بسم الله الرحمن الرحیم

Design Pattern

راه حل هایی برای حل مشکل هایی است که در یک نرم افزار به وجود می اید و دلیل استفاده این است که اگر از این روش ها استفاده کنیم دیگر به این مشکل ها نمی خوریم.

الگوهای طراحی به سه دسته بندی تقسیم می شوند.

* Creational Design Patterns : الگو های طراحی سازنده
* Structural Design Patterns : الگو های طراحی ساختاری
* Behavioral Design Patterns : الگو های طراحی رفتاری

مورد اول راه حل هایی برای حل مشکلات ابجکت های یک کلاس است که معروف های این مورد singleton, factory, abstract factory,builder

دسته دوم (الگو های ساختاری) :برای مدیریت روابط بین کلاس ها و شی ها هست درواقع رابطه ی بین کلاس ها و object ها را هندل می کند . که معروف ترین ها هم adapter,Facade ,proxy,bridge,composit .و ....

دسته سوم مریوط به تعامل اشیا با یکدیگراست که معروف ترین ها هم strategy و observer و chain of responsibility هستند.

Creational : Singleton :

در این متد singleton از تولید object های جدید جلوگیری می کند. یعنی مثلا برای data base connection نیازی نیست 500 تا ابجکت جدید بسازیم که هر کدام از نوع data base connection باشند و زور زیادی به سرور و data base وارد شود.

یعنی مثلا برای اینکه نتوانیم شی جدیدی از روی یک object بسازیم باید ابتداعا constructor آن کلاس را از نوع private در نظر بگیریم و به جاش یک instance می سازیم که private static instance است و یک تابع

Public static function getInstance را می سازیم و بررسی می کنیم که اگر instance ای وجود نداشت یک instance بساز در غیر این صورت همان instance را برمیگرداند. با این کار تنها یک instance از آن کلاس داریم.

حال به بررسی design pattern های موجود در ری اکت می پردازیم که در ابتدا render prop را بررسی می کنیم.

Render prop تکنیکی است برای اشتراک کد در بین کامپوننت ها است ( states )

به این صورت است که زمانی که یک component ای را به عنوان یک فرزند فراخوانی می کنیم در داخل ورودی های این کامپوننت یک چیزی را رندر کنیم به این صورت

Const App =()=>

{

<h1>I am Father</h1>

<Child1 render={()=><h3>Iam Child</h3>} />

}

Const Child=({render})=>

{

<h1>Hello</h1>

<div>{render}</div>

}