VueJS

Grocery Shopping List

Getting started

- Create a new page and add the following to the head section to link up the library:
- `<script src="js/vue.min.js"></script>`
- Can you remember what kind of tag needs to go into the body to be the part of the page that Vue can control?

Getting started

- Add the following to the body:

- ```

      <h2>{{ title }}</h2>

  ```

- There are two important parts here. The ID is what will define the **app** in the code.

- The **title** is a variable that we can update through code, note it's inside of two moustache brackets!
  ```

Getting started

- Before the closing body add script tags and then add this inside:

```
new Vue({
        el: '#app',
        data: {
            title: 'Your Page'
        }
});
```

- Here we define the **app** as the bit that Vue is going to take control of.
- The **title** is a variable that is stored in the data of this app.
- Save your page as **data_binding.html** and test it

Data Binding

- You may remember that this is known as data binding.

- This is one way data binding, but it can be two way, let's explore that. First add some CSS in style tags to the head:

- ```
  .container {
        width: 50%;
        margin: 20px auto 0px auto;
  }
  ```
  ```

Two way data binding

- Now choose save as and name the next file **binding_2_way.html**.
- Add the following to the HTML inside the container:
- ```

          <hr/>
          Change the title of your shopping list here
          <input v-model="title" />

  ```
- OK no more code, just test this and change the title in the input field.
- The code in red links this up with the title variable!
  ```

Creating a list

- Now choose save as and name the next file **list1.html**.
- Add the following to the HTML under the <h2> title:

```
<ul>
        <li v-for="item in items">
              {{ item.text }}
        </li>
</ul>
```

- The **v-for** is going to loop through a list of **items** and place every item in the items list into the **item.text**

Creating a list

- Update the data in the vue:

```
data: {
        newItem: '',
        items: [{
                text: 'Bananas',
                checked: true
        }, {
                text: 'Apples',
                checked: false
        }],
        title: 'My Shopping List'
  }
```

- There's the list of items and the text that will be put into our shopping list
- Save and view in the browser

Creating a list

- Save the page as **list2.html** and add the following CSS:

```
.removed label {
        text-decoration: line-through;
}
ul li {
        list-style-type: none;
}
```

- You will have noticed that our data has a true or false state for the checked item, let's add that to the HTML

Creating a list

- Update the unordered list as follows:

- ```
  <ul>
        <li v-for="item in items" :class="{ 'removed': item.checked }">

                <label>
                    <input type="checkbox" v-model="item.checked"> {{ item.text }}
                </label>

        </li>
  </ul>
  ```

- The new elements that you haven't seen before are in red. Test in the browser
  ```

Creating a list

- This code is easier to explain so let's start here:
- `<input type="checkbox" v-model="item.checked">`
- The checkbox is to take it's value from the item.checked, you can see that in the data:{} and items.
- Bananas is true and Apples, false.

Creating a list

- This code is easy to see if you look at the console and see the code update as you click the checkbox on and off:
- `<li v-for="item in items" :class="{ 'removed': item.checked }">`
- The class ‘removed’ is only added if this item is checked
- Look at the removed class and it simply does a strike through!

Updating the list

- OK save the page and save as '**list3.html**'.

- ```

          <input v-model="newItem"
  @keyup.enter="addItem" placeholder="add shopping
  list item" type="text" class="form-control">

  ```

- So this element is going to be a newItem, we already have an empty variable for that in the data.

- When the user presses enter this input text will be a new shopping item
  ```

Updating the list

- Make sure you add a comma , after the data closing bracket, then add this.

```
methods: {
        addItem: function() {
                var text;
                text = this.newItem.trim();
                if (text) {
                        this.items.push({
                                text: text,
                                checked: false
                        });
                        this.newItem = '';
                }
        }
}
```

- This is the **addItem** that is called when the user presses enter

- It takes the **newItem** text and trims any extra trailing spaces, then it makes a new item in the **items** list with the text. Finally clearing the newItem text ready for next time.

Going further

- We have learned a lot today
- Go back over every stage and make sure you understand what is happening.
- Look for where the JS code in the script tags talks to the HTML code on display