11 best practices every React developer should follow to don't make the horror project

Front-End

Farid Karami

Should avoid hard coding

هرگز متن هایی که در پروژه استفاده می کنید را بصورت مستقیم درون تگ ها قرار ندهید.

<Title>Welcome to my website</Title>

این کار علاوه بر اینکه باعث ناخوانایی کد می شود وقتی که بخواهید پروژه را چند زبانه کنید نیز وقت زیادی از شما خواهد گرفت و تغییرات را سخت و زمانبر خواهد کرد.
تمامی محتواهای متنی خود را درون یک فایل JSON یا JS تعریف کرده و با استفاده از پیاده سازی یک سناریو از آنها استفاده کنید.

<Title>{locales.title}</Title>

Front-End

Farid Karami

Avoid using Index as Key for map

در صورتی که از index برای مقدار key استفاده شود چنانچه طول آرایه تغییر کند مقادیری که بعنوان key تعیین کرده اید ممکن است دیگر unique نباشند.

```
<Component key={index} />
```

می توان از این کار اشتباه به عنوان یک anti-pattern نام برد و بجای آن چنانچه آرایه را از یک سرویس دهنده می گیرید به احتمال زیاد فیلد id دارد یا از پکیجی مانند nanoid استفاده کنید و خودتان id منحصر فرد به خانه های آرایه اضافه کنید و از آن برای مقداردهی به key استفاده نمایید.

```
<Component key={item.id} />
```

Error Boundaries

یکی از نکات مهمی که هر برنامه نویسی باید رعایت کند بخش مدیریت خطا می باشد و برای این کار از try...catch استفاده می کنید.

```
try { } catch (error) { }
```

اما زمانی که در حال نوشتن JSX هستید استفاده از try...catch مناسب نیست و به جای آن از مفهومی به نام Error Boundary استفاده کنید و برای اینکار هم می توانید یک سناریو بسازید یا از پکیج react-error-boundary استفاده کنید.

```
<ErrorBoundary fallback={<div>Error message</div>}>
    <Component />
</ErrorBoundary>
```

Try to make fewer States

ابتـدا سـعی کنیـد از State هـای کمتری اسـتفاده کنید چون هر چه تعداد State ها بیشـتر باشـد برنامه ای به مراتب پیچیده تری خواهید داشت.

```
const [firstName, setFirstName] = useState('');
const [lastName, setLastName] = useState('');
```

چنانچـه بـه State هـای زیـادی نیـاز داریـد سـعی کنیـد آنهـا را بـر اسـاس نـوع ارتباطـی کـه بـا یکدیگر دارند در Object های با نام مناسب دسته بندی نمایید.

```
const [userInformation, setUserInformation] = useState({
        firstName: '',
        lastName: '',
})
```

Maximum 4 parameters

هرگز نباید یک Function یا Component بیش از 4 پارامتر ورودی بگیرد و باید آن را اصلاح کنید تا از پیچیدگی برنامه جلوگیری نمایید.

```
<Component a={1} b={2} c={3} d={4} e={5} f={6} g={7} h={8} />
```

چنانچه یک Function یا Component بیش از 4 پارامتر داشته باشد بی شک چندین کار را به آن محول کرده اید. توجه داشته باشید که اگر قرار است یک پکیج توسعه دهید پیروی کردن از این نکته کاملا اختیاری است و به عوامل و شرایط مختلفی بستگی دارد.

```
<Component a={1} b={2} />
```

Use the klona package

در Shallow copy یک متغیر ساخته می شود و به مکانی در حافظه، که مقدار متغیر قبلی در آن قرار گرفته است، اشاره می کند و چنانچه مقدار متغیر اول تغییر کند، متغیر دوم نیز تغییر می کند و به همین ترتیب اگر مقدار متغیر دوم تغییر کند، مقدار متغیر اول هم تغییر می گند.

برای Deep copy روش های بسیاری وجود دارد و هر یک ممکن است در شرایط مختلف جوابگوی نیاز شما نباشد اما یکی از راه های مطمئن که Performance مناسبی هم دارد استفاده از پکیج klona می باشد.

```
const newObject = klona(oldObject);
```

Use the react-window package

در نرم افزارهای تحت وب بهتر است همان مقدار محتوایی که داخل صفحه نمایشگر کاربر قابل نمایش است را Render کنید و کاربر هر چقدر Scroll کرد باقی اطلاعات را Render خواهید کرد.

