RESPONSIVE WEB APPS PORTFOLIO

David Adams

216110104 SIT120 Responsive web apps

Responsive Web Apps Portfolio

Introduction

This portfolio is a culmination of the weekly tasks within SIT120 Responsive Web Apps, and details each week's tasks and concepts with code and detailed web images to justify and give evidence on the work summated in this document. Over the unit's course I have learned many different things when developing web applications and the types of frameworks and methodologies to send and receive communication. Furthermore, this unit covers the Vue framework which is displayed within this portfolio detailing my journey on learning the concepts and methodologies of creating web applications.

Justification

My aim for this unit is a Distinction level, however due to the handling of this unit and the complexity level at which it is asking for I will be settling for a credit. Throughout this portfolio I have detailed and documented the process and concepts of each task along with reflections and the description of each concept which will allow people to use the information and implement it themselves. All tasks and weekly concepts have been demonstrated and achieved along with screenshots and code snippets as well as a GitHub link for access to show the concepts that have been created.

GitHub link: https://github.com/Davym8/Responsivewebapps

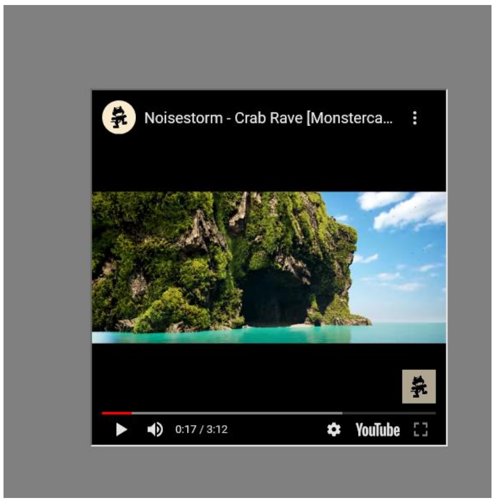
This link will show each week as well as the project which will be available to access and view the code. The GitHub contains each weekly tasks and content with code detailing what I did for each week, furthermore the code snippets regarding week 6 and 7 are also available in the improved version to detail the progress for those two weeks implementing the tasks.

Week 1 Responsive Web Apps

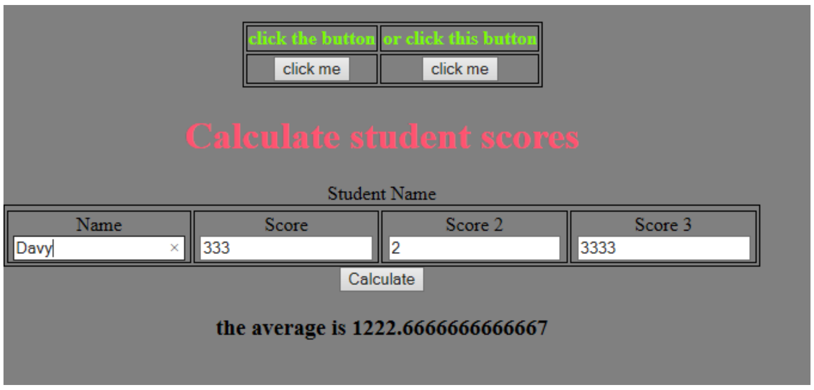
Task 1: Create a HTML webpage

GitHub link: https://github.com/Davym8/Webappstask1.git

Embedded video



Calculator using JavaScript



click the button	or click this button
click me	click me

Calculate student scores

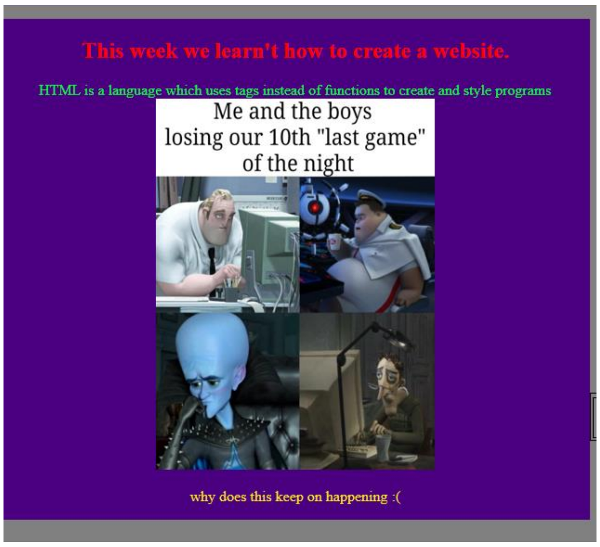
Student Name

Name	Score	Score 2	Score 3
Davy	333	2	3333

Calculate

the average is 1222.6666666666667

Images

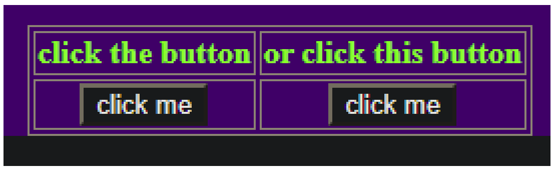


This week we learn't how to create a website.

HTML is a language which uses tags instead of functions to create and style programs

why does this keep on happening :(

Hyperlinks and table



click the button	or click this button
click me	click me

Reflection

In this task I used a number of layouts and elements to create a rudimentary basic layout, since this is the first time, I have ever created a website the complexity and understanding are still new to me. However, during the process I was able to create a basic layout using multiple styles of CSS and HTML design properties. Featured above in the images are examples of using embedded video links, imagery, JavaScript functions, tables and

My process

Hyperlinks which demonstrate basic principles within Website development and which all websites have.

For this task in order to complete the required elements a did a lot of research on the foundation of a website as it comes first when designing a website.

First: get a layout of the page you want to design, where everything goes, how it will flow, what will happen.

Second: organise the information in which areas you will apply them to, such as relationships, inheritance and website manipulation of data.

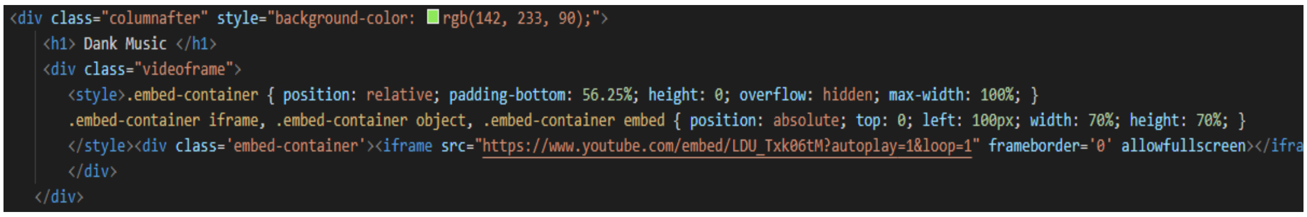
Third: construct specific elements so that they can be partitioned and modified, such as identity tags.

Fourth: insert the required functions and make sure that CSS and scripts have access and connections to the correct data.

Useful elements

In order to gain an understanding on how design and layout works for a website lookup CSS and HTML design and implementation such as YouTube and w3school.

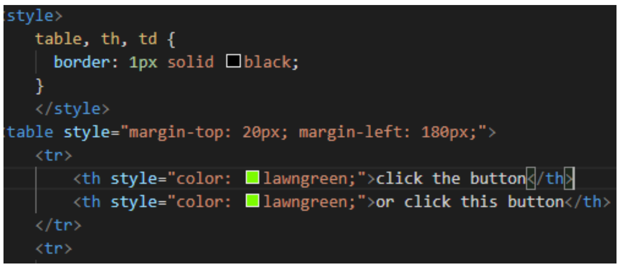
Useful information within website development is learning to embed data from different sources, embedding is the process of using other elements within the HTML page that are prepackaged such as Videos, JavaScript functions and API's. In this case embedding A video from YouTube is a simple procedure. Using inline styling is a proper method of overriding default styling techniques so that you can decide on how it is displayed within a website.



```
<div class="columnafter" style="background-color: 🟩rgb(142, 233, 90);">
    <h1> Dank Music </h1>
    <div class="videoframe">
      <style>.embed-container { position: relative; padding-bottom: 56.25%; height: 0; overflow: hidden; max-width: 100%; }
      .embed-container iframe, .embed-container object, .embed-container embed { position: absolute; top: 0; left: 100px; width: 70%; height: 70%; }
      </style><div class='embed-container'><iframe src="https://www.youtube.com/embed/LDU_Txk06tM?autoplay=1&loop=1" frameborder='0' allowfullscreen></ifra
      </div>
  </div>
```

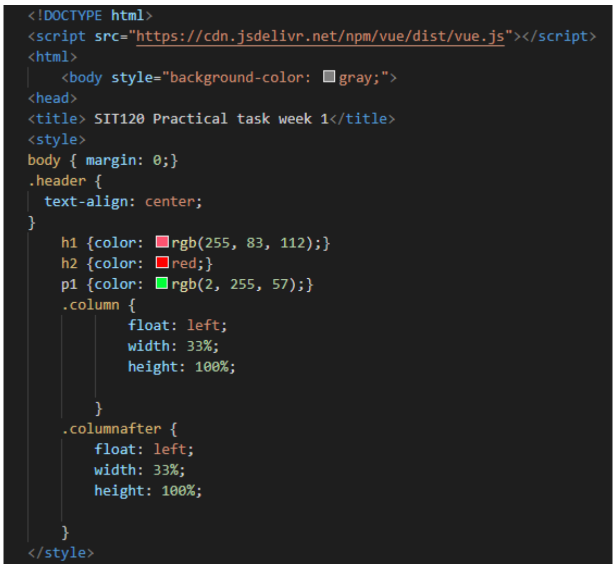
Task 2: Adding CSS to the website

Within this task CSS is a styling language which allows us to represent our webpages in a creative manner. Inline CSS is the manner of adding styling techniques within the HTML page alongside the HTML structure.



```
style>
   table, th, td {
     border: 1px solid black;
   }
   </style>
table style="margin-top: 20px; margin-left: 180px;">
   <tr>
        <th style="color: lawngreen;">click the button</th>
        <th style="color: lawngreen;">or click this button</th>
   </tr>
   <tr>
```

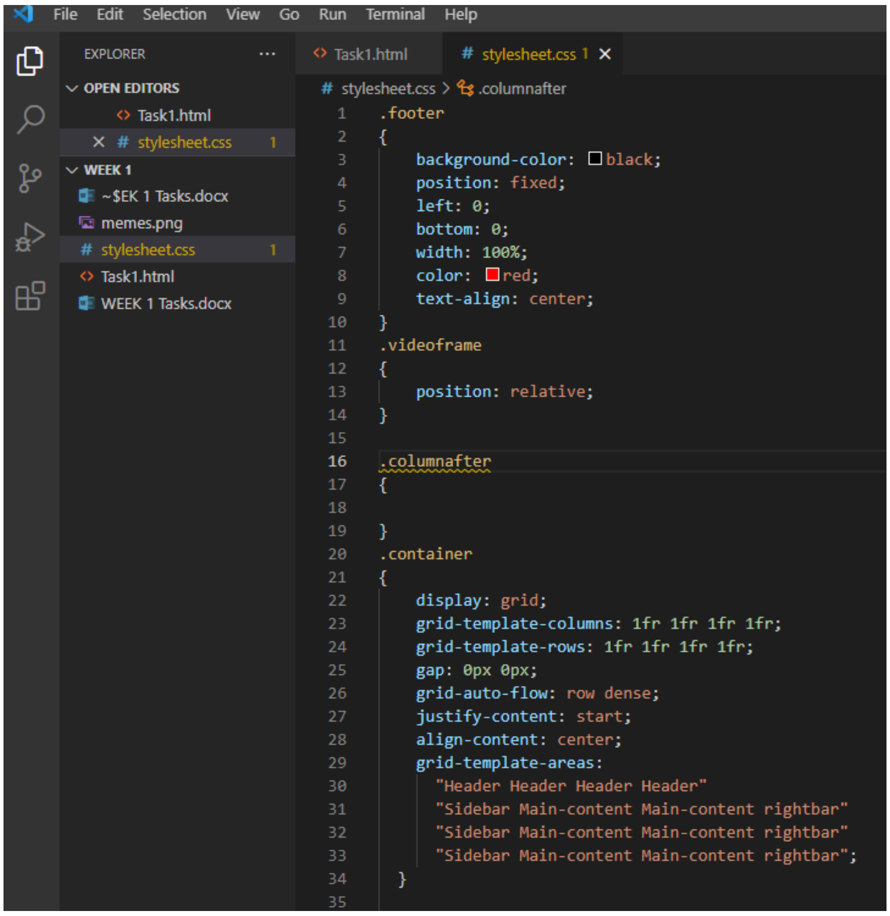
Internal CSS is the method of adding sections of styling techniques with reference to the location within the HTML document. Below demonstrates the use of adding certain styling techniques that are internal, with each being referenced by a .ID and {} curly brackets.



```
<!DOCTYPE html>
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
<html>
    <body style="background-color: gray;">
<head>
<title> SIT120 Practical task week 1</title>
<style>
body { margin: 0;}
.header {
  text-align: center;
}
    h1 {color: rgb(255, 83, 112);}
    h2 {color: red;}
    p1 {color: rgb(2, 255, 57);}
    .column {
            float: left;
            width: 33%;
            height: 100%;

        }
    .columnafter {
        float: left;
        width: 33%;
        height: 100%;

    }
</style>
```

External CSS is where best practice is done, by creating CSS stylesheets externally we can modify it, use it for multiple things as well as create a clean and ideal format.

Example of external styling

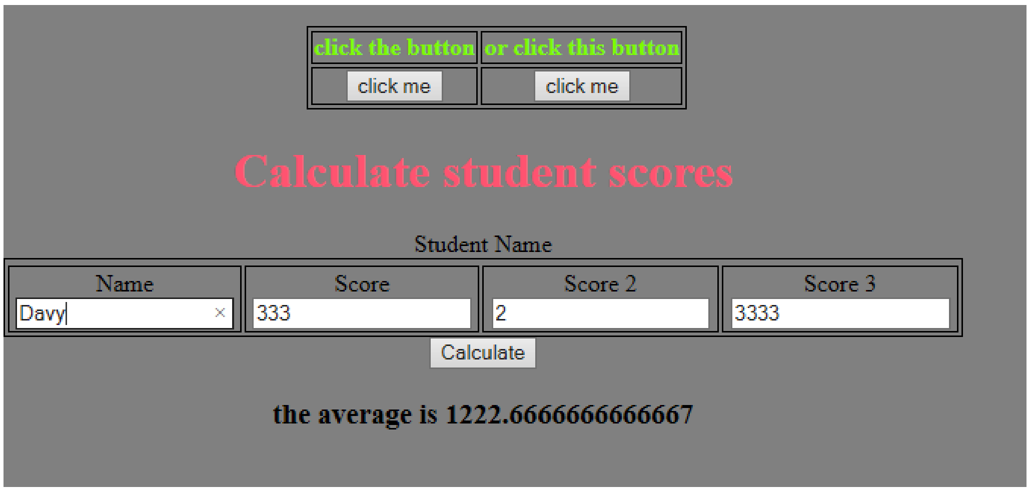


```css
.footer
{
    background-color: black;
    position: fixed;
    left: 0;
    bottom: 0;
    width: 100%;
    color: red;
    text-align: center;
}
.videoframe
{
    position: relative;
}

.columnafter
{

}
.container
{
    display: grid;
    grid-template-columns: 1fr 1fr 1fr 1fr;
    grid-template-rows: 1fr 1fr 1fr 1fr;
    gap: 0px 0px;
    grid-auto-flow: row dense;
    justify-content: start;
    align-content: center;
    grid-template-areas:
      "Header Header Header Header"
      "Sidebar Main-content Main-content rightbar"
      "Sidebar Main-content Main-content rightbar"
      "Sidebar Main-content Main-content rightbar";
  }

```

Task 3: JavaScript

JavaScript is an object-oriented programming language used within web applications and severs to allow interaction and creation of functionable programs.

