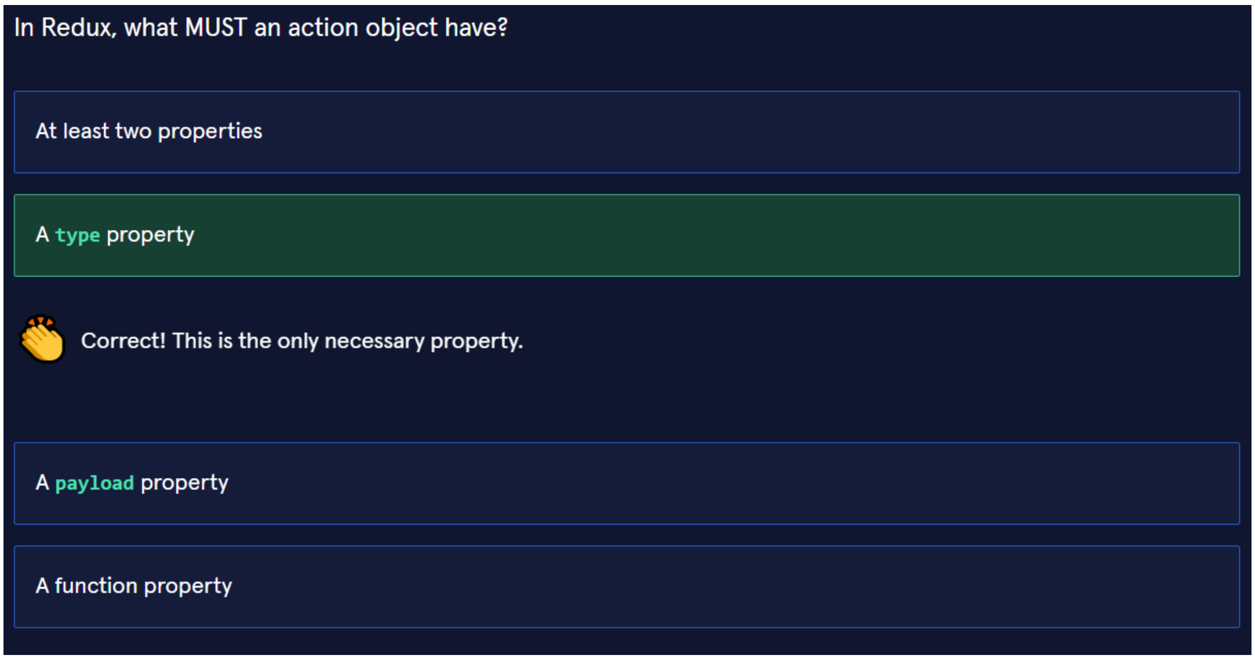
QUIZ


In Redux, what MUST an action object have?

