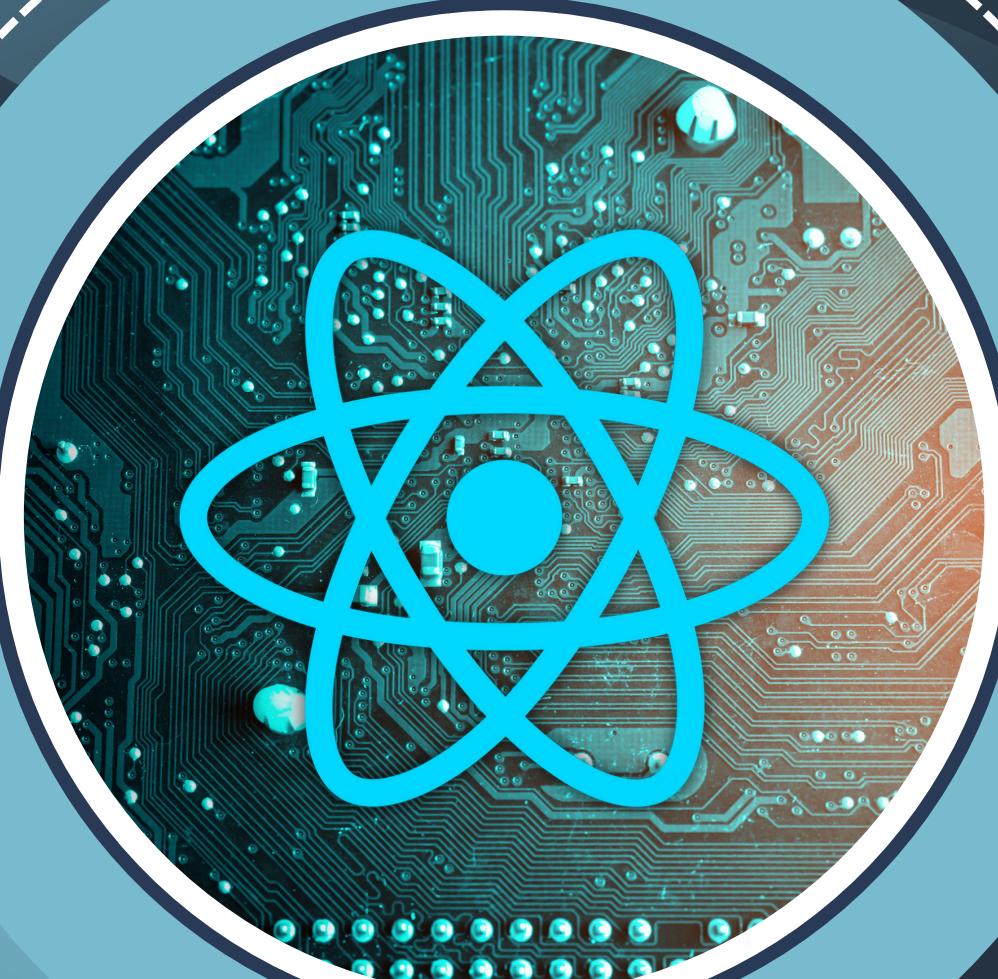


# Observer

دیزاین پترن هادر React

Behavioral Patterns

به زبان ساده همراه با مثال های عملی  
کاربرد دیزاین پترن ها برای برنامه نویسان React



# Behavioral Patterns

## Observer



الگوی Observer برای تعریف وابستگی "یک به چند" بین اشیا استفاده می شود. در این الگو، زمانی که وضعیت یک شیء (موضوع یا Subject) تغییر کند، همه اشیاء وابسته به آن (ناظران یا Observers) مطلع شده و به روزرسانی می شوند.

فرض کنید می خواهیم ابزاری شبیه به Zustand یا Redux یا State Management پیاده سازی کنیم، اما بدون شک چیزی که ما می سازیم، با چیزی که این کتابخانه های بزرگ ساخته اند اصلا قابل قیاس نیست، اما این مثال علاوه بر اینکه باعث می شود این الگو را به خوبی یاد بگیرید، دیدگاه خوبی هم راجع به نحوه کارکرد این کتابخانه ها به شمامی دهد.

# بخش Observable

این کلاس وظیفه مدیریت Observer ها و اطلاع رسانی تغییرات را بر عهده دارد.



```
type Observer = () => void;

class Observable<T> {
  private observers: Observer[] = [];
  private state: T;

  constructor(initialState: T) {
    this.state = initialState;
  }

  getState(): T {
    return this.state;
  }

  setState(newState: T): void {
    this.state = newState;
    this.notify();
  }

  subscribe(observer: Observer): void {
    this.observers.push(observer);
  }

  unsubscribe(observer: Observer): void {
    this.observers = this.observers.filter(obs => obs !== observer);
  }

  private notify(): void {
    this.observers.forEach(observer => observer());
  }
}

export default Observable;
```

