 # 리액트란?

- React 는 UI 를 구현하는 JavaScript 라이브러리
- 웹 앱(Web App) 또는 네이티브 앱(Native App)
- 유지보수를 쉽게 , DOM 관리
- 성능최적화 쉽게
- 컴포넌트에 집중
- 대부분 공식 라이브러리가 없음 (높은 자유도)
- 자바스크립트 친화적 es6 기반로 배우기가 쉽다

 # 앵귤러

- 자체 내장된 기능이 많음
- 다양한 공식 라이브러리가 존재
- TypeScript 가 거의 강제적
- 성숙하나 인지도 측면에서는 성장 단계
- 배우기가 어렵다
- **Directive** 사용

```
<ul class="heroes">
  <li *ngFor="let item in list">
    [class.selected]="item === selectedItem" (click)="onSelect(item)"
  </li>
</ul>
```

 # 뷰

- 사용하기 쉬움 (리액트와 앵귤러보다)
- 웹팩없이 사용가능
- HTML 을 템플릿처럼 활용
- 앵귤러와 비슷한 구조 (ngFor -> v-for)
- **Directive** 사용

```
<ol>
  <li v-for="item in list">{{ item }}</li>
</ol>
```

 # Svelte

Svelte 는 빠른 웹 어플리케이션을 구축하기 위한 도구

react 진도

1. 리액트에 필요한 자바스크립트
2. 리액트 작성법 (state , props)
3. hooks - useState / useRef / useEffect
4. 스타일 :sass , styeld-component
5. 리액트에서 ajax 요청방법 (axios / fetch)
6. 성능관련 : useMemo / useCallback
7. 상태분리 useReducer
8. Single Page Application: spa - router
9. 상태관리 (context , redux)

 # react 에 필요한 es6(ecmascript) 정리하기

1. let / const

let 은 변수에 재할당이 가능하지만, const 는 변수 재선언, 재할당 모두 불가능 / {} scope

2. 템플릿 리터럴(Template literal) 새로운 문자열 표기법

1) backtick (`) 2) 변수 처리는 ${변수}

3. 삼항연산자와 &&연산자 , ||연산자

조건 ? true 일때 : false 일때 같다 === / 다르다 !== (react 에서) true && 결과 / (false , null , undefined) || 결과

4. 화살표 함수

function(){} -> () => { ... } # 매개변수가 없을 경우 function(x){} -> (x) => { ... } # 매개변수가 한 개인 경우, 소괄호를 생략 function(x,y){} -> (x, y) => { ... } # 매개변수가 여러 개인 경우, 소괄호를 생략할 수 없다.

#. 배열.메서드() 중 불변성 유지

5. push : 배열 뒷부분에 값을 삽입 (배열의 값이 변경된다)

let arr = [1, 2, 3, 4]; / arr.push(5);

6. concat : 다수의 배열을 합치고 병합된 배열의 사본을 반환 **기존의 배열은 건드리지 않음 ** 불변성유지

const arr = [0, 1, 2]; const arr1 = arr.concat(3); // 결과 [0, 1, 2, 3]

7. slice

slice(startIndex, endIndex) 배열의 startIndex 부터 endIndex 까지(endIndex 는 불포함)에 대한 shallow copy 를 새로운 배열 객체로 반환 **기존의 배열은 건드리지 않음 **

8. map (반복문)

```
배열.map((요소, 인덱스 ) => {
    return 요소
});
```

8. filter

```
filter 도 map 함수와 마찬가지로 콜백함수를 인자로 받는데 모든 원소를 한번씩 돌리면서 콜백함수의 몸체부분에서
true 를 반환하는 원소들만 걸러줌  결과 배열
const newArr = arr.filter( xx => 조건 )
```

9. find (findIndex)

```
형식은 filter 와 비슷 , 결과는 하나의 값
const newArr = arr.find( xx => 조건 )
```

10. 객체

```
객체를 선언 할 때에는 이렇게 { } 문자 안에 원하는 값들을 넣준다
키: 원하는 값
const dog = { name: '멍멍이', age: 2   };
dog.name , dog.age
```

11. 비구조할당 (구조분해)

```
const object = { a: 1, b: 2 , c:10 };
const { a, b , c} = object
```

12. spread... (전개 연산자)

```
const ani = ['개', '고양이', '참새'];
const ani1 = [...ani, '비둘기'];
```

13. new Set() : 동일한 값을 하나로 만들어줌

14. reduce

```
배열.reduce((누적값, 현잿값, 인덱스, 요소) => {
return 결과
}, 초기값)  초기값 생략하면 1
```

15. forEach

```
배열.forEach( (요소) => { } )
```

16. 검색에 필요한 명령어

```
string.indexOf(찾을문자열)
toLowerCase() 소문자변환
정규표현식을 사용한 match() 함수
```