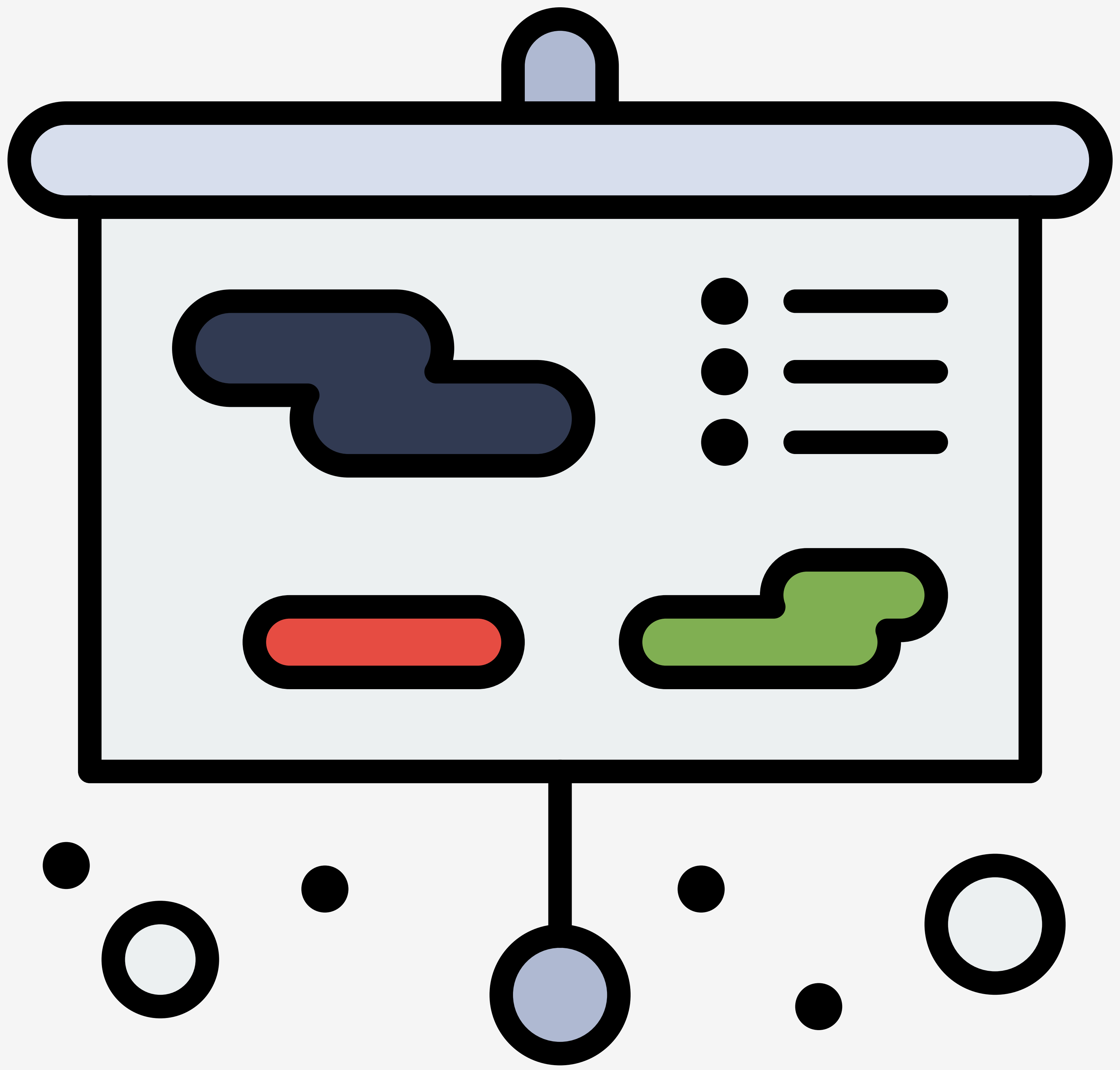
WEB PRODUCTION

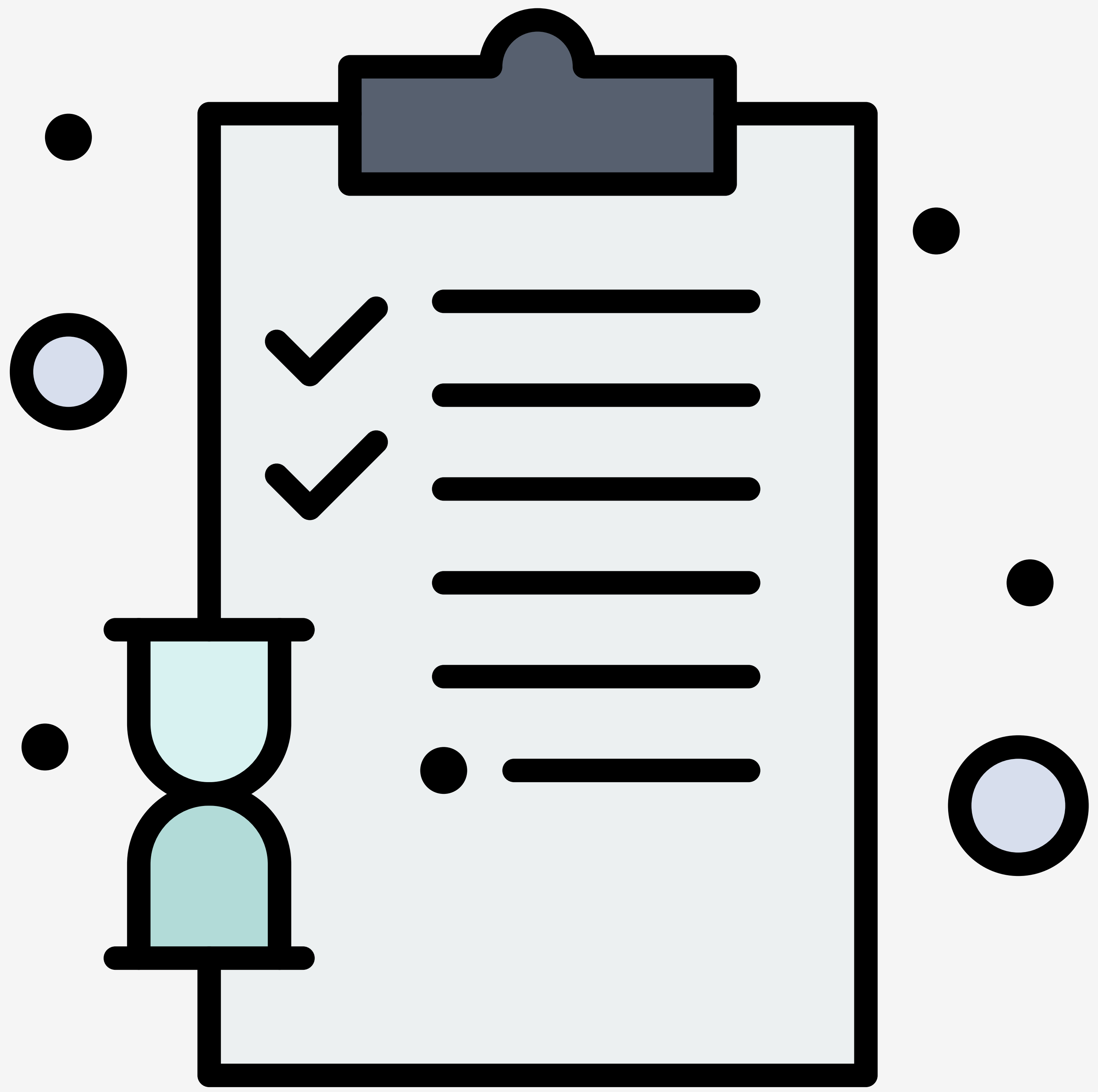
Lecture 5

TODAY'S TOPICS



- Managing Classes
- Transitions
- localStorage
- Project: Event Calendar

ANNOUNCEMENTS

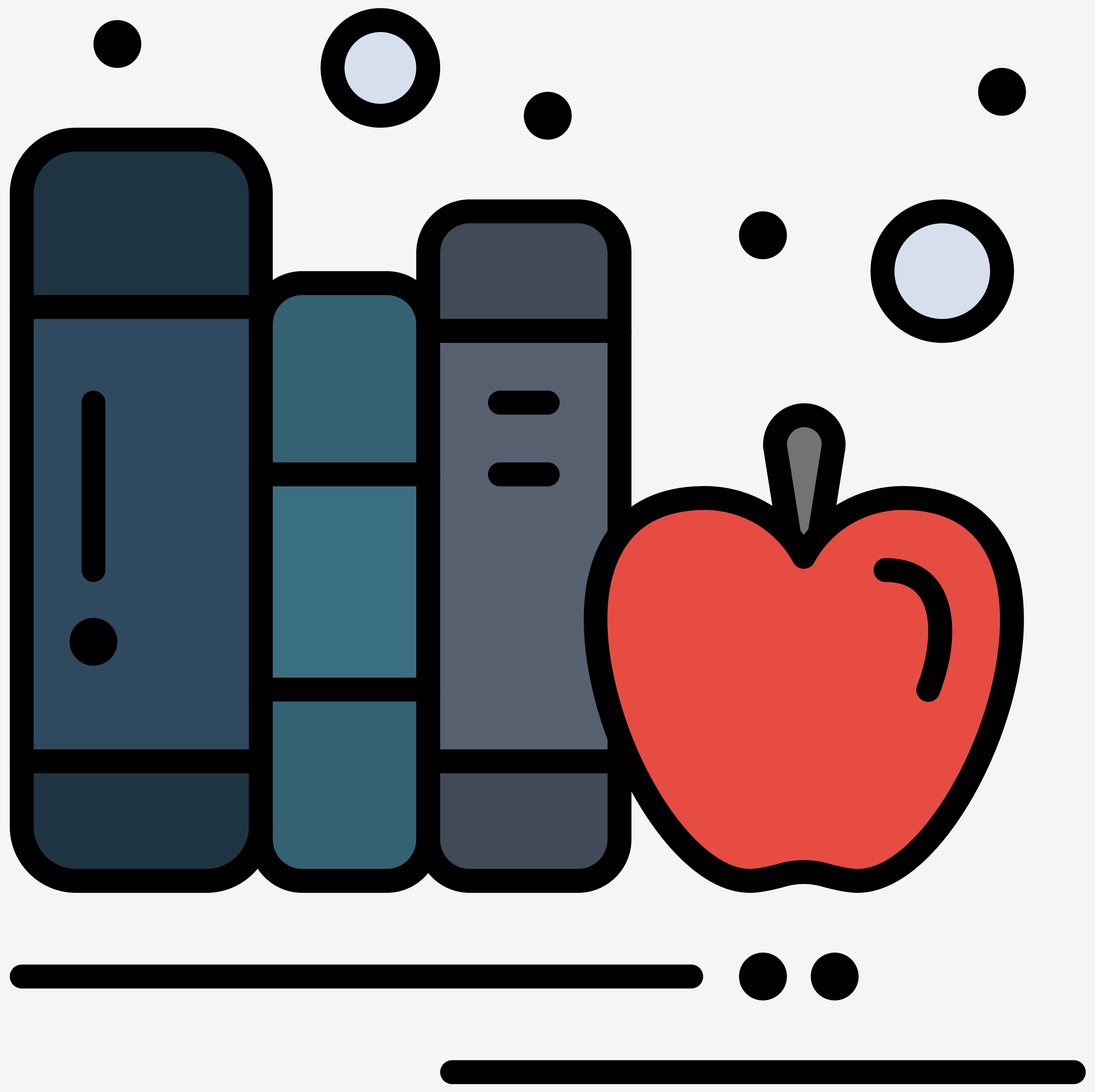


- Sign-in Sheet
- Recording
- Schedule Changes

QUESTIONS

MANAGING CLASSES

BINDING CLASS



- It is possible to add the `v-bind` on the class attribute
