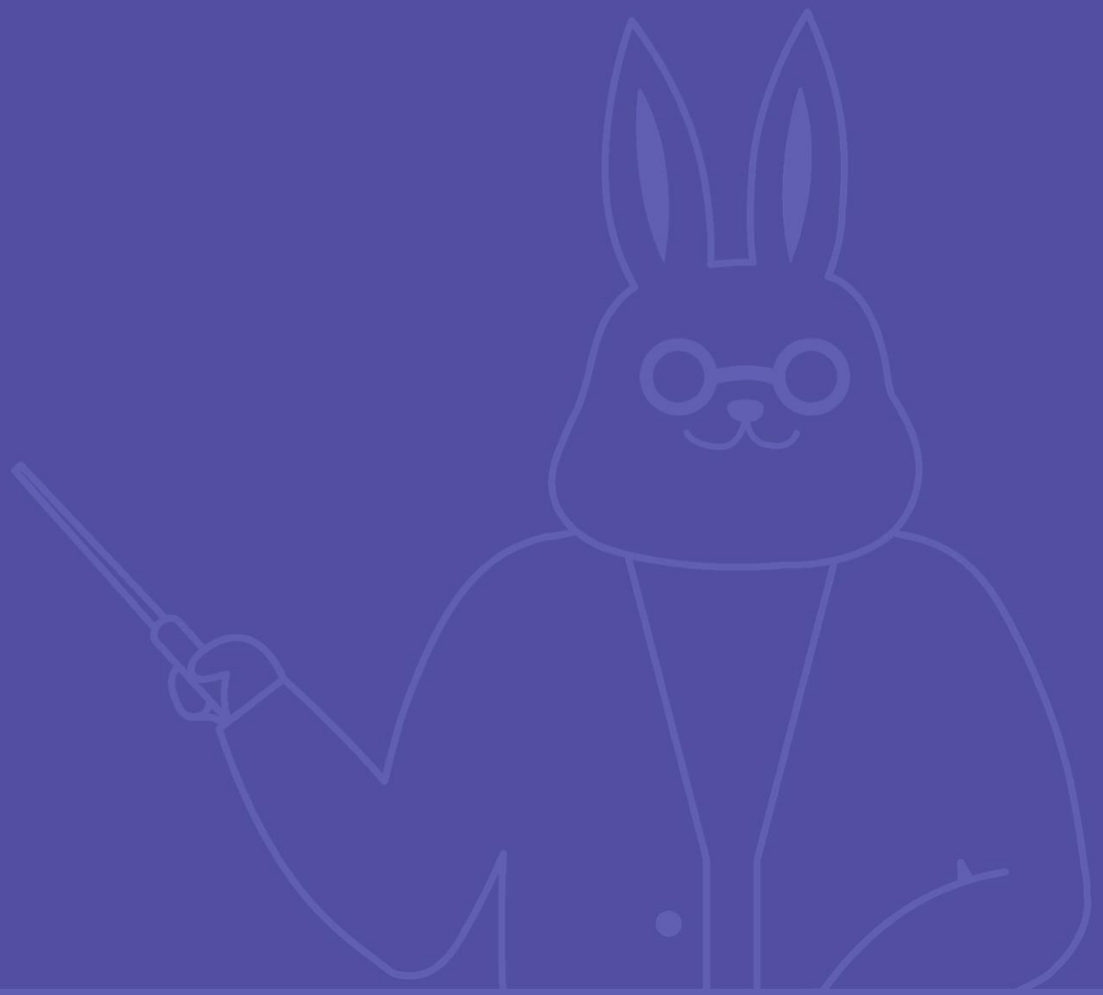
React 기초 I

02 최신 JavaScript 문법 알아보기



var →
const & let

Array 메소드
(forEach, map, filter등)

Arrow function
(() => {})

Destructuring assignment
(const {a, b, c} = obj)

Shorthand property names
(const obj = {a, b, c})

Spread Syntax
(const a = {...b, ...c})

Template literals
(const a = `hello ${name}!`)

Optional chaining
(const a = obj?.b?.c)

var → const & let

코드

```
//! Past
var a;
var b = 1;

//* Now
const c = 2;
let d = 3;
d = 4;

//! Not available
const e = 5;
e = 6;
```

const는 한 번 선언하면 값을 바꿀 수 없는 상수,

let은 선언과 변경이 자유로운 변수를 의미합니다.

또한 const로 선언된 변수는 같은 스코프(중괄호) 내에서 중복된 이름을 가질 수 없습니다.

Array 메소드(forEach)

Past

```
var a = [0, 1, 2, 3, 4, 5];
for (var i = 0; i < a.length; i++) {
  var item = a[i];
  console.log(item);
}
```

Now

```
const b = [0, 1, 2, 3, 4, 5];
b.forEach(function (item) {
  console.log(item);
});
```

Array.forEach는 배열의 요소를 이용해 순차적으로 함수를 실행하는 메서드(method)입니다.

