삼육대학교 SW융합교육원 김진호

자바스크립트 기초교육

<class>
</body>
<head>
<title>

05. function (함수)

5-1. function definition

5-2. function call

5-3. arrow function

5-4. various type of functions

5-5. first class object

5. function

5-1. function definition (함수 정의)

5-1-1. function declaration (함수 선언문)

```
function hello(name) {
    return `${name}님 안녕하세요!`;
}
```

- 함수 선언문에서는 함수의 이름을 생략할 수 없다.

5. function

5-1. function definition (함수 정의)

5-1-2. function expression (함수 표현식)

```
var hello = function (name) {
    return `${name}님 안녕하세요!`;
}
```

- 함수 표현식에서는 함수명을 생략할 수 있다.

5. function

5-1. function definition (함수 정의)

5-1-3. function hoisting (함수 호이스팅)

```
// 함수 참조
console.log(hello);
console.log(hi);

// 함수 호출
console.log(hello('홍길동'));
// TypeError: hi is not a function
// console.log(hi('홍길동'));

// 함수 선언문
function hello(name) {
    return `${name}님 안녕하세요!`;
}

// 함수 표현식
var hi = function(name) {
    return `${name} 안녕~`;
}
```

함수 선언문은 런타임 이전 자바스크립트 엔진에 의해 먼저 실행 된다. 따라서 함수 선언문 이전에 함수를 참조할 수 있으며 호출할 수도 있다. 함수 선언문이 코드의 선두로 끌어 올려진 것처럼 동작하는 자바스크립트 고유의 특징을 함수 호이스팅이라고 한다.

변수 할당문의 값은 할당문이 실행되는 시점, 런타임에 평가되므로 함수 표현식의 함수 리터럴도 할당문이 실행되는 시점에 평가되어 함수 객체가 된다.

함수 표현식으로 정의한 함수는 반드시 함수 표현식 이후에 참조 또는 호출해야 한다.

5. function

5-2. function call (함수 호출)

5-2-1. parameter and arguments

- **매개변수는 함수 몸체 내부에서만 참조할 수 있다.**
- **함수는 매개변수의 개수와 인수의 개수가 일치하는지 체크하지 않는다.**
  - 인수가 부족해서 할당되지 않은 매개변수의 값은 undefined이다.
  - 매개변수보다 인수가 더 많은 경우 초과된 인수는 무시된다.
- **모든 인수는 암묵적으로 arguments 객체의 프로퍼티로 보관된다.**
  - 가변인자 함수 구현 시 유용하게 사용 된다.
- **인수를 전달하지 않았을 경우, undefined를 전달하였을 경우 ES6에서 도입된 매개변수 기본값을 사용할 수 있다.**
- **매개변수의 최대 개수에 대해 명시적인 제한은 없다.**
  - 하지만 이상적인 함수는 한 가지 일만 해야 하며 가급적 작게 만들어야 하므로 최대 3개 이상을 넘지 않는 것을 권장한다.

5. function

5-2. function call (함수 호출)

5-2-2. return

- 반환문은 리턴 키워드 뒤에 오는 값을 반환한다.
- 반환 값을 명시적으로 지정하지 않으면 undefined가 반환된다.
- 반환문을 생략할 수도 있다. 이때도 암묵적으로 undefined를 반환한다.

5. function

5-3. arrow function (화살표 함수)

5-3-1. arrow function basic syntax (화살표 함수 기본 문법)

- ES6에서 도입된 화살표 함수는 function 키워드 대신 화살표를 사용해 좀 더 간략한 방법으로 함수를 선언할 수 있다.
- 화살표 함수는 항상 익명 함수로 정의한다.
- 본문이 한 줄인 함수를 작성할 때 유용하다.

5. function

5-3. arrow function (화살표 함수)

5-3-1. arrow function basic syntax (화살표 함수 기본 문법)

```
var message;

// 기존 function 정의
message = function () {
    return "Hello World!";
};

console.log(message());

// function 키워드 생략 가능
message = () => {
    return "Arrow Function!";
};

console.log(message());

// 명령문이 하나만 있는 경우 중괄호 생략 가능
// 이 때 함수 몸체 내부의 문이 값으로 평가 될 수 있는 표현식인 문이라면 암묵적으로 반환 된다.
// return 키워드 생략 가능
message = () => "Arrow Functions are Simple!";

console.log(message());

// 매개변수가 있을 경우
message = (val1, val2) => "Arrow " + val1 + val2;

console.log(message('Function', '!'));

// 매개변수가 하나면 소괄호 생략 가능
// 매개변수가 없거나 여러 개일 경우 생략 불가하다
message = val => "Arrow " + val;

console.log(message('Functions are GOOD!!!'));
```

5. function

5-4. various type of functions (다양한 타입의 함수)

5-4-1. immediately invoked function expression (즉시 실행 함수)

```
(function() {
    console.log('익명 즉시 실행 함수! 함수 정의와 동시에 호출!');
})();
```

- 함수 정의와 동시에 즉시 호출되는 함수로 단 한 번만 호출 되며 다시 호출할 수 없다.

