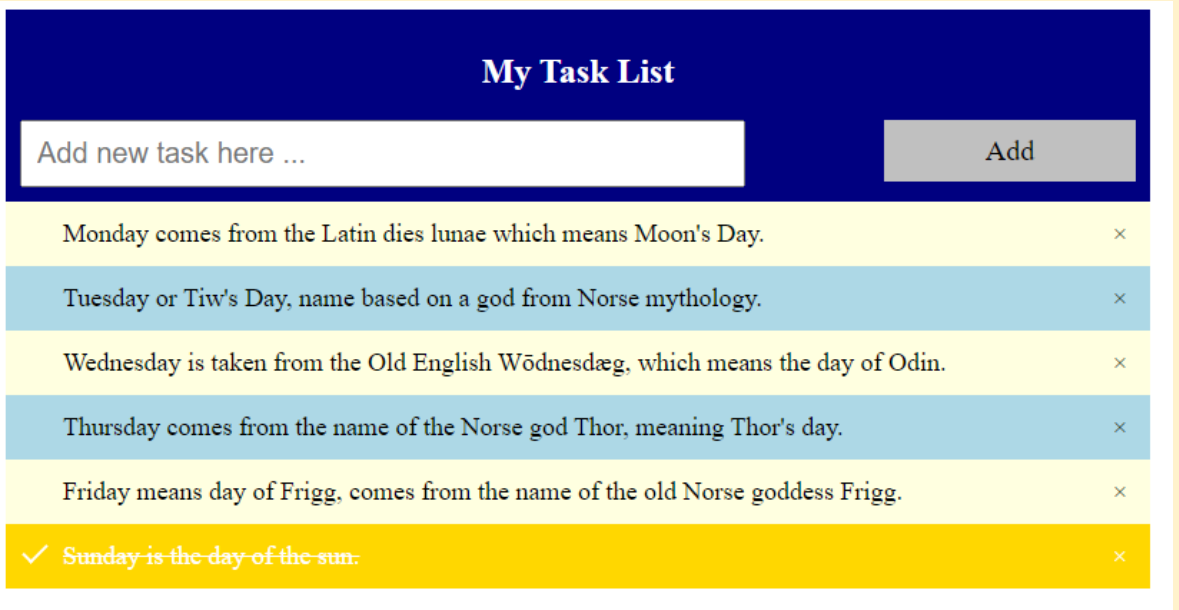
Task List on JavaScript

By the end of this session you should have…

- Created the “task manager” element
- This could include…
  - Showing a list
  - Adding to a list
  - Editing a list item
  - Deleting a list item
- These pages do not need a database backend.
- All the data can be stored locally
  - meaning the changes made while using your application would disappear when it is closed.
- Remember this is a prototype. We are not expecting a fully working system. Just a demonstration to how it might work that could be shown to an end user.
- You could also be….
  - Continuing to populate your pages/screen.
  - Identify any missing components (images, text, etc..) you may need to complete this assignment.

Let’s do a Task List in JavaScript

- This is a simple step-by-step for making a very basic “to do” list, using JavaScript.
- It allows the user to review, add, and delete entries.
- This does not save the users input after the page has been closed.



First step – setting up the page - HTML

- This is the HTML we will be using.
- You will see that we have a DIV **class** called “**header**”, but also has an **id** of “**myDiv**”
  - “**header**” **class** is for the formatting of the DIV
  - “**myDiv**” **id** is to relate to the JavaScript that that will control the list (to be created later)
- Then we have an **input box**, with the id of “**myInput**”, and some text that is found within it
- After that is a **button** called “**addBtn**” with the label “**Add**”
- Finally there is a list of task that have already been added to the page (to make it look partially functional)

```
<DOCTYPE html>
<html>
<head>

<link rel="stylesheet" type="text/css" href=
"css/MyCSS.css">

</head>

<body>

<div id="myDiv" class="header">
    <h2> My Task List </h2>
    <input type="text" id="myInput" placeholder="Add
    new task here ...">

    <span onclick="newElement()" class="addBtn"> Add
    </span>
</div>

<ul id="myList">
    <li>Monday comes from the Latin dies lunae which
    means Moon's Day.</li>
    <li> Tuesday or Tiw's Day, name based on  a god
    from Norse mythology.</li>
    <li> Wednesday is taken from the Old English
    Wōdnesdæg, which means the day of Odin. </li>
    <li> Thursday comes from the name of the Norse
    god Thor, meaning Thor's day. </li>
    <li> Friday means day of Frigg, comes from the
    name of the old Norse goddess Frigg.</li>
    <li> And how about Saturday and Sunday? </li>
</ul>

</body>
</html>
```

