# 📌 React 사전 문법 개념 정리

## 🚀 ES6 (ECMAScript 6)란?

### 📖 개념

- **ES6(ECMAScript 2015)** 는 JavaScript의 최신 문법을 포함한 표준 스펙입니다.
- **React와 함께 사용할 경우 코드의 가독성과 유지보수성을 높이는 데 필수적**입니다.

### ✅ ES6 주요 기능

1. **let & const** → 블록 스코프 변수를 선언하는 키워드
2. **템플릿 리터럴** → 백틱(`)을 사용한 문자열 표현
3. **화살표 함수 (Arrow Function)** → 짧고 간결한 함수 표현식
4. **비구조화 할당 (Destructuring Assignment)** → 객체/배열에서 값 추출
5. **Spread & Rest 연산자** → 객체와 배열 다루기
6. **Promise & async/await** → 비동기 프로그래밍 개선

✅ ES6를 이해하면 React 코드 작성이 더욱 쉬워집니다.

---

## 🚀 1. 문자열 표현식 (String Template Literals)

### 📖 개념

- 템플릿 리터럴(Template Literals)을 사용하면 **문자열과 변수를 쉽게 조합**할 수 있습니다.
- 백틱(`)과 ${} 문법을 사용하여 표현 가능`

### 📌 사용 예시

```
const name = "John";
const age = 25;
console.log(`안녕하세요, 제 이름은 ${name}이고 나이는 ${age}살입니다.`);
```

✅ 기존 + 연산자보다 가독성이 뛰어나며, 여러 줄 문자열도 쉽게 표현 가능

### 📌 React에서 활용 예시

```
const user = { name: "Alice", age: 30 };
return <h1>{`안녕하세요, ${user.name}님!`}</h1>;
```

✅ JSX에서 문자열과 변수를 조합하여 동적인 UI를 만들 때 사용됨

---

## ⚡ 2. 화살표 함수 (Arrow Function)

### 📖 개념

- **기존의 function 표현식을 더 간결하게 작성할 수 있는 ES6 문법**
- `this` 바인딩 문제가 없고, 짧고 직관적인 문법 제공

### 📌 기존 함수 표현식

```
function greet(name) {
  return `Hello, ${name}!`;
}
console.log(greet("Alice"));
```

### 📌 화살표 함수(Arrow Function) 적용

```
const greet = name => `Hello, ${name}!`;
console.log(greet("Alice"));
```

✅ **함수의 길이를 줄이고 가독성을 높일 수 있음**

### 📌 React에서 활용 예시

```
const Button = ({ onClick }) => (
  <button onClick={onClick}>Click me</button>
);
```

✅ React의 이벤트 핸들러를 정의할 때 유용하게 사용됨

---

## 💡 3. Spread 연산자 (`...`)

### 📖 개념

- 객체나 배열을 복사하거나 새로운 값을 추가할 때 사용
- 불변성을 유지하며 데이터를 변경하는 데 유용

### 📌 배열에서 사용

```
const numbers = [1, 2, 3];
const newNumbers = [...numbers, 4, 5];
console.log(newNumbers); // [1, 2, 3, 4, 5]
```

### 📌 객체에서 사용

```
const user = { name: "Alice", age: 25 };
const updatedUser = { ...user, age: 30 };
console.log(updatedUser); // { name: "Alice", age: 30 }
```

### 📌 React에서 활용 예시

```
const user = { name: "Alice", age: 25 };
const newUser = { ...user, location: "Seoul" };
console.log(newUser); // { name: "Alice", age: 25, location: "Seoul" }
```

✅ React의 `state`를 업데이트할 때 `spread` 연산자를 활용하여 **불변성 유지**

---

## 🔥 4. Rest 매개변수 (`...args`) + 비구조화할당

### 📖 개념

- **함수에서 여러 개의 인수를 배열로 받을 때 사용**

### 📌 사용 예시

```
function sum(...numbers) {
  return numbers.reduce((acc, num) => acc + num, 0);
}
console.log(sum(1, 2, 3, 4, 5)); // 15
```

✅ 매개변수 개수를 모를 때 유용하게 활용 가능

---

# 🛠 5. `filter` 메서드 (배열 필터링)

### 📖 개념

- **배열에서 특정 조건을 만족하는 요소들만 필터링하여 새로운 배열 생성**

### 📌 사용 예시

```
const numbers = [1, 2, 3, 4, 5];
const evenNumbers = numbers.filter(num => num % 2 === 0);
console.log(evenNumbers); // [2, 4]
```

