3주차 (4/18)

▼ 목차

1번 실습 (가변수, 불변변수)

자바스크립트에는 총 3가지의 변수 선언 방법이 있습니다.

변수 선언

Aa Name	Scope	재정의	재선언
var	Function Scoped	변경 가능	변경 가능
let	Block Scoped	변경 가능	변경 불가능
const	Block Scoped	변경 불가능	변경 불가능

```
let x = 1;
const y = 2;
var z = 3;
console.log(x); 1
console.log(y); 2
console.log(z); 3

x = 12;
y = 43;
console.log(x); 12
console.log(y); // error

let x = 3;
const y = 8;
// both error
var z = 54;
console.log(x); // error
console.log(y); // error
console.log(z); // 54
```

💡 `const`로 선언된 배열 또는 객체는 내용을 바꿀 수 있다

```
const cars = ["Saab", "Volvo", "BMW"];
cars = ["Toyota", "Volvo", "Audi"];    // ERROR

// You can change an element:
cars[0] = "Toyota";

// You can add an element:
cars.push("Audi");
```

▼ 실습 정답 코드

```
let x = 2;
const y = 2;

x = 4;
// y = 4;

document.write(x);
document.write(y);
```

2번 실습 (import, export, export default)

개발하는 애플리케이션의 크기가 커지면 파일을 여러 개로 분리해야 하는 상황이 옵니다. 이때 분리되는 파일 각각을 '모듈(module)'이라고 부릅니다.

선언부 앞에 export 붙이기

변수나 함수, 클래스를 선언할 때 맨 앞에 `export`를 붙이면 내보내기가 가능합니다.

아래 내보내기는 모두 유효합니다.

```
// 배열 내보내기
export let name = ['Daniel', 'John', 'Doe']

// 상수 내보내기
export const CAPITAL_CITY = 'Seoul';

// 클래스 내보내기
export class User {
  constructor(name) {
    this.name = name;
  }
}`
```

선언부와 떨어진 곳에 export 붙이기

선언부와 `export`가 떨어져 있어도 내보내기가 가능합니다.

아래 예시에선 함수를 먼저 선언한 후, 마지막 줄에서 내보냅니다.

```
function sayHi(user) {
  alert(`Hello, ${user}!`);
}

function sayBye(user) {
  alert(`Bye, ${user}!`);
}

export {sayHi, sayBye}; // 두 함수를 내보냄
```

import 문법

무언갈 가져오고 싶다면 아래와 같이 이에 대한 목록을 만들어 `import {...}` 안에 적어주면 됩니다.

```
import {sayHi, sayBye} from './say.js';

sayHi('John'); // Hello, John!
sayBye('John'); // Bye, John!
```

가져올 것이 많으면 `import * as <obj>` 처럼 객체 형태로 원하는 것들을 가지고 올 수 있습니다. 예시를 살펴보겠습니다.

```
import * as say from './say.js';

say.sayHi('John');
say.sayBye('John');
```

import ‘as’

`as` 를 사용하면 이름을 바꿔서 모듈을 가져올 수 있습니다.

`sayHi` 를 `hi` 로, `sayBye` 를 `bye` 로 이름을 바꿔서 가져와 봅시다.

```
import {sayHi as hi, sayBye as bye} from './say.js';
hi('John'); // Hello, John!
bye('John'); // Bye, John!
```

Export ‘as’

`export`에도 `as`를 사용할 수 있습니다.

`sayHi`와 `sayBye`를 각각 `hi`와 `bye`로 이름을 바꿔 내보내 봅시다.

```
export {sayHi as hi, sayBye as bye};
```

이제 다른 모듈에서 이 함수들을 가져올 때 이름은 `hi`와 `bye`가 됩니다.

```
import * as say from './say.js';

say.hi('John'); // Hello, John!
say.bye('John'); // Bye, John!
```

export default

모듈은 크게 두 종류로 나뉩니다.

1. 복수의 함수가 있는 라이브러리 형태의 모듈
2. 개체 하나만 선언되어있는 모듈

모듈은 `export default`라는 특별한 문법을 지원합니다. `export default`를 사용하면 '해당 모듈엔 개체가 하나만 있다’는 사실을 명확히 나타낼 수 있습니다.

내보내고자 하는 개체 앞에 `export default`를 붙여봅시다.

```
export default function User(name) { // export 옆에 'default'를 추가해보았습니다.
 console.log(name);
}

// or

function User(name) {
  console.log(name);
}

export default User;
```

파일 하나엔 대개 `export default`가 하나만 있습니다.