- This allows for classes to be dynamically added to an element
- It possible to have a class attribute and a `v-bind:class` attribute on the same element

V-BIND CLASS

```html
<div id="app">
  <button class="btn" :class="button">
    Button
  </button>
</div>
```

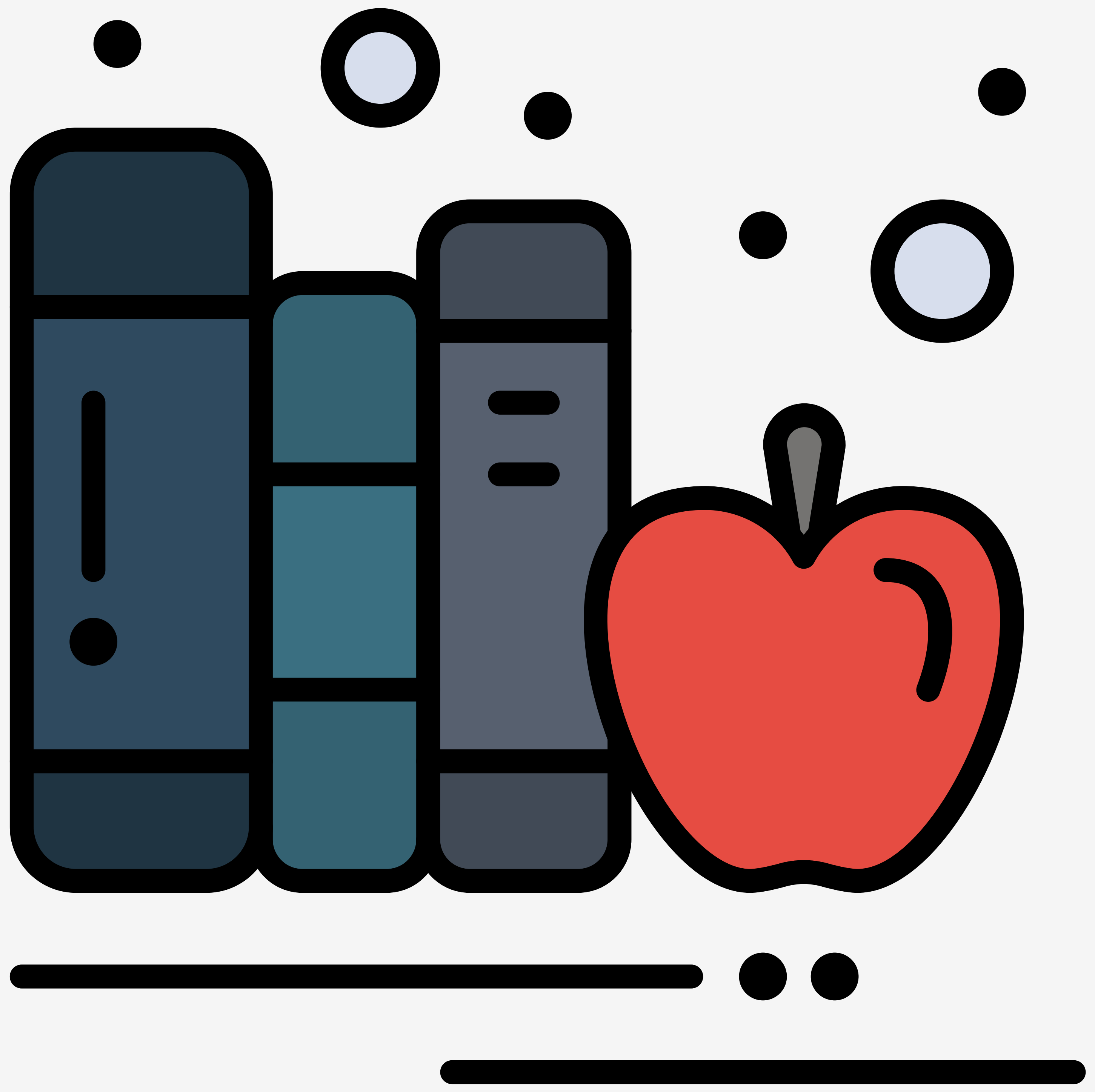
```js
new Vue({
  el: '#app',
  data: {
    button: 'btn-primary'
  }
})
```