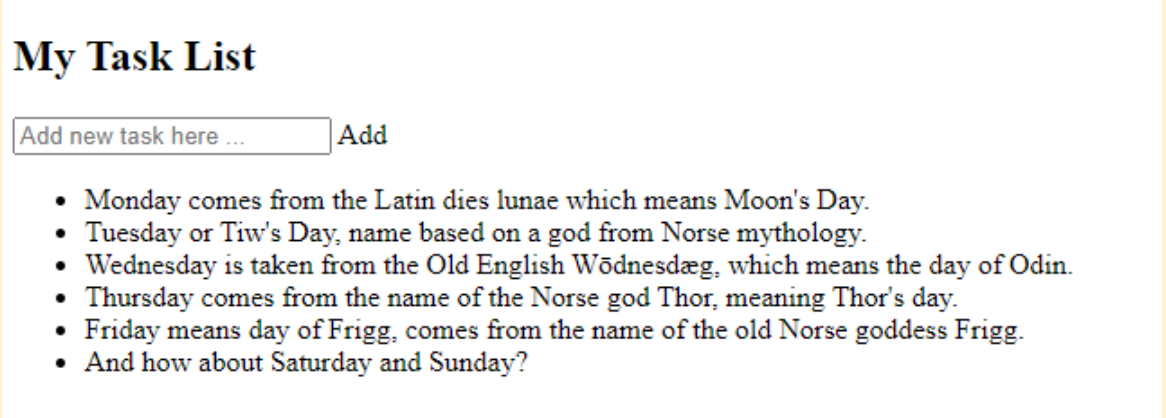
First step – setting up the page - HTML

- There is no functionality but it does show the content the page will contain

- Back in your code. You now need to link your CSS to the page.

- The CSS we will be using is called **“MyCSS.css”**

- The line of code that is missing need to be placed in the **<head>**
  - *<link rel="stylesheet" type="text/css" href="MyCSS.css">*



First step – setting up the page – CSS

- This is the first part of the CSS we will be using
- This will cover just the header and the input text box
- The next few pages will explain the code (in sections)

```css
body {
    min-width: 450px;
    max-width: 800px;
}

.header {
    background-color: navy;
    padding: 10px;
    color: white;
    text-align: center;
}

.header:after {
    content: "";
    display: table;
    clear: both;
}
```

```css
input {
    margin: 0;
    width: 65%;
    padding: 10px;
    float: left;
    font-size: 20px;
}

.addBtn {
    padding: 10px;
    width: 20%;
    background: silver;
    color: black;
    float: right;
    text-align: center;
    font-size: 20px;
    cursor: pointer;
}

.addBtn:hover {
    background-color: grey;
    color: white;
}
```

First step – setting up the page – CSS

- First we set the constraints for the “**body**” of the page. We want this to be able to expand but not by too much. So this is limited to be no more than **800** pixels wide, but no smaller than **450** pixels.
  - Anything smaller than 450 pixels messes up the clean formatting.