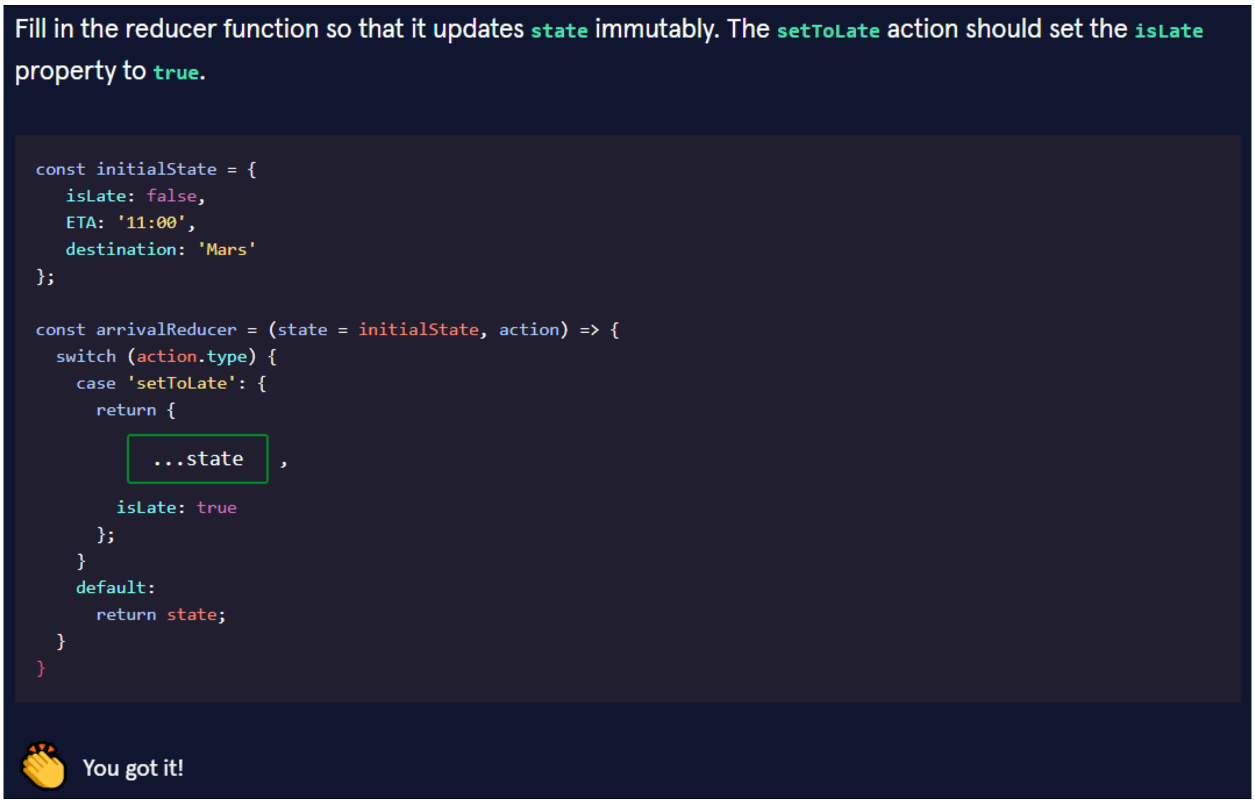
At least two properties

[✓ A `type` property]

Correct! This is the only necessary property.