이렇게 `default`를 붙여서 모듈을 내보내면 중괄호 `{}` 없이 모듈을 가져올 수 있습니다.

```
import User from './user.js'; // {User}가 아닌 User로 클래스를 가져왔습니다.

User("John")
```

▼ 실습 정답 코드

```
import euroToWon from "./exchange/euro.js";
import { dollarToWon } from "./exchange/dollar.js";
import { yuanToWon, yenToWon } from "./exchange/asia.js";

console.log(euroToWon(100));
console.log(dollarToWon(200));
console.log(yuanToWon(300));
console.log(yenToWon(400));
```

3번 실습 (Template Literal)

백틱엔 특별한 기능이 있습니다. 표현식을 `${…}`로 감싸고 이를 백틱으로 감싼 문자열 중간에 넣어주면 해당 표현식을 문자열 중간에 쉽게 삽입할 수 있죠. 이런 방식을 템플릿 리터럴(template literal)이라고 부릅니다.

```
function sum(a, b) {
  return a + b;
}

alert(`1 + 2 = ${sum(1, 2)}.`); // 1 + 2 = 3.
```

백틱을 사용하면 문자열을 여러 줄에 걸쳐 작성할 수도 있습니다.

```
let guestList = `손님:
 * John
 * Pete
 * Mary
`;

alert(guestList); // 손님 리스트를 여러 줄에 걸쳐 작성함`
```

▼ 실습 정답 코드

```
let learn = '자바스크립트';
let year = 3;

let sentence = `나는 ${learn}를 ${year}년째 공부중입니다.`;
document.write(sentence);
```

4번 실습 (Arrow function)

```
let func = (arg1, arg2, ...argN) => expression;
```

```
let func = function(arg1, arg2, ...argN) {
  return expression;
};
```

▼ 실습 정답 코드

```
const x = (x, y) => x + y;

document.write(x(5, 5));
```

5번 실습 (Class)

Class 생성하기

```
class Person {
  // ...
}
let Daniel = new Person();
console.log(Daniel);
```

Class 초기값 설정해주기

Constructor(생성자)를 이용하면 class 객체의 초기 값을 설정해 줄 수 있습니다.

```
class Person {
    constructor (name, age, city) {
        this.name = name;
        this.age = age;
        this.city = city;
    }
}

let Daniel = new Person('Daniel','25','Seoul');

console.log(Daniel);
```

Class 매서드 사용하기 (기능)

class에서 설정한 초기값을 접근해 특정 기능을 하는 메서드를 만드는 것도 가능하다.

class안에 function 형식으로 만들어준 뒤 해당 메서드를 호출하기만 하면 된다.

```
class Person {
  constructor (name,age, city) {
    this.name = name;
    this.age = age;
    this.city = city;
  }

  moveCity(movedCity) {
    return this.city = movedCity;
  }
}

let Daniel = new Person('Daniel','25','Seoul');
console.log(Daniel.moveCity("Suwon"));
```

▼ 실습 정답 코드

```
class Car {
  constructor(name, year) {
    this.name = name;
    this.year = year;
  }
}

let myCar1 = new Car("Ford", 2014);
let myCar2 = new Car("Audi", 2019);

document.write(myCar1.name);
document.write(myCar1.year);

document.write(myCar2.name);
document.write(myCar2.year);
```

6번, 7번, 8번 실습 (forEach, map, reduce)

forEach()

`forEach()`는 주어진 `callback`을 배열에 있는 각 요소에 대해 오름차순으로 한 번씩 실행합니다.

```
let names = ["Daniel", "John", "Doe"];

names.forEach((e) => console.log(e))
```

```
let names = ["Daniel", "John", "Doe"];

function callNames(names) {
  console.log(names)
}

names.forEach(names.forEach(callNames))
```

map()

`map()` 메서드는 배열 내의 모든 요소 각각에 대하여 주어진 함수를 호출한 결과를 모아 새로운 배열을 반환합니다.

```
const array1 = [1, 4, 9, 16];

// pass a function to map
const map1 = array1.map(x => x * 2);

console.log(map1);
// expected output: Array [2, 8, 18, 32]
```

reduce()

`reduce()` 메서드는 배열의 각 요소에 대해 주어진 **리듀서**(reducer) 함수를 실행하고, 하나의 결과값을 반환합니다.

`array .reduce( function(total, currentValue, currentIndex, arr), initialValue )`

```
const arr = [1, 2, 3, 4, 5];
const result = arr.reduce((acc, cur) => { return acc += cur; });
console.log(result);  // 15
```

▼ 6번 실습 정답 코드

```
let fruits = ["apple", "orange", "cherry"];

function myFunction(item, index) {
  document.write(index + ":" + item + "<br>");
}

fruits.forEach(myFunction);
```

▼ 7번 실습 정답 코드

```
let persons = [
  { firstname: "Malcom", lastname: "Reynolds" },
  { firstname: "Kaylee", lastname: "Frye" },
  { firstname: "Jayne", lastname: "Cobb" },
];

function getFullName(item) {
  let fullname = item.firstname + " " + item.lastname;
```

```
  return fullname;
}

document.write(persons.map(getFullName));

document.write("<br>");
let numbers = [4, 9, 16, 25];
let x = numbers.map(Math.sqrt);
document.write(x);
```

▼ 8번 실습 정답 코드

```
var numbers = [175, 50, 25];

document.write(numbers.reduce(myFunc));

function myFunc(total, num) {
  return total - num;
}
```