1~2. 리액트 학습 준비 (실습)

Prof. Seunghyun Park (sp@hansung.ac.kr)

Division of Computer Engineering

한성대학교 HANSUNG UNIVERSITY

학습 목표: 2. 리액트를 위한 자바스크립트

- ES2015+
  - 변수와 상수
  - 함수
  - 화살표 함수

실습1: 변수와 상수 – var, let, const 비교하기 (ch02-01-1~3)

- 문자열을 매개변수로 전달받고, 해당 문자열을 console에 출력하는 함수
  - 문제1. 할당 없이 선언
  - 문제2. 선언 및 값 할당 후에 (1) 값을 변경하고, (2) 재선언
  - 문제3. 키워드 없이 할당 후 재선언
  - 문제4. 블록 안에서 선언하고, 블록 밖에서 참조

실습1-1: var

문제1. 할당 없이 선언
문제2. 선언 및 값 할당 후에 (1) 값을 변경하고, (2) 재선언
문제3. 키워드 없이 할당 후 재선언
문제4. 블록 안에서 선언하고, 블록 밖에서 참조

```
/* ch02-01-1.html */

// 할당 없이 선언
var i1;
console.log(i1);
```

```
/* ch02-01-1.html */

// 선언 및 값 할당
var i2 = 10;
console.log(i2);

// 값 할당 (변경)
i2 = 20;
console.log(i2);

// 변수 재선언
var i2 = 30;
console.log(i2);
```

```
/* ch02-01-1.html */

// var 없이 선언
i3 = 40;
console.log(i3);

변수 재선언
var i3 = 50;
console.log(i3);
```

```
/* ch02-01-1.html */

// 블록 안에서 선언
if (true) {
  var i4 = 60;
}

// 블록 밖에서 참조
console.log(i4);
```

실습1-2: let

문제1. 할당 없이 선언
문제2. 선언 및 값 할당 후에 (1) 값을 변경하고, (2) 재선언
문제3. 키워드 없이 할당 후 재선언
문제4. 블록 안에서 선언하고, 블록 밖에서 참조

```
/* ch02-01-2.html */

// 할당 없이 선언
let i1;
console.log(i1);
```

호이스팅 허용여부 확인

```
/* ch02-01-2.html */

// 선언과 값 할당
let i2 = 10;
console.log(i2);

// 변수 값 변경
i2 = 20;
console.log(i2);

// 변수 재선언
let i2 = 30;
console.log(i2);
```

```
/* ch02-01-2.html */

// var/let 없이 선언
i3 = 40;
console.log(i3);

// 변수 재선언
let i3 = 50;
console.log(i3);
```

```
/* ch02-01-2.html */

if (true) {
  // 블록 안에서 선언
  let i4 = 60;
}

// 블록 밖에서 참조
console.log(i4);
```

실습1-3: const

문제1. 할당 없이 선언
문제2. 선언 및 값 할당 후에 (1) 값을 변경하고, (2) 재선언
문제3. 키워드 없이 할당 후 재선언
문제4. 블록 안에서 선언하고, 블록 밖에서 참조

```
/* ch02-01-3.html */

// 할당 없이 선언
const i1;
console.log(i1);
```

호이스팅 허용여부 확인

```
/* ch02-01-3.html */

// 선언과 값 할당
const i2 = 10;
console.log(i2);

// 상수 값 변경
i2 = 20;
console.log(i2);

// 상수 재선언
const i2 = 30;
console.log(i2);
```

```
/* ch02-01-3.html */

// var/const 없이 선언
i3 = 40;

// 상수 재선언
const i3 = 50;
console.log(i3);
```

```
/* ch02-01-3.html */

if (true) {
  // 블록 안에서 선언
  const i4 = 60;
}

// 블록 밖에서 참조
console.log(i4);
```

실습2: 템플릿 문자열

```
/* ch02-02-1.html */

let i1 = 10;
let i2 = 20;
let i3 = 30;
```

```
10 + 20 + 30 = 60
10, 20, 30을 모두 더하면 60입니다.
```

일반 문자열로 출력

```
/* ch02-02-1.html */
```

템플릿 문자열로 출력

```
/* ch02-02-2.html */
```

실습3: 함수

- 문자열을 매개변수로 전달받고, 해당 문자열을 console에 출력하는 함수
  - 문제1. 함수 선언문으로 함수를 정의하고, 함수 호출하기
    - 함수명: func1
  - 문제2. 함수 표현식으로 함수를 정의하고, 함수 호출하기
    - 함수명: func2
  - 문제3. 화살표 함수로 함수를 정의하고, 함수 호출하기
    - 함수명: func3

```
/* ch02-03-func-1.html */

function func1(str){   // 함수 선언문
  console.log(str);
}

func1("Hello, func1");
```

```
Hello, func1
```

실습3-1: 함수 (1)

문제1. 함수 선언문으로 함수를 정의하고, 함수 호출하기 (함수명: func1)
문제2. 함수 표현식으로 함수를 정의하고, 함수 호출하기 (함수명: func2)
문제3. 화살표 함수로 함수를 정의하고, 함수 호출하기 (함수명: func3)

함수 선언문

```
/* ch02-03-func-1.html */

function func1(str){
  console.log(str);
}

func1("Hello, func1");
```

```
Hello, func1
```