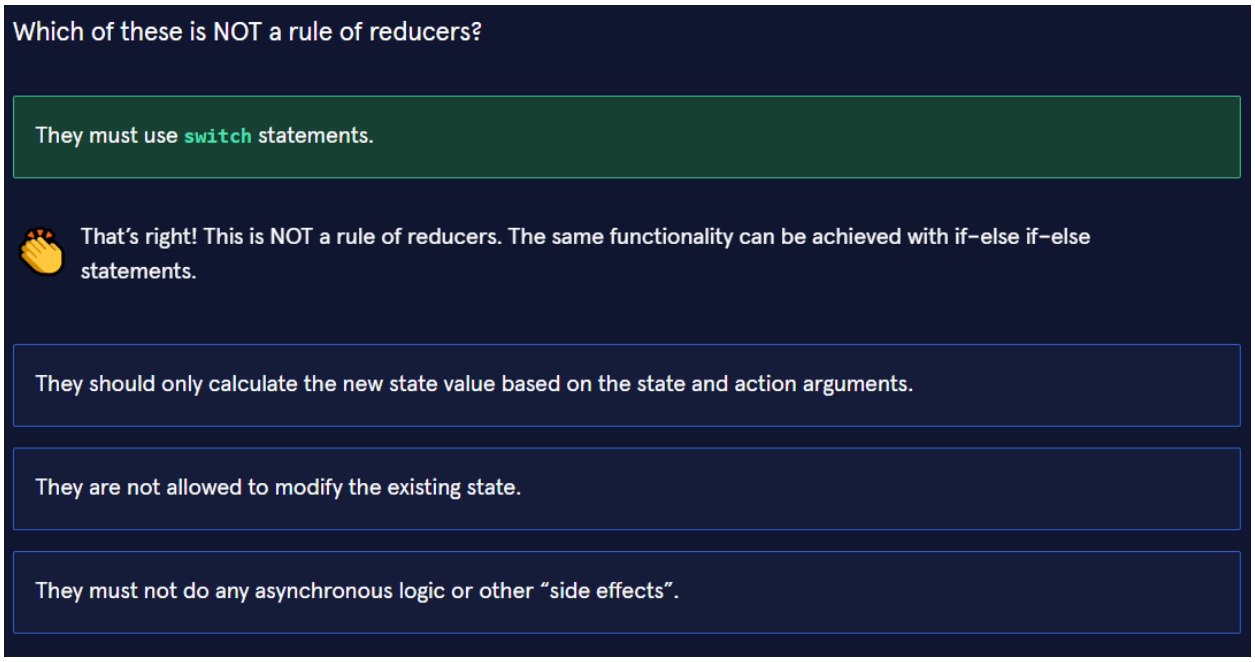
A `payload` property

A function property


Fill in the reducer function so that it updates `state` immutably. The `setToLate` action should set the `isLate` property to `true`.

```
const initialState = {
   isLate: false,
   ETA: '11:00',
   destination: 'Mars'
};

const arrivalReducer = (state = initialState, action) => {
  switch (action.type) {
    case 'setToLate': {
      return {
        ...state ,
        isLate: true
      };
    }
    default:
      return state;
  }
}
```

You got it!



Which of these is NOT a rule of reducers?

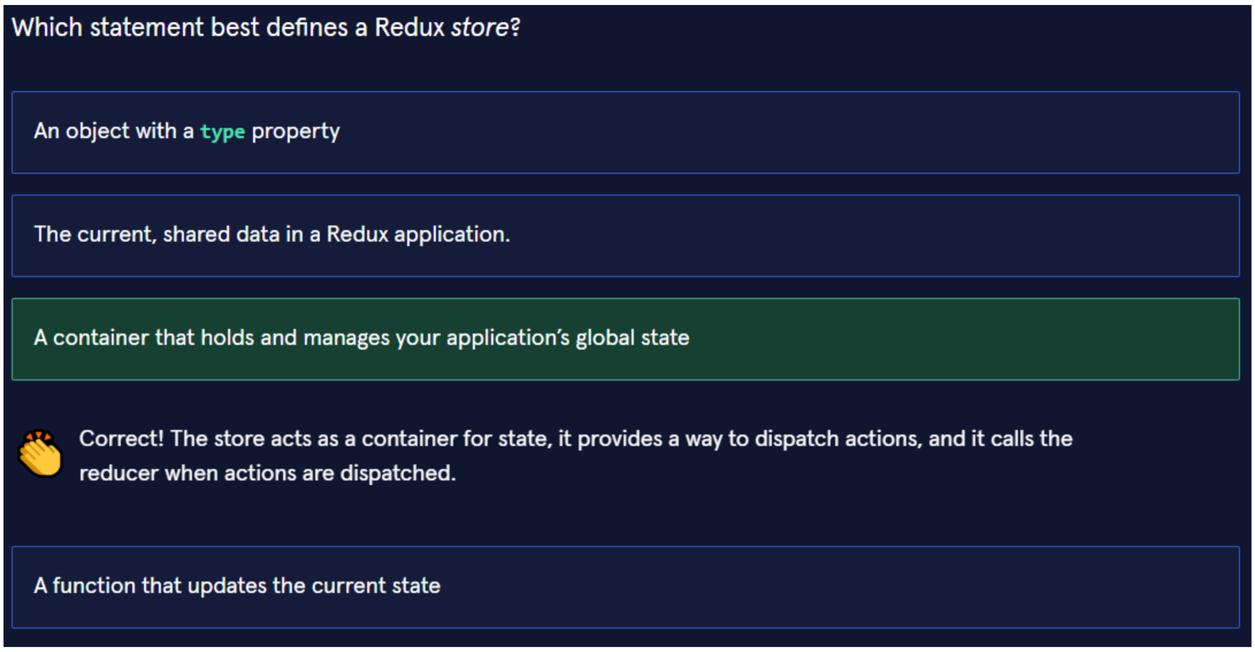
[✓ They must use `switch` statements.]