Within this task I created a calculator which gets the average of a user's score

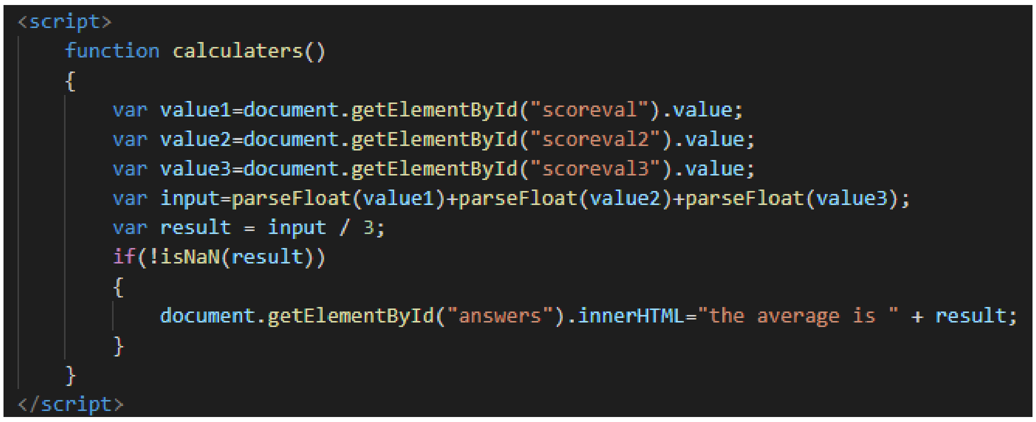


Student Name

Name	Score	Score 2	Score 3
Davy	333	2	3333

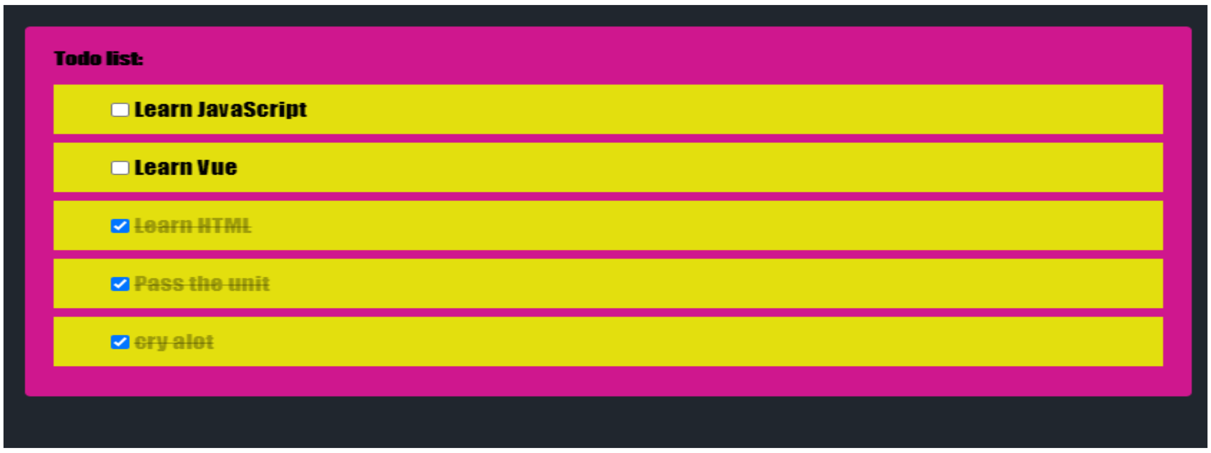
the average is 1222.6666666666667

Implementation within the HTML document, this is an example of internal JavaScript function.


```
<script>
    function calculaters()
    {
        var value1=document.getElementById("scoreval").value;
        var value2=document.getElementById("scoreval2").value;
        var value3=document.getElementById("scoreval3").value;
        var input=parseFloat(value1)+parseFloat(value2)+parseFloat(value3);
        var result = input / 3;
        if(!isNaN(result))
        {
            document.getElementById("answers").innerHTML="the average is " + result;
        }
    }
</script>
```

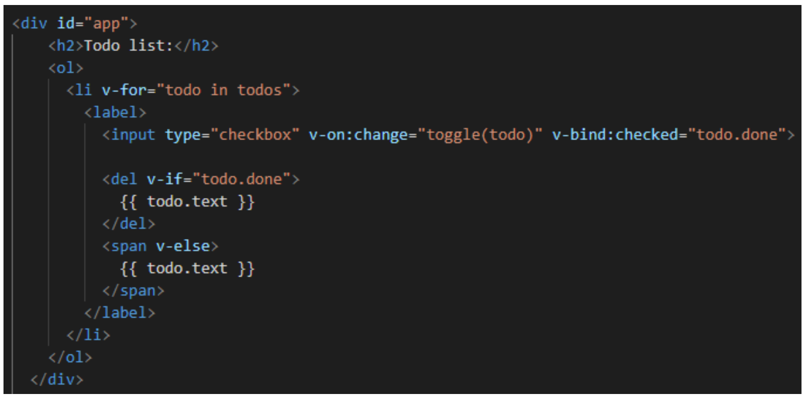
Task 4: Vue.js framework

Vue js is a different programming language for web applications that follow the same idea of JavaScript. In task 4 we will be using Vue to create a simple to do application.

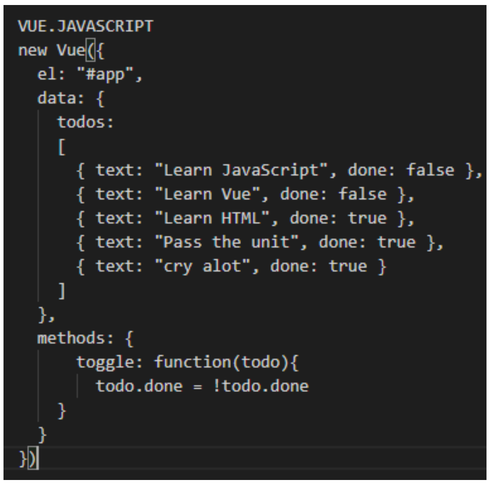

Todo list:

- [ ] Learn JavaScript
- [ ] Learn Vue
- [x] ~~Learn HTML~~
- [x] ~~Pass the unit~~
- [x] ~~cry alot~~

Code implementation


```
<div id="app">
    <h2>Todo list:</h2>
    <ol>
      <li v-for="todo in todos">
        <label>
          <input type="checkbox" v-on:change="toggle(todo)" v-bind:checked="todo.done">

          <del v-if="todo.done">
            {{ todo.text }}
          </del>
          <span v-else>
            {{ todo.text }}
          </span>
        </label>
      </li>
    </ol>
  </div>
```

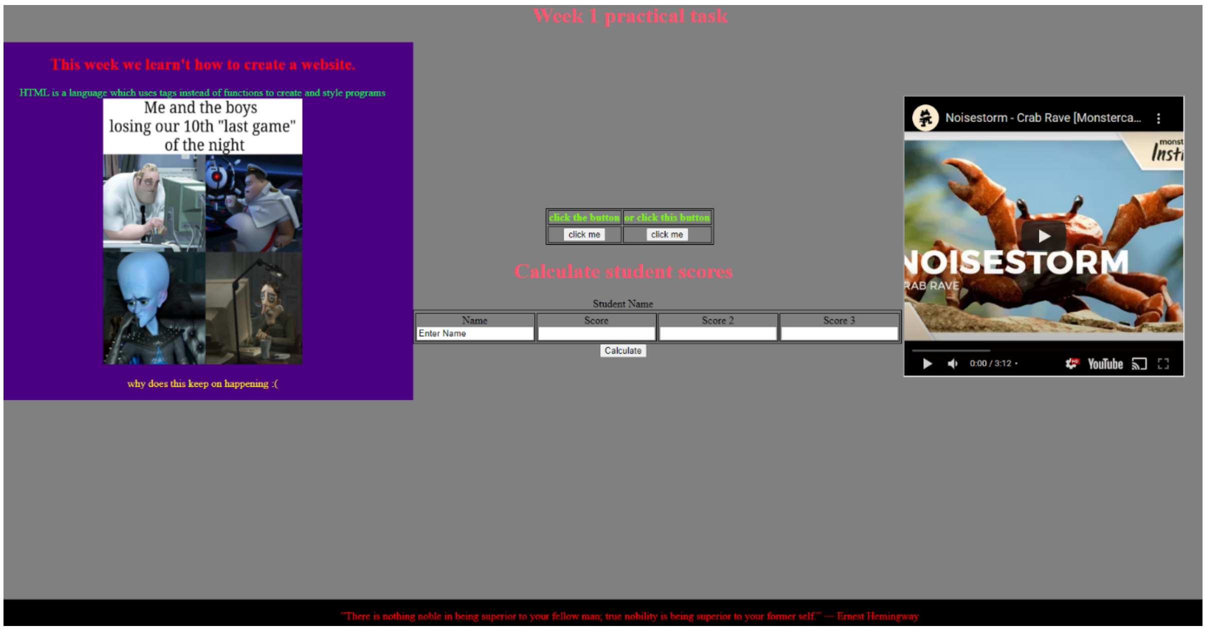
Vue app which renders the TODO function.



```
VUE.JAVASCRIPT
new Vue({
  el: "#app",
  data: {
    todos:
    [
      { text: "Learn JavaScript", done: false },
      { text: "Learn Vue", done: false },
      { text: "Learn HTML", done: true },
      { text: "Pass the unit", done: true },
      { text: "cry alot", done: true }
    ]
  },
  methods: {
      toggle: function(todo){
        todo.done = !todo.done
    }
  }
})
```

Using Vue framework in my opinion is a decent language offering simple and efficient modelling of applications as well as offering ease of integration within projects. By simply defining the app and creating identification tags the app can function in a tree like structure. However due to the nature of the inheritance it makes it difficult to use widely within a document. Overall, the language Is good for performance and interaction within the DOM.

Final output for webpage for week 1



Week 1 practical task

This week we learn't how to create a website.

HTML is a language which uses tags instead of functions to create and style programs

Me and the boys
losing our 10th "last game"
of the night

why does this keep on happening :(

click the button	or click this button
click me	click me

Calculate student scores

Student Name

Name	Score	Score 2	Score 3
Enter Name			

Calculate

Noisestorm - Crab Rave [Monsterca...
NOISESTORM
RAB RAVE
0:00 / 3:12
YouTube

"There is nothing noble in being superior to your fellow man; true nobility is being superior to your former self." — Ernest Hemingway

Weekly Reflection

Creating a html page by scratch with having a little bit of knowledge in high school was hard and I needed to brush up and learn through YouTube videos and read the guides on websites detailing how to create a website. For starters I knew that I needed to get a layout going with the website such as a header, footer, columns, rows link fields, body and various other templates. Next is to get the titles set, links and text were set in as well as an image.

So, in retrospect doing this task with no clue on website creation I'm pretty satisfied with what I've done so far however I do need to watch more tutorials and research design and coding syntax. The code so far is similar to other languages I've studied however I've found that the Vue, html and CSS are somewhat messy in encapsulating all the required designs. Which I'm still getting used to wrapping my head around. All in all, the website uses the CSS file and a file for images and within the file I've managed to create a simple website detailing stuff. I Believe this task was good in how it introduced basic website coding.

Week 2 Responsive Web Apps

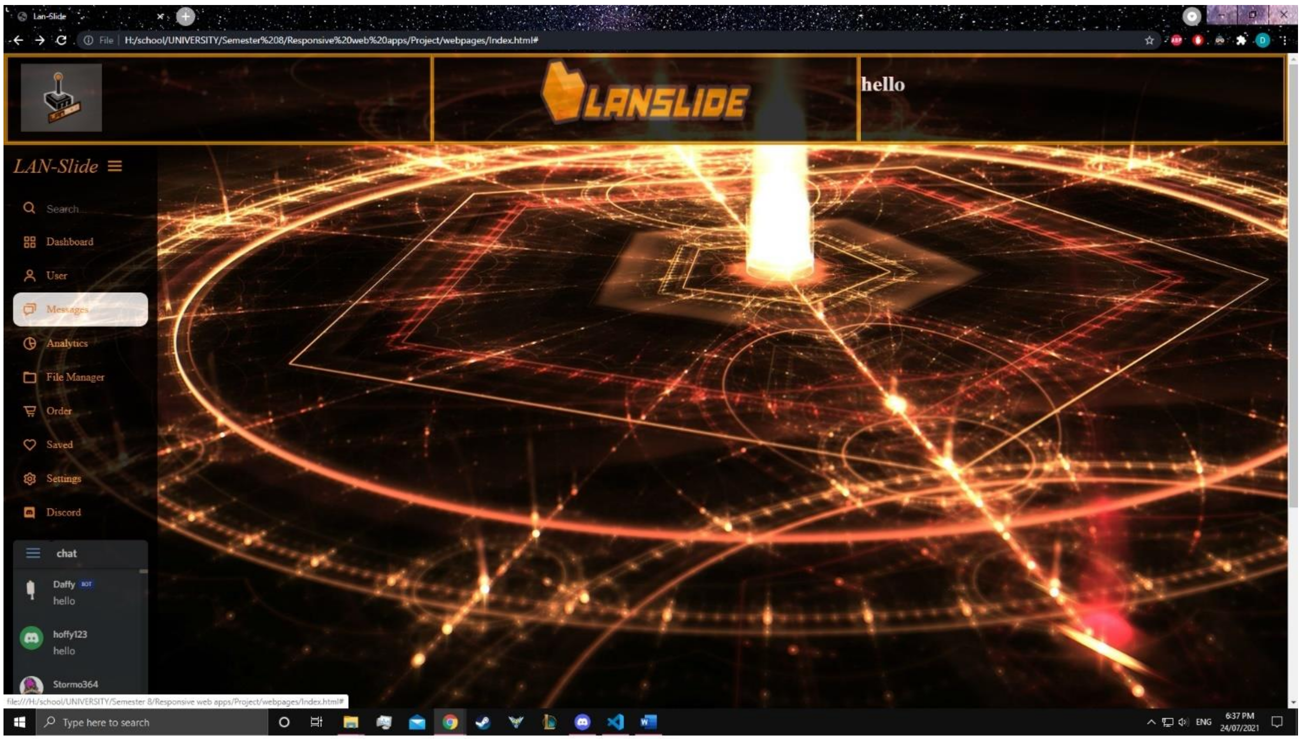
Task 1:

responsive analysis on the deakin website using desktop shows the images and canvas and divisions resize and form to the browser. However while on phone the website is resized and uses a scroll feature to go through the content with a drop down menu featuring the links. Through both devices they use percentages to form the page based on the device dimensions.

Cloud deakin website is broad and open with links and and are resized especially with pdf files and images. However in the mobile version the links are all condensed and compartmentalised within the frame.