V-BIND CLASS

```
<div id="app">
  <button class="btn" :class="'btn-' + button">
    Button
  </button>
</div>
```

```
new Vue({
  el: '#app',
  data: {
    button: 'primary'
  }
})
```

OBJECT SYNTAX



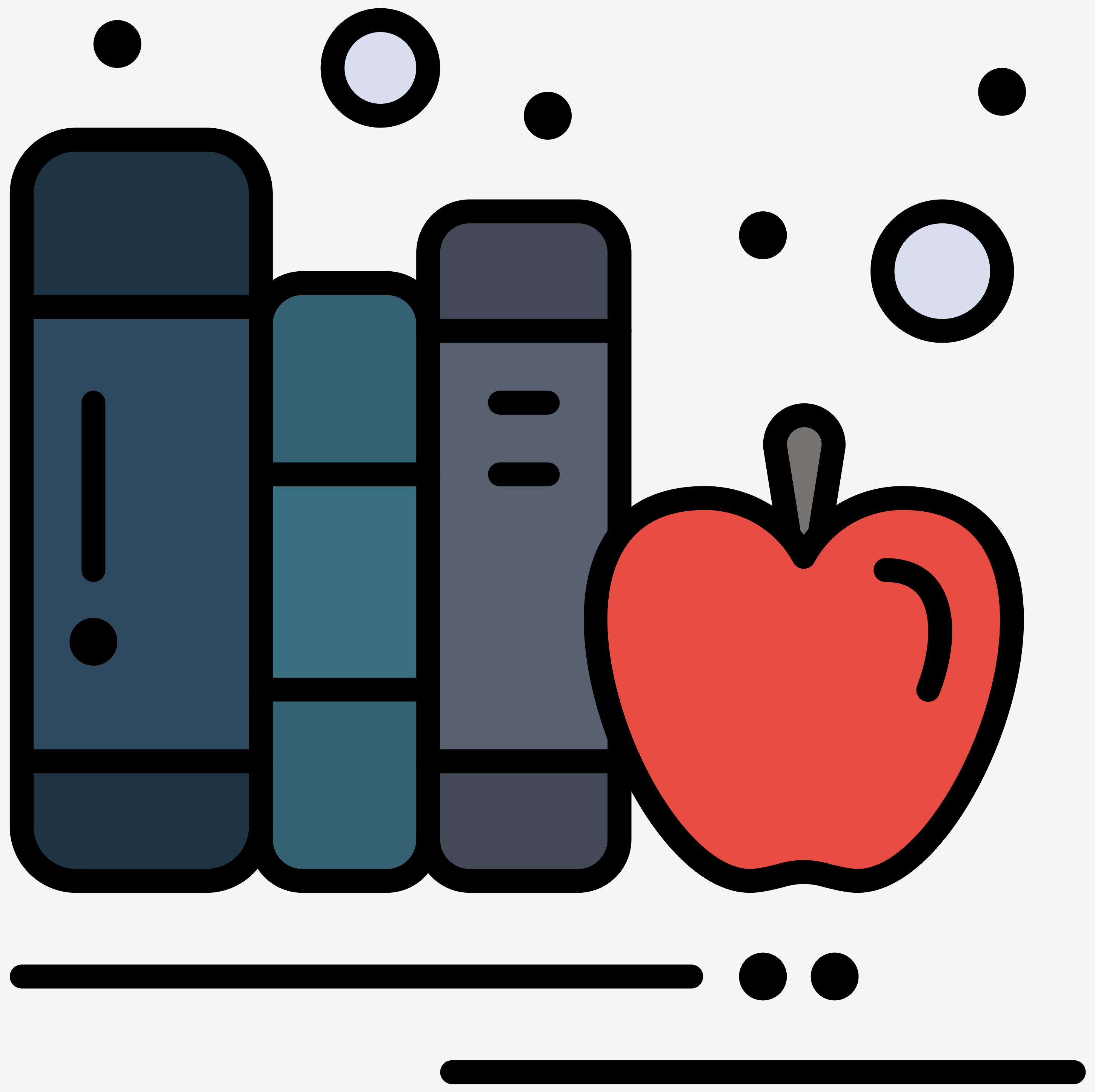
- When using the `v-bind` directive, it is possible to use an object syntax
- This allows classes to be added based on the truthiness of the provided expression
- The key will be the class
- The value will be the expression
- If the expression is truthy, then the class will be added

OBJECT SYNTAX

```html
<div id="app">
  <button class="btn"
    :class="{ 'btn-primary': isPrimary }">
    Button
  </button>
</div>
```

```js
new Vue({
  el: '#app',
  data: {
    isPrimary: true
  }
})
```