- The “**header**” formats how the header of the page looks (colour and formatting)

- The “**header:after**” is important as this stops the list being directly after the header text. This creates some space to aid with formatting.

```
body {
    text-align: center;
    min-width: 450px;
    max-width: 800px;
}

.header {
    background-color: navy;
    padding: 10px;
    color: white;
    text-align: center;
}

.header:after {
    content: "";
    display: table;
    clear: both;
}
```

First step – setting up the page – CSS

- This part of the code deals with the formatting of the text box where we can add our own inputs

- It is set to **width: 65%** of the screen so that it never overlap with the add button

- **Float: left** also support it not overlapping the button

```
input {
    margin: 0;
    width: 65%;
    padding: 10px;
    float: left;
    font-size: 20px;
}
```

First step – setting up the page – CSS

- The 2 sections here deal with the button
- First is the formatting of the button in general
- Also note that we set the **cursor: pointer**, so when a mouse goes over it the user knows that this is a button.
- The second part is the formatting of the button when the mouse hovers over it
- It basically changes colour

```
.addBtn {
    padding: 10px;
    width: 20%;
    background: silver;
    color: black;
    float: right;
    text-align: center;
    font-size: 20px;
    cursor: pointer;
}

.addBtn:hover {
    background-color: grey;
    color: white;
}
```

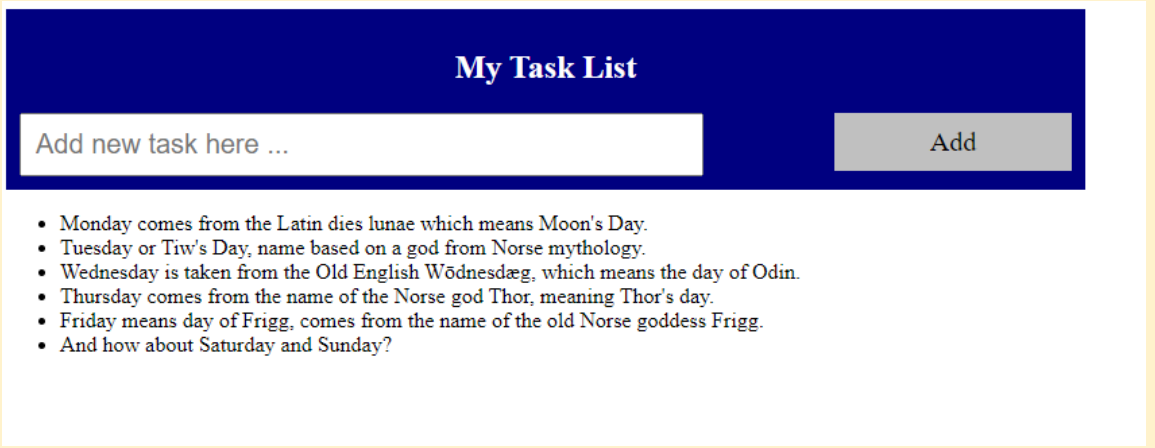
What this will look like in the browser

- If this code is opened in a browser it will look like this…

- You will notice that we now have colour for the header part of the document

- But nothing works
  - We can add text
  - But the button does nothing when clicked
  - No new items get added to the list

- Also we still require some formatting for the list items



Next step – setting up the page – CSS

- This is the second part of the CSS we will be using
- This will cover the list elements
- This code is added directly after the previous section of code in your CSS
- The next few pages will explain the code (in sections)

```css
ul {
    margin: 0;
    padding: 0;
}

ul li {
    cursor: pointer;
    position: relative;
    padding: 12px 8px 12px 40px;
    list-style-type: none;
    background: lightblue;
    font-size: 18px;

    -webkit-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
}

ul li:nth-child(odd) {
    background: lightyellow;
}

ul li:hover {
    background: dodgerblue;
}

ul li.checked {
    background: gold;
    color: white;
    text-decoration: line-through;
}
```

```css
ul li.checked::before {
    content: '';
    position: abasolute;
    border-color: white;
    border-style: solid;
    border-width: 0 2px 2px 0;
    top: 10px;
    left: 16px;
    transform: rotate (45deg);
    height: 15px;
    width: 7px;
}

.close {
    position: absolute;
    right: 0;
    top: 0;
    padding: 12px 16px 12px 16px;
}

.close:hover {
    background-color: navy;
    color: white;
}
```

