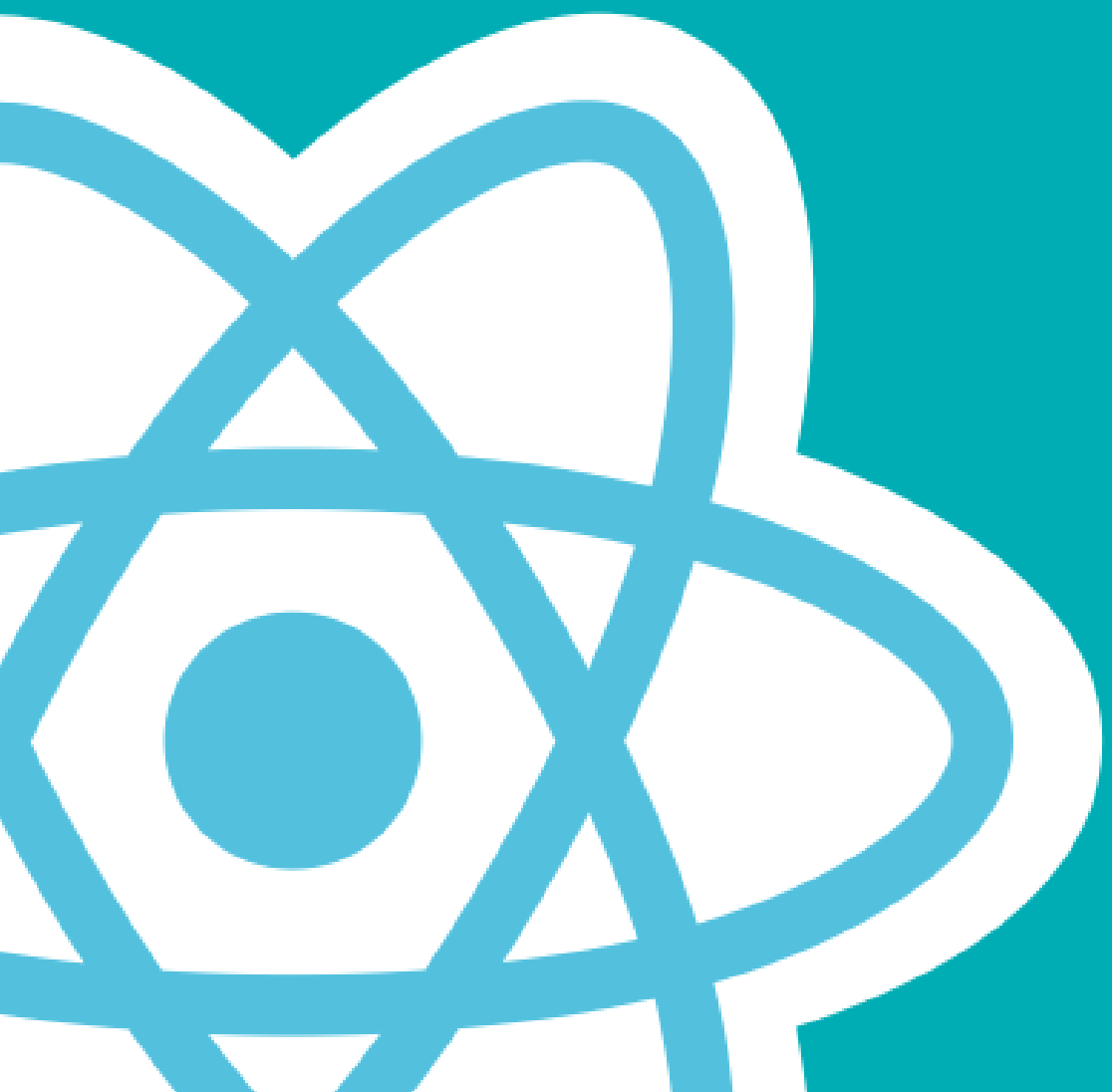