ARRAY SYNTAX



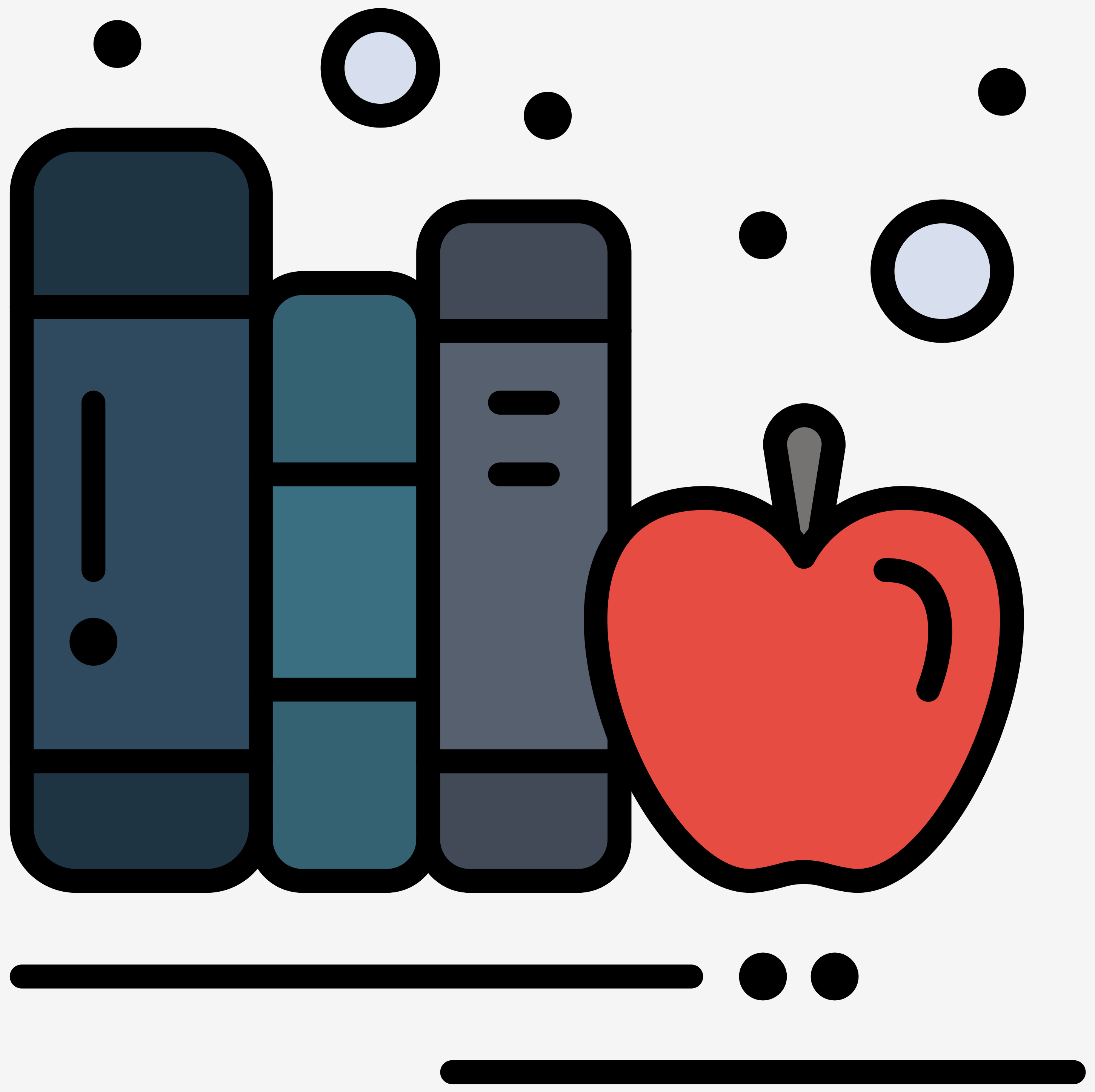
- When using the `v-bind` directive, it is possible to use an array syntax
- This allows multiple classes to be added dynamically
- Each array item will be an expression which will evaluate to be a class name

ARRAY SYNTAX

```
<div id="app">
  <button class="btn"
    :class="[ buttonType, buttonSize ]">
    Button
  </button>
</div>
```

```
new Vue({
  el: '#app',
  data: {
    buttonType: 'btn-primary',
    buttonSize: 'btn-lg'
  }
})
```

TOGGLING CLASSES



- Using a combination of the `v-bind` and the `v-on` directives it is possible to toggle between classes
- A computed property can be used to programatically choose which class to display
- The ternary operator can be used as a single line conditional statement

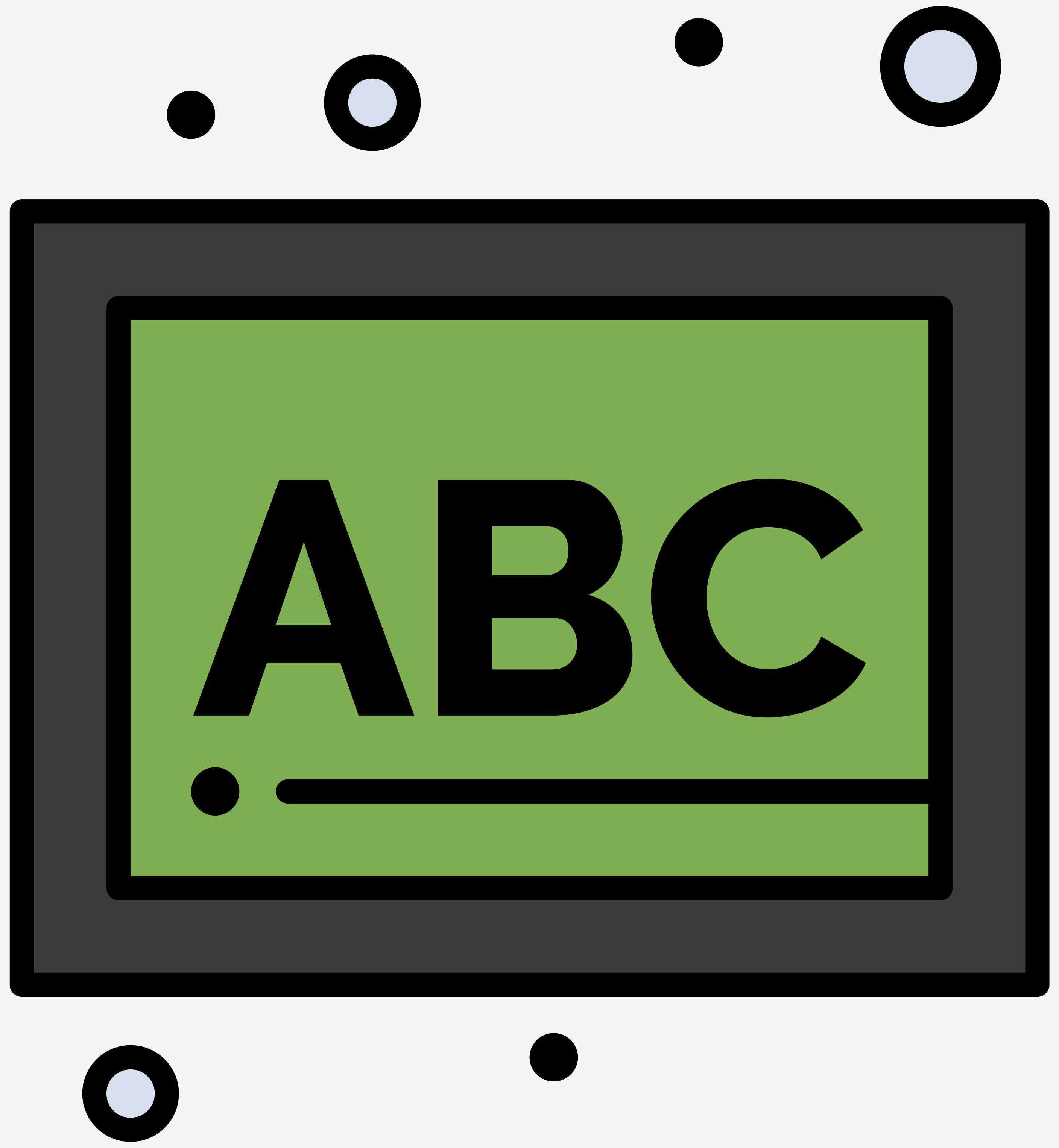
TOGGLING CLASSES

```
<div id="app">
  <button class="btn"
    :class="buttonClass"
    @click="outline = !outline">
    Button
  </button>
</div>
```

```
new Vue({
  el: '#app',
  data: {
    outline: true
  },
  computed: {
    buttonClass: function () {
      return this.outline ? 'btn-outline-primary' : 'btn-primary'
    }
  }
})
```