Task 2: improve webpage

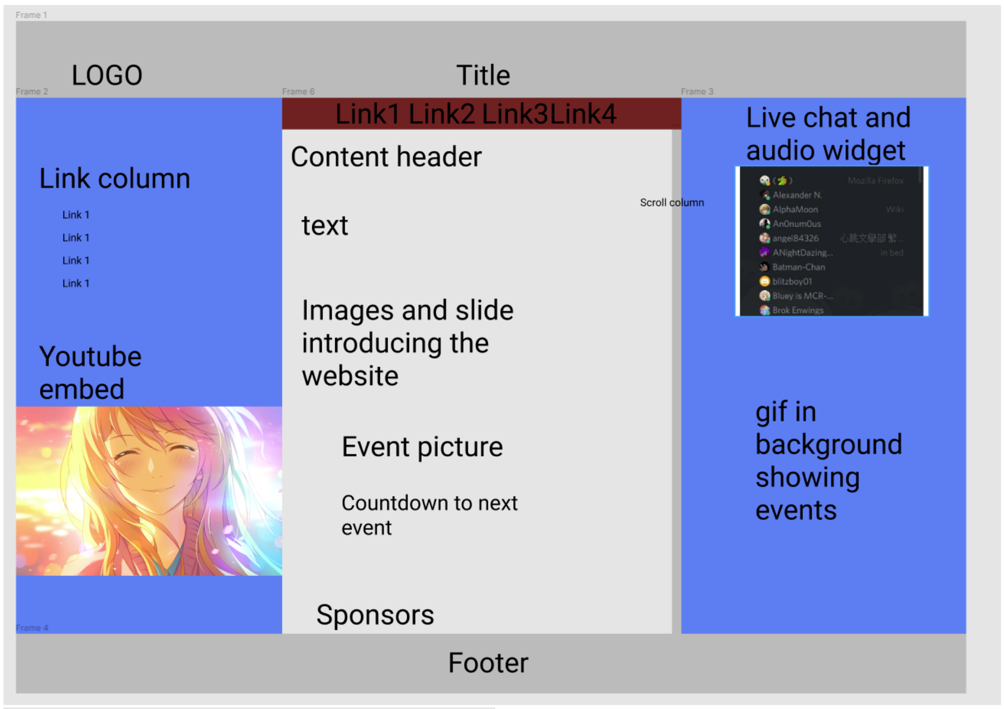
Within this task I updated the page layout using grid, navigation with a sidebar colour of the them Is orange and black, and adding an embedded discord bot as well as creating functionable control where you can toggle the sidebar.

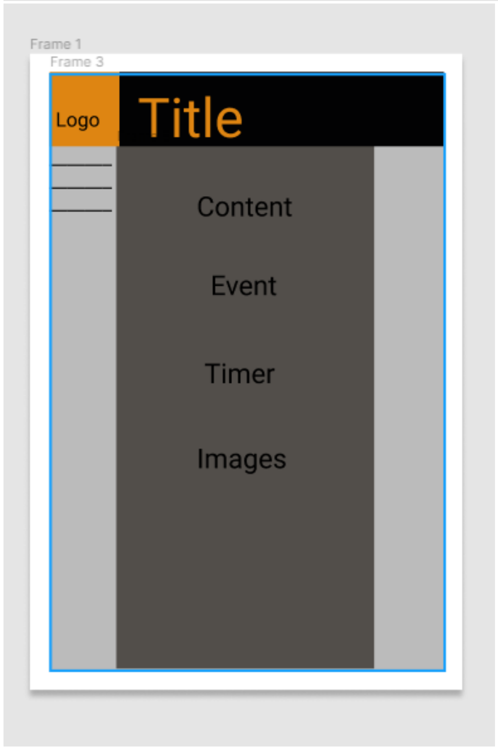


With this design it will be used within my project as a template and prototype of the project, as I said in week 1 format and layout is the most important step, by designing a foundation we can then implement iterative properties and functions within the document.

Task 3: User stories and UI/UX design

Here shows the design using figma to show what the prototype will be like. This also includes the mobile version.





The figma design is an illustration of a rough draft of the prototype to the final project. This rough draft done in week two looks decent with regards to the constraints of time and having little experience with the product. By doing this work it shows what type of website will be implemented however this is just a representation and not the final product.

User Stories: The user stories give insight and purpose of the website and its functions and what would matter to the client, this will illustrate the process and priority in which iterations of the website will be done.

User objective	Completion Criteria	Constraints	Priority

As an avid gamer, I would like the website to display the games being played and have the option to view the Lan's schedule	1.User should be able to see the types of games and schedule of the event. 2.event should be easy to see and access through the website. 3. Once viewed, the website should give options to the user to see the contents and details. 4.as the user it should give the option to enlist in the games and tournaments 5. there should be a search option.	Lan schedule and timing. Games available. Styling techniques. Content tbd varying between each Lan. **Examples**	**Priority: 1** High Priority
		Create a listed view, Create a popup view when searched Allow content of the game and previous event to be displayed.	Completion of this task is among the first to be done due to importance and design necessity.
User objective	**Completion Criteria**	**Constraints**	**Priority**
As the owner of the service, I would like the website to have a dynamic and responsive element such as slideshow, event content and design structure.	1.User should see video of past events, as well as slideshow 2.website should be interactive for the user, such as video and imagery used dynamically 3. simple and easy to see content 4.user interface can adapt to the device 5. enable plugins and embedded objects.	CSS and styling may interfere with website. Insufficient content of events Limited graphics **Examples**	**Priority: 1** High Priority
		Creating a slideshow Allowing video playback or gif. Changing the content displayed	Completion should be in the end process as artistic design is after layout

User objective	**Completion Criteria**	**Constraints**	**Priority**

As an event coordinator, I would like the events to be easily accessed and give information when clicked on.	1. Create content visible to users. 2. allow information available when clicked. 3. event coordinator should have access to change the content. 4. should give users to opt into the event. 5. should allow the ability to view past events	Lan schedule and timing. Games available. Styling techniques. Content tbd varying between each lan. Database to hold the information **Examples**	**Priority: 1** High Priority
		Create dynamic view and plugin to view in others pages as well as options to link other websites. Give content subject to change and listing to other events.	Completion Supplement to content being added, however the template and design should be down first.

User objective	Completion Criteria	Constraints	Priority
As the owner I would want the website to be able to play content from previous events on the front page	1. allow content to play on website opening. 2. should show sponsorships and advertise the content. 3. must be creatively engaging. 4. allow prizes and content to be displayed. 5. subject to change with content and event.	Lan schedule and Event contents Plugin and responsive template will be optimised for devices. **Examples**	**Priority: 1** High Priority
		Allow dynamic view to play previous content through Gif or slideshow	Completion should be expected within the second iteration.

User objective	Completion Criteria	Constraints	Priority
As the owner I would like a registration and customer login section so that we can have members join.	1. should allow users to sign up and login. 2. allow users to register for events. 3. should allow a secure connection. 4. should be https enabled. 5. should have security to protect data and stop hackers.	Database to save user input. Have security features. Quick and seamless login and usability. **Examples**	**Priority: 1** High Priority
		Allow access through other methods such as social media sites.	Completion should be expected within the third iteration.

User objective	Completion Criteria	Constraints	Priority
As the owner I want sponsors to be acknowledged and allow for further sponsors to apply.	1. should show sponsors. 2. should allow sponsorship. 3. should give details on their sponsorship contributions	Sponsor input and cooperation. Design constraints based on sponsors **Examples**	**Priority: 2** Medium Priority
		Display sponsors in images, display the contributions for the events.	Completion should be expected within the third iteration.

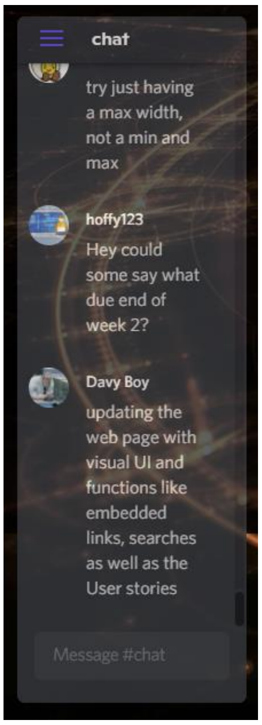
User objective	Completion Criteria	Constraints	Priority
As a sponsor I would like to see content related to us as well as our contribution	1. should show sponsorship images and content 2. should have links to sponsors. 3. allow prizes to be shown as well as competition stats.	Sponsorship deals, and desired style, competitions Content. **Examples**	**Priority: 2** Medium Priority
		Links to sponsors. Content displayed through website such as images Competition prizes. Embedded audio and video	Completion Third iteration

User objective	Completion Criteria	Constraints	Priority
As a user I would like the navigation to be easier to see and UI to be accessible from multiple places.	1. should allow link column and embedded content to access sites. 2. should allow dynamic and responsive elements based on device and software. 3. should give users option to change view. 4. should not have cluttered link bar. 5. colours should be easy to see and eye candy.	Link amount, link categories and required links, must be amenable to other webpages. **Examples**	**Priority: 2** mid Priority
		Change colour gradient. Change link style. Separate links based on categories. Size changes.	Completion should be expected within the third iteration.

User objective	Completion Criteria	Constraints	Priority
As a user I would want links to software and hardware which would be used.	1. should allow users download and get software. 2. give users hardware options. Within the website listing where to get it. 3. should have access to third party sites. 4. should give access to organisers and events.	Embedded access options. Support downloads. Give hardware options. **Examples**	**Priority: 2** Medium Priority
		Gives software available to download. Embed similar websites related to landslide so that users can get involved.	Completion in third iteration.
User objective	**Completion Criteria**	**Constraints**	**Priority**
As the owner I would want customer and event players to leave comments and reviews.	1. should allow users to leave comments. 2. allows advertisements 3. allow statistics to show consumers. 4. give options to see comments 5. allow forums.	Consumer input could be skewed Reliability of statistics based on users. **Examples**	**Priority: 2** Medium Priority
		Create forums Give tickets features For problem solving Have a Q&A site	Completion should be expected within the third iteration.

Task 4: Use graphics, media and API's

In task four I implemented embedded discord widget as well as updating the graphics and API for it, furthermore adding the styles and theme for the website.


chat

try just having a max width, not a min and max

hoffy123
Hey could some say what due end of week 2?

Davy Boy
updating the web page with visual UI and functions like embedded links, searches as well as the User stories

Using graphics and media as well as APIs were an important learning curve, this allows me to implement it within my final project, such as media files, sound files, API calls to discord, a set countdown clock and progress bar are instrumental to how my website functions.

Reflection

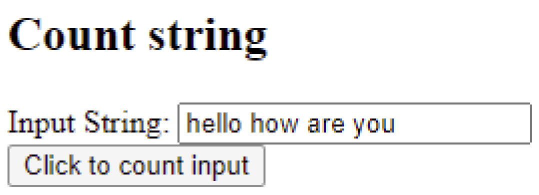
Within this week's tasks I got to learn what responsive web applications are and how they are created, by reacting to users' devices and applications we can manipulate the web applications to form to the user's device and program application. By investigating other websites, we can see how they are able to form to mobile phones and browsers. In week two learning from week 1 I believe the best method to iteratively work on a website is to start from the foundation, in this case I designed a project template with the layout and UI set and graphics implemented. User stories are set as to the client and businesses needs and gains perspective and interest on what they want. Particularly from stake holders on all aspects such as customers/gamers, business owner, organisers and advertisers.

Overall week two's task has allowed me to understand concepts related to responsive design and how they can be manipulated to cater to the user's device and software application. While furthermore implementing new methods and elements within HTML, JavaScript and CSS to improve my webpage.

Week 3 Responsive Web Apps

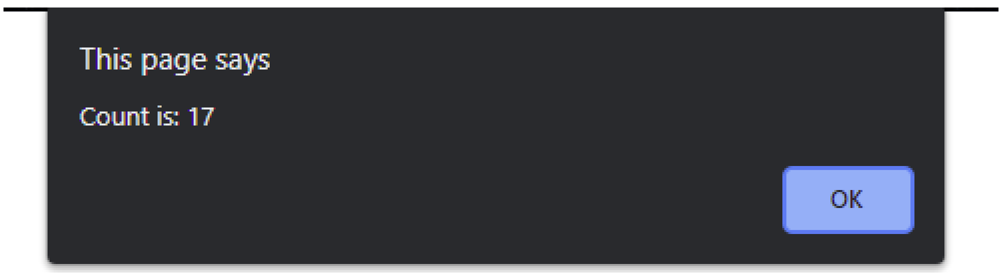
Task 1: JavaScript string methods

JavaScript is the programming language of the internet and is used in backend and front-end development. String methods are pretty simple with string we can manipulate them to count them or capitalize each word or other things. In this example I got user input and calculated the length of the input.


Count string

Input String: hello how are you

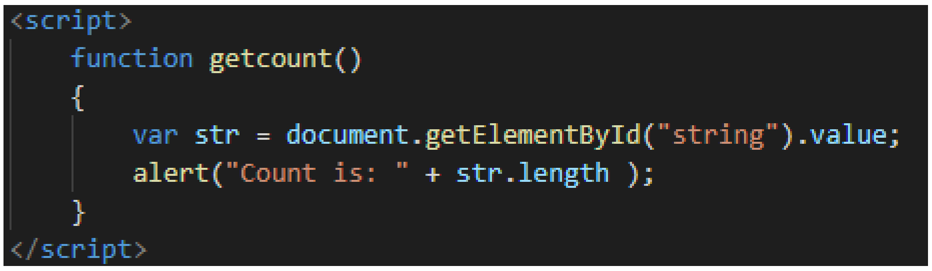
With this function when we input anything on the keyboard it will be considered as a string which is stored as an array of chars, the function will then count the array to get the total.


This page says

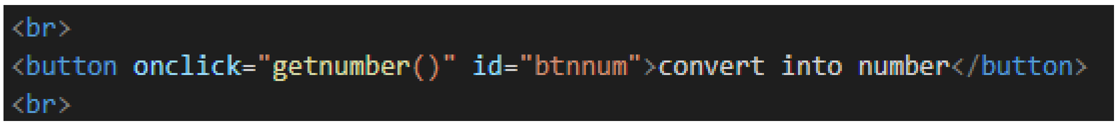
Count is: 17

This will then output the total.

The code to develop this is a simple function with two lines inside.

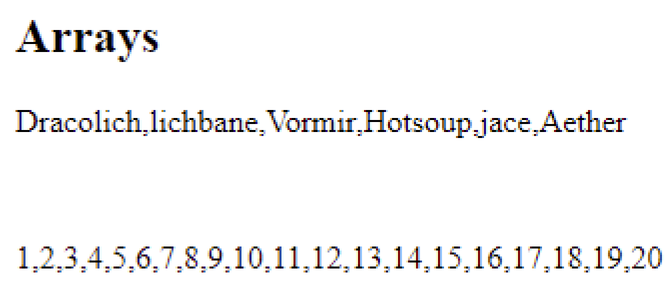

```
<script>
    function getcount()
    {
        var str = document.getElementById("string").value;
        alert("Count is: " + str.length );
    }
</script>
```

You assign this function to a specific element by using the ID.


```
<br>
<button onclick="getnumber()" id="btnnum">convert into number</button>
<br>
```

Task 2: JavaScript numbers and arrays

Creating numbers and arrays are quite simple, an array is a data structure that is a series of memory allocation which can hold information and data. It can be an array of any types such as int, float, string, char and bool.



Arrays

Dracolich,lichbane,Vormir,Hotsoup,jace,Aether

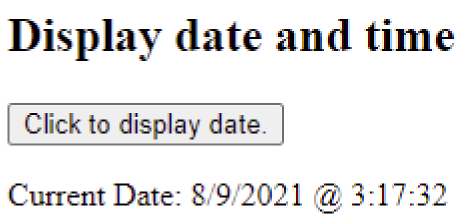
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20

As we can see we have an array of numbers and an array of strings displaying on the webpage.