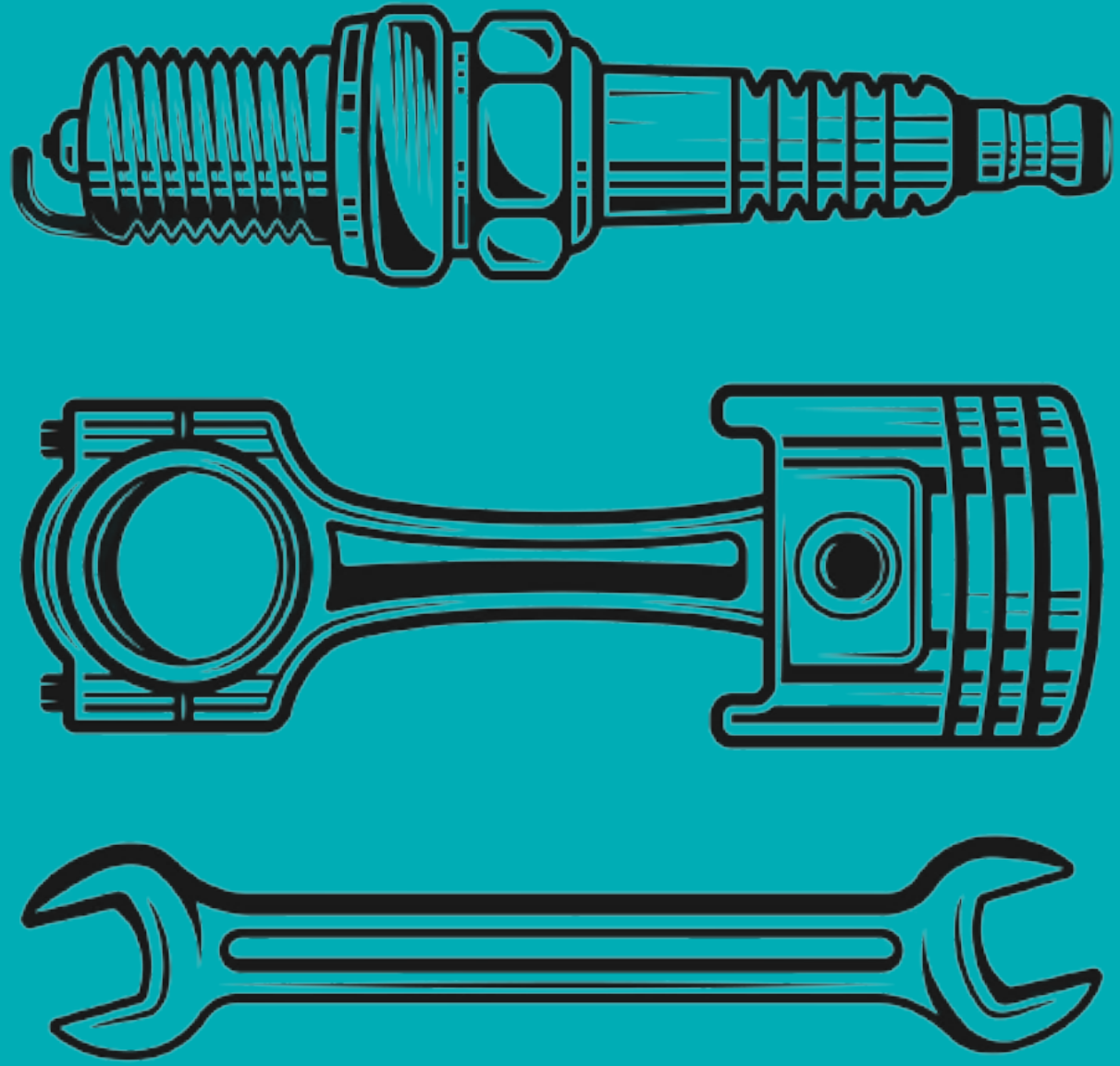
نزار پروژه ات منفجر بشه!