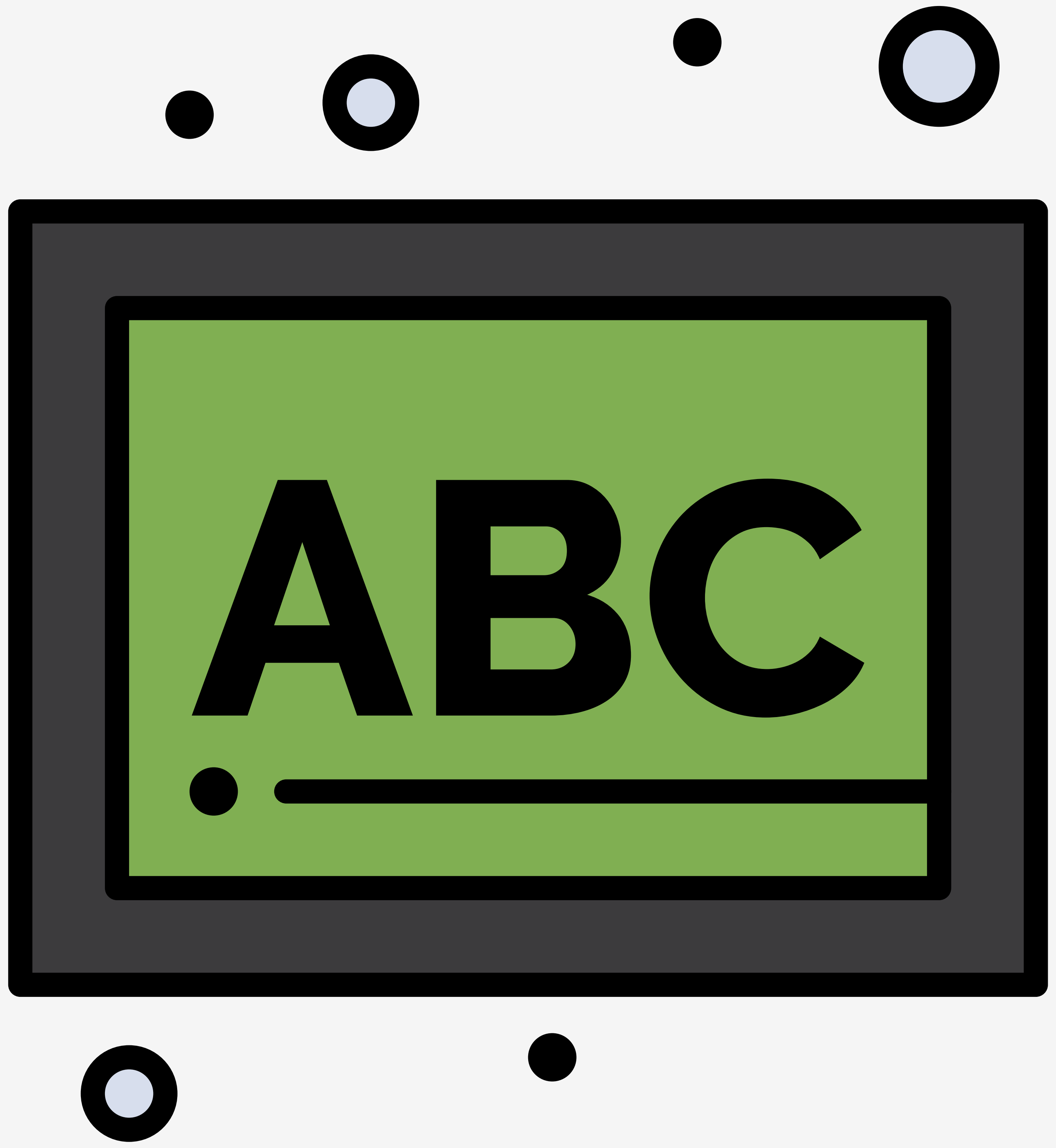
TRANSITIONS

TRANSITIONS

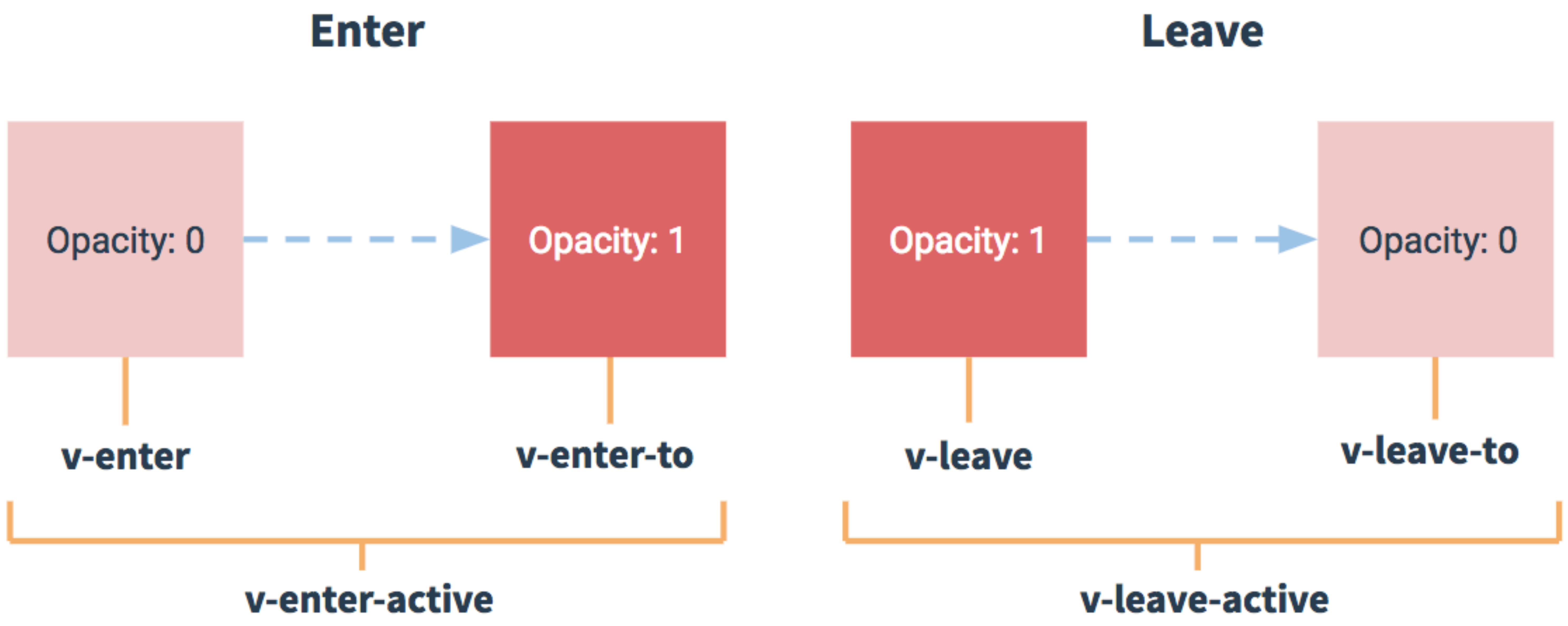


- The `transition` wrapper component allows adding entering / leaving transitions
- The `transition` wrapper can be used on any element with the `v-if` or `v-show` directives or dynamic components
- The `name` attribute is used to provide a name to the transition and determine the classes to use

