**<이화이언 4차 수습과제>**

#0강

HTML

->바디 부분이 바뀜

SPA

->복잡한 웹사이트 관리

SCRIPT LANGUGE

자바스크립트의 사용 범위 넓어짐

(ES6 문법)

자료형(DATA TYPE)

-> Dynamic typing ( let사용 )

“ or ‘ 사용

undefined ->정의x

Java script 연산자

대입연산자 - 변수의 값을 대입하기 위해

오른쪽 -> 왼쪽

증가 연산자 ++/ 감소 연산자 --

postfix a++ / Prefix: ++a

a==b : a와b가 같다.

a!=b : a와 b가 다르다

일치연산자 - 엄격

a===b : a와 b가 값과 자료형 모두 같다

a!==b : a와 b가 값이나 자료형이 같지 않다

이진 논리 연산자

a&&b : a와 b가 모두 true일 경우에만 true

a||b : a 또는 b가 true일 경우 true

Javascript 함수

1. function statement

ex)function sum(a,b)

{return a+b;}

2. arrow function expression

ex) const multyply = (a,b) =>

{ return a\*b;}

실습 -> node.js와 npm(node.js 설치하면 자동으로 다운됨) 깔기 - 16번대

버전확인!

$ npm --version

#1강

리액트 - 사용자의 웹사이트의 상호작용을 돕는 인터페이스를 만들기 위한 js

라이브러리? - 자주 사용되는 기능들을 정리해 모아 놓은 것

사용자 인터페이스 - UI라이브러리 - 화면을 만들기 위한 기능들을 모아놓은것

프레임워크 vs 라이브러리

=> 프로그램 흐름의 제한권 차이

웹사이트의 작동 원리와 흐름을 함께 이해해야함!

장점: 빠른 업데이트 & 렌더링 속도, 재사용성, 개발 기간 단축, 유지보수 용이,

활발한 지식공유&커뮤니티

단점: 방대한 학습량, 계속 바뀜 - 꾸준히 공부해야함, 높은 상태관리 복잡도

#2강

index.html

<html>

<head>

</head>

<body>

</body>

</html>

cra

간편하게 한번에 처리 유용

방법

$ npx create-react app <your-project-name>

-경로 변경

$ cd my-app

#3강

JSX - java script의 확장 문법

:js + XML/HTML

-역할

createELement -> JSX를 JS로 바꿔줌

리액트에서 JSX를 쓰는것 필수는 아님!

JSX의 장점

-> 필수는 아니나 장점 때문에 권장

간결함, 가독성 향상, 버그를 발견하기 쉬움, injection attacks 방어

JSX의 사용법

중간에 js 쓰고싶다?

-> 중괄호 사용 {}

-태그 속성에 값 넣는 방법

큰따옴표 사이에 문자열, 중괄호 사이에 js코드 넣기

-자식 정의

#4강

elements - 리액트 앱을 구성하는 가장 작은 블록들

리액트 엘 -랜더링> 돔 엘리멘트

r.e -> js 객체 형태로 존재

react -> immutable

im - not , mutable - 변할수 있는

=> 불벼성

elements 생성후에는 자식이나 속성을 바꿀 수 없다!

elemnt 랜더링 - Root DOM Node -최상

#5강

Components :리액트는 컴포넌트로 구성

계속 새로운 컴포넌트를 만듦 가능

반복적으로 사용 -> 시간,비용 --, 유지보수 용이 가능

입력: props - 출력: react element

Props : prop이 여러 개

prop: property, 속성

리액트 컴포넌트의 속성

- 중요 특징

Read-Only : 읽을 수만 있다, 값을 변경x

자바스크립트 함수의 속성

함수가 pure = 입력값을 변경X, 같은 입력값에 대해서는 항상 같은 출력값을 리턴

모든 리액트 컴포넌트는 그들의 props에 관해서는 pure!

props를 직접 바꿈x, 같은 props에 대해서는 항상 같은 결과

함수가 impure = 입력값을 변경

리액트 컴포넌트 = 함수와 같은 기능 수행

-추가적인 기능 존재

react.component를 상속 받음

-component의 이름 : 대문자!

-component 렌더링

컴포넌트로부터 엘리먼트를 생성

-component 합성

여러개의 컴포넌트를 합 -> 하나의 컴포넌트를

-component 추출

큰 컴포넌트에서 일부를 추출 -> 새로운 컴포넌트를 만듦

( 재사용성 향상, 개발 속도 향상)

곧바로 재사용이 가능한 형태로 추출 ++

#6강

state: 상태

in 리액트

state = 리액트 컴포넌트의 상태, 리액트 컴포넌트의 변경 가능한 데이터

->개발자가 정의, 렌더링이나 데이터 흐름에 사용되는 값만 state에 포함

하나의 자바스크립트 객체

lifecycle :생명주기

컴포넌트가 생성되는 시점과 사라지는 시점이 정해져 있음

출생: mounting / 인생: updating / 사망: Unmounting -상위 컴포넌트에서 현재 컴포넌트를 더 이상 화면에 표현하지 x 때

\*\*계속 존재X 시간의 흐름에 따라 생성 & 업데이트 .. 사라짐

#7강

hooks

함수 컴포넌트는 코드 간결, 별도로 state 정의 or 생명주기에 맞추어 코드 실행X

-> 지원 -> hook

함수 컴포넌트 -> 클래스 컴포넌트의 기능을 동일 구현 가능

**useState()**

state를 사용 위함

const

[변수명, set함수명] = useState(초기값);

**useEffect()**

Side effect를 수행 위함

세 가지 생명주기 함수와 동일한 기능을 수행 가능

useEffect(이펙트 함수, 의존성 배열);

**useMemo()**

Memoized value를 리턴

memoization: 연산량이 많은 함수의 호출 결과를 저장 -> 같은 입력 값으로 함수를 호출 -> 이전의 결과를 바로 반환 (불필요 계산--)

-> 결과값 : memoized value

if) 의존성 배열x? - 매 렌더링마다 함수가 실행됨 (의미x)

if) 의존성 배열 = 빔 ? - 컴포넌트 마운트 시에만 호출

(대부분의 경우 의존성 배열에 값을 넣음)

**useCallback()**

값이 아닌 함수를 반환

**useRef()**

Reference(특정 컴포넌트에 접근할 수 있는 객체)를 사용하기 위함

refContainer = useRef(초깃값);

내부의 데이터가 변경됨 -> 알림X

if 돔로드가 연결 혹은 분리 -> 코드를 실행?

Callback ref

- 규칙

최상위 레벨에서만 호출

훅은 컴포넌트가 렌더링 될 때마다 매번 같은 순서로 호출

리액트 함수 컴포넌트에서만 훅을 호출

일반적인 자바스크립트 함수에서 호출x

+eslint-plugin-react-hooks

훅의 규칙을 따르도록 강제해주는 플러그

자동으로 경고 표시 해주어 고칠 방법 제안

직접 훅을 만들 경우

-> 여러 컴포넌트에서 반복적으로 사용되는 로직을 훅으로 만들어 재사용

-커스텀 훅의 이름: use로 시작!!!!

#8강

Event: 사건

처리 = 핸들링

DOM - onclick을 통해 전달

사용하는 방법이 다름?

->이벤트 이름의 표기법 & 함수 전달 방식에서의 차이.

-camel Case: 카멜 표기법

첫 글자: 소문자로 시작 / 중간의 새로운 글자: 대문자

-Event Handler

특정 이벤트가 발생했을 때 이벤트 처리 함수

-Arguments :주장, 논쟁, 말다툼

함수에 주장할 내용, 함수에 전달할 데이터

-parameter

매개변수