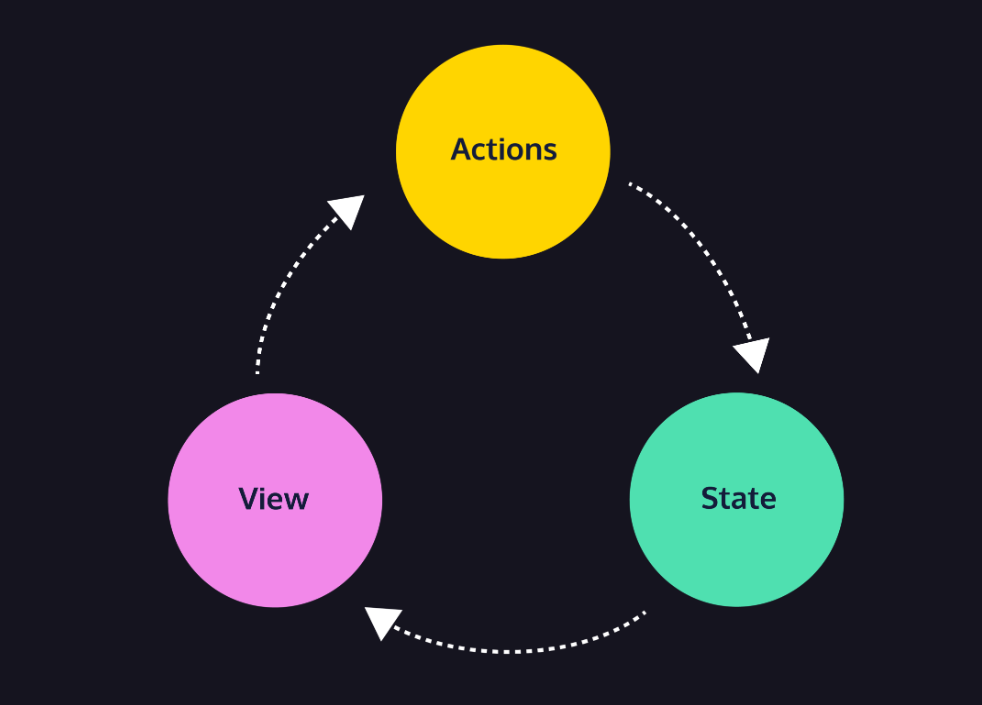
Redux

Redux is a state management library that follows a pattern known as the [Flux architecture](https://facebookarchive.github.io/flux/docs/in-depth-overview/). In Flux and Redux, shared information is consolidated within a single object instead of being scattered across individual components. Components receive data to render and can request changes using actions, which are events triggered by user interactions or other events. The state is available throughout the application, and updates are handled in a predictable manner, with components being notified whenever a change occurs.



### One-Way Data Flow

In most applications, there are three parts:

* **State** – the current data used in the app
* **View** – the user interface displayed to users
* **Actions** – [events](https://www.codecademy.com/resources/docs/javascript/events) that a user can take to change the state
* The state holds the current data used by the app’s components.
* The view components display that state data.
* When a user interacts with the view, like clicking a button, the state will be updated in some way.
* The view is updated to display the new state.

### State

State in a web application represents the current information that drives the application’s behavior and appearance. It acts as a centralized source of data, storing the essential details of the application at any given moment.

With Redux, the state can be any JavaScript type, including number, string, boolean, array, and object.

Here’s an example state for a to-do app:

const state = [ 'Print trail map', 'Pack snacks', 'Summit the mountain' ];

### Actions

Most well-designed applications will have separate components that need to communicate and share data.

A to-do list might have an input field where the user can type in a new to-do item. The application might transfer this data from the input field, add it to an array of all to-dos, and render them as text on the screen. This entire interaction can be defined as an **action**. Actions describe an event or an action that has occurred and provide information about what needs to be updated in the application’s state. In short, actions are how Redux manages and update the state.

In Redux, actions are represented as plain JS

Preview: Docs Loading link description

[objects](https://www.codecademy.com/resources/docs/javascript/objects)

. Here’s what that action might look like:

const action = {  
  type: 'todos/addTodo',  
  payload: 'Take selfies'  
};

* Every action must have a type property with a string value. This describes the action.
* Typically, an action has a payload property with an object value. This includes any information related to the action. In this case, the payload is the to-do text.
* When an action is generated and notifies other parts of the application, we say that the action is *dispatched*.

“Remove all to-dos”. This requires no payload because no additional information is needed:

* const action = {  
    type: 'todos/removeAll'  
  }
* “Remove the ‘Pack snacks’ to-do”:
* const action = {  
    type: 'todos/removeTodo',  
    payload: 'Pack snacks'  
  }