Array.forEach에 실행할 콜백 함수는 현재 값, 인덱스, forEach를 호출한 배열을 매개변수로 받습니다.

또한 함수 내에서 따로 return을 할 필요가 없습니다.

Array 메소드(map)

Past

```
var c = [0, 1, 2, 3, 4, 5];
var newC = [];
for (var i = 0; i < a.length; i++) {
  var item = c[i];
  newC.push(item * 2);
}
```

Now

```
const d = [0, 1, 2, 3, 4, 5];
const newD = d.map(function (item) {
  return item * 2;
});
```

Array.map은 배열의 요소를 이용해 순차적으로 함수를 실행하여 새로운 배열을 반환하는 메서드입니다.

Array.map에 실행할 콜백 함수는 현재 값, 인덱스, map을 호출한 배열을 매개변수로 받습니다.

함수내에서 반드시 새로운 값을 return을 해주어야 합니다.

Array 메소드(filter)

Past

```
var c = [0, 1, 2, 3, 4, 5];
var newC = [];
for (var i = 0; i < a.length; i++) {
  var item = c[i];
  if (item > 3) {
    newC.push(item);
  }
}
```

Now

```
const d = [0, 1, 2, 3, 4, 5];
const newD = d.filter(function (item)
{
  return item > 3;
});
```

Array.filter는 배열의 요소를 이용해 순차적으로 함수를 실행하여 조건을 통과하는 요소를 모아 새로운 배열로 반환하는 메서드입니다.

Array.filter에 실행할 콜백 함수는 현재 값, 인덱스, map을 호출한 배열을 매개변수로 받습니다.

함수 내에서 false를 return할 경우 새로운 배열에서 제외되고 true를 return할 경우 새로운 배열에 추가됩니다.

Arrow function

코드

```
//* arrow function
const c = (x, y) => {
  console.log(x, y);
};
c(5, 6);

const d = (x, y) => console.log(x, y);
d(7, 8);
```

Arrow Function(화살표 함수)은 function 표현보다 구문이 짧은 함수 표현입니다.

function이라는 키워드를 생략하고 매개변수를 받은 뒤 => 를 써주는 형태입니다.

중괄호({})를 열어 로직을 작성할 수 있으며 return 값만 존재하는 짧은 함수의 경우 중괄호와 return을 생략하고 바로 return할 값을 입력할 수도 있습니다.

Destructuring assignment(Object)

Past

```
var a = { x: 1, y: 2, c: 3 };
var x = a.x;
var y = a.y;
var z = a.z;
```

Now

```
const b = { i: 1, j: 2, k: 3 };
const { i, j, k } = b;
```

Destructuring Assignment(구조 분해 할당)은 객체나 배열을 해체하여 개별 변수에 담을 수 있게 하는 표현식입니다.

기존에는 객체에 담긴 값을 각각 새로운 변수로 선언하기 위해 왼쪽 예시와 같이 작성하였지만 Destructuring Assignment 표현을 이용하여 더 간편하게 코드를 작성할 수 있습니다.

Destructuring assignment(Array)

Past

```
var c = [1, 2, 3];
var c0 = c[0];
var c1 = c[1];
var c2 = c[2];
```

Now

```
const d = [1, 2, 3];
const [d0, d1, d2] = d;
```

Destructuring Assignment(구조 분해 할당)은 객체나 배열을 해체하여 개별 변수에 담을 수 있게 하는 표현식입니다.

기존에는 배열에 담긴 값을 각각 a, b에 담기 위해 왼쪽 예시와 같이 표현하였지만
Destructuring Assignment 표현을 이용하여 더 간편하게 코드를 작성할 수 있습니다.

Shorthand property names

코드

```
const username = "김레이서";
const age = 21;
const school = "엘리스";

//! Past
var person1 = { username: username, age: age, school: school };

//* Now
const person2 = { username, age, school };
```

Shorthand property names(단축 속성명)을 이용해 새로운 객체 선언을 간편하게 할 수 있습니다. 새로 선언하는 object에 key값과 동일한 변수명을 가진 변수를 할당할 경우 value 값을 생략해서 적을 수 있습니다.

Spread Syntax(배열)

코드

```
const numbers = [1, 2, 3];
//* 1.
function getSum(...n) {
  let sum = 0;
  n.forEach((item) => {
    sum += item;
  });
  return sum;
}
//* 2.
getSum(...numbers);
//* 3.
const newNumbers = [0, ...numbers, 4, 5, 6];
```

Spread Syntax(전개 구문)은 배열이나 객체를 전개할 때 사용합니다.

기본적으로 배열이나 객체에 ...을 붙여 사용합니다.

함수 호출 및 선언, 배열 선언, 객체 선언 시 다양하게 활용 가능합니다.
왼쪽의 예시는 Spread Syntax를 배열에 적용한 코드입니다.

Spread Syntax(객체)

코드

```
const user = { name: "김레이서", age: 23, school: "엘리스" };
//* 4.
const newUser = { ...user, grade: 3, age: 24 };
```

위의 예시는 Spread Syntax를 객체에 적용한 코드입니다.
참고로 두 객체를 합성할 때 겹치는 key가 있을 경우 **나중에 오는 값**이 들어갑니다.