That's right! This is NOT a rule of reducers. The same functionality can be achieved with if–else if–else statements.

They should only calculate the new state value based on the state and action arguments.

They are not allowed to modify the existing state.

They must not do any asynchronous logic or other "side effects".



Which statement best defines a Redux *store*?

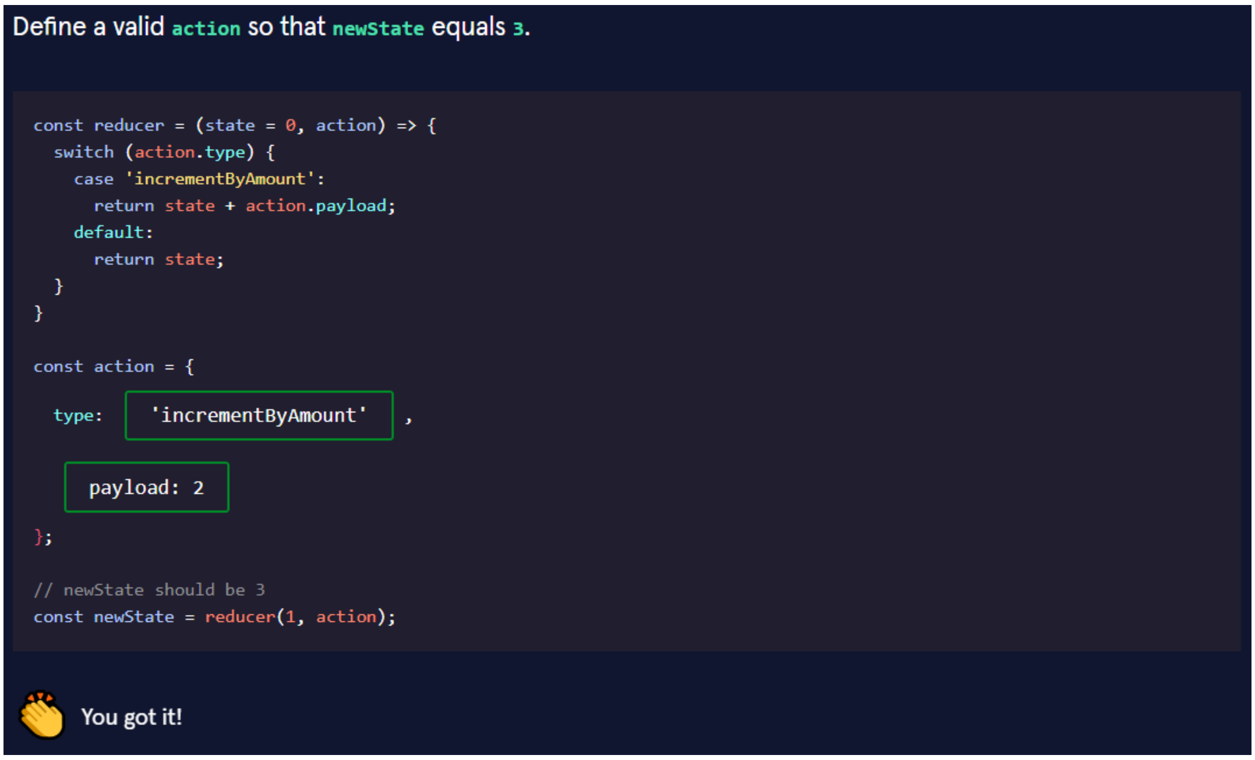
An object with a `type` property

The current, shared data in a Redux application.

[✓ A container that holds and manages your application's global state]

Correct! The store acts as a container for state, it provides a way to dispatch actions, and it calls the reducer when actions are dispatched.