Next step – setting up the page – CSS

- First this setups the this **Unordered List** (**ul**).
- The first part ensures that the list is started far over to the left hand side. Without setting the **margins** and **padding** being set to **0** it would float over to the right slightly.
- "**ul li**" is setting the layout of each object on the list.
- **"list-style-type: none"** removed the bullet points from the list.
- After setting up the links we have 4 lines of code for the different browsers this could be used within. Without these being added you would not be able to select any of the items within the list.
- The final part, "**ul li:nth-child(odd)**", is there to allow us to have every odd numbered row in the list to have a different background colour. So this will give you an alternate colour scheme.

```
ul {
    margin: 0;
    padding: 0;
}

ul li {
    cursor: pointer;
    position: relative;
    padding: 12px 8px 12px 40px;
    list-style-type: none;
    background: lightblue;
    font-size: 18px;

    -webkit-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
}

ul li:nth-child(odd) {
    background: lightyellow;
}
```

Next step – setting up the page – CSS

- This next part of the code deals with the hover over and “checked” state. The “checked” state is what happens when the user click on an item in the list.
- When the JavaScript has been added we want it to change colour and put a tick beside the item when the user clicks on them to say they are “done”.
- The first section of this code states how the item should be formatted when they have been clicked as “done”.
- The second is how it looks when it has not be selected as “done”.
- The section “**content: ‘ ‘:**” is there because we want to have a “tick” appear next to the text if the task has been “done”. But before they have been done we do not want to see anything, so we want this to be empty.
- “**transform: rotate (45degs)**” is just formatting the symbol of the tick. In reality we are not using a tick symbol but a backwards L (⌋), and then rotating it slightly so it appears as a “tick”.

```
ul li:hover {
    background: dodgerblue;
}

ul li.checked {
    background: gold;
    color: white;
    text-decoration: line-through;
}

ul li.checked::before {
    content: '';
    position: abasolute;
    border-color: white;
    border-style: solid;
    border-width: 0 2px 2px 0;
    top: 10px;
    left: 16px;
    transform: rotate (45deg);
    height: 15px;
    width: 7px;
}
```

Next step – setting up the page – CSS

- This final section of code is to do with a “X” that we want to appear next to each task, which will function as a “delete” button

- With this included if the user wants to remove an item they will click on the “X” and it will be removed from the list

- This will appear when we create the JavaScript code

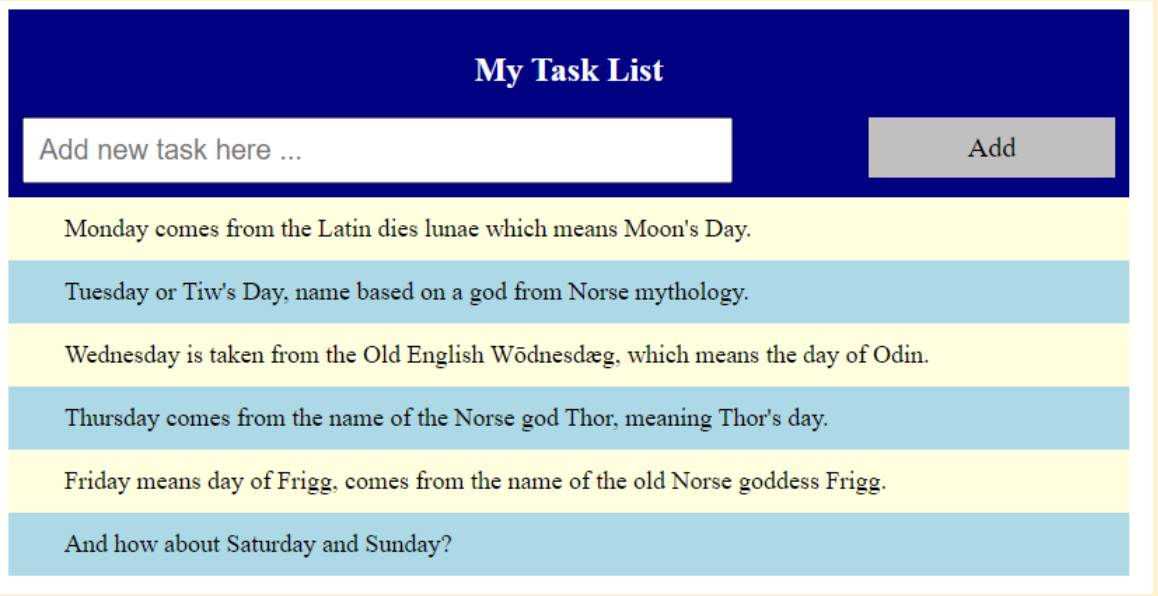
```
.close {
    position: absolute;
    right: 0;
    top: 0;
    padding: 12px 16px 12px 16px;
}

.close:hover {
    background-color: navy;
    color: white;
}
```