Using number methods, I used the input to convert to a float value which will display any number present in float form or if there isn't it will give me NaN. Simple method using parse method with numbers to illustrate JavaScript functions.

Task 3: Get and Set methods

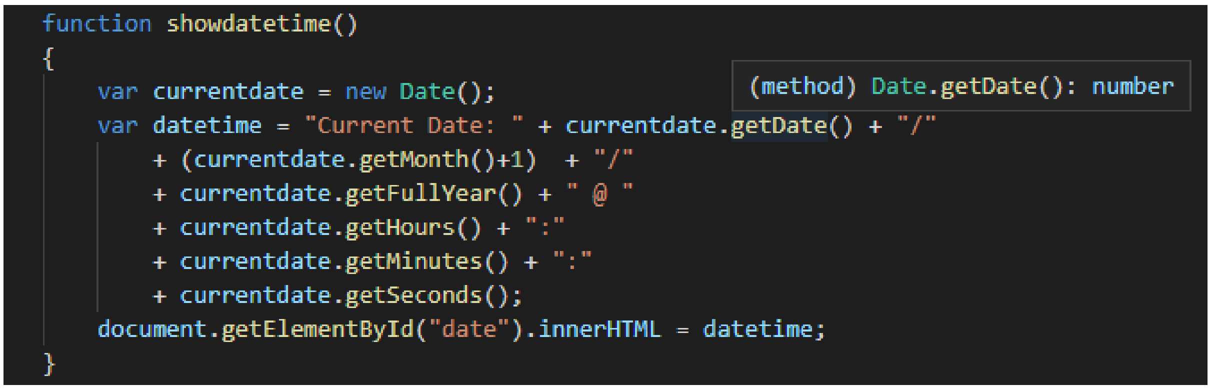
Get and Set are methods that can obtain and send data. To a specified variable or object.



Display date and time

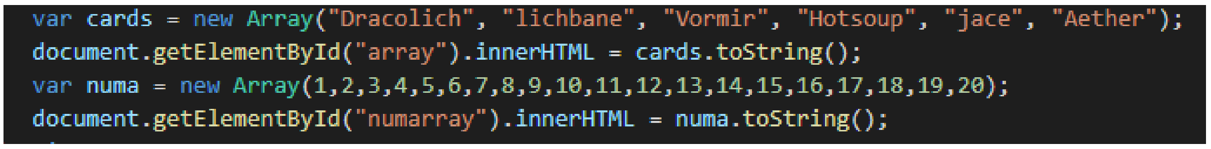
Current Date: 8/9/2021 @ 3:17:32

In this instance we are getting the time which is most possibly using Unix timestamp and formatting it to display how we want it.



```
function showdatetime()
{
    var currentdate = new Date();
    var datetime = "Current Date: " + currentdate.getDate() + "/"
        + (currentdate.getMonth()+1)  + "/"
        + currentdate.getFullYear() + " @ "
        + currentdate.getHours() + ":"
        + currentdate.getMinutes() + ":"
        + currentdate.getSeconds();
    document.getElementById("date").innerHTML = datetime;
}
```

Set methods is when we set data to an object or variable



```
var cards = new Array("Dracolich", "lichbane", "Vormir", "Hotsoup", "jace", "Aether");
document.getElementById("array").innerHTML = cards.toString();
var numa = new Array(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20);
document.getElementById("numarray").innerHTML = numa.toString();
```

In this instance we are setting an object type array with to a variable called cards and numa since JavaScript only allows var, let and const. set them and then we get the element and cast it to a string.

Task 4: Vue framework

Computed properties

With the next task we will be exploring Vue framework which will detail the design and syntax of creating functions using Vue.

```
<div id="example">
  <p>base message: "{{ message }}"</p>
  <p>reversed message: "{{ messagereversed }}"</p>
</div>
var vm = new Vue({
  el: '#example',
  data: {
    message: 'Hello how are you'
  },
  computed: {
    //getter
    messagereversed: function () {

      return this.message.split('').reverse().join('')
    }
  }
})
```

Computed property is an instance of a particular type/function that allows it to transform and perform calculations on data which can be reused.

Watchers is a method on listening to events and functions where you can perform operations on changing data.

In this example above we are reversing the message by assigning the example ID tag to the Vue application, this will then return the message in reverse order by splitting it then reverse joining it together again.

Class and style binding

In order to bind data and manipulate elements we use v-bind to handle them, Vue provides features to toggle classes in order to activate certain elements.

```
<div
  class="stat"
  v-bind:class="{ active: Active, 'text-danger': Error }"
></div>
```

And the following data:

```
data: {
  Active: true,
  Error: false
}
```

It will render:

```
<div class="stat active"></div>
```

Object syntax is to bind the class and then that class and data will then be used by the called sections within the html file. On binding classes, we can also return an object by binding to a computed property and apply a list of classes as well so that data can be manipulated.

```
Vue.component('my-component', {
  template: '<p class="foo bar">Hi</p>'
})
<my-component class="classID"></my-component>
<p class="This is the rendered document">Hi</p>
<my-component v-bind:class="{ active: Active }"></my-component>
<p class=" classID active">Hi</p>
```

On binding inline styles its best to bind directly to make the object cleaner and will be used in conjunction with computed properties.

```
<div v-bind:style="styles"></div>
data: {
  styles: {
    color: 'blue',
    fontSize: '12px'
  }
}
```

Conditional rendering

The conditional rendering v-if is like a normal if statement where it conditionally applies a method or value where it will only work if it is a Boolean logical value. The V is just used to say that it's a Vue function syntax.

```
<div v-if="type === 'Cake'">
Cake
</div>
<div v-else-if="type === 'Pie'">
Pie
</div>
<div v-else-if="type === 'Ice-Cream'">
Ice-Cream
</div>
<div v-else>
It is neither
</div>
```

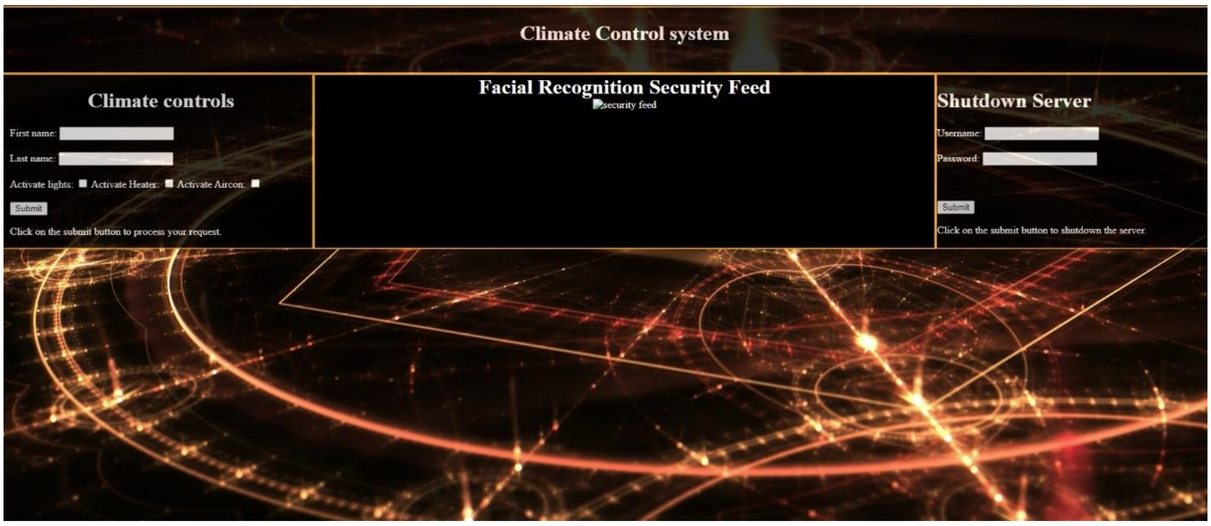
V-for can also be used to loop through an object

```
<ul id="v-for" class="example">
  <li v-for="Values">
    {{ value }}
  </li>
</ul>
new Vue({
  el: '#v-for',
  data: {
    object: {
      title: 'Start of list',
      author: 'Mr Cupcakes',
      publishedAt: '2021-09-08'
    }
  }
})
```

v-for block we can also have access to properties and be allowed to express that to html.

Form input bindings

To make a two-way data connection we can use v-model which automatically picks the correct way to update the element based on the input type.


Climate Control system
Climate controls
First name:
Last name:
Activate lights: Activate Heater: Activate Aircon:
Submit
Click on the submit button to process your request.
Facial Recognition Security Feed
security feed
Shutdown Server
Username:
Password:
Submit
Click on the submit button to shutdown the server.

Here we are using forms to send messages using requests, while simultaneously running a video feed to the website.

v-model uses different properties and produces different events for different input elements: text and text area elements use value property and input event.

checkboxes and radio buttons use checked property and change event. Select fields use a value as a prop and change as an event.

Vue components

Vue components are reusable functions which can be used in order to structure the components properly we register the functions to set on how we use them either in global scope or local scope.

Another insightful component is props which are custom attributes that allow us to dynamically cast values which will then become property of the component connected to it.

When the component grows its best practice to segregate.

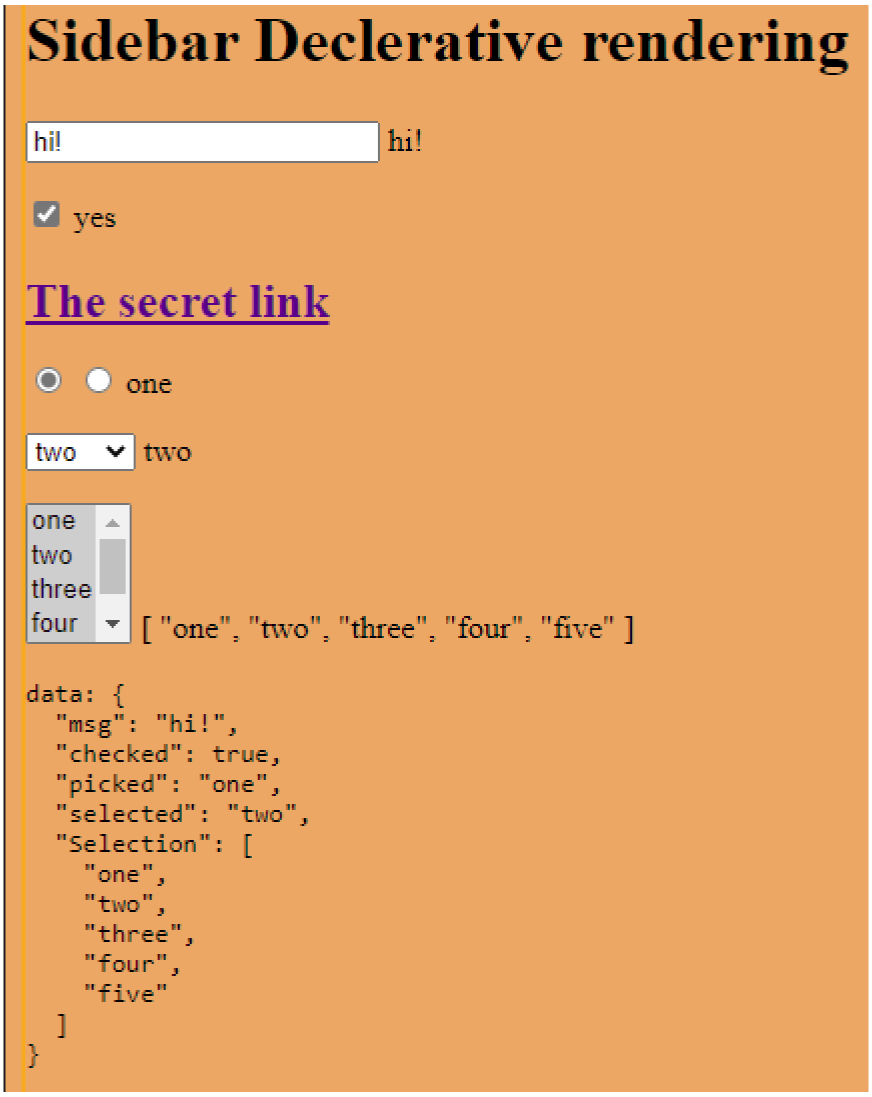
Reflection

The methods we covered on Vue is important and discusses the basic operations on how Vue is implemented directly with CSS and HTML by providing simple and ergonomic design to create complex functions. Such as properties and watchers to render and allow the DOM to be manipulated. The Vue components to style and bind elements within Vue either inline or externally offer simple processes to achieve a clean and working environment. This task has helped me understand the concepts on creating a website with simple and clean architecture.

Week 4 Responsive Web Apps

Task 1: JavaScript string methods

Declarative rendering is the ability to bind and set data using Vue framework so that it can render to the DOM, honestly Vue is hard at first if you don't know the basics but with experience and learning the fundamentals it gradually becomes easier to understand and how to implement the code. In this case we are using v-models, if statements, check elements and others.



Sidebar Declerative rendering

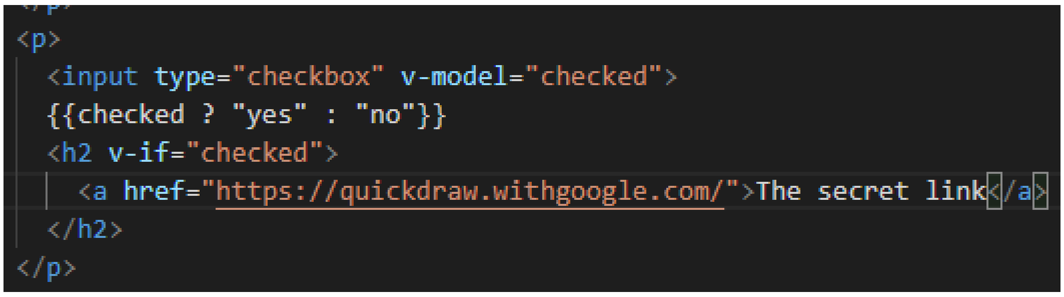
["one", "two", "three", "four", "five"]

```
data: {
  "msg": "hi!",
  "checked": true,
  "picked": "one",
  "selected": "two",
  "Selection": [
    "one",
    "two",
    "three",
    "four",
    "five"
  ]
}
```

Within the webpage, I implemented rendering to display the information that is being changed within the app to show what is happening behind the user interface and what it will display to the DOM.

Task 2: Conditionals and loops

V-if is an if statement where we can display, change or create elements and data based on conditions that are Boolean.



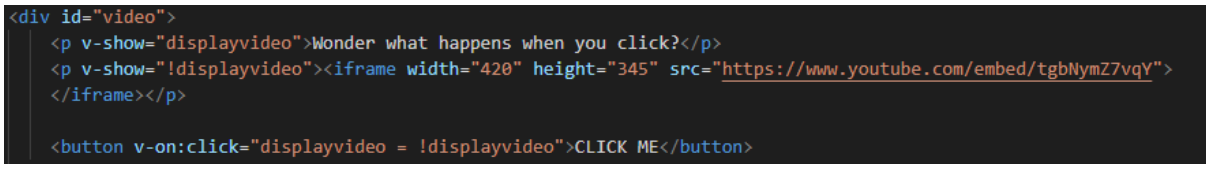
```
<p>
  <input type="checkbox" v-model="checked">
  {{checked ? "yes" : "no"}}
  <h2 v-if="checked">
    <a href="https://quickdraw.withgoogle.com/">The secret link</a>
  </h2>
</p>
```

V-else

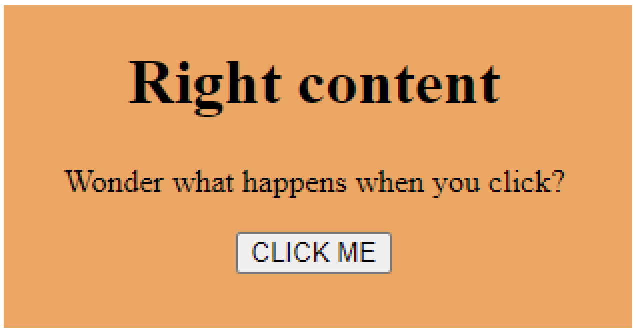
In this case when the condition is met, we can display a href link. And V-else conditionally renders something if the other condition is not met.