팁: 이해가 어렵다면 배열의 대괄호([]), 객체의 중괄호({})를 없애준다고 생각해보세요.

Template literals

코드

```
//! Past
const text1 = "Hello " + name;

//* Now
const text2 = `Hello ${name}`;
```

Template literals(템플릿 리터럴)은 표현식을 허용하는 문자열 리터럴입니다.

기본적으로 ` `(back quote)로 감싸 문자열을 표현합니다.

문자열 내에 표현식을 사용하기 위해서는 ${ }(달러와 중괄호)로 표기합니다.

Optional chaining

코드

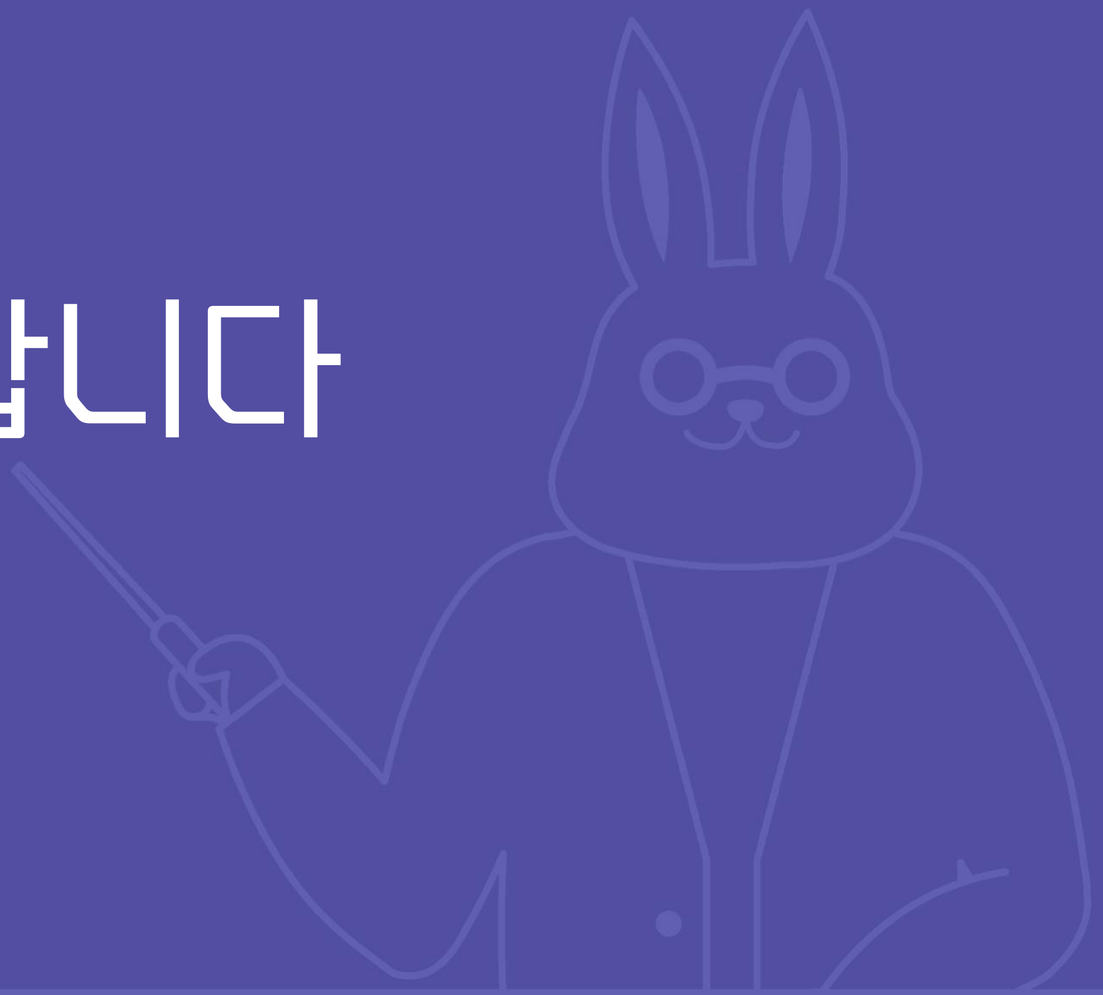
```
//! Past
var x;
if(a && a.b && a.b.c) {
  x = a.b.c;
}

//* Now
const y = a?.b?.c;
```

Optional chaining 연산자는 객체나 변수에 연결된 다른 속성 참조할 때 유효한 속성인 지 검사하지 않고 값을 읽을 수 있도록 해줍니다. 만약 유효한 속성이 아닐 경우 에러를 발생시키지 않고 `undefined`를 반환해줍니다.

배열의 경우 `array?.[index]` 와 같이 사용할 수 있습니다.

감사합니다



크레딧

/* elice */

코스 매니저

콘텐츠 제작자
마로

강사
마로

감수자