What this will look like in the browser

- You will notice that we now have colour for all items

- But still nothing works
  - We can add text
  - But the button does nothing when clicked
  - No new items get added to the list



Next step – setting up the page – JavaScript

- We are going to look at the JavaScript in two parts
- This is added to your HTML code directly after the lists
- This part will look at selecting, and deleting tasks from the list
- The second part will look at adding to the list
- Here is the code for the selecting & deleting tasks
- Add this before your </html> tag in the end of the index.html
  - <script src="scripts/MyJS.js"> </script>

```
var myNodeList = document.getElementsByTagName("LI");

for (var i = 0; i < myNodeList.length; i++) {
    var span = document.createElement("SPAN");
    var txt = document.createTextNode("\u00D7");
    span.className = "close";
    span.appendChild(txt);
    myNodeList[i].appendChild(span);
}

var close = document.getElementsByClassName("close");

for (var i = 0; i < close.length; i++) {
    close[i].onclick = function() {
        var div = this.parentElement;
        div.style.display = "none";
    }
}

var list = document.querySelector('ul');
list.addEventListener('click',function(ev) {
    if(ev.target.tagName === 'LI') {
        ev.target.classList.toggle('checked');
    }
}, false);
```

Next step – setting up the page – JavaScript

- This first section is all about creating the little “X” close symbol to appear beside each item in the list
- “**var MyNodeList**” is counting how many items there are in the list
- “**var i**” is a counter for the FOR loop
- “**var span**” the holding the details for how the “X” icon should look (the formatting)
- “**var txt**” is the “X” icon (the image)
- So this code is looking at every line in the list and adding a “X” button to the end of each one of them.

```
var myNodeList = document.getElementsByTagName("LI");

for (var i = 0; i < myNodeList.length; i++) {
    var span = document.createElement("SPAN");
    var txt = document.createTextNode("\u00D7");
    span.className = "close";
    span.appendChild(txt);
    myNodeList[i].appendChild(span);
}
```

Next step – setting up the page – JavaScript

- This next section of code is dealing with the functionality of the “X” button
- “**var close**” is holding the class details from “close” (in the CSS)
- Then, through a loop, it checks to see if the “X” has been selected. If it has then it stops displaying that item
- The last part of this code deals with adding a tick beside the tasks that have been selected
- So it has an event listener to watch for a mouse action, that when an item in the list has been clicked on it runs the “checks” class in the CSS

```
var close = document.getElementsByClassName("close");

for (var i = 0; i < close.length; i++) {
    close[i].onclick = function() {
        var div = this.parentElement;
        div.style.display = "none";
    }
}

var list = document.querySelector('ul');
list.addEventListener('click',function(ev) {
    if(ev.target.tagName === 'LI') {
        ev.target.classList.toggle('checked');
    }
}, false);
```