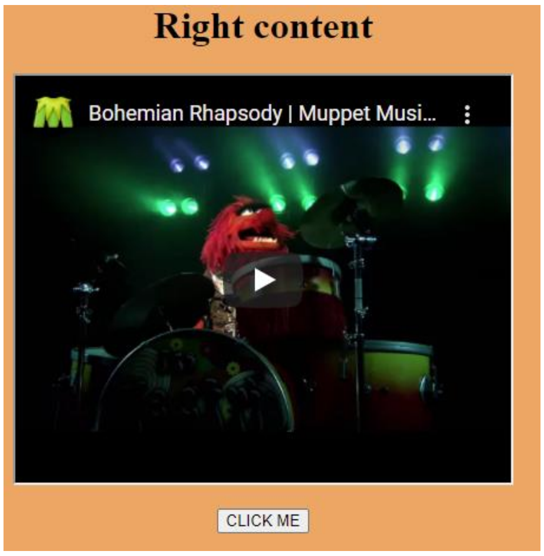
V-show

V-show displays elements to the DOM, when a condition is met by changing CSS and HTML properties, if the condition is true, it will make it visible and if not, it will make it invisible



```
<div id="video">
    <p v-show="displayvideo">Wonder what happens when you click?</p>
    <p v-show="!displayvideo"><iframe width="420" height="345" src="https://www.youtube.com/embed/tgbNymZ7vqY">
    </iframe></p>

    <button v-on:click="displayvideo = !displayvideo">CLICK ME</button>
```

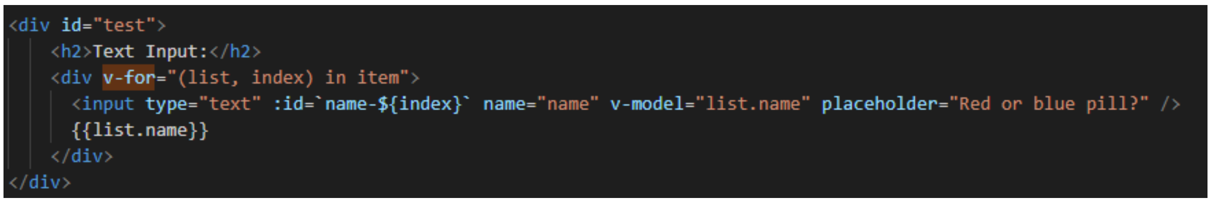




Here we see what happens when the v-show is used to dynamically change the webpage and rendering an embedded video.

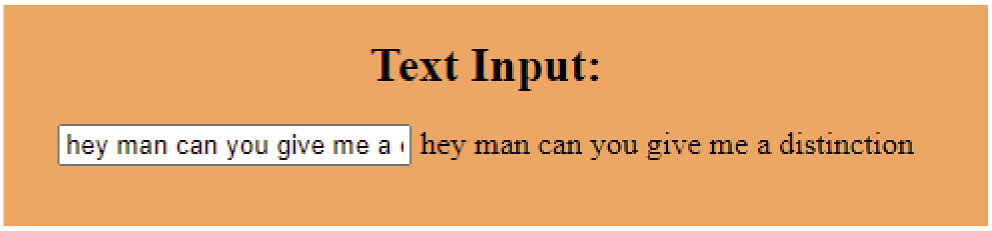
V-for

V-for renders a list of data within an array or objects, by binding the data to elements within the HTML document we can render it in view to the UI.



```
<div id="test">
    <h2>Text Input:</h2>
    <div v-for="(list, index) in item">
      <input type="text" :id=`name-${index}` name="name" v-model="list.name" placeholder="Red or blue pill?" />
      {{list.name}}
    </div>
</div>
```

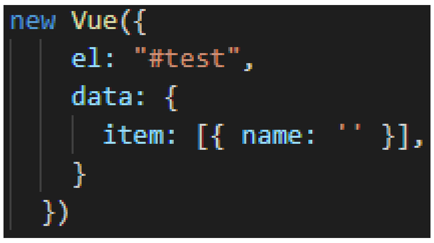
The v-for list will render to the DOM the typed-out string so that you can see it update live as your typing.



Text Input:

hey man can you give me a (hey man can you give me a distinction

This is implemented externally using a Vue app



```
new Vue({
    el: "#test",
    data: {
      item: [{ name: '' }],
    }
  })
```

Task 3: Discussed with Shang as I attend Thursday evening tutorials due to time constraints and its eventful, with the project I got decent advice and what I should do to implement my project.

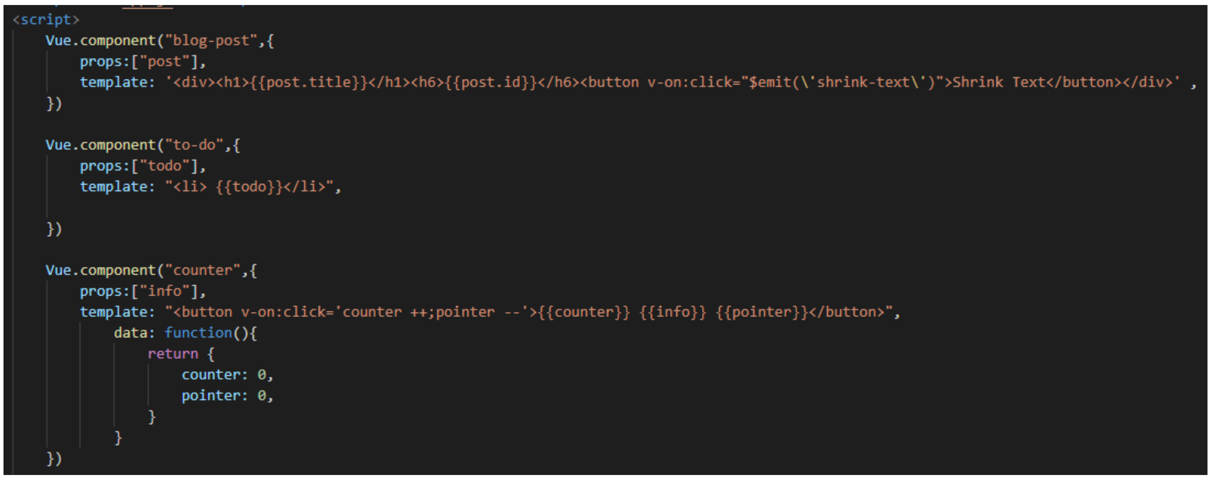
Reflection

Within the weeks task I implemented a simple responsive grid layout with multiple Vue components such as v-if, form, else, show and for loops along with embedding and layout. By using these elements and methods we can learn on how to manipulate and create layout modification by using applications. Overall, I believe I need to study and learn more about the concepts of Vue frameworks and implementation methods to better improve my website designs. During the project I hope to implement these elements such as v-for, v-if and v-show to create a dynamic and responsive design.

Week 5 Responsive Web Apps

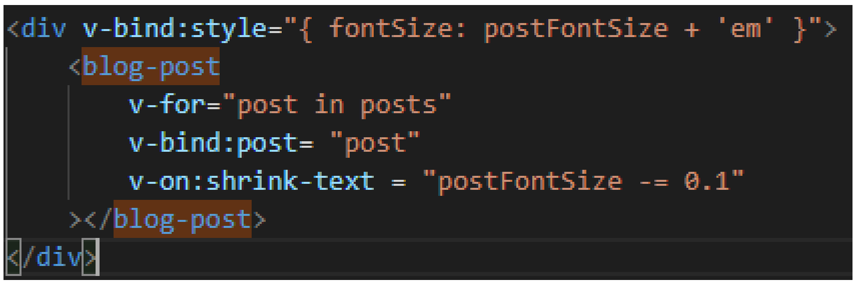
Task 1: Composing Components

In Vue a component is a reusable Vue instance with a modular design where we can customize HTML and CSS within a template, we can define our component and initialise it within HTML tags which will then be loaded up into the html DOM.



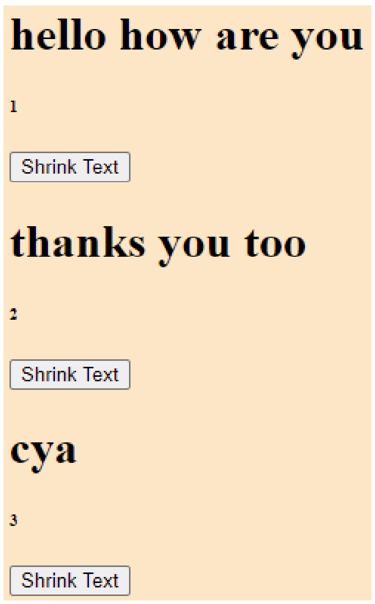
```
<script>
    Vue.component("blog-post",{
        props:["post"],
        template: '<div><h1>{{post.title}}</h1><h6>{{post.id}}</h6><button v-on:click="$emit(\'shrink-text\')">Shrink Text</button></div>' ,
    })

    Vue.component("to-do",{
        props:["todo"],
        template: "<li> {{todo}}</li>",

    })

    Vue.component("counter",{
        props:["info"],
        template: "<button v-on:click='counter ++;pointer --'>{{counter}} {{info}} {{pointer}}</button>",
            data: function(){
                return {
                    counter: 0,
                    pointer: 0,
                }
            }
    })
```

We call a component and use a template to inject html structure within the division to modify the DOM.

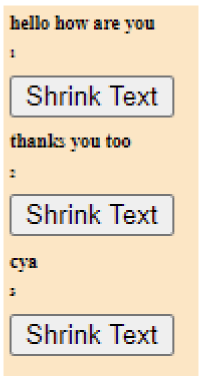


```
<div v-bind:style="{ fontSize: postFontSize + 'em' }">
    <blog-post
        v-for="post in posts"
        v-bind:post= "post"
        v-on:shrink-text = "postFontSize -= 0.1"
    ></blog-post>
</div>
```

Call blog-post and it sets the component div into that section, and what this does is shrink the text.



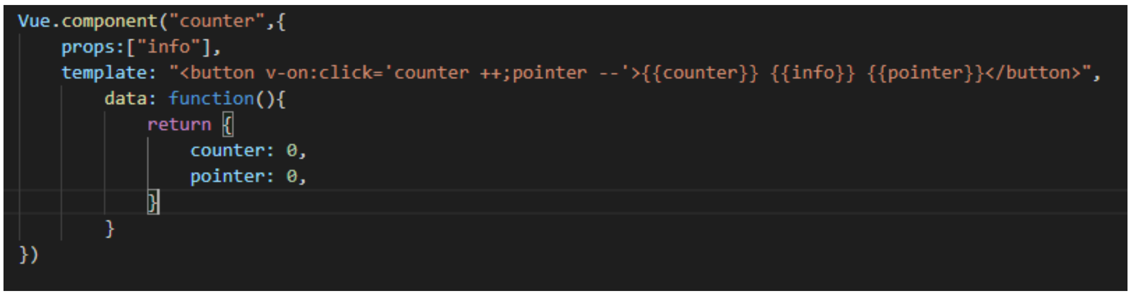
→



Task 2: Exploring the framework

Vue components are methods that create custom elements. This component involves using a template to modify data on the webpage by processing the props using a function.

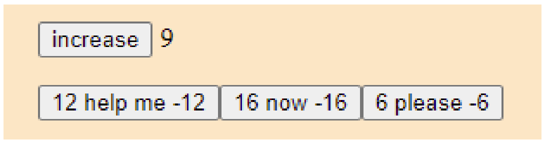
Code:



```
Vue.component("counter",{
    props:["info"],
    template: "<button v-on:click='counter ++;pointer --'>{{counter}} {{info}} {{pointer}}</button>",
        data: function(){
            return {
                counter: 0,
                pointer: 0,
            }
        }
})
```

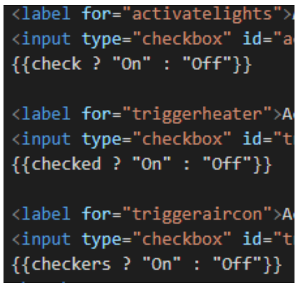
Within the template section we modify the data to display the counter and pointer and update it when the button is clicked.

Output:



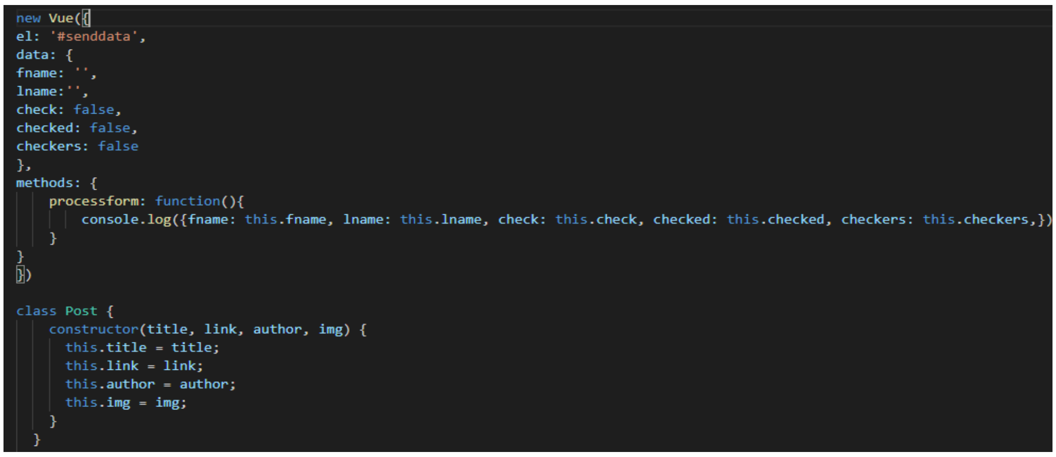
The output shows that we can modify each button respectively.

We can also use custom V-forms to process data and switch between the data that is displayed onscreen.