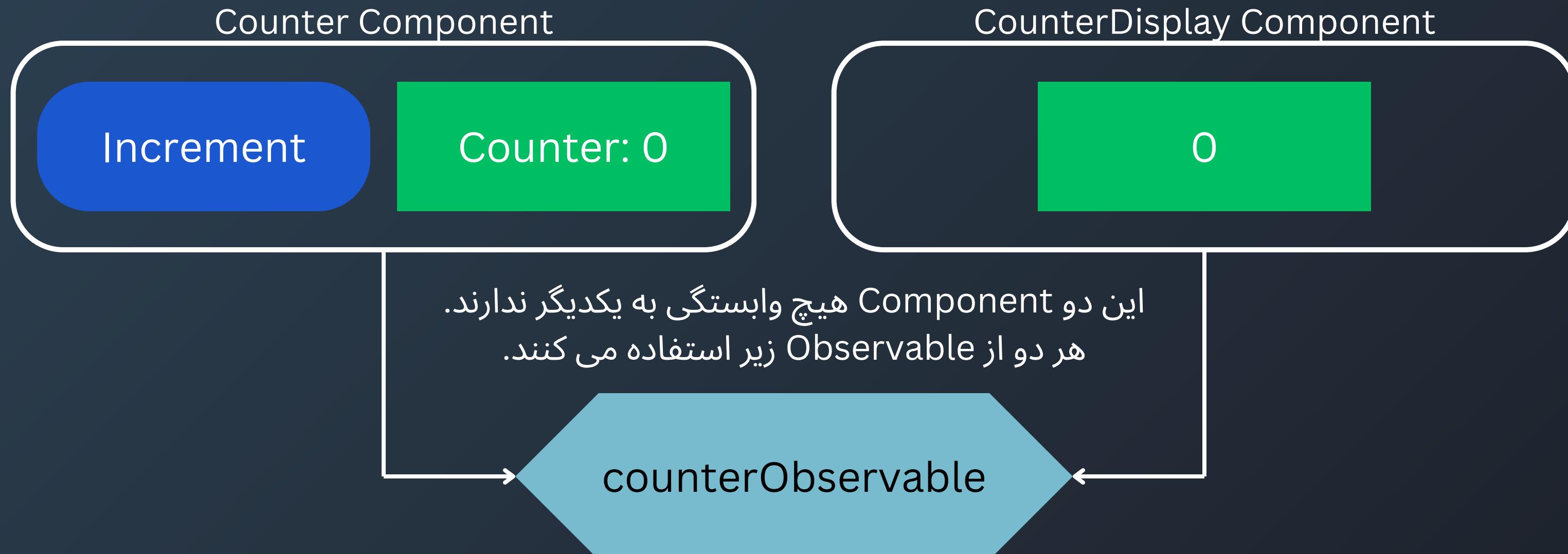
- لیستی از Observer ها را مدیریت می کند.
- دارای Method هایی برای افزودن و حذف Observer ها می باشد.
- با تغییر وضعیت، همه Observer ها را مطلع می کند.

عبارة Observable به معنای ناظر، دیده بان، مراقب می باشد.  
به بیانی ساده کلاس Observable مانند مدیر مجموعه ای از  
دیده بان ها عمل می کند، آن ها را اضافه و حذف می کند و در  
صورت تغییر وضعیت، آن ها را مطلع می کند.



# ساخت Counter

می خواهیم یک دکمه Counter مقدار یک را افزایش دهیم، همچنین یک State CounterDisplay نیز می سازیم که مقدار همان State را نمایش می دهد.



# بخش counterObservable

در این بخش یک نمونه Observable ایجاد می کنیم.



## counterObservable



```
import Observable from './observable.class';
export const counterObservable = new Observable(0);
```

# بخش Counter

در این بخش Counter اول خود را که Component نام دارد و دارای یک تگ Button برای افزایش مقدار State و یک تگ Span برای نمایش مقدار State می باشد را به شکل زیر می سازیم.



```
import { useState, useEffect } from 'react';
import { counterObservable } from './pattern/counter.observable';

const Counter: React.FC = () => {
  const [counter, setCounter] = useState(counterObservable.getState());

  useEffect(() => {
    const updateState = () => setCounter(counterObservable.getState());
    counterObservable.subscribe(updateState);
  }, []);

  const increment = () => {
    counterObservable.setState(counter + 1);
  };

  return (
    <div>
      <button onClick={increment}>Increment</button>
      &nbsp;
      <span>Counter: {counter}</span>
    </div>
  );
};

export default Counter;
```

# بخش CounterDisplay

در این بخش CounterDisplay نام دارد و فقط با استفاده از یک تگ H1 مقدار State را نمایش می دهد را به شکل زیر من سازیم.

## CounterDisplay



The screenshot shows a code editor window with a dark theme. At the top left, there are three small colored dots (red, yellow, green). The main area contains the following code:

```
import { useState, useEffect } from 'react';
import { counterObservable } from './pattern/counter.observable';

const CounterDisplay: React.FC = () => {
  const [counter, setCounter] = useState(counterObservable.getState());

  useEffect(() => {
    const updateState = () => setCounter(counterObservable.getState());
    counterObservable.subscribe(updateState);

    return () => counterObservable.unsubscribe(updateState);
  }, []);

  return <h1>{counter}</h1>;
};

export default CounterDisplay;
```

# طرز استفاده از CounterDisplay و Counter

در پایان به شکل زیر از CounterDisplay و Counter استفاده می کنیم.



## ObserverDesignPattern

```
import Counter from './Counter';
import CounterDisplay from './CounterDisplay';

const ObserverDesignPattern = () => {
  return (
    <div>
      <Counter />
      <hr />
      <CounterDisplay />
    </div>
  );
};

export default ObserverDesignPattern;
```