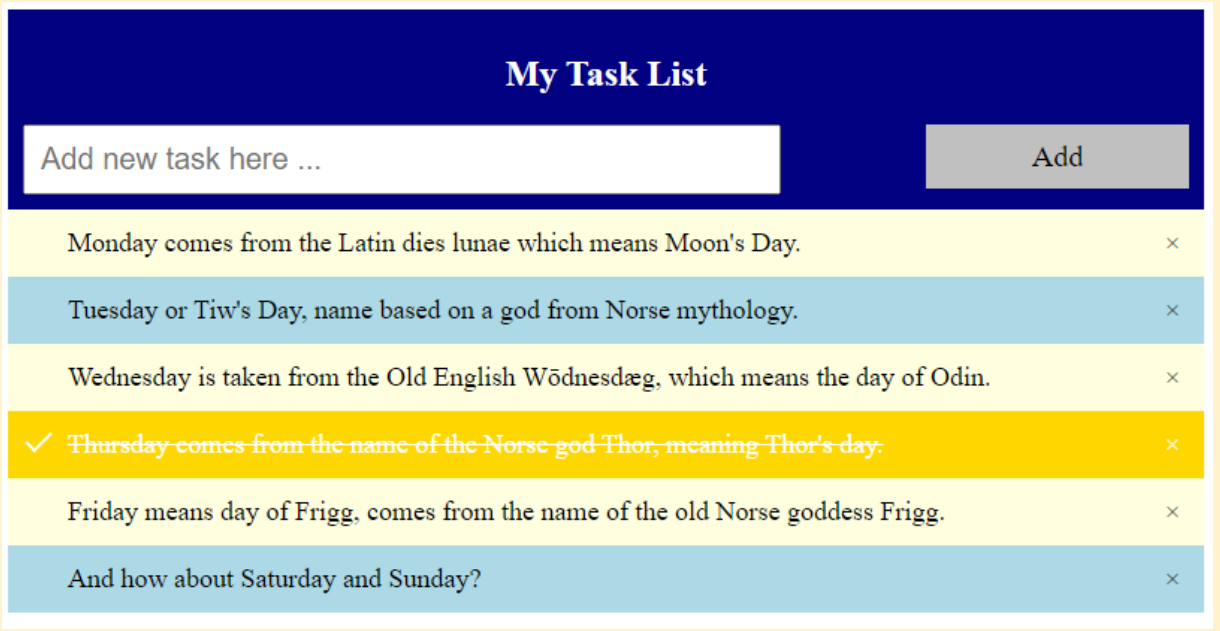
What this will look like in the browser

- If this code is opened in a browser it will look like this …

- You will notice that we are able to click on items within the list and they change state (turn gold, tick appears)

- We also have “X” icons next to each item, which when clicked on they remove the item from the list

- At this point we still can not add tasks to the list (the next section)



Next step – setting up the page – JavaScript

- This is the next part of the JavaScript code
- This deals with adding tasks to the list
- This is added to the end of the previous sections of code
- The next few pages will explain the code (in sections)

```
function newElement() {
    var li = document.createElement("li");
    var inputValue = document.getElementById(
    "myInput").value;
    var t = document.createTextNode(inputValue);
    li.appendChild(t);

    if (inputValue === '') {
        alert("you must write something!");
    } else {
        document.getElementById("myList").
        appendChild(li);
    }
    document.getElementById("myInput").value = "";

    var span = document.createElement("SPAN");
    var txt = document.createTextNode("\u00D7");
    span.className = "close";
    span.appendChild(txt);
    li.appendChild(span);

    for(var i = 0; i < close.length; i++) {
        close[i].onclick = function() {
                var div = this.parentElement;
                div.style.display = "none";
        }
    }
}
```

Next step – setting up the page – JavaScript

"**var li**" is creating a new element (item in the list)
"**var inputValue**" is holding the text that has been entered into the text box.
"**var t**" is what hold the value that appears in the list.

This code is allowing the "X" to appear next to the new item in the list.

This is the instructions for what happens when the user clicks on the "X".

```
function newElement() {
    var li = document.createElement("li");
    var inputValue = document.getElementById(
    "myInput").value;
    var t = document.createTextNode(inputValue);
    li.appendChild(t);

    if (inputValue === '') {
        alert("you must write something!");
    } else {
        document.getElementById("myList").
        appendChild(li);
    }
    document.getElementById("myInput").value = "";

    var span = document.createElement("SPAN");
    var txt = document.createTextNode("\u00D7");
    span.className = "close";
    span.appendChild(txt);
    li.appendChild(span);

    for(var i = 0; i < close.length; i++) {
        close[i].onclick = function() {
                var div = this.parentElement;
                div.style.display = "none";
        }
    }
}
```

This is checking to see if anything was typed into the text box. If not then it will bring up a message telling the user to write something.

