Concentration Lab using Angular

# Welcome

Having Visual Studio installed with help a lot, but you can do this lab with notepad and a web browser.

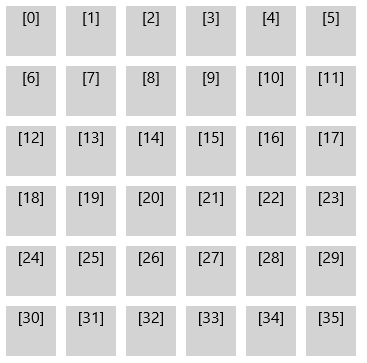
Each section is associated with a folder of code and you will need to make changes to enable the functionality. The following section often contains the completed work from the previous section, so you can sometimes cheat if you get stuck.

By the end of this lab you will have a working Concentration game built using Angular.

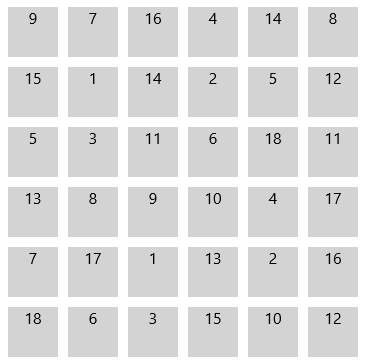
# Baseline

1. Open the files in the ‘1. Get Started’ folder
2. Review index.html, site.js, and site.css
3. Notice the **ng-app**, **ng-controller**
4. Notice the **$scope.greeting** being set and the **{{greeting}}** binding

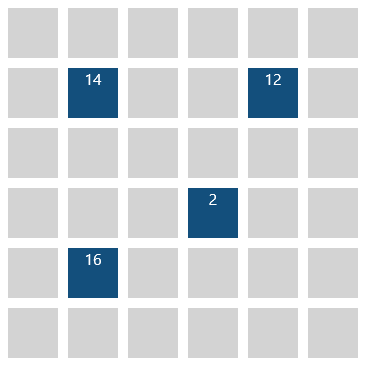
# Grid

1. Open the files in the ‘2. Grid’ folder
2. site.js: Update the controller to call the **generateCardGridMethod** for a 6x6 grid and set the result in a **cardGrid** variable on the scope.
3. index.html: Update the HTML to **ng-repeat** across rows of cards.
4. index.html: Update the HTML to display the number in each **card**.  
     
     
   

# Paired Deck

1. Open the files in the ‘3. Paired Deck’ folder.
2. index.html: Update the HTML to display the card value in each card start. The value should appear as a 1 based index.  
     
   

# Card Reveal

1. Open the files in the ‘4. Card Reveal’ folder.
2. site.js: Update the JavaScript to expose a **toggleCard** method on the scope. The method should take a card object and set the **revealed** property to **true**.
3. index.html: Update the HTML to call the **toggleCard** method each time a card is clicked, passing in the appropriate card data instance.
4. index.html: Update the **card** and **card-value** div’s with the **revealed** class when the card **revealed** property is **true**.  
     
   

# Pairing

### Selecting 2 cards

1. Open the files in the ‘5. Pairing’ folder.
2. site.js: Add 2 card selection variables to the controller to track the first and second selections made by the user (**cardSelection1**, **cardSelection2**).
3. site.js: Update the **toggleCard** method to set the **cardSelection1** first and reveal the card.
4. site.js: Update the **toggleCard** method to set the **cardSelection2** second and reveal the card.