TRANSITIONS



- There are six classes that can be applied during entering and leaving transitions
  - `v-enter`
  - `v-enter-active`
  - `v-enter-to`
  - `v-leave`
  - `v-leave-active`
  - `v-leave-to`


Enter
Leave
Opacity: 0
Opacity: 1
Opacity: 1
Opacity: 0
v-enter
v-enter-to
v-leave
v-leave-to
v-enter-active
v-leave-active

TRANSITIONS

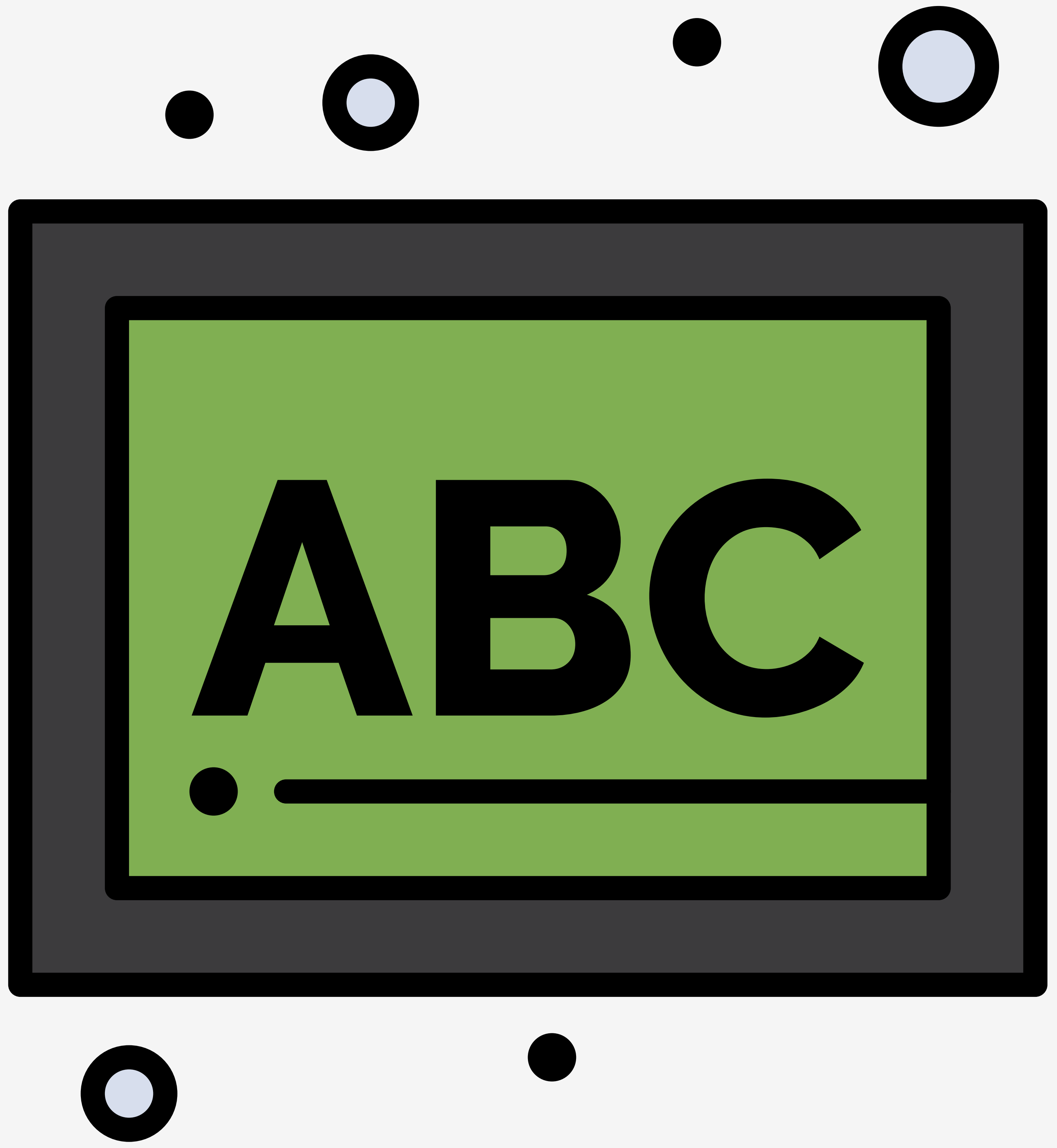
```html
<div id="app">
  <transition name="fade">
    <p v-show="show">hello</p>
  </transition>
</div>
```

```js
new Vue({
  el: '#app',
  data: {
    show: true
  }
})
```

```css
.fade-enter, .fade-leave-to {
  opacity: 0;
}

.fade-enter-active, .fade-leave-active {
  transition: 0.5s;
}
```

TRANSITIONS



- Custom transition classes can use with the following transition attributes:
  - `enter-class`
  - `enter-active-class`
  - `enter-to-class`
  - `leave-class`
  - `leave-active-class`
  - `leave-to-class`
- These attributes are handy when using animation libraries

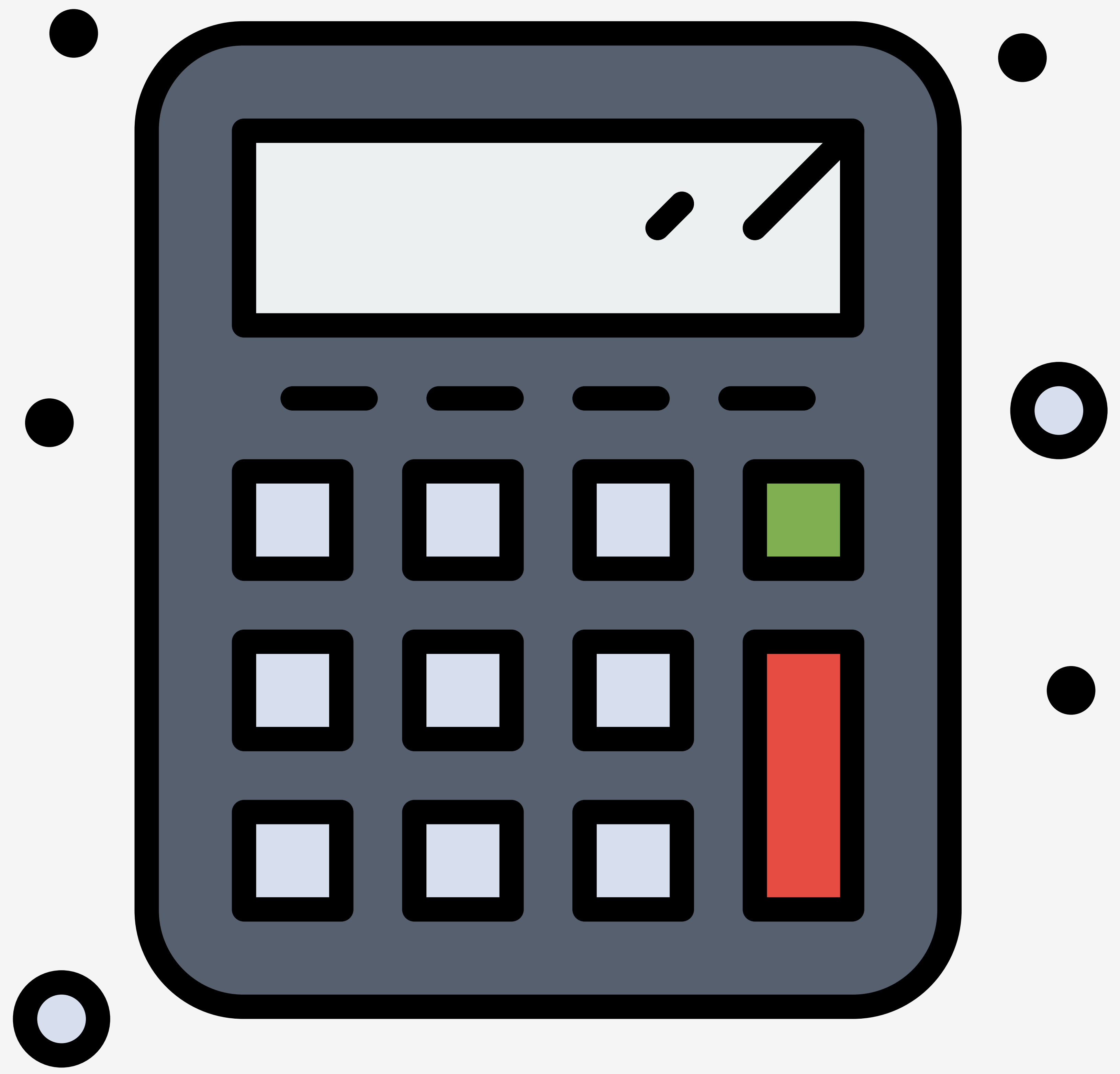
TRANSITIONS

```html
<!-- using animate.css -->
<div id="app">
  <transition  enter-active-class="animated tada"
  leave-active-class="animated bounceOutRight">
    <p v-if="show">hello</p>
  </transition>
</div>
```

```js
new Vue({
  el: '#app',
  data: {
    show: true
  }
})
```