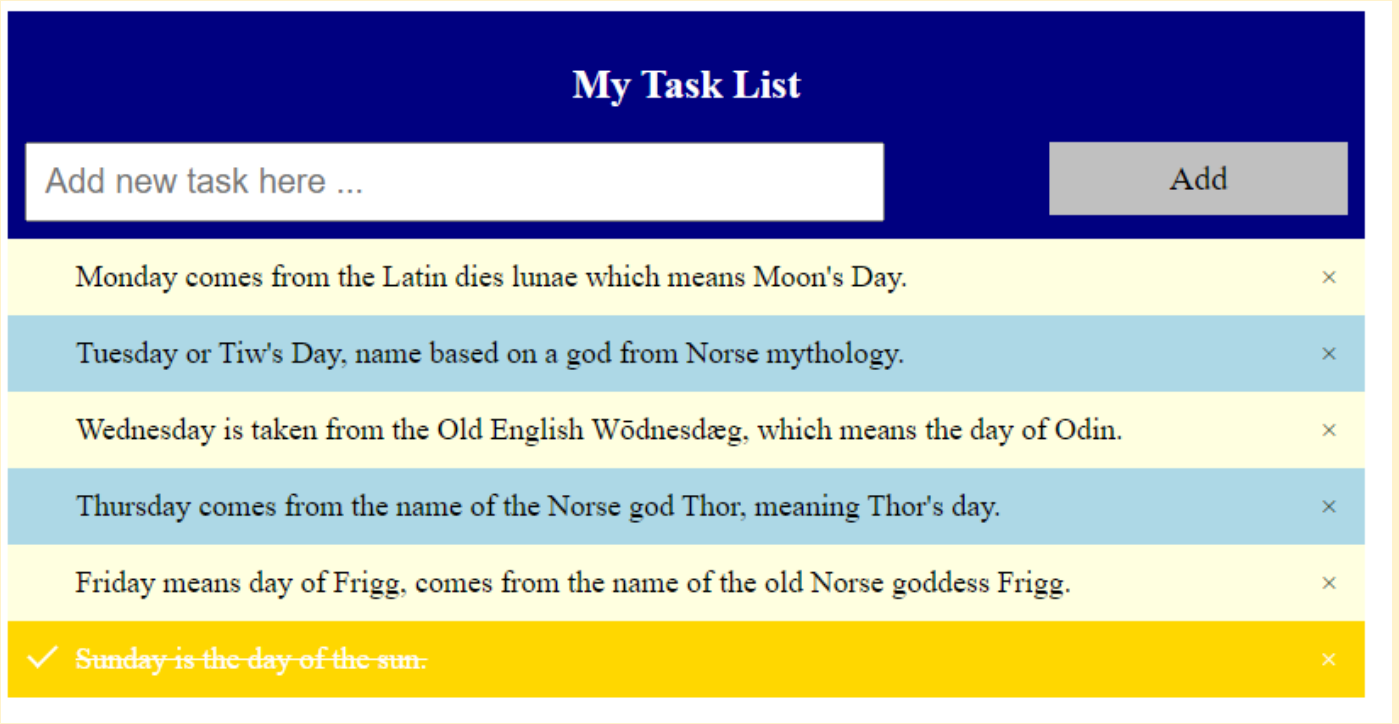
If something has been written in the text box then add it to the list called "**myList**".

This just resets the textbox to being blanks, ready for the next entry.

What this will look like in the browser

- This should now be fully functional

- You should be able to
  - Add new task
  - Complete tasks
  - Remove tasks



What happens next?

- Ensure by next week you have …
  - Continue to collected all missing elements you require (images, text, etc…)
  - Continue to populate your pages/screens
  - Where possible, to start to put together your submission
    - even if it is just setting up a temporary documents with all the heading added so you can just “fill in the gaps” as you go along
  - Prepare your prototype so someone can start testing it