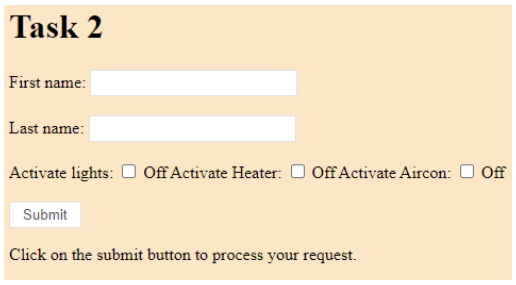
```
<label for="activatelights">
<input type="checkbox" id="a
{{check ? "On" : "Off"}}

<label for="triggerheater">A
<input type="checkbox" id="t
{{checked ? "On" : "Off"}}

<label for="triggeraircon">A
<input type="checkbox" id="t
{{checkers ? "On" : "Off"}}
```

This allows us to toggle the data and switch between the two elements.



```
new Vue({
el: '#senddata',
data: {
fname: '',
lname:'',
check: false,
checked: false,
checkers: false
},
methods: {
    processform: function(){
        console.log({fname: this.fname, lname: this.lname, check: this.check, checked: this.checked, checkers: this.checkers,})
    }
}
})

class Post {
    constructor(title, link, author, img) {
      this.title = title;
      this.link = link;
      this.author = author;
      this.img = img;
    }
  }
```

Output



Task 2

First name:

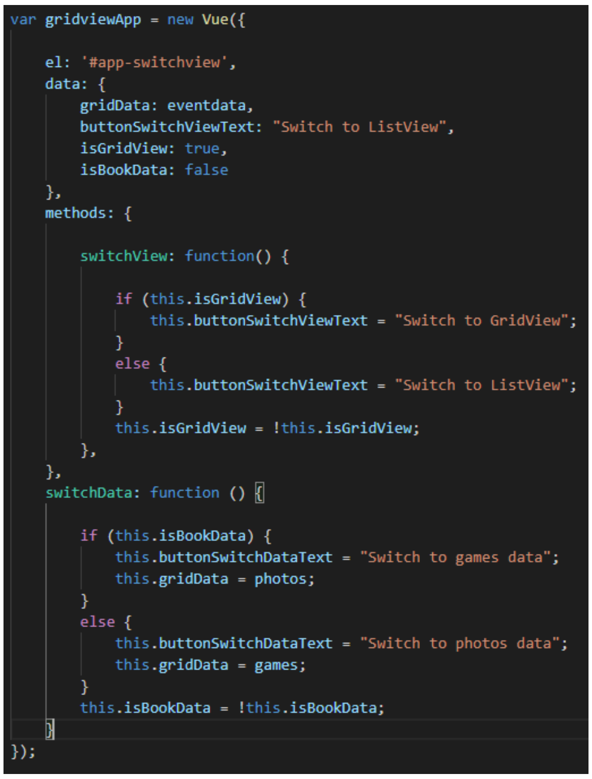
Last name:

Activate lights: ☐ Off Activate Heater: ☐ Off Activate Aircon: ☐ Off

Click on the submit button to process your request.

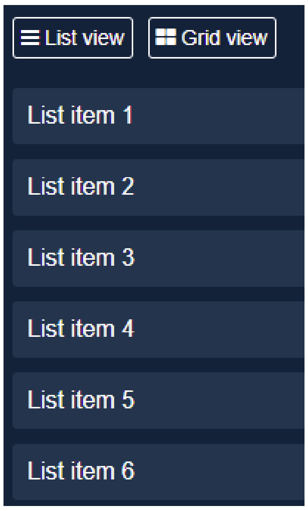
Task 3: handling user input

Handling user input is a way to make reactive elements within the HTML document to change depending on what methods or items are listening for user input.



```
var gridviewApp = new Vue({

    el: '#app-switchview',
    data: {
        gridData: eventdata,
        buttonSwitchViewText: "Switch to ListView",
        isGridView: true,
        isBookData: false
    },
    methods: {

        switchView: function() {

            if (this.isGridView) {
                this.buttonSwitchViewText = "Switch to GridView";
            }
            else {
                this.buttonSwitchViewText = "Switch to ListView";
            }
            this.isGridView = !this.isGridView;
        },
    },
    switchData: function () {

        if (this.isBookData) {
            this.buttonSwitchDataText = "Switch to games data";
            this.gridData = photos;
        }
        else {
            this.buttonSwitchDataText = "Switch to photos data";
            this.gridData = games;
        }
        this.isBookData = !this.isBookData;
    }
});
```

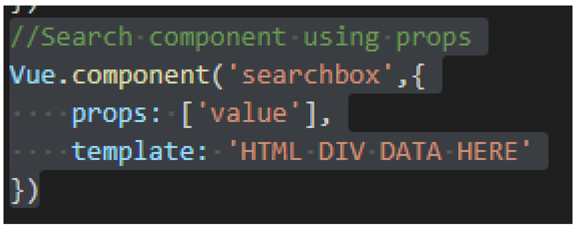
This code snippet illustrates how the HTML document would switch between list or grid view and what items you would want to display.



as we can see in this output, we can switch between list view and grid view interchangeably with handling user input, this can also be done with components to signal which component will be used by user input as well.

Task 4: Learn components

Component registration is the practice of naming conventions for our components that will either be used in global or local scopes; this is done to organise and call different HTML designs when these components are activated.



```
//Search component using props
Vue.component('searchbox',{
    props: ['value'],
    template: 'HTML DIV DATA HERE'
})
```

In this code snippet we can create components of Vue that can be later called and mixed in with other Vue apps. this is a simple and efficient method on rendering data to the DOM based on user input and conditions.

Props are custom attributes you can register to a component or app and it will become a property of that instance.

Dynamic and Async components allows us to save and cache the state in which certain elements were in in order to allow functionality for the user and keep properties alive or archived.

Custom events work by emitting an event when the listener reacts to user input, this is used to notify child or parent components that something has changed and will be updated.

Reflection

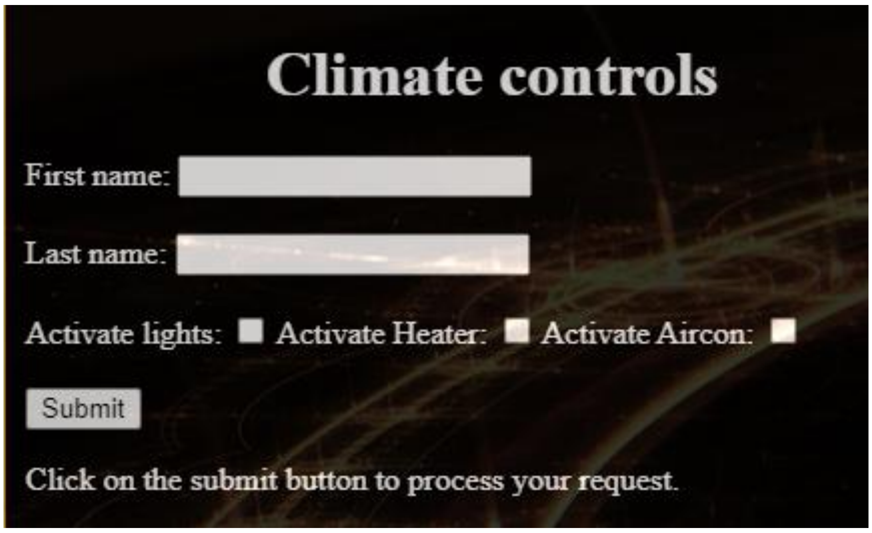
Week five has been an interesting and complex learning curve where implementation of components can be mixed in with other Vue apps and we can supply templates in order to change functionality of the structure of the website. Components allow us to have inheritance and abstraction within our website where we can modify and overload certain aspects to create a more responsive and reactive design. I found this incredibly interesting and important concept to learn. Overall, the week has been productive and has shown me how to use user interaction within the website and handle object notation. For my two components I will be using templates and props to handle some information and website function. For custom events such as onclick or change will be incredibly useful to allow functionality and reactiveness with user input. This week has been a hard but convenient week since it has allowed me to learn more design concepts that I can add to my website.

Week 6 Responsive Web Apps

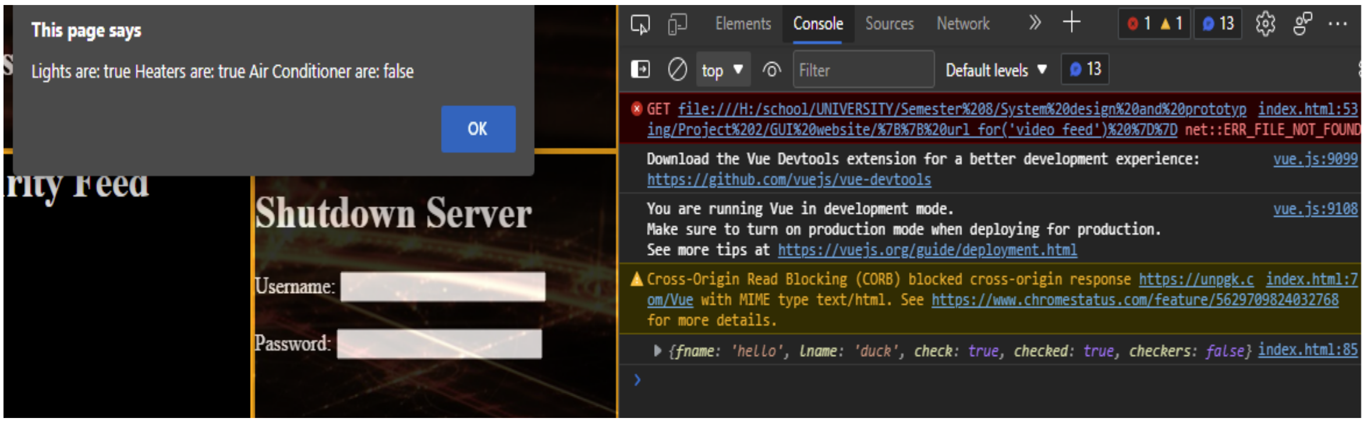
Task 1: V-model for handling user inputs

V-model is a Vue designed form handling function that allows two-way data binding between inputs or components. This can also be used with HTTP requests to send and receive data based on the value inside the form.

UI for the form data



Output



Lights are: true Heaters are: true Air Conditioner are: false

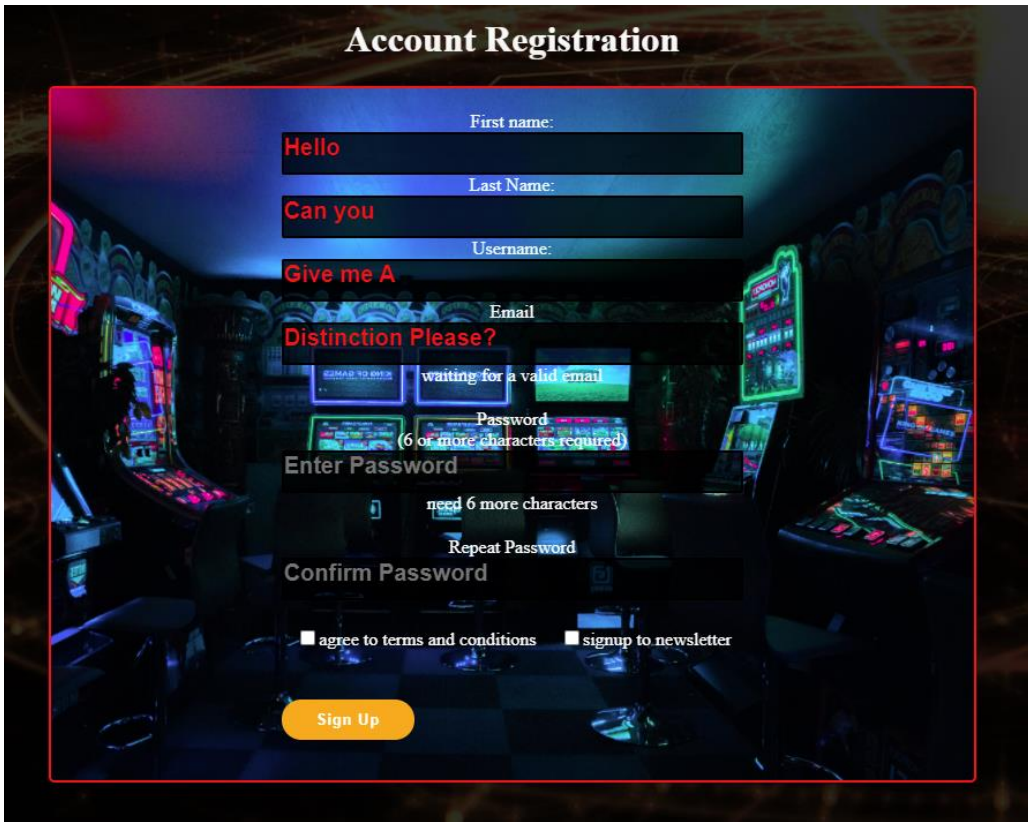
```
GET file:///H:/school/UNIVERSITY/Semester%208/System%20design%20and%20prototyp  index.html:53
ing/Project%202/GUI%20website/%7B%7B%20url_for('video_feed')%20%7D%7D net::ERR_FILE_NOT_FOUND
Download the Vue Devtools extension for a better development experience:  vue.js:9099
https://github.com/vuejs/vue-devtools
You are running Vue in development mode.                                  vue.js:9108
Make sure to turn on production mode when deploying for production.
See more tips at https://vuejs.org/guide/deployment.html
Cross-Origin Read Blocking (CORB) blocked cross-origin response https://unpgk.c  index.html:7
om/Vue with MIME type text/html. See https://www.chromestatus.com/feature/5629709824032768
for more details.
 ▶ {fname: 'hello', lname: 'duck', check: true, checked: true, checkers: false} index.html:85
```

In the output we are alerting the webpage to make a post telling us what has happened within the form data, we can also write the values of the form and post it to the terminal. Additionally, we can also post form data to send information.

This has been used in a flask server to send information and data in order to activate certain functions within a device to control HVAC system.

Task 2: checkbox

I will be using a checkbox within my registration and other parts of the website to get authentication from users based on terms and conditions as well as validation on preferences and rule obligations.


Account Registration

First name:
Hello

Last Name:
Can you

Username:
Give me A

Email
Distinction Please?

waiting for a valid email

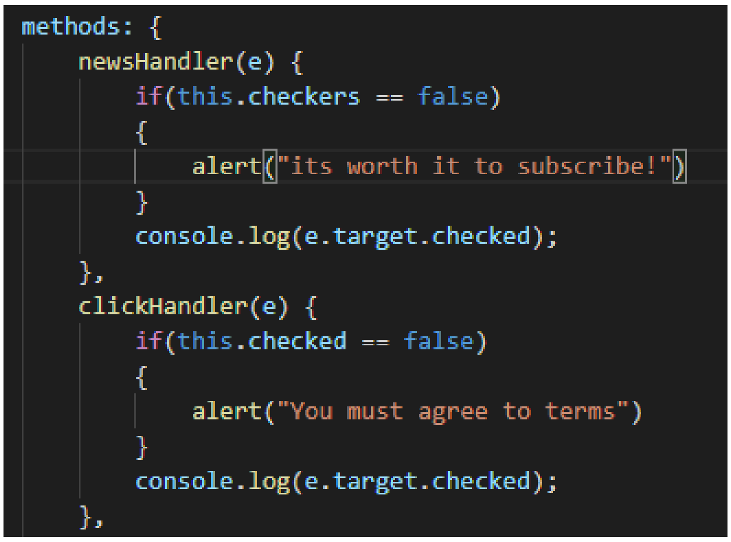
Password
(6 or more characters required)
Enter Password

need 6 more characters

Repeat Password
Confirm Password

☐ agree to terms and conditions ☐ signup to newsletter

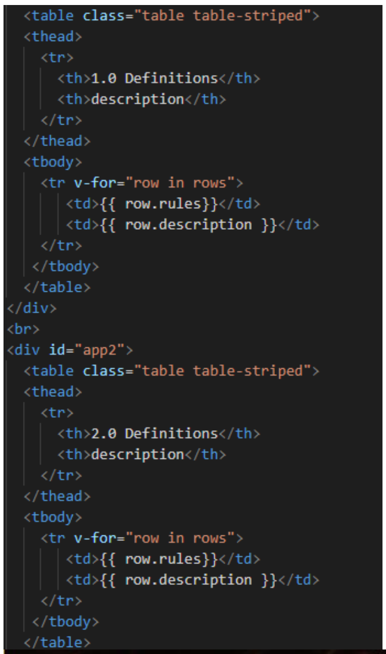
When the checkboxes are true the user will be allowed to register an account if they have not agreed to the TOC they will not be allowed to progress.


```
methods: {
    newsHandler(e) {
        if(this.checkers == false)
        {
            alert("its worth it to subscribe!")
        }
        console.log(e.target.checked);
    },
    clickHandler(e) {
        if(this.checked == false)
        {
            alert("You must agree to terms")
        }
        console.log(e.target.checked);
    },
```

This code snippet shows how the implementation of the checkbox will be used as a prototype for now, full implementation within the project later.