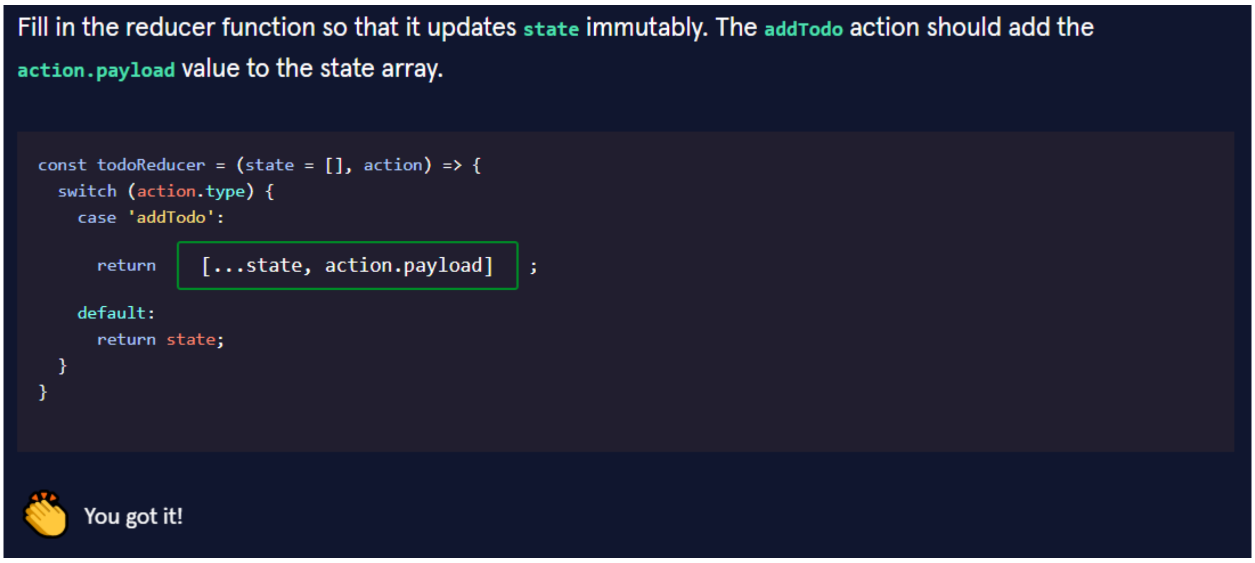
A function that updates the current state



Define a valid `action` so that `newState` equals `3`.

```
const reducer = (state = 0, action) => {
  switch (action.type) {
    case 'incrementByAmount':
      return state + action.payload;
    default:
      return state;
  }
}

const action = {
  type:  'incrementByAmount' ,
    payload: 2
};

// newState should be 3
const newState = reducer(1, action);
```

