MODULE 4

# Event Handling in Vue.js

# Event Design Pattern (Named)

```javascript
let changeButton = document.getElementById('change-greeting');
```

1. A DOM element where we want to listen for events

```javascript
changeButton.addEventListener('click', (event) => {
  changeGreeting();
});
```

2. A specific event that we want to listen to

```javascript
function changeGreeting() {
  let greetingHeader = document.getElementById('greeting');
  greetingHeader.innerText = 'Goodbye';
}
```

3. A function that holds the logic that we want to execute

# Event Design Pattern (Anonymous)

- the name of the event
- the type of data structure used to represent key properties of the event
- the object that will 'emit' or 'publish' the event

1. A DOM element where we want to listen for events
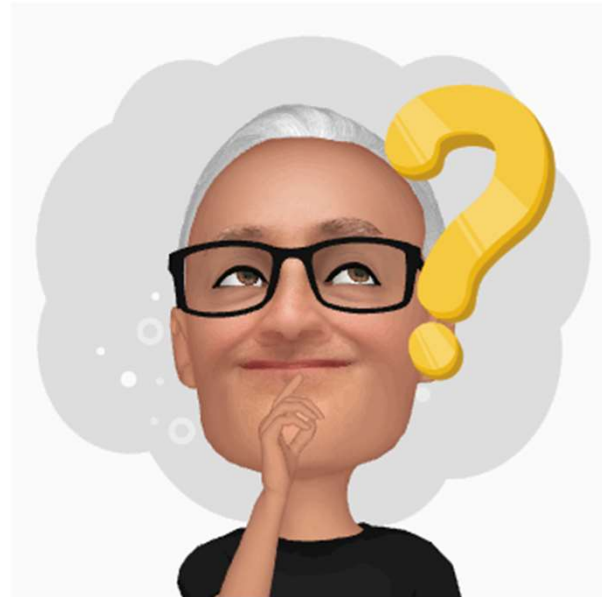
```
let p = document.getElementsByTagName('p')[0];

p.addEventListener("click", function(event) {
    this.innerHTML = "Paragraph Clicked!";
});
```

3. A function that holds the logic that we want to execute

2. A specific event that we want to listen to

# Wayback Machine

- How did we handle actions in C#/Java?

# Adding Methods to Components

```
export default {
    name: 'order-form',
    methods: {
        displayFormData() {
            // Method logic goes here
        },
        anotherMethod() {
            // Method logic goes here
        }
    }
};
```

# Adding Methods to Components

```
export default {
    name: 'order-form',
    methods: {
        displayFormData() {
            // Method logic goes here
        },
        anotherMethod() {
            // Method logic goes here
        }
    }
};
```

```
<a v-on:click="displayFormData">Show Form</a>


<input type="text" name="firstName" v-on:change="anotherMethod" />
```

# Special Data

- <textarea **v-on:keyup.enter="processEnterKey"**></textarea>
  - .enter
  - .tab
  - .delete (captures both "Delete" and "Backspace" keys)
  - .esc
  - .space
  - .up
  - .down
  - .left
  - .right
- <input type="text" v-on:keyup.74="handleJ" />

# Mouse Events

- .left
- .right
- .middle
- &lt;li v-on:click.left="updateTotal" **v-on:click.right="activateContextMenu"**&gt;Add to Cart&lt;/li&gt;
- &lt;li v-on:dblclick.left="doubleTotal" **v-on:dblclick.right="doubleContextMenu"**&gt;Add to Cart&lt;/li&gt;

# Event Arguments

- Inline Event Handlers
  - `<a href="#" id="increase" class="btn" v-on:click="counter += 1">`
  - Increase
  - `</a>`

- Passing Arguments
  - `<a href="#" id="increase" class="btn" v-on:click="updateCounter(1)">`
  - Increase
  - `</a>`

- v-on shorthand is @
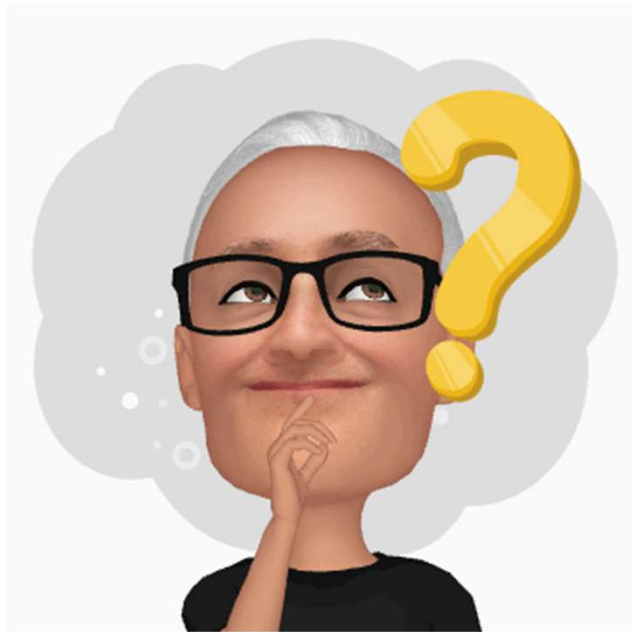  - `<a href="#" id="increase" class="btn" @click="counter += 1">`

# Event Object

- We still have access to the event object

- Implicitly
  - `<a href="#" id="increase" class="btn" v-on:click="updateCounter">`
- Explicitly
  - `<a href="#" id="increase" class="btn" v-on:click="updateCounter($event)">`

```
methods: {
  updateCounter(event) {
    this.counter += event.target.id === "increase" ? 1 : -1;
  }
}
```

# LET'S CODE!

# WHAT QUESTIONS DO YOU HAVE?

Reading for tonight:
**Vue.js Testing**