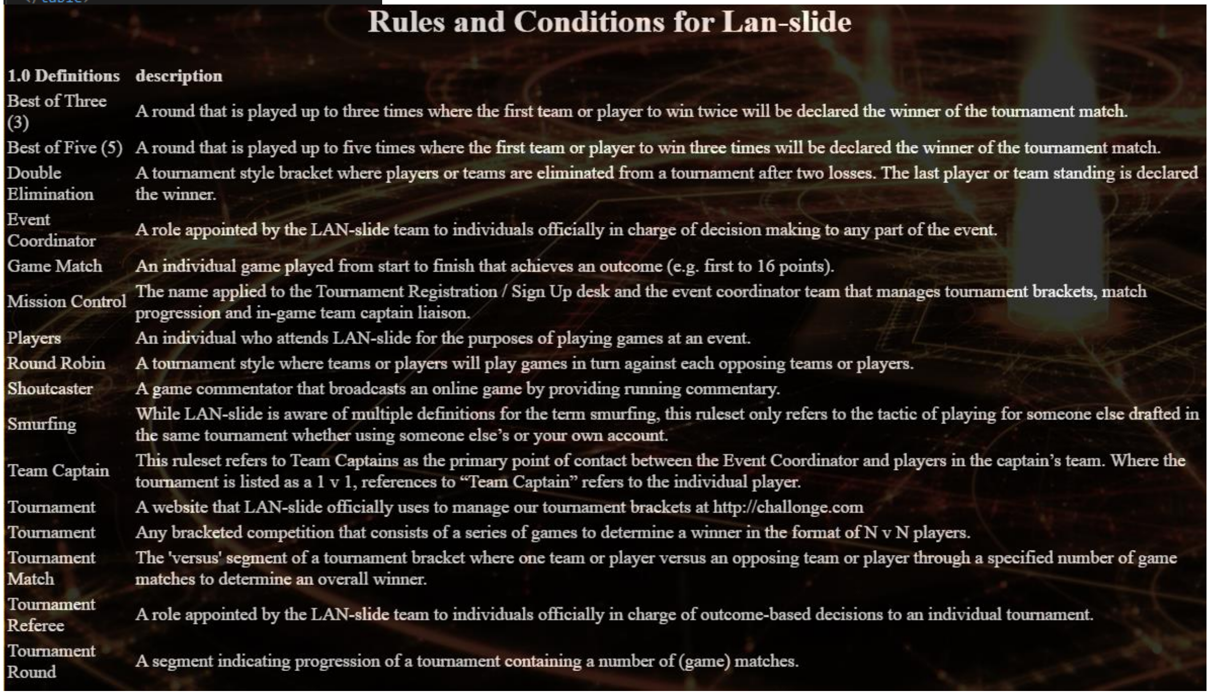
Task 3: V-For

V-for is a for loop within Vue that can be used to dynamically render data and objects to the DOM as well as modify HTML layout and design. Within my website I will be using it to render data and information within list format in order to project the information in an Organised and aesthetically pleasing manner.



```
  <table class="table table-striped">
  <thead>
    <tr>
      <th>1.0 Definitions</th>
      <th>description</th>
    </tr>
  </thead>
  <tbody>
    <tr v-for="row in rows">
       <td>{{ row.rules}}</td>
       <td>{{ row.description }}</td>
    </tr>
   </tbody>
  </table>
</div>
<br>
<div id="app2">
  <table class="table table-striped">
  <thead>
    <tr>
      <th>2.0 Definitions</th>
      <th>description</th>
    </tr>
  </thead>
  <tbody>
    <tr v-for="row in rows">
       <td>{{ row.rules}}</td>
       <td>{{ row.description }}</td>
    </tr>
   </tbody>
  </table>
```

This code snippet shows the use of v-for in the HTML section we are rendering the v-for within a table which will implement new tables and rows until it completes all elements within the dictionary.



Rules and Conditions for Lan-slide

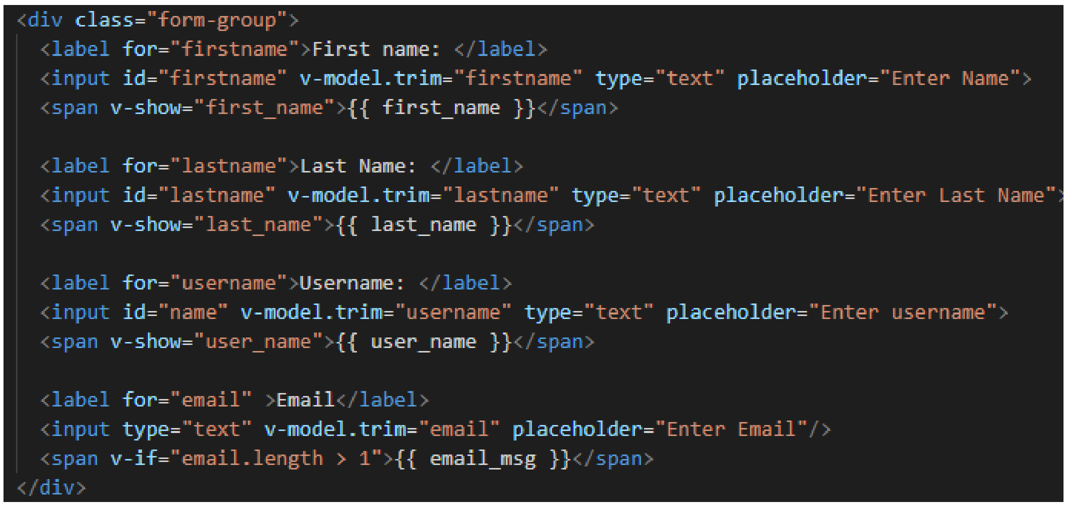
1.0 Definitions	description
Best of Three (3)	A round that is played up to three times where the first team or player to win twice will be declared the winner of the tournament match.
Best of Five (5)	A round that is played up to five times where the first team or player to win three times will be declared the winner of the tournament match.
Double Elimination	A tournament style bracket where players or teams are eliminated from a tournament after two losses. The last player or team standing is declared the winner.
Event Coordinator	A role appointed by the LAN-slide team to individuals officially in charge of decision making to any part of the event.
Game Match	An individual game played from start to finish that achieves an outcome (e.g. first to 16 points).
Mission Control	The name applied to the Tournament Registration / Sign Up desk and the event coordinator team that manages tournament brackets, match progression and in-game team captain liaison.
Players	An individual who attends LAN-slide for the purposes of playing games at an event.
Round Robin	A tournament style where teams or players will play games in turn against each opposing teams or players.
Shoutcaster	A game commentator that broadcasts an online game by providing running commentary.
Smurfing	While LAN-slide is aware of multiple definitions for the term smurfing, this ruleset only refers to the tactic of playing for someone else drafted in the same tournament whether using someone else's or your own account.
Team Captain	This ruleset refers to Team Captains as the primary point of contact between the Event Coordinator and players in the captain's team. Where the tournament is listed as a 1 v 1, references to "Team Captain" refers to the individual player.
Tournament	A website that LAN-slide officially uses to manage our tournament brackets at http://challonge.com
Tournament	Any bracketed competition that consists of a series of games to determine a winner in the format of N v N players.
Tournament Match	The 'versus' segment of a tournament bracket where one team or player versus an opposing team or player through a specified number of game matches to determine an overall winner.
Tournament Referee	A role appointed by the LAN-slide team to individuals officially in charge of outcome-based decisions to an individual tournament.
Tournament Round	A segment indicating progression of a tournament containing a number of (game) matches.

This is the output shown on the website using v-for rendering, it renders the dictionary with ids and data. A more complex V-for will be used to generate types of viewing formats for users to browse information and data and will allow options to choose certain seating and memberships.

Task 4: Modifiers within project

A modifier is a suffix which you can add onto a Vue directive or similar Vue functions to add extra functionality and design within Vue.

In some instances, we will be using “trim” which gets rid of whitespace in input areas automatically. And “lazy” we will be using so that it syncs after event changes are made.



```
<div class="form-group">
  <label for="firstname">First name: </label>
  <input id="firstname" v-model.trim="firstname" type="text" placeholder="Enter Name">
  <span v-show="first_name">{{ first_name }}</span>

  <label for="lastname">Last Name: </label>
  <input id="lastname" v-model.trim="lastname" type="text" placeholder="Enter Last Name"
  <span v-show="last_name">{{ last_name }}</span>

  <label for="username">Username: </label>
  <input id="name" v-model.trim="username" type="text" placeholder="Enter username">
  <span v-show="user_name">{{ user_name }}</span>

  <label for="email" >Email</label>
  <input type="text" v-model.trim="email" placeholder="Enter Email"/>
  <span v-if="email.length > 1">{{ email_msg }}</span>
</div>
```

The trim modifier works within this code snippet to get rid of whitespace within the input fields.

Reflection

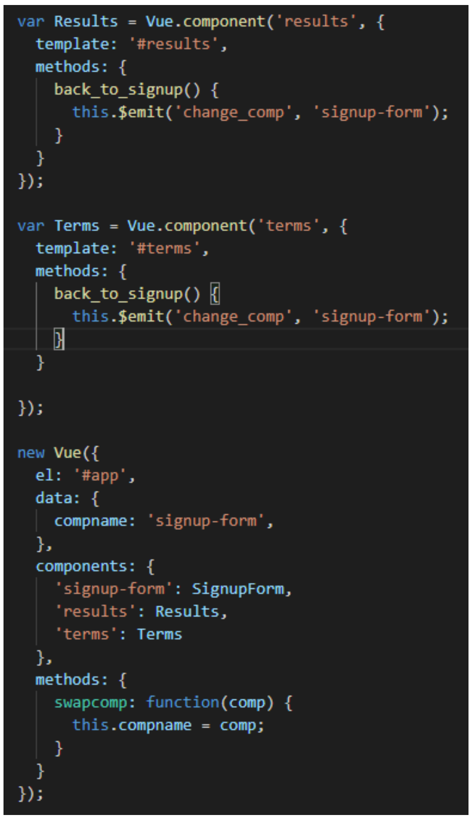
This week has been about implementing Vue concepts within my website, along the way in my website I have implemented various Vue applications to handle functionality within the project. Vue modifiers for user input and event handling to react to user input has been most useful, furthermore creating dictionary within Vue apps has allowed me a clean and decluttered method of loading information and content. V-model is also being used to handle registration and login so that user can have access to the site, it will then be sent using a post request handled by the form. This week has been challenging as I have had to implement various Vue components that I did not really need to add however, it has been beneficial to learn new concepts and ideas.

Week 7 Responsive Web Apps

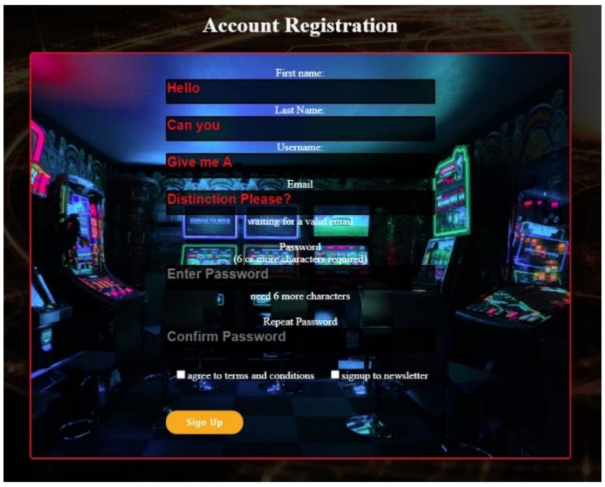
Task 1: Local and global registration

Registration is the method in using different Vue components that render and process various data and functions. Global registration is done by using a prefix of Vue.component to the application which will make it a JavaScript object that can be used globally. In order to make it a local component we use Var and "name" = Vue.component. This will allow us to create a local component. Components are hierarchy based which means they act like a family tree structure where child components are inherited by parent components and can be used by the parents. The parent component is usually the root instance.

Below is a code snippet of local registration of components being used within registration page.



```
var Results = Vue.component('results', {
  template: '#results',
  methods: {
    back_to_signup() {
      this.$emit('change_comp', 'signup-form');
    }
  }
});

var Terms = Vue.component('terms', {
  template: '#terms',
  methods: {
    back_to_signup() {
      this.$emit('change_comp', 'signup-form');
    }
  }

});

new Vue({
  el: '#app',
  data: {
    compname: 'signup-form',
  },
  components: {
    'signup-form': SignupForm,
    'results': Results,
    'terms': Terms
  },
  methods: {
    swapcomp: function(comp) {
      this.compname = comp;
    }
  }
});
```



Account Registration

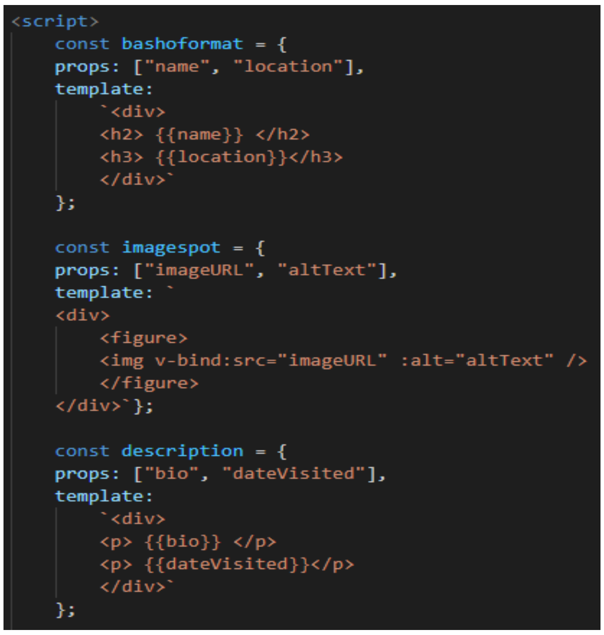
First name:
Hello
Last Name:
Can you
Username:
Give me A
Email
Distinction Please?
waiting for a valid email
Password
(6 or more characters required)
Enter Password
need 6 more characters
Repeat Password
Confirm Password

This is the output used for local and global registration of components and route application. The application uses three components to process

Task 2: Coding Props

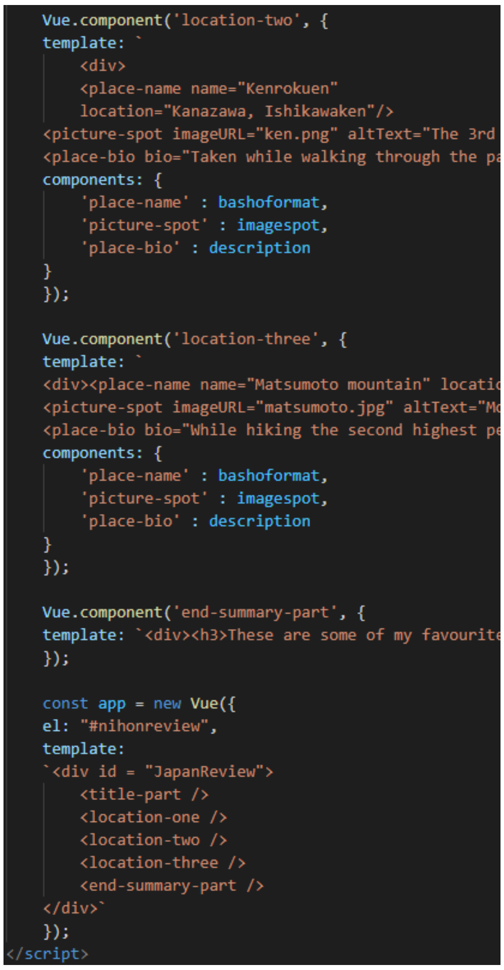
Props are a way we can pass data through inherited components by binding them to variables. This is done using a family tree data structure. The data flows down to the root component.