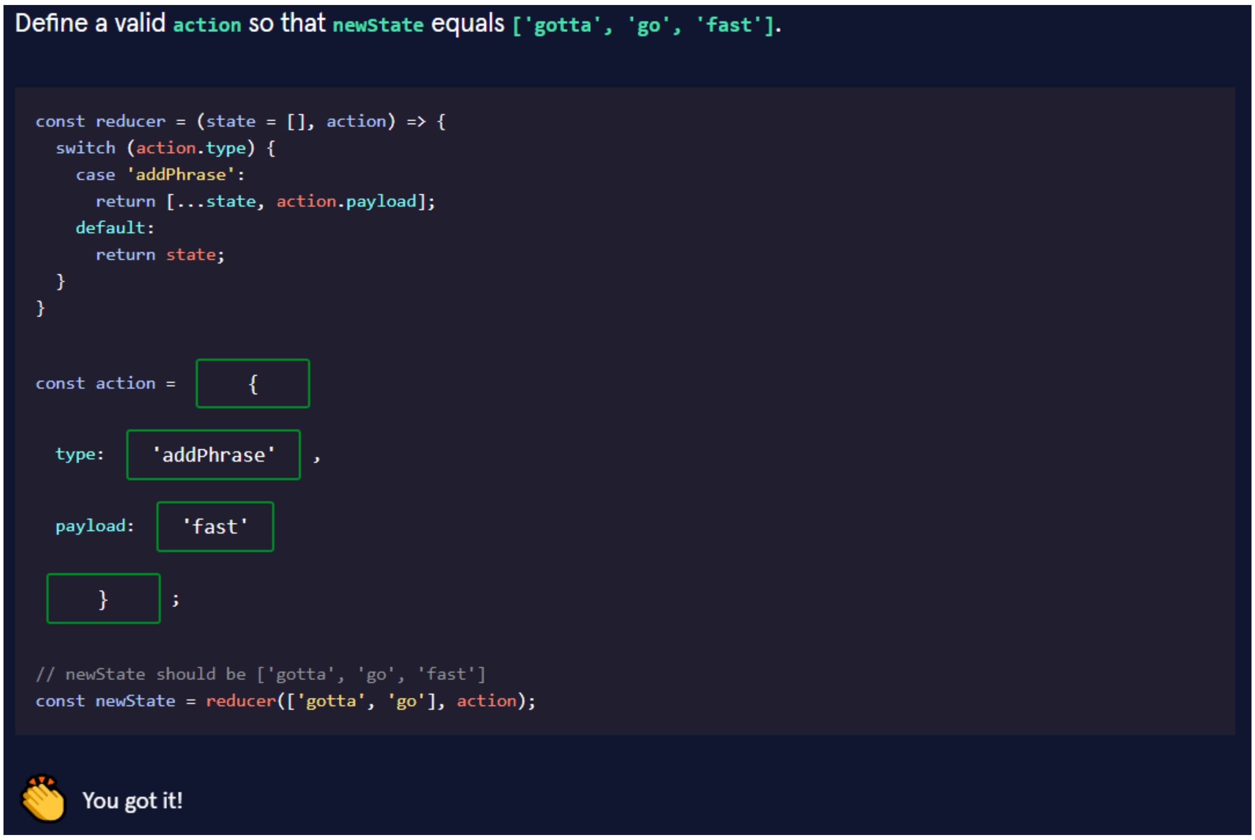
👏 You got it!



Fill in the reducer function so that it updates `state` immutably. The `addTodo` action should add the `action.payload` value to the state array.

```
const todoReducer = (state = [], action) => {
  switch (action.type) {
    case 'addTodo':
      return  [...state, action.payload] ;
    default:
      return state;
  }
}
```

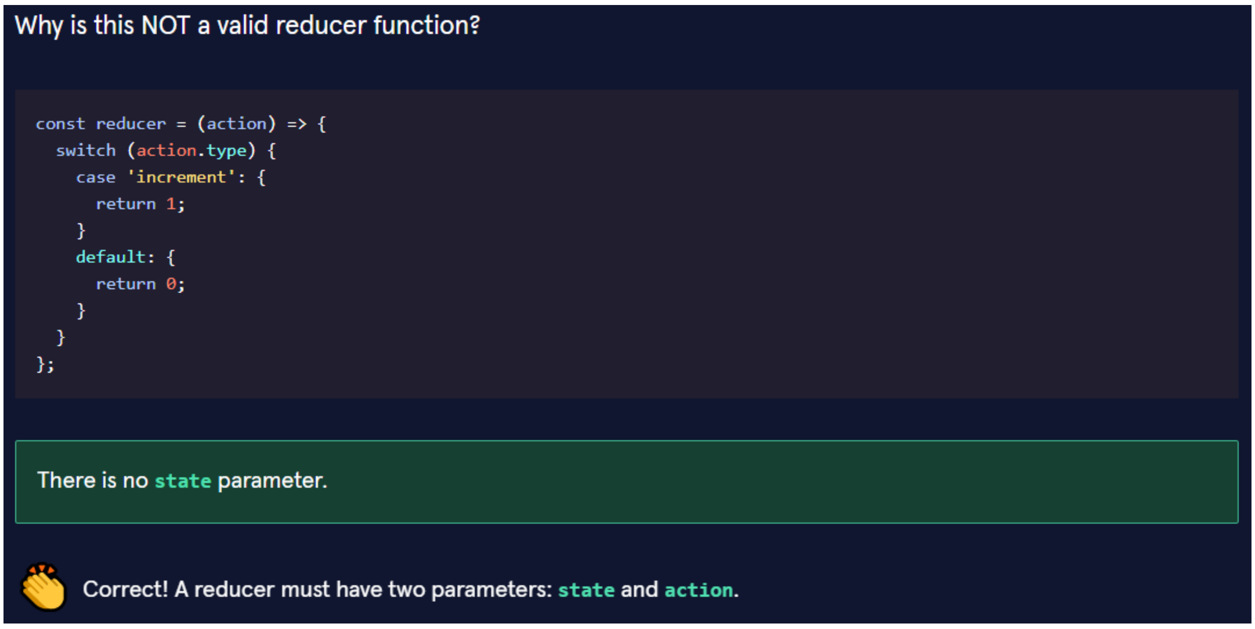
👏 You got it!