راهش Error Boundaries هست...

"Error Boundaries" در ReactJS مکانیزمی است که به شما اجازه می‌دهد خطاها در یک قسمت خاص از کد را مدیریت کنید تا این خطاها به تمام بخش‌های برنامه‌ی شما گسترش نیابند. به این ترتیب، اگر یک خطا در یک کامپوننت رخ دهد، اثرات آن تا حد ممکن به کامپوننت‌های دیگر نمی‌رسد و برنامه ادامه می‌یابد.



چه مواقعی ازش استفاده کنم!؟

- **در کامپونت‌های مستقل:** ممکن است بخواهید یک Error Boundary را برای کامپونت‌هایی که به صورت مستقل در سرتاسر برنامه استفاده می‌شوند، ایجاد کنید. این‌طوری خطاهای احتمالی در این کامپونت‌ها به سایر بخش‌های برنامه گسترش نخواهد یافت.
- **در کامپونت‌های دارای ارتباط با شبکه:** اگر کامپونت‌هایی در برنامه دارای ارتباط با سرویس‌های شبکه یا API هستند، ممکن است خطاهای مرتبط با اتصال به شبکه رخ دهد. با ایجاد یک Error Boundary در اطراف این کامپونت‌ها، می‌توانید این خطاها را مدیریت کنید.
- **در کامپونت‌های دینامیک:** اگر کامپونت‌هایی وجود دارند که بر اساس داده‌های دینامیک رندر می‌شوند و خطاهایی ممکن است در این فرآیند رخ دهند، می‌توانید یک Error Boundary را برای مدیریت این خطاها در نظر بگیرید.

چطور ازش استفاده کنم؟

برای ساخت یک Error Boundary در کامپوننت‌های تابعی، از Hook useEffect و یک متغیر state استفاده می‌کنیم.
در اینجا یک مثال از ساخت یک Error Boundary در یک کامپوننت تابعی در ReactJS آمده است:

Step 1

```
1 import React, { useState, useEffect } from 'react';
```

Step 2

```
1 function MyErrorBoundary({ children }) {
2
3   const [hasError, setHasError] = useState(false);
4
5 }
```

Step 3

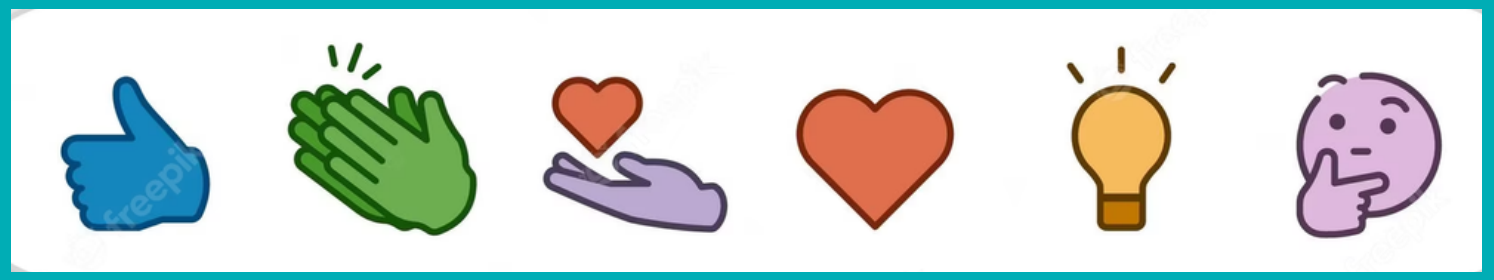
```
function MyErrorBoundary({ children }) {
  const [hasError, setHasError] = useState(false);

  useEffect(() => {
    // componentDidCatch معادل با
    window.addEventListener('error', handleError);
    return () => {
      window.removeEventListener('error', handleError);
    };
  }, []);

  const handleError = (errorEvent) => {
    console.error("Error caught by Error Boundary:", errorEvent.error);
    setHasError(true);
  };

  if (hasError) {
    // توانید یک UI در اینجا می طراحی کنید خطا نمایش برای مناسب
    return <h1>Something went wrong.</h1>;
  }

  return children;
}
```

Step 4

```
1 function App() {
2   return (
3     <MyErrorBoundary>
4       {/* کامپوننت‌های دیگری که ممکن است خطا داشته باشند */}
5     </MyErrorBoundary>
6   );
7 }
8
9 export default App;
```

در این مثال، از Hook useEffect برای افزودن و برداشتن یک Event Listener بر روی ویندو (Window) استفاده می‌کنیم. وقتی که خطا در کامپوننت‌های داخلی رخ دهد، Event Listener فراخوانی می‌شود و متد handleError اجرا می‌شود که خطا را مدیریت می‌کند. سپس کامپوننت MyErrorBoundary متناسب با وضعیت خطا، یک UI مناسب را نمایش می‌دهد.

ممنون که تا اینجا همراهی کردی رفیق خوشحال میشم نظرت رو درباره پست کامنت کنی 😀



از اینا یادت نره :)