Within my project I will be using props to pass information and data to certain templates in order to render different information based on user input. For this example, I will demonstrate displaying some of my fondest memories while in Japan.



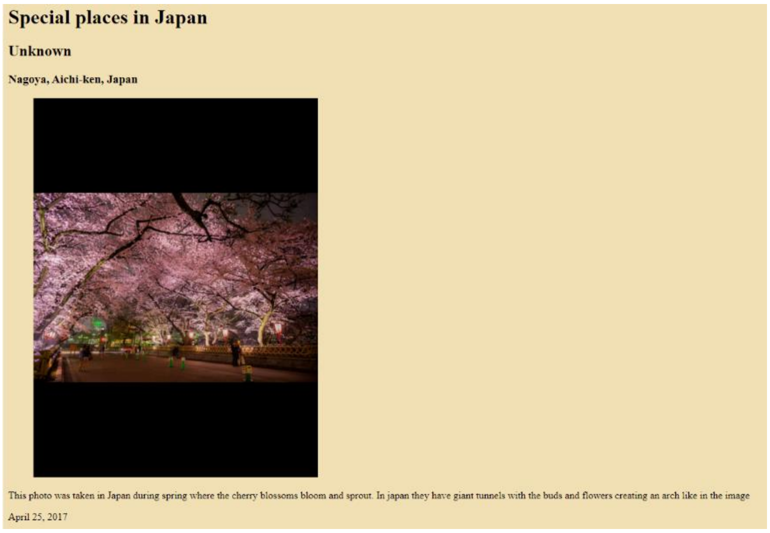
```
<script>
    const bashoformat = {
    props: ["name", "location"],
    template:
        `<div>
        <h2> {{name}} </h2>
        <h3> {{location}}</h3>
        </div>`
    };

    const imagespot = {
    props: ["imageURL", "altText"],
    template: `
    <div>
        <figure>
        <img v-bind:src="imageURL" :alt="altText" />
        </figure>
    </div>`};

    const description = {
    props: ["bio", "dateVisited"],
    template:
        `<div>
        <p> {{bio}} </p>
        <p> {{dateVisited}}</p>
        </div>`
    };
```

Here we have code which implements prop data and uses it within a template to render HTML code structure.



```
    Vue.component('location-two', {
    template: `
        <div>
        <place-name name="Kenrokuen"
        location="Kanazawa, Ishikawaken"/>
    <picture-spot imageURL="ken.png" altText="The 3rd
    <place-bio bio="Taken while walking through the pa
    components: {
        'place-name' : bashoformat,
        'picture-spot' : imagespot,
        'place-bio' : description
    }
    });

    Vue.component('location-three', {
    template: `
    <div><place-name name="Matsumoto mountain" locatio
    <picture-spot imageURL="matsumoto.jpg" altText="Mo
    <place-bio bio="While hiking the second highest pe
    components: {
        'place-name' : bashoformat,
        'picture-spot' : imagespot,
        'place-bio' : description
    }
    });

    Vue.component('end-summary-part', {
    template: `<div><h3>These are some of my favourite
    });

    const app = new Vue({
    el: "#nihonreview",
    template:
    `<div id = "JapanReview">
        <title-part />
        <location-one />
        <location-two />
        <location-three />
        <end-summary-part />
    </div>`
    });
</script>
```

And here we can see the tree structure in action as it works going down the top will be displayed first and the bottom last.



Special places in Japan

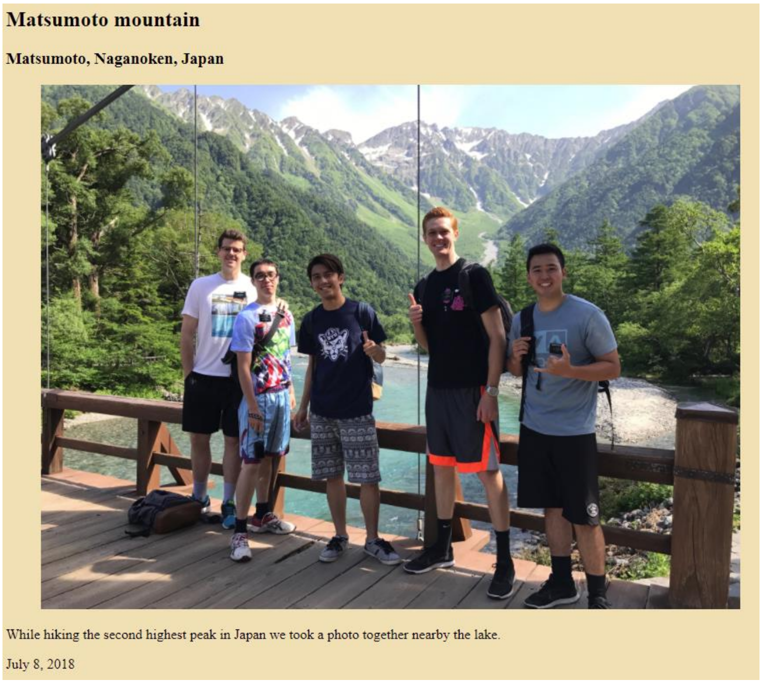
Unknown

Nagoya, Aichi-ken, Japan

This photo was taken in Japan during spring where the cherry blossoms bloom and sprout. In japan they have giant tunnels with the buds and flowers creating an arch like in the image

April 25, 2017

Here is the first template that is rendered by the component.



Matsumoto mountain

Matsumoto, Naganoken, Japan

While hiking the second highest peak in Japan we took a photo together nearby the lake.

July 8, 2018

Here is the third spot.

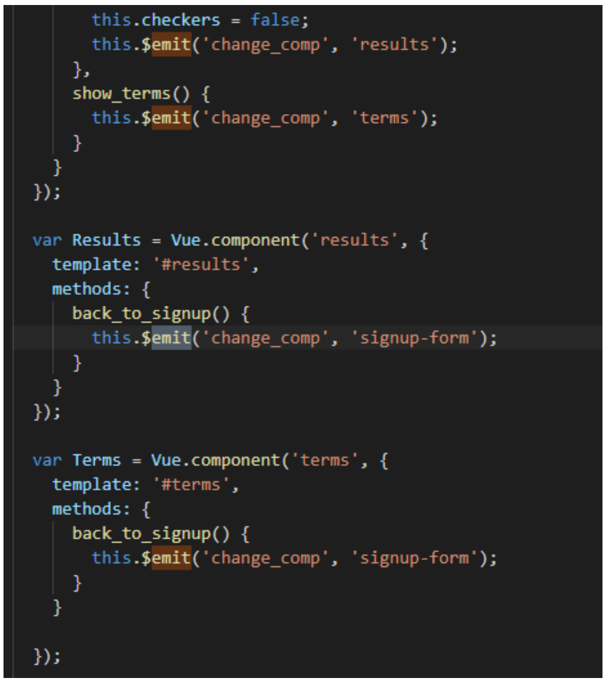
As we can see we have props that are being manipulated and used multiple times to display different information and render different elements and formats to the DOM.

Task 3: Custom Events

Custom events are a way to allow component and two-way communication within a website. Custom components allow us to create various and customizable components that can react based on listening to certain or various events.

In order to create custom events, we use the $emit and append it on our objects. We can use Inline emit options API emit and compositional emits.

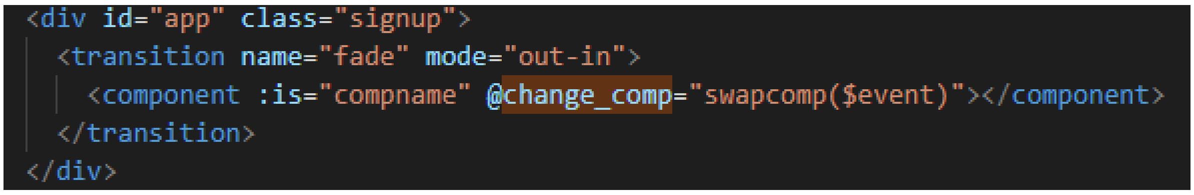
In my project I am using emit to render the registration data on the screen



```
      this.checkers = false;
      this.$emit('change_comp', 'results');
    },
    show_terms() {
      this.$emit('change_comp', 'terms');
    }
  }
});

var Results = Vue.component('results', {
  template: '#results',
  methods: {
    back_to_signup() {
      this.$emit('change_comp', 'signup-form');
    }
  }
});

var Terms = Vue.component('terms', {
  template: '#terms',
  methods: {
    back_to_signup() {
      this.$emit('change_comp', 'signup-form');
    }
  }

});
```

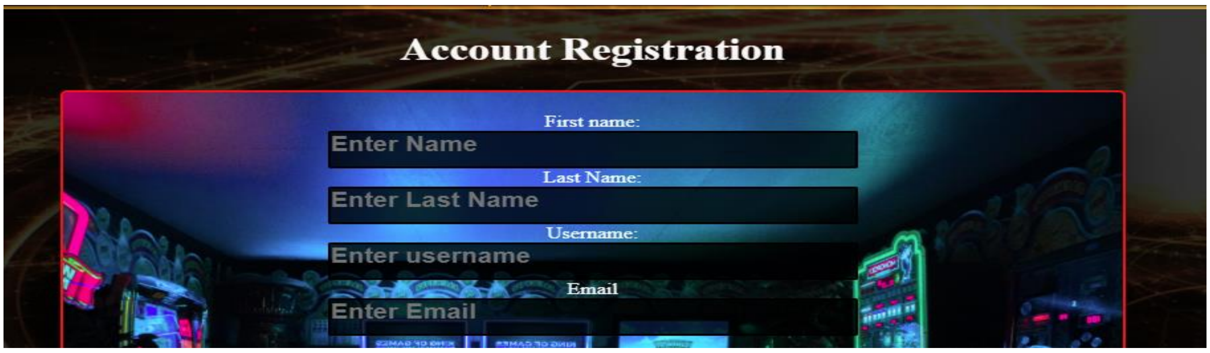
Here we are rendering the signup form for registration when we change states within the webpage in Vue the 'change_comp' and 'signup-form' are also called in the HTML document to register that specific button.

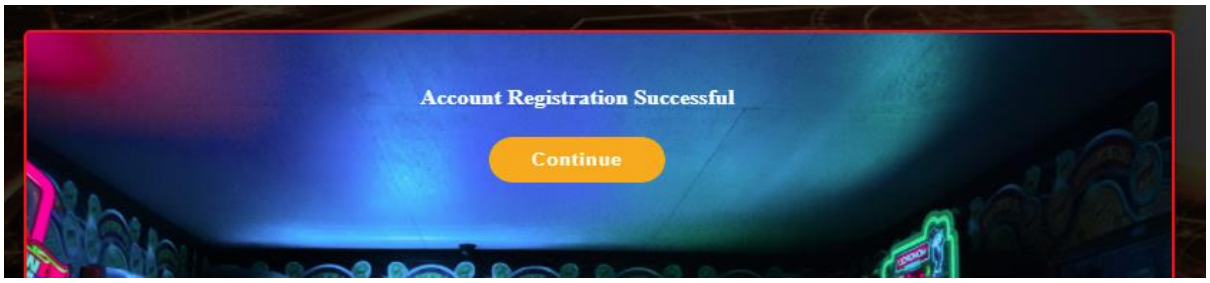
HTML section



```
<div id="app" class="signup">
  <transition name="fade" mode="out-in">
    <component :is="compname" @change_comp="swapcomp($event)"></component>
  </transition>
</div>
```

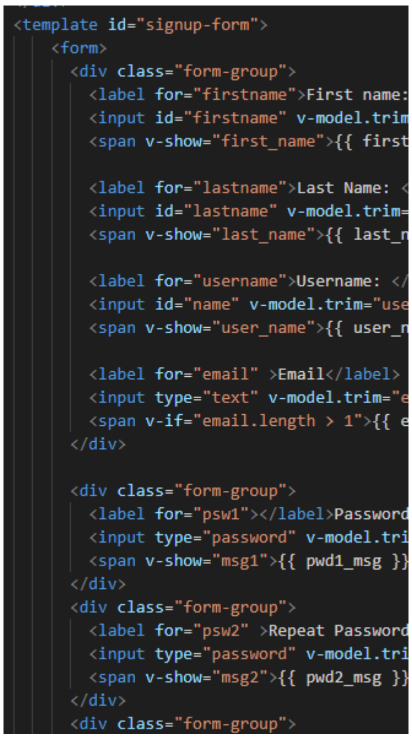
In this snippet we call @change and call the event we wish to swap between. Custom events allow us to handle specific and reactive event handling this allows complex and detailed customization within Vue and is a powerful and efficient tool. Below illustrates the custom event transition between the two.





Task 4: Vue slots

Vue slots are a way to identify sections where Vue can place any content within a components body. We can also call these slots and override them to allow different elements inside.



```
<template id="signup-form">
    <form>
      <div class="form-group">
        <label for="firstname">First name:
        <input id="firstname" v-model.trim
        <span v-show="first_name">{{ first

        <label for="lastname">Last Name: <
        <input id="lastname" v-model.trim=
        <span v-show="last_name">{{ last_n

        <label for="username">Username: </
        <input id="name" v-model.trim="use
        <span v-show="user_name">{{ user_n

        <label for="email" >Email</label>
        <input type="text" v-model.trim="e
        <span v-if="email.length > 1">{{ e
      </div>

      <div class="form-group">
        <label for="psw1"></label>Password
        <input type="password" v-model.tri
        <span v-show="msg1">{{ pwd1_msg }}
      </div>
      <div class="form-group">
        <label for="psw2" >Repeat Password
        <input type="password" v-model.tri
        <span v-show="msg2">{{ pwd2_msg }}
      </div>
      <div class="form-group">
```

We can name our slots with various names as well as setting inline slots to change things.

Reflection

This week's task was pretty nice since I've implemented these concepts already within my website project. Each topic discussed within this week has been interesting and helpful in designing my website using Vue concepts. The most beneficial aspect are custom events, dynamic components and slots and have been used extensively within my project to render different elements and have a responsive website.

Summary

In summary this portfolio covers from week 1 to week 7 detailing the process of learning and developing multiple components and concepts of Vue and responsive website applications. This has been a tough and difficult unit learning multiple coding languages such as CSS, HTML, Vue, JavaScript and backend implementation such as JSON and databases. Overall, I found this unit informative and interesting on how web applications and backend communication works with website and the internet. Furthermore, this unit in lockdown has not offered the opportunity for me to learn by experience and through others as I did not have the opportunity to attend the lectures. Thanks for reviewing my work, it's a bit long but I hope it allows me to achieve a reasonable grade.