با استفاده از پکیج react-window می توانید به راحتی اینکار را انجام دهید و با ادغام آن با Pagination و... یک تجربه فوق العاده برای کاربر خلق کنید. بعنوان مثال فرض کنید در سناریو Pagination به صورت 20 تا 20 تا اطلاعات را از سرویس دهنده می گیرید، با استفاده از این پکیج چنانچه اندازه نمایشگر کاربر فقط می تواند 5 سطر را نمایش دهد فقط همان 5 المان Render می شود و باقی آن با Scroll کردن Render خواهد شد.

Axios interceptors

مدیریت خطاهایی که در زمان کار با API رخ خواهند داد یکی از مهم ترین کارهایی است باید انجام دهید.
حتما خطاهایی که شکل سراسری دارند را مانند خطای 500 و 403 و... را به صورت کلی با استفاده از interceptor های Axios هندل کنید.

در Axios از Request Interceptor برای زمانی که Request را ارسال می کنید استفاده کنید و مثلا چنانچه در Cookie یا Local Storage یک Token وجود دارد آنرا به Request خود اضافه کنید یا هر تغییر دیگری را در تمامی Request های خود اعمال کنید.
همچنین از Response Interceptor برای زمانی که Response سمت شما آمده است استفاده کنید و چنانچه هر نوع خطایی رخ داده است آن را به صورت سراسری و کلی هندل نمایید و هرگز اینکار را به صورت تک تک برای هر API انجام ندهید.

Use the fontfaceobserver package

وقتی بسیاری از سایت ها را می بینید ابتدا دارای یک font-family هستند سپس فونت آنها تغییر می کند و این موضوع رابطه زیادی با سرعت اینترنت دارد و اگر سرعت اینترنت شما پایین باشد برای چندین ثانیه ظاهر سایت به دلیل عدم Load شدن فونت مطلوب نیست.

اگر خواستید بدانید که چه زمانی فونت مد نظرتان Load شده است و بر اساس آن مثلا یک Component را نمایش دهید می توانید از پکیج fontfaceobserver استفاده کنید.

```
const font = new FontFaceObserver('My Family', { weight: 400 });
font.load().then(function () {
  console.log('Font is available');
}, function () {
  console.log('Font is not available');
});
```

Lodash

حتما یک زمان را به یادگیری و مطالعه توابعی که در Lodash وجود دارند اختصاص دهید.
استفاده از آن باعث می شود حجم کد نویسی شما کمتر شود و چنانچه از توابع این کتابخانه استفاده می کنید خیالتان راحت است توابع آن به درستی کار می کنند و درگیر توابعی نمی شوید که خودتان توسعه داده اید و ممکن است دارای مشکلاتی باشند.

در رابطه با Lodash صحبت های زیادی می شود که از آن استفاده کنیم یا خیر و جواب آن اینست که چنانچه تنها به چند (بین یک الی سه) تابع آن نیاز دارید می توانید توابع مورد نظر را خودتان بنویسید اما در اینجا راجع به این حالت صحبت نمی کنیم و چنانچه به آن تسلط داشته باشید می توانید از آن بعنوان بستر اصلی کارهایتان استفاده کنید و حداکثر بهره مندی را از آن ببرید.

