Week 3 Task 3

Task 1

JavaScript 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.

Task 2

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

Using Arrays, we can put anything into an array and have the contents printed out simple process to populate an array.

Task 4

Get and set methods for date and time using the get methods when the button is clicked It will say the current time with the calendar year and exact time.

Task 5

Computed properties

```
<div id="example">
  <p>Original message: "{{ message }}"</p>
  <p>Computed reversed message: "{{ reversedMessage }}"</p>
</div>
var vm = new Vue({
  el: '#example',
  data: {
    message: 'Hello'
  },
  computed: {
    // a computed getter
    reversedMessage: function () {
      // `this` points to the vm instance
      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.

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="static"
  v-bind:class="{ active: isActive, 'text-danger': hasError }"
></div>
```

And the following data:

```
data: {
  isActive: true,
  hasError: false
}
```

It will render:

```
<div class="static 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>'
})
```

Then add some classes when using it:

```
<my-component class="baz boo"></my-component>
```

The rendered HTML will be:

```
<p class="foo bar baz boo">Hi</p>
```

The same is true for class bindings:

```
<my-component v-bind:class="{ active: isActive }"></my-component>
```

When `isActive` is truthy, the rendered HTML will be:

```
<p class="foo bar 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="styleObject"></div>
data: {
  styleObject: {
    color: 'red',
    fontSize: '13px'
  }
}
```

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.

```
<div v-if="type === 'A'">
  A
</div>
<div v-else-if="type === 'B'">
  B
</div>
<div v-else-if="type === 'C'">
  C
</div>
<div v-else>
  Not A/B/C
</div>
```

You can also use `v-for` to iterate through the properties of an object.

```
<ul id="v-for-object" class="demo">
  <li v-for="value in object">
    {{ value }}
  </li>
</ul>
new Vue({
  el: '#v-for-object',
  data: {
    object: {
      title: 'How to do lists in Vue',
      author: 'Jane Doe',
      publishedAt: '2016-04-10'
    }
  }
})
```

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

Event handling

Event handling is when the Vue method is set to listen to DOM events and will run functions and code when triggered.

```
<div id="example-1">
  <button v-on:click="counter += 1">Add 1</button>
  <p>The button above has been clicked {{ counter }} times.</p>
</div>
var example1 = new Vue({
  el: '#example-1',
  data: {
    counter: 0
  }
})
```

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.

`v-model` internally uses different properties and emits different events for different input elements:

- text and textarea elements use `value` property and `input` event;
- checkboxes and radiobuttons use `checked` property and `change` event;
- select fields use `value` as a prop and `change` as an event.

The Vue website dictates the usage of automatic input bindings with Vue functions and html code.

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

```
<blog-post
  v-for="post in posts"
  v-bind:key="post.id"
  v-bind:title="post.title"
  v-bind:content="post.content"
  v-bind:publishedAt="post.publishedAt"
  v-bind:comments="post.comments"
></blog-post>
```

`This is a bad example`

```
<blog-post
  v-for="post in posts"
  v-bind:key="post.id"
  v-bind:post="post"
></blog-post>
Vue.component('blog-post', {
  props: ['post'],
  template: `
    <div class="blog-post">
      <h3>{{ post.title }}</h3>
      <div v-html="post.content"></div>
    </div>
  `
})
```

This is a good example it shows how Vue can be used to section html code within Vue components to be used within HTML interchangeably

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.