Define a valid `action` so that `newState` equals `['gotta', 'go', 'fast']`.

```
const reducer = (state = [], action) => {
  switch (action.type) {
    case 'addPhrase':
      return [...state, action.payload];
    default:
      return state;
  }
}

const action = { 

  type: 'addPhrase' ,

  payload: 'fast' 

 } ;

// newState should be ['gotta', 'go', 'fast']
const newState = reducer(['gotta', 'go'], action);
```

You got it!

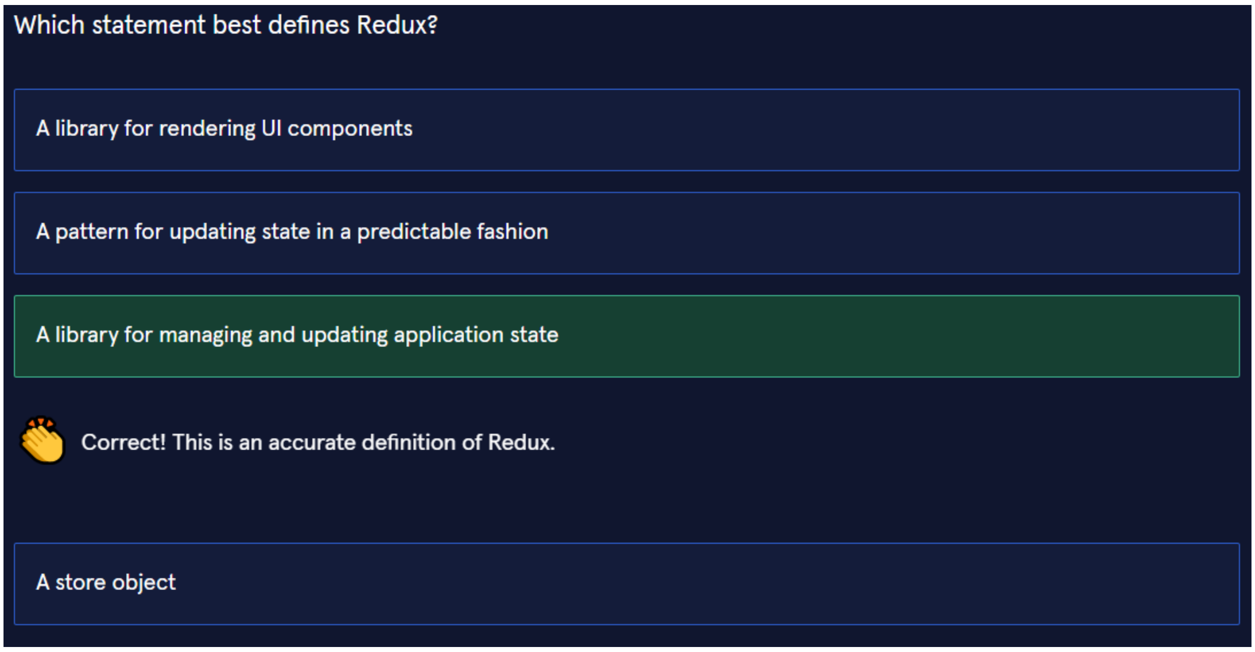


Why is this NOT a valid reducer function?

```
const reducer = (action) => {
  switch (action.type) {
    case 'increment': {
      return 1;
    }
    default: {
      return 0;
    }
  }
};
```

