Vue Lifecycle Methods

Component Lifecycle

- **Created**
  - after its data has been made "reactive"
  - not yet part of DOM

- **Mounted**
  - inserted into the DOM of its parent or
    replaces the element referred to by its `el` property

- **Updated**
  - any of its props or data change

- **Destroyed**
  - removed from DOM

Lifecycle Methods

- Defined as top-level properties in component instance definitions

- Automatically called at specific points in their lifecycle

- Most commonly used are **mounted** and **updated**

  - correspond to React lifecycle methods
    componentDidMount and **componentDidUpdate**

beforeCreate & Created

- A component instance is consider to be created after its data has been made "reactive"

- At this point it is not yet part of DOM

- Template compilation into a `render` method occurs between `created` and `beforeMount`

beforeMount & mounted ...

- A component instance is mounted when
it is inserted into the DOM of its parent or
replaces the element referred to by its `el` property

- **beforeMount** can be use used to modify DOM
before component instance is rendered for first time

- **mounted** can be use used to modify DOM
after component instance is rendered for first time

  - the most frequently used lifecycle method

  - common place for making REST calls that get data needed by component

... beforeMount & mounted

- **mounted** can be called before all of its child components have been mounted

- To do something after all child components have been mounted call **this.$nextTick(fn)** inside **mounted**, passing it a function it will call

  - can also be used to execute code after batched DOM updates have completed

6

`beforeUpdate` & `updated`

- A component instance is updated when any of its props or data change

- These methods are sometimes used for debugging purposes

- **`updated`** can be used to modify DOM after a prop or data change

`beforeDestroy` & `destroyed`

- A component instance is destroyed when it is removed from DOM

- **`beforeDestroy`** is sometimes used to unsubscribe from events and data streams

activated & deactivated

- A component can be marked as "keep-alive"
  so its state is maintained after being removed from DOM

- Provides a performance benefit when
  the same component instance will be rendered again later

  - for example, components rendered in a tab interface

- To make a component "keep-alive", wrap in a `<keep-alive>` element

- After a keep-alive component is rendered for first time

  - when removed from DOM its `deactivated` lifecycle method is called

  - when added to DOM again its `activated` lifecycle method is called

Exercise

- Start dog app server
  - cd to top project directory
  - `npm run serve`
- Browse localhost:8080
- Add **mounted** method to **Dogs** component
- Set **dogs** array in component data
  to contain some predefined dogs
  - example: `this.dogs = [{name: 'Maisey'}, {name: 'Oscar'}];`
  - in real usage this might call a REST service to get dogs
- Verify that predefined dogs appear in UI