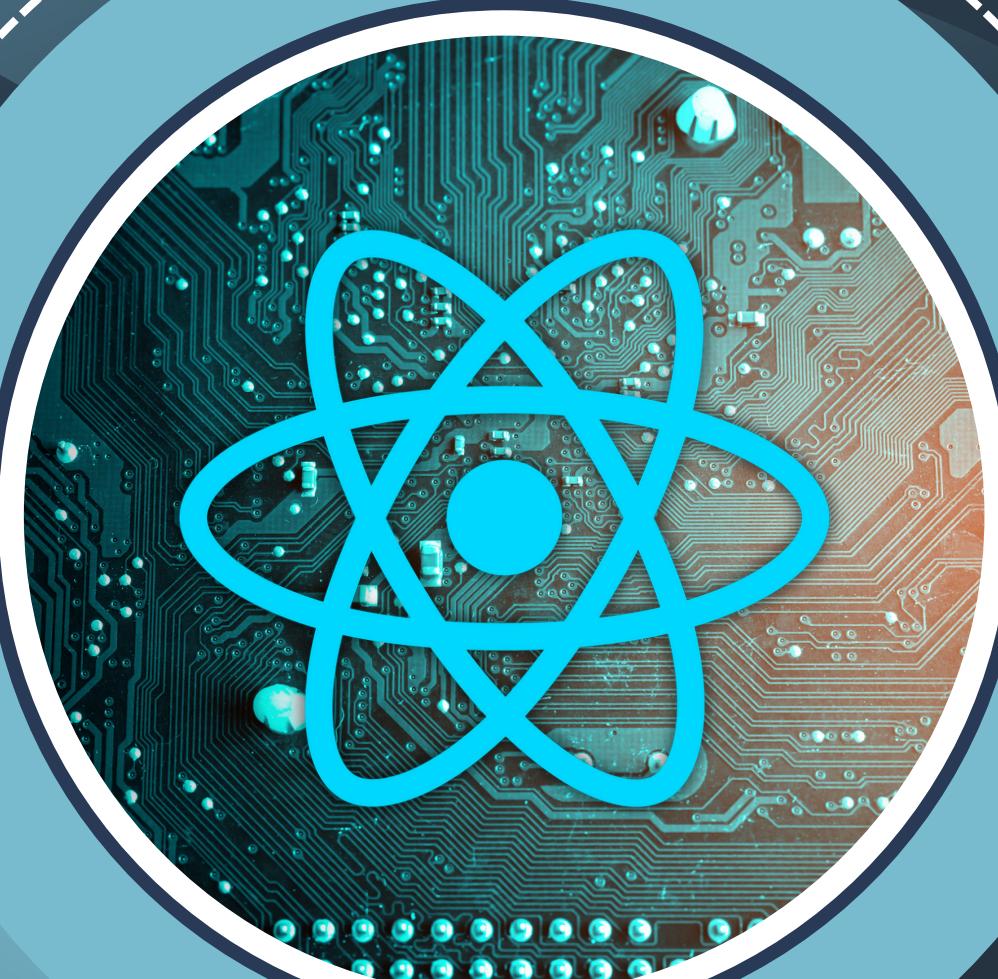


# Flyweight

## دیزاین پترن هادر React

Structural Patterns

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



# Structural Patterns

## Flyweight



الگوی Flyweight برای کاهش مصرف حافظه و بهینه سازی استفاده از منابع، با اشتراک گذاری داده های مشترک بین اشیاء مشابه استفاده می شود. این الگو در جاهایی که تعداد زیادی از اشیاء مشابه داریم، کاربرد دارد.

فرض کنید یک تگ Canvas داریم که قرار است کلی دایره با اندازه ها و رنگ های تصادفی در آن رسم شود، اما برای مدیریت بهتر حافظه آن دایره هایی که شعاع و رنگ یکسان دارند را مجددا نسازیم و در جایی ذخیره کنیم تا از فراخوانی مجدد کدهای آن جلوگیری کنیم.



# بخش IShape

ابتدا یک Interface به شکل زیر برای Shape ایجاد می کنیم.  
یک Method با نام draw داریم که دارای یک ورودی ctx که در واقع همان تگ Context است و دو ورودی دیگر به نام های x و y برای محل قرارگیری دایره.



IShape

```
export default interface IShape {  
    draw(ctx: CanvasRenderingContext2D, x: number, y: number): void;  
}
```

# بخش Circle

این Class وظیفه رسم کردن دایره را بر اساس ورودی های موجود بر عهده دارد.  
در این بخش اصطلاحا Concrete می گوییم که اشیاء اشتراکی را پیاده سازی  
می کند.



```
import IShape from './shape.interface';

export default class Circle implements IShape {
    constructor(private radius: number, private color: string) {}

    draw(ctx: CanvasRenderingContext2D, x: number, y: number): void {
        ctx.beginPath();
        ctx.arc(x, y, this.radius, 0, 2 * Math.PI);
        ctx.fillStyle = this.color;
        ctx.fill();
        ctx.closePath();
    }
}
```

# بخش ShapeFlyweightFactory

در این بخش پس از ساخت هر دایره آن را در حافظه ذخیره می کنیم و پس از آن چنانچه دایره تکراری با شعاع و رنگ یکسان درخواست شد دیگر مجددا آن را نمی سازیم و همان دایره ای که در حافظه داریم را برمی گردانیم.



## ShapeFlyweightFactory

```
import Circle from "./circle.concrete";
import IShape from "./shape.interface";

type CircleKey = string;

export default class ShapeFlyweightFactory {
  private static circleMap: Record<CircleKey, Circle> = {};

  static getCircle(radius: number, color: string): IShape {
    const key = `${radius}-${color}`;
    if (!this.circleMap[key]) {
      console.log("Create a new circle: ", key);
      this.circleMap[key] = new Circle(radius, color);
    } else {
      console.log("Return of existing circle: ", key);
    }
    return this.circleMap[key];
  }
}
```

# طرز استفاده از ShapeFlyweightFactory

در پایان به شکل زیر درخواست سه دایره را می دهیم و نکته مهم این است که با توجه به منطقی که پیاده سازی کردیم چون دایره سوم دارای شعاع و رنگ یکسان با دایره اول است لذا همان دایره اول برگردانده می شود.



## FlyweightDesignPattern

```
import { useEffect, useRef } from "react";
import ShapeFlyweightFactory from "./pattern/shape-flyweight-factory";

const FlyweightDesignPattern = () => {
  const canvasRef = useRef<HTMLCanvasElement>(null);

  useEffect(() => {
    const canvas = canvasRef.current;
    if (!canvas) return;
    const ctx = canvas.getContext("2d");
    if (!ctx) return;

    const circle1 = ShapeFlyweightFactory.getCircle(50, "red");
    circle1.draw(ctx, 100, 100);

    const circle2 = ShapeFlyweightFactory.getCircle(30, "blue");
    circle2.draw(ctx, 200, 100);

    const circle3 = ShapeFlyweightFactory.getCircle(50, "red");
    circle3.draw(ctx, 300, 100);
  }, []);

  return (
    <canvas ref={canvasRef} width={400} height={200} style={{ border: "1px solid black" }} />
  );
};

export default FlyweightDesignPattern;
```

