MODULE 3

Event Handling in Vue.js

Event Design Pattern (Named)

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

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


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

1. A DOM element where we want to listen for events
2. A specific event that we want to listen to
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

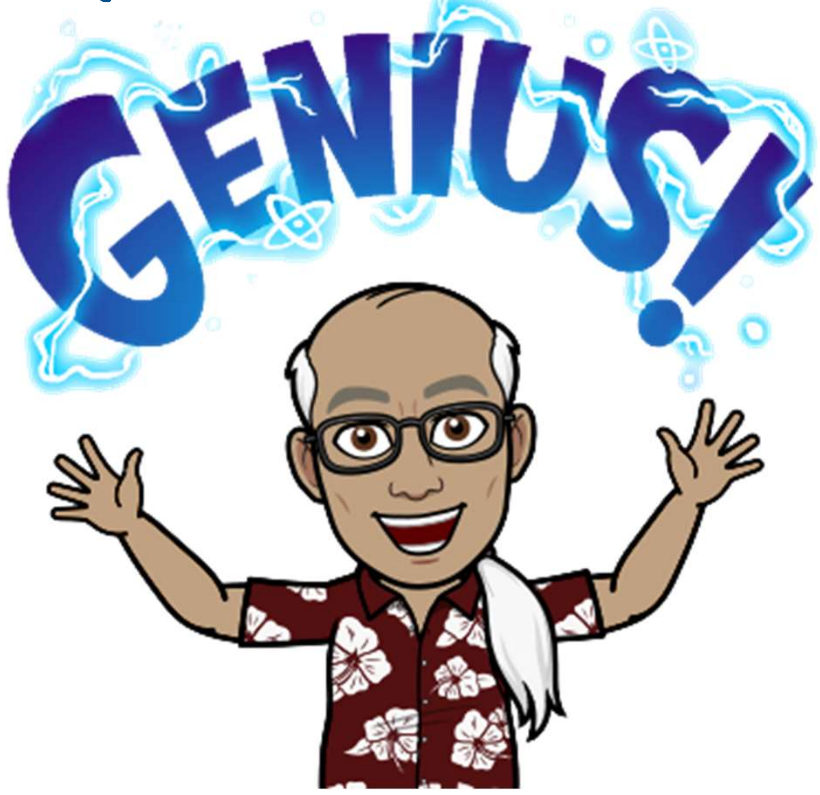
```
let p = document.getElementsByTagName('p')[0];

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

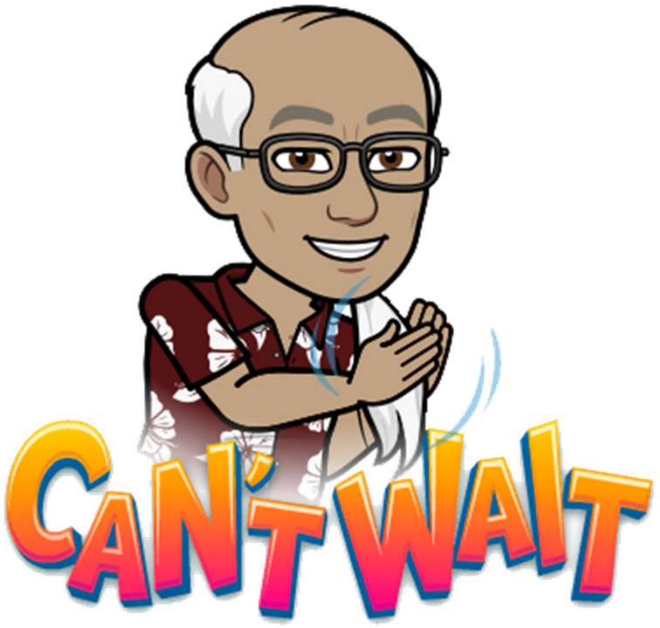
```
export default {
  name: 'order-form',
  methods: {
    displayFormData() {
      // Method logic goes here
    },
    anotherMethod() {
      // Method logic goes here
    }
  }
};
```

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" />~~
- <input type="text" v-on:keyup.shift.j="handleJ" />

Mouse Events

- .left
- .right
- .middle
- <li v-on:click.left="updateTotal" **v-on:click.right="activateContextMenu"**>Add to Cart</li>
- <li v-on:dblclick.left=“doubleTotal" **v-on:dblclick.right=“doubleContextMenu"**>Add to Cart</li>

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 Component Communication