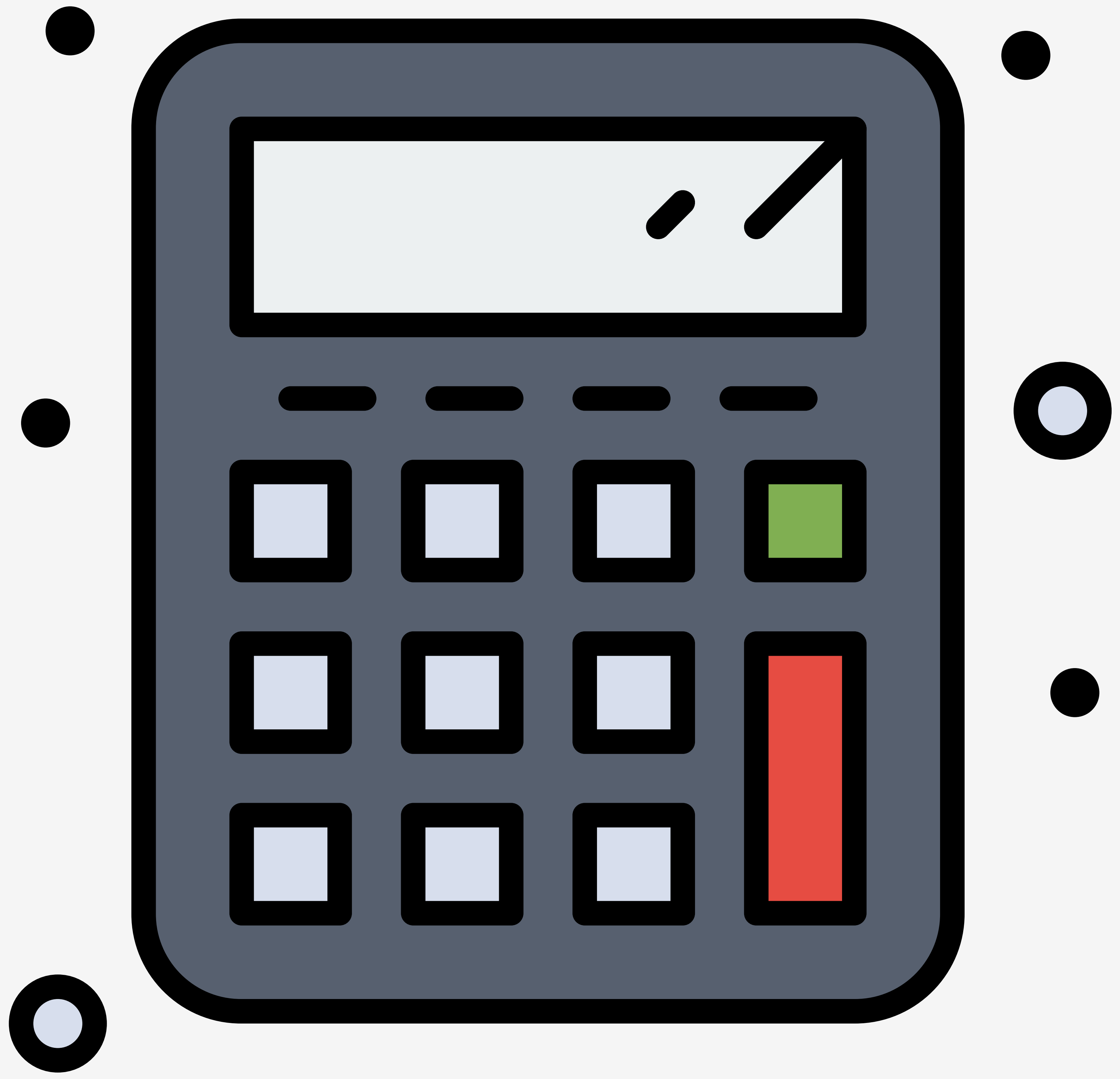
LOCALSTORAGE

LOCALSTORAGE



- Most web application require data to be persistent across pages
- One method is to use localStorage
- Data in localStorage is stored in the browser and remains until deleted by the user or the application
- Data is store in key / value pairs
- Both key and value must be a string

LOCALSTORAGE



- The `setItem()` function is used to add data to localStorage
- The `getItem()` function is used to retrieve data from localStorage
- The `JSON.stringify()` function can be used to store objects or array in localStorage
- The `JSON.parse()` function is used to convert a string back to an object or array

