**S02-E02:Events In React Way**

وقتی داریم یک event مینویسیم باید بصورت callback function بنویسیم چرا که میخواهیم بعدا اجرا شود.

<button

  onMouseEnter={() => alert("Mouse Entered!")}

  key={tab.id}

  className={activetab === tab.id ? "active" : ""}

>

 <span>{tab.title}</span>

 <span className="tab-indicator"></span>

</button>

طبق یک قاعده کلی:

هر وقت خواستیم به handler هامون یک چیزی پاس بدیم، اون رو بصورت arrow function مینویسیم.

function App() {

  const activetab = 1;

  const handleActive = (id)  => {

   console.log("clicked!!" , id)

  }

  return (

    <div className="tab">

      <div className="tab\_\_header">

        {tabData.map((tab) => (

          <button

            onClick={() => handleActive(tab.id)}

            key={tab.id}

            className={activetab === tab.id ? "active" : ""}

          >

            <span>{tab.title}</span>

            <span className="tab-indicator"></span>

          </button>

        ))}

      </div>

   <div className="tab\_\_content">{tabData[activetab - 1].content} </div>

    </div>

  );

}

export default App;

در کد بالا، id رو از داخل Button (arrow func) پاس دادیم به handleActive.

**S02-E03:What is state in React.js ?**

کامپوننت های ما یک سری data در خودشون دارند که این data شامل state و props هست.

این دیتا ها در کنار logic ای که ما مینویسیم UI برنامه ما را می سازند.

نکته:

یعنی خروجی ما بر اساس state و props هست که کنترل میشه.

State و props چه تفاوت هایی باهم دارند؟

* State یک دیتای internal هست ولی props یک دیتای external هست چون از پدر به فرزند منتقل میشه.
* State ها حتما و حتما توسط خود اون کامپوننت (که صاحب اون state هست) باید update بشه ولی props ها توسط کامپوننت پدر update میشن

و میشه اینطوری گفت که props ها استیت کامپوننت پدر هستند و چون استیت یک کامپوننت دیگر هستند باید توسط خود اون کامپوننت آپدیت بشن.

* Props ها Immutable یا read only هستند و ما اجازه Update کردن اون ها رو نداریم ولی state هارو در داخل اون برنامه و داخل اون کامپوننت میتونیم آپدیت کنیم.

نکات:

* کامپوننت های ما فقط زمانی re-render میشن که یا state و یا props اون ها تغییر کنه.
* State ها دیتاهای internal اون کامپوننت هستند که که ما، بین render های متوالی بهش دسترسی داریم.
* State ها، مغز یا memory اون کامپوننت هستند که بین render های متوالی نیاز هست که به این دیتا دسترسی داشته باشیم که بتونیم اطلاعات state مون رو آپدیت کنیم.

**S02-E04:What is state in React.js ?**

چرا به state ها نیاز داریم؟

ما به state ها نیاز داریم چون :

* اطلاعات کامپوننتمون رو توش ذخیره میکنیم
* در render های متوالی بهش دسترسی داشته باشیم
* با آپدیت شدن state بتونیم re-render رو داشته باشیم
* و در نهایت، کامپوننت UI ای که داره رو Update میکنه

React reacts to state changes

نکته:

هوک ها رو در داخل nested function ها یا داخل condition ها یا داخل loop ها نمیتونیم بذاریم.

**S02-E05:Do not mutate state in react.js**

نکته:

ما اجازه نداریم به صورت دستی state های خودمون رو تغییر بدیم.باید از setter function(setState func) کمک بگیریم.

State ها اگر تغییر کنن منجر به re-render شدن و در نهایت منجر به تغییر UI میشن.

حالا واقعا چرا نباید mutate کرد؟

Obj ها reference type هستند.وقتی آبجکت ما (state) ما mutate میشه، رفرنس ما تغییر نمیکنه و وقتی ما یک پراپرتی اون رو تغییر میدیم ولی رفرنس اون آبجکت ثابت هست و همون رفرنس قبلی رو داره.

راه حل چیه؟

وقتی ما یک property از یک obj رو mutate میکنیم فقط داریم property هاش رو تغییر میدیم و بنابراین رفرنس تغییر نمیکنه.

برای اینکه Re-render اتفاق بیوفته تو بحث obj ها و Array ها ما باید یک Obj یا یک array جدید ایجاد کنیم که یک دیتای جدید با یک رفرنس جدید ساخته بشه.

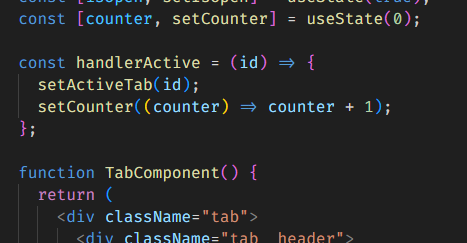
**S02-E06:Update state based on previous state in React.js**

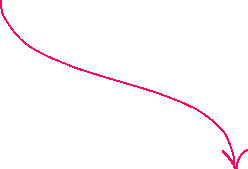
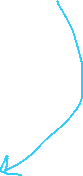
setState هایی که مینویسیم Async هستند. یعنی در آنِ واحد انجام نمیشن.

نکته:

وقتی مقدار state بعدی به state قبلی وابسته است، باید بیایم از روش callback method استفاده کنیم.



قلهمه جا میتونیم از callback func استفاده کنیم حتی جایی که به مقدار قبلیش وابسته نباشه.



اون counter ای که به عنوان آرگومان ورودی میگیره رو در داکیومنت ها با عنوان prevState میشناسند و این مقدار state قبلی ما هست.

مقدار state بعدی ما counter (با فلش آبی) هست.