### 📌 React에서 활용 예시

```
const users = [
  { id: 1, name: "Alice", active: true },
  { id: 2, name: "Bob", active: false },
];
const activeUsers = users.filter(user => user.active);
console.log(activeUsers); // [{ id: 1, name: "Alice", active: true }]
```

✅ 리스트 데이터를 필터링하여 화면에 보여줄 때 유용함

---

# 💡 6. `map` 메서드 (배열 변환)

### 📖 개념

- **배열의 각 요소를 변환하여 새로운 배열을 생성**

### 📌 사용 예시

```
const numbers = [1, 2, 3, 4];
const squaredNumbers = numbers.map(num => num ** 2);
console.log(squaredNumbers); // [1, 4, 9, 16]
```

### 📌 React에서 활용 예시 (리스트 렌더링)

```
const users = ["Alice", "Bob", "Charlie"];
return (
  <ul>
    {users.map(user => (
      <li key={user}>{user}</li>
    ))}
  </ul>
);
```

✅ React에서 리스트 데이터를 UI로 변환할 때 필수적인 개념

---

## ⚡ 7. Promise

### 📖 개념

- **비동기 작업을 처리할 때 사용하는 객체**
- `resolve()` 또는 `reject()`를 통해 성공 또는 실패를 반환

### 📌 사용 예시

```
const fetchData = new Promise((resolve, reject) => {
  setTimeout(() => {
    resolve("데이터 로드 완료");
  }, 2000);
});

fetchData.then(data => console.log(data)); // "데이터 로드 완료" (2초 후)
```

✅ 서버 요청, 파일 읽기 등 시간이 걸리는 작업을 처리할 때 사용됨

---

## 🔥 8. `async` & `await`

### 📖 개념

- **비동기 코드를 더 쉽게 작성할 수 있도록 도와주는 문법**

### 📌 사용 예시

```
async function fetchData() {
  const response = await fetch("https://jsonplaceholder.typicode.com/posts/1");
  const data = await response.json();
  console.log(data);
}
fetchData();
```

✅ `await` 키워드를 사용하면 비동기 코드를 동기적인 코드처럼 작성할 수 있음

---

## 💡 9. 비구조화 할당 (Destructuring Assignment)

### 📖 개념

- **객체 또는 배열의 값을 변수에 쉽게 할당하는 문법**
- React에서 props와 state를 다룰 때 필수적으로 사용됨

### 📌 객체 비구조화 할당

```
const user = { name: "Alice", age: 25 };

// 기존 방식
const name = user.name;
const age = user.age;

// 비구조화 할당 방식
const { name, age } = user;
console.log(name, age); // Alice 25
```

✅ `const { name, age } = user;`를 사용하면 객체의 속성을 손쉽게 변수에 할당 가능

### 📌 React에서 활용 예시 (props 비구조화 할당)

```
function Greeting({ name }) {
  return <h1>안녕하세요, {name}님!</h1>;
}
```

✅ `{ name }`을 직접 받아 props에서 값을 추출하는 방식으로 가독성을 높일 수 있음

### 📌 배열 비구조화 할당

```
const numbers = [1, 2, 3];
const [first, second, third] = numbers;
console.log(first, second, third); // 1 2 3
```

✅ 배열의 요소를 변수로 빠르게 할당 가능

### 📌 React에서 활용 예시 (useState 활용)

```
const [count, setCount] = useState(0);
```

✅ `useState`에서 반환된 배열을 비구조화 할당을 통해 쉽게 사용할 수 있음

---

## 🎯 10. 정리

✅ **문자열 표현식** → 백틱을 활용하여 가독성 높은 문자열 생성 가능

✅ **화살표 함수 (Arrow Function)** → 짧고 직관적인 함수 표현 방식

✅ **Spread 연산자** → 객체와 배열을 복사하거나 업데이트할 때 필수

✅ **Rest 매개변수** → 함수의 인자 개수를 동적으로 받을 때 유용

✅ **filter()** → 리스트 데이터를 특정 조건에 맞게 필터링할 때 사용

✅ **map()** → 배열을 변환하여 UI 리스트를 만들 때 필수

✅ **Promise & async/await** → 비동기 작업을 처리할 때 중요한 개념

✅ **비구조화 할당** → 객체 및 배열에서 값을 쉽게 추출하여 코드 가독성 향상