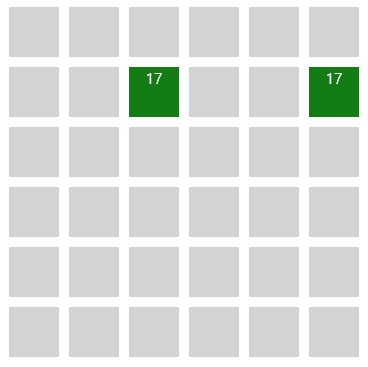
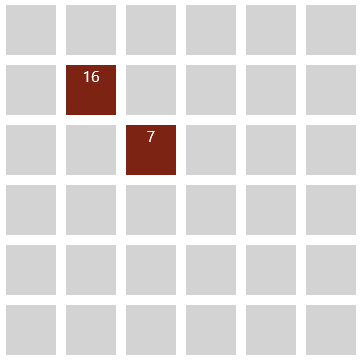
### Matches

1. site.js: Update the **toggleCard** method so that when the user has selected 2 cards and their values match, then both cards **matched** property is set to **true**.
2. site.js: Update the **toggleCard** method to clear the selection when the user matches two cards.
3. index.html: Update the HTML to set the **matched** class when a card’s **matched** property is **true**.

### Mismatches

1. site.js: Update the **toggleCard** method so that when the user has selected 2 cards and their values don’t match, then both cards **mismatched** property is set to **true**.
2. site.js: Update the **toggleCard** method so that when the user has selected 2 cards and their values don’t match, then both cards **revealed** property is set to **false**.
3. site.js: Update the **toggleCard** method to clear the selection when the user mismatches two cards.
4. index.html: Update the HTML to set the **mismatched** class when a card’s mismatched property is true.

### UI timing

1. site.js: Update the **toggleCard** method so that after the second card is selected and revealed, the matching code is within a **timeout** function of 500 milliseconds.
2. site.js: Update the **toggleCard** method so that after the two cards are mismatched, the hiding code and selection clearing is within a **$timeout** function of 500 milliseconds.  
     
     
     
   

# Scoreboard

1. Open the files in the ‘6. Scoreboard’ folder.

### Create templates

1. index.html: Move the **<div class=”scoreboard”>** element into the scoreboard.html template.
2. index.html: Move the first **<div class=”player”>** element into the player.html template. Delete the second **player** div.

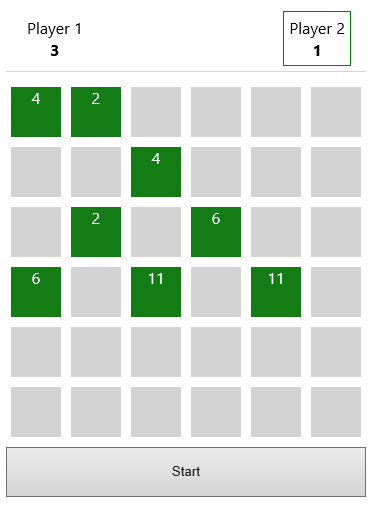
### Make general purpose

1. index.html: Update the **player.html** template to reference **player** rather than **player1**.
2. index.html: Update the **scoreboard.html** template to have two instances of the **<player>** element.
3. index.html: Update the **scoreboard.html** template to set the **player** attribute on each **<player>** element to **player1** and **player2** respectively.
4. index.html: Update the **scoreboard.html** template to set the **currentPlayer** attribute on each **<player>** element to **currentPlayer**.

### Review the directive code

1. site.js: Review the **scorecard** and **player** directives to see how they use **templateUrl** to find their templates.
2. site.js: Review the **player** directive to see how isolated **scope** allows a different **player** and **currentPlayer** to be used.

### Implement turn taking

1. site.js: Implement the **nextTurn** function to toggle the **currentPlayer** between **player1** and **player2**.
2. site.js: Review the **startGame** method and notice it is called at the end of the controller being run.  
     
   

# Game Winner (Lab Challenge 1)

Update the scoreboard to indicate which player won after the game is complete. This will require authoring new HTML in the scoreboard template and new logic in site.js.

# Who’s Match? (Lab Challenge 2)

Update the matching logic to indicate which player made each match.

# Who’s Match? (Lab Challenge 3)

Create a match history directive that lists the matches date & time, the scores, and who won. Update the game to record the winner of each match.