There is no `state` parameter.

Correct! A reducer must have two parameters: `state` and `action`.

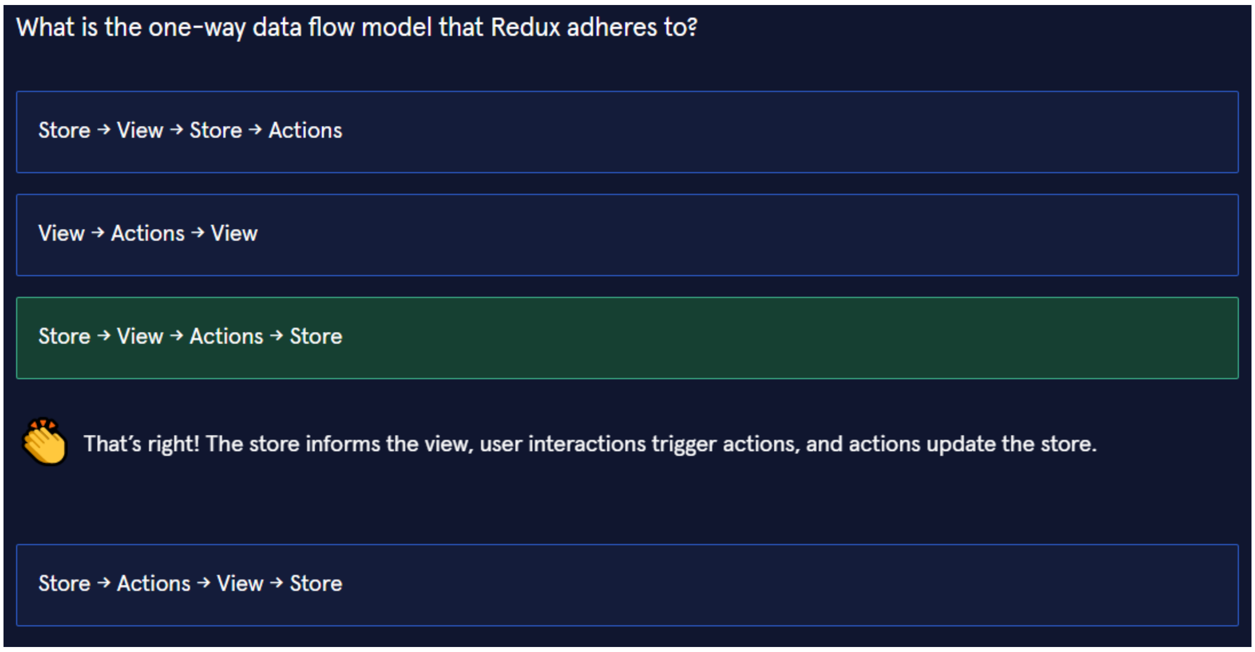


Which statement best defines Redux?

- A library for rendering UI components
- A pattern for updating state in a predictable fashion
- [✓ A library for managing and updating application state]

👏 Correct! This is an accurate definition of Redux.

- A store object



What is the one-way data flow model that Redux adheres to?

- Store → View → Store → Actions
- View → Actions → View
- [✓ Store → View → Actions → Store]

👏 That’s right! The store informs the view, user interactions trigger actions, and actions update the store.

- Store → Actions → View → Store