LOCALSTORAGE

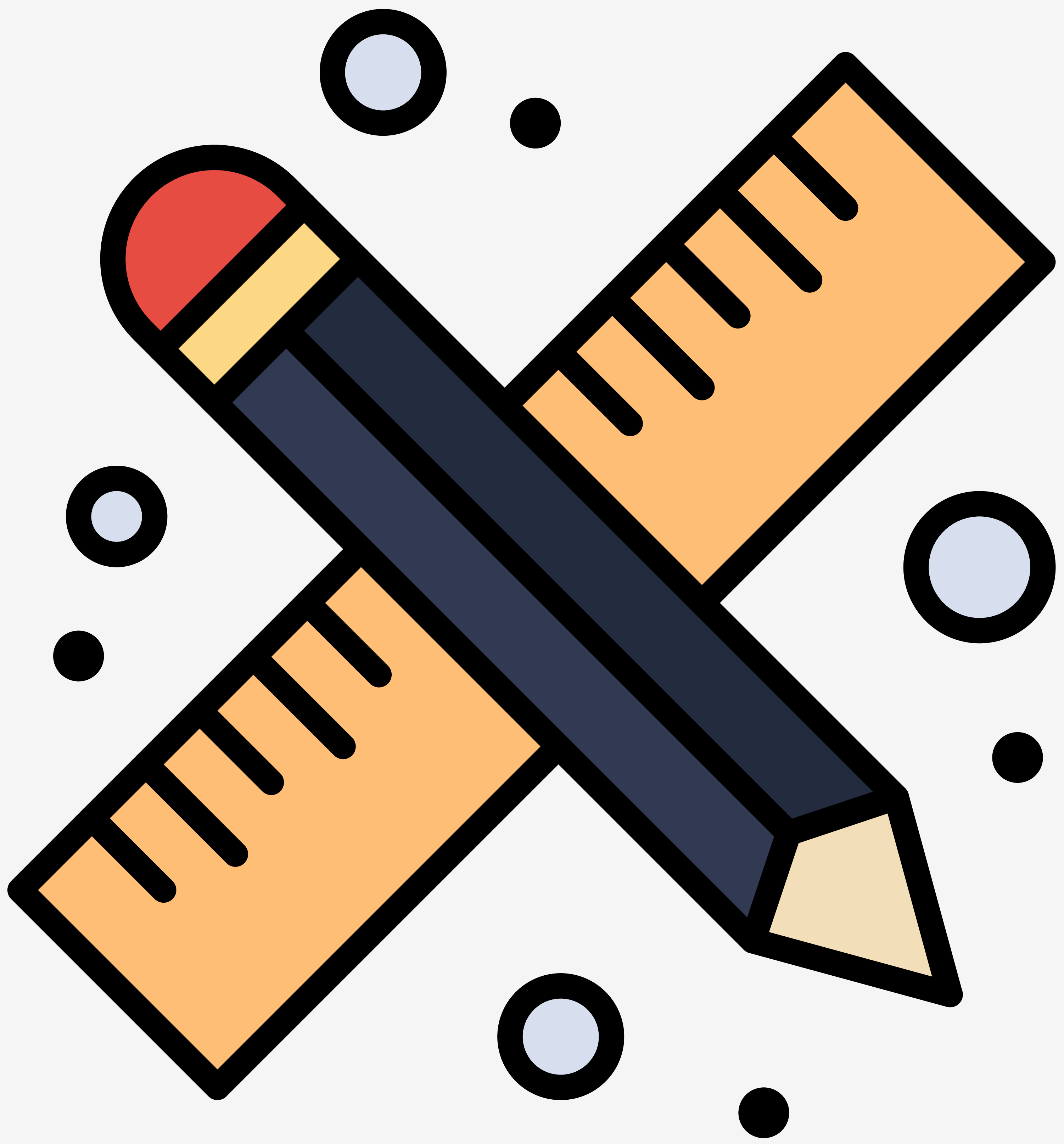
```
// storing a string
localStorage.setItem('name', 'Michael')
localStorage.getItem('name') // Michael

// storing an object
const string = JSON.stringify({
  name: 'Michael',
  title: 'Professor'
})
localStorage.setItem('data', string)

const data = localStorage.getItem('data')
data.name // Michael
```

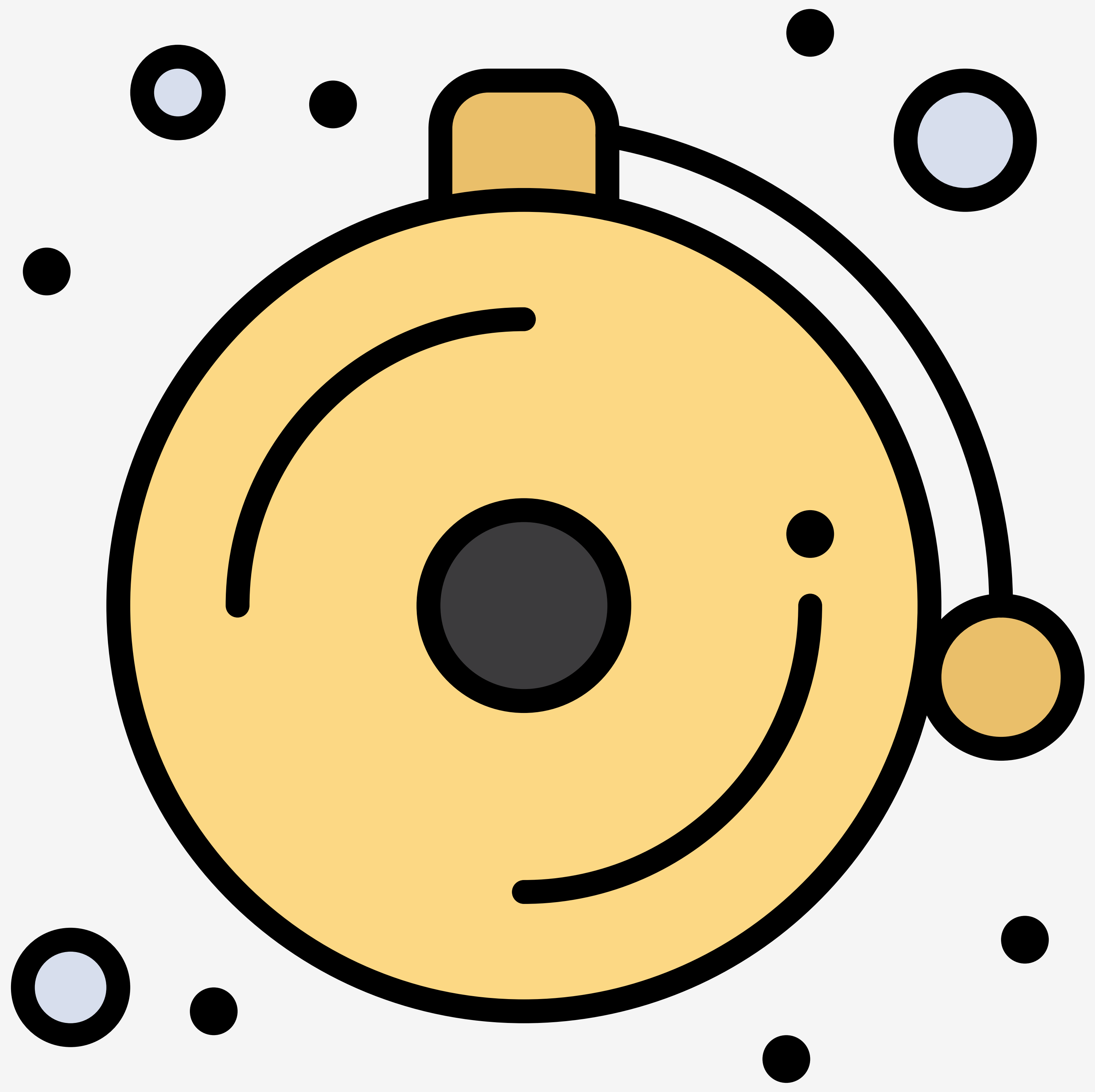
HANDS-ON

EVENT CALENDAR - MIDTERM



- *GITHUB CLASSROOM ASSIGNMENT*
- Create calendar web app
- Display a monthly calendar with user created events
- Allow user to change the month being displayed
- Allow user to add new events and edit or delete existing events
- Submit the URL to your repository
- *DUE:* **Mon. Mar 2 @ 11:59 PM**

NEXT TIME...



- Vue CLI