**S1-E5: React Source Files**

نکات:

* ما در react با DOM سروکار نداریم و با Virtual DOM سرو کار داریم.
* ما در کل پروژه فقط و فقط یک فایل index.html داریم که همان DOM هست.
* داخل فولدر public، فونت ها، imageها و icon ها و موارد عمومی پروژه رو قرار میدیم.
* در ری اکت برای کال کردن فانکشن App به این صورت مینویسیم:

<App/>

**S1-E6: Separation of componenet**

App پدر همه ی کامپوننت هاست.

نکته:

* ما در یک فایل، میتونیم چندین export داشته باشیم و فقط یک export default میتونیم داشته باشیم
* ما 2 جور Import داریم:

1: import default

2:Named import که از {} استفاده میکنیم

**S1-E7: Component in react.js**

Component چیست؟

* کامپوننت ها ، بلاک ها یا اجزاء سازنده اپ ما هستند.
* شامل logic وdata و appearance آن قسمت از برنامه هستند.
* Reusable هستند و میتونن به صورت تو در تو استفاده بشن
* Component ها میتونن دیتا رو انتقال بدن

اگر parent-child باشن میتونن ولی اگر sibling باشن نمیتونن

* Componenet های ما یا Function هستند یا class
* بصورت pascal case نوشته میشن (ProductListOne)
* ما در React فراخوانی دیگه نداریم
* Component ها میتونن بهم دیتا پاس بدن

**S1-E8: CourseCard component markup**

jSX چیست؟

این مواردی که ما در کامپوننت ها return میکنیم jsx هست

ساختارش شبیه HTML هست.(JS and XML)

**S1-E9: Extract CourseList Component**

در این قسمت فقط یک قسمتی از کد را که میشد بصورت جداگانه درآورد، در یک کامپوننت مجزا نوشتیم.

**S1-E10: Render list in react.js**

در این قسمت یک تکه دیگر از کد را تبدیل به کامپوننتی به نام CouorseCard کردم.

**S1-E11: JSX in React.js**

React در پشت صحنه از Babel استفاده میکنه برای اینکه HTML ما رو در پشت صحنه به کد هایی حاوی

React.createElement() تبدیل کنه.

Declarative syntax چیست؟

توصیف میکنه که کامپوننت ما چجوریه، چجوری کار میکنه و چه شکل و شمایلی باید داشته باشه

این کامپوننتی که ما در نهایت یک JSX رو ریترن میکنه، UI و Logic برنامه رو مشخص میکنه

پس JSX یک extension هست که میتونیم توش HTML,CSS,JS بنویسیم.

Imperative چیست؟ Declarative چیست؟

In Declarative : What we want to do 

ما در React روش Declarative رو داریم.یعنی کد های ما به پیچیدگی vanilla JS نیست.

In Imperative: How to do things

ما در vanilla Js از روش imperative استفاده میکردیم.

یعنی دونه دونه المنت ها رو سلکت میکردیم،بعد به DOM،append میکردیم و...

Imperative کد هامون سخت و پیچیده میشه

نکته:

در حالتی که ما با React کار میکنیم ما هیچ وقت DOM رو تاچ نمیکنیم؛ یعنی از createElement و … استفاده نمیکنیم.

**S1-E12: Props in component**

Props ها

* دیتاهایی هستند که از کامپوننت پدر به فرزند منتقل میشه که بتونیم شکل و شمایل کامپوننت رو کنترل کنیم
* و در نهایت component ما بر اساس این بتونه دیتاهایی dynamic ای رو داخل خودش داشته باشه.
* مثل پارامتر هایی بود که ما به فانکشن ها پاس می دادیم.
* Props های ما میتونن هر نوع دیتایی باشن مثل Array ,function ,number ,object,other components
* بر اساس این props ها، کامپوننت ها میتونن توسط پدر، از نظر شکل و شمایل و عملکرد کنترل بشن
* Props ها یک سری آبجکت هستند

**S1-E13: Extract CourseCard component**

در این قسمت کامپوننت CourseCard که شامل دو قسمت CourseCardBody و CourseCardFooter بود را بصورت کامپوننت های جداگانه در خود فایل کامپوننت نوشتیم نه در یک فایل مجزا.

**S1-E14: Key in React.Js**

چرا Key ها در زمانی که استفاده میشه حتما باید یونیک باشه؟و چرا نباید از Index استفاده کرد؟

سناریویی رو در نظر بگیرید که روی یک سری آیتم map زدیم و طبیعتا هر کدام از محصولات باید یک key منحصر به فرد داشته باشند تا ری اکت بعدا یک سر نخ از اون محصولی که میخوایم روش یه کاری انجام بدیم (مثل ویرایش یا حذف و...) داشته باشه.

حالا قاعده کلی این هست که این key رو تا جایی که میشه باید id در نظر بگیریم نه index.

به این خاطر که اگر key مون رو index در نظر بگیریم با اتفاقاتی مثل حذف یکی از محصولات یا sort محصولات، شماره index ها تغییر میکنه و به این ترتیب کل محاسبات ما تغییر میکنه.

مگر اینکه جایی از Index استفاده کنیم که بدونیم جاش قرار نیست تغییر کنه.