Validation

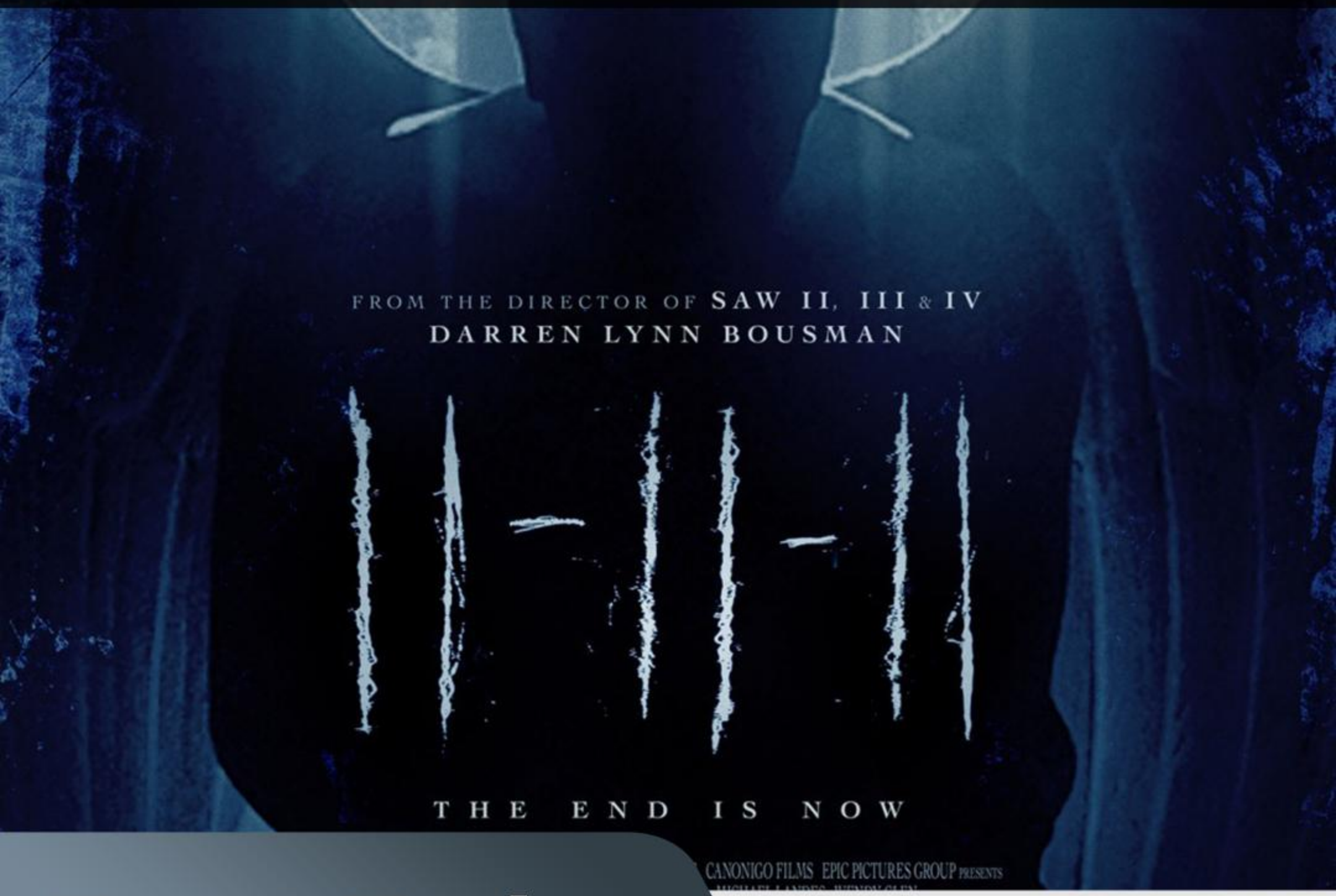
به کار بردن اعتبار سنجی می تواند تا حد زیادی شرط ها و خطاهای برنامه را کاهش دهد.
برای اینکار همیشه از کتابخانه های مطرحی که وجود دارند استفاده کنید و هرگز خودتان دستی اینکار را انجام ندهید.

از جمله کتابخانه های معروف برای اعتبار سنجی می توان به پکیج های Yup و Zod اشاره کرد. در حال حاظر پکیج Zod دارای ویژگی های نسبتا بهتری نسبت به سایر رقبای خود می باشد. در ادامه دو نمونه از کاربرد های این پکیج ذکر شده است.

```
import { z } from "zod";

z.string().endsWith(".com", { message: 'Only .com domains allowed' });
z.date().min(new Date("1992-10-24"), { message: 'Too old' });
```

در پایان یک فیلم ترسناک با عنوان 11-11-11 از Darren Lynn Bousman کارگردان مجموعه فیلم های ترسناک Saw براتون آوردم که در تاریخ 2011/11/11 منتشر شده دعوت می کنم که هم لذت ببرید و هم بترسید تا همیشه این 11 نکته ای که ذکر شد را رعایت کنید.



Front-End

Farid Karami