함수 표현식

```
/* ch02-03-func-1.html */

func2("Hello, func2");
```

```
Hello, func2
```

화살표 함수

```
/* ch02-03-func-1.html */

func3("Hello, func3");
```

```
Hello, func3
```

실습3-2: 함수의 호이스팅 확인

문제1. 함수 선언문의 호이스팅 확인 (함수명: func4)
문제2. 함수 표현식의 호이스팅 확인 (함수명: func5)
문제3. 화살표 함수의 호이스팅 확인 (함수명: func6)

함수 선언문

```
/* ch02-03-func-2.html */

func4("Hello, func4");

function func4(str){
  console.log(str);
}
```

함수 표현식

```
/* ch02-03-func-1.html */

func5("Hello, func5");

const func5 = function(str){
  console.log(str);
}
```

화살표 함수

```
/* ch02-03-func-1.html */

func6("Hello, func6");

const func6 = (str) => {
  console.log(str);
}
```

실습4-1: 화살표 함수 (1)

- 다음 조건에 따라 화살표 함수를 구현하고, 결과를 console에 출력
  - 문제1. 매개변수를 2개 전달 받아서, 이 값을 더한 값을 반환하는 화살표 함수 구현
    - 함수명: `aFunc1`
  - 문제2. 매개변수 없이 함수 내부에서 선언한 상수 2개의 값을 더한 값을 반환하는 화살표 함수
    - 함수명: `aFunc2`
  - 문제3a. 매개변수를 1개 전달 받아서, 이 값에 200을 더한 값을 반환하는 화살표 함수 구현
    - 함수명: `aFunc3a`
  - 문제3b. 문제3a에서 매개변수의 ( ) 를 생략
    - 함수명: `aFunc3b`

실습4: 화살표 함수 (2)

- 다음 조건에 따라 화살표 함수를 구현하고, 결과를 console에 출력
  - 문제4a. 문제3a (매개변수 1개 전달 받아서, 이 값에 200을 더한 값을 반환하는 화살표 함수)의 함수 정의를 1줄로 구현
    - 함수명: `aFunc4a`
  - 문제4b. 문제4a의 함수 정의에서 `{ }`와 `return` 구문 생략하여 구현
    - 함수명: `aFunc4b`
  - 문제5. 문제4b의 화살표 함수의 호이스팅 여부 확인
    - 함수명: `aFunc5`

실습4: 화살표 함수 (1)

문제1. 매개변수를 2개 전달 받아서, 이 값을 더한 값을 반환하는 화살표 함수 구현 (함수명: aFunc1)
문제2. 매개변수 없이 함수 내부에서 선언한 상수 2개의 값을 더한 값을 반환하는 화살표 함수 (함수명: aFunc2)

```
/* ch02-04-a-func-1.html */



console.log(`aFunc1(100, 200): ${aFunc1(100, 200)}`);
```

```
aFunc1(100, 200): 300
```

```
/* ch02-04-a-func-1.html */



console.log(`aFunc1(100, 200): ${aFunc1(100, 200)}`);
```

```
aFunc2(): 300
```

실습4: 화살표 함수 (2)

문제3a. 매개변수를 1개 전달 받아서, 이 값에 200을 더한 값을 반환하는 화살표 함수 구현 (함수명: aFunc3a)
문제3b. 문제3a에서 매개변수의 ( ) 를 생략 (함수명: aFunc3b)

```
/* ch02-04-a-func-1.html */

console.log(`aFunc3a(100): ${aFunc3a(100)}`);
```

```
/* ch02-04-a-func-1.html */

console.log(`aFunc3b(101): ${aFunc3b(101)}`);
```

```
aFunc3a(100): 300
```

```
aFunc3b(101): 301
```

실습4: 화살표 함수 (3)

문제3a. 매개변수를 1개 전달 받아서, 이 값에 200을 더한 값을 반환하는 화살표 함수 구현 (함수명: aFunc3a)
문제3b. 문제3a에서 매개변수의 ( ) 를 생략 (함수명: aFunc3b)

```
/* ch02-04-a-func-1.html */

console.log(`aFunc3a(100): ${aFunc3a(100)}`);
```

```
/* ch02-04-a-func-1.html */

console.log(`aFunc3b(101): ${aFunc3b(101)}`);
```

```
aFunc3a(100): 300
```

```
aFunc3b(101): 301
```

실습4: 화살표 함수 (4)

문제4a. 문제3a (매개변수 1개 전달 받고, 200을 더한 값을 반환하는 화살표 함수)의 함수 정의를 1줄로 구현 (함수명: aFunc4a)
문제4b. 문제4a의 함수 정의에서 { }와 return 구문 생략하여 구현 (함수명: aFunc4b)
문제5. 문제4b의 화살표 함수의 호이스팅 여부 확인 (함수명: aFunc5)

```
/* ch02-04-a-func-1.html */

console.log(`aFunc4a(100): ${aFunc4a(100)}`);
```

```
aFunc4a(100): 300
```

```
/* ch02-04-a-func-1.html */

console.log(`aFunc4b(101): ${aFunc4b(101)}`);
```

```
aFunc4b(101): 301
```

```
/* ch02-04-a-func-1.html */

console.log(`aFunc5(100): ${aFunc5(100)}`);
const aFunc5 = val => val + 200;
```