5. function

5-4. various type of functions (다양한 타입의 함수)

5-4-2. recursive function (재귀 함수)

```
function factorial(n) {
    // n이 1 이하일 때 재귀 호출을 멈춘다.
    if(n <= 1) return 1;
    // 재귀 호출
    return n * factorial(n - 1);
}
```

- 함수가 자기 자신을 호출하는 것을 재귀 호출이라고 한다.
- 재귀 호출을 수행하는 함수인 재귀 함수는 반복되는 처리를 위해 사용한다.
- 재귀 함수는 반복 되는 처리를 반복문 없이 구현할 수 있다는 장점이 있지만 무한 반복에 빠질 위험이 있고 이로 인해 스택 오버플로 에러를 발생시킬 수 있으므로 주의한다.
  - 반복문보다 재귀 함수 사용이 더 직관적으로 이해하기 쉬울 때만 한정적으로 사용하는 것이 바람직하다.

5. function

5-4. various type of functions (다양한 타입의 함수)

5-4-3. nested function (중첩 함수)

```
function outer() {
    var outerVal = '외부 함수';

    function inner() {
        var innerVal = '내부 함수';
        // 외부 함수의 변수를 참조할 수 있다.
        console.log(outerVal, innerVal);
    }

    inner();
}

outer();
```

- 함수 내부에 정의된 함수를 중첩 함수 또는 내부 함수라고 한다.
- 중첩 함수를 포함하는 함수는 외부 함수라고 한다.
- 일반적으로 중첩 함수는 자신을 포함하는 외부 함수를 돕는 헬퍼 함수의 역할을 한다.

5. function

5-4. various type of functions (다양한 타입의 함수)

5-4-4. callback function (콜백 함수)

- 함수의 매개변수를 통해 다른 함수의 내부로 전달되는 함수를 콜백 함수라고 한다.
- 매개변수를 통해 함수의 외부에서 콜백 함수를 전달 받은 함수를 고차 함수라고 한다.
- 콜백 함수는 고차 함수에 전달 되어 헬퍼 함수의 역할을 한다. 즉, 고차 함수는 콜백 함수를 자신의 일부분으로 합성한다.
- 함수의 변하지 않는 공통 로직은 미리 정의해두고 경우에 따라 변경되는 로직은 추상화해서 함수 외부에서 내부로 전달하는 방식이다.
- 콜백 함수는 함수형 프로그래밍 패러다임뿐만 아니라 비동기 처리(이벤트, 타이머, ajax)에 활용되는 중요한 패턴이다. 또한 배열 고차 함수(map)에서도 사용 된다.

5. function

5-4. various type of functions (다양한 타입의 함수)

5-4-4. callback function (콜백 함수)

```
// 전달 받은 값을 증가 시켜주는 함수
function increase (value) {
    return value + 1;
}

// 전달 받은 값을 감소 시켜주는 함수
function decrease (value) {
    return value - 1;
}

// 전달 받은 함수에 전달 받은 값을 적용 시켜주는 고차 함수
function apply(func, value) {
    // 고차 함수는 매개변수를 통해 전달 받을 콜백 함수의 호출 시점을
결정해서 호출한다.
    // 콜백 함수는 고차 함수에 의해 호출되며 이때 고차 함수는 필요에
따라 콜백함수에 인수를 전달할 수 있다.
    return func(value);
}

// 고차 함수로 콜백 함수를 전달하며 호출
console.log(apply(increase, 5));
console.log(apply(decrease, 5));
```

5. function

5-4. various type of functions (다양한 타입의 함수)

5-4-5. pure and impure function (순수 함수와 비순수 함수)

- 순수 함수 : 외부 상태에 의존하지도 않고 변경하지도 않는 함수
- 비순수 함수 : 외부 상태에 의존하거나 외부 상태를 변경하는 함수
- 함수 외부 상태의 변경을 지양하는 순수 함수를 사용하는 것이 좋다.

5. function

5-4. various type of functions (다양한 타입의 함수)

5-4-5. pure and impure function (순수 함수와 비순수 함수)

```
var cnt = 0;

// 순수 함수는 최소 하나 이상의 인수를 전달 받으며 인수의 불변성을 유지한다.
function increase(n) {
    return ++n;
}

// 순수 함수가 반환한 결과값을 변수에 재할당해서 상태를 변경
cnt = increase(cnt);
console.log(cnt);

cnt = increase(cnt);
console.log(cnt);

// 비순수 함수
function decrease() {
    return --cnt;   // 외부 상태에 의존하며 외부 상태를 변경한다.
}

// 비순수 함수는 외부 상태(cnt)를 변경하므로 상태 변화를 추적하기 어려워진다.
cnt = decrease();
console.log(cnt);

cnt = decrease();
console.log(cnt);
```

5. function

5-5. first class object (일급 객체)

5-5-1. first class object (일급 객체)

■ **일급 객체란?**

1. 무명의 리터럴로 생성할 수 있다. 즉, 런타임에 생성이 가능하다.
2. 변수나 자료구조(객체, 배열 등)에 저장할 수 있다.
3. 함수의 매개변수에 전달할 수 있다.
4. 함수의 반환값으로 사용할 수 있다.

5. function

5-5. first class object (일급 객체)

5-5-1. first class object (일급 객체)

```
// 1. 무명의 리터럴로 생성할 수 있다.
// 2. 변수에 저장할 수 있다.
var hello = function () {
    return '안녕하세요!';
};
// 2. 객체에 저장할 수 있다.
var obj = { hello };

// 3. 함수의 매개변수에 전달할 수 있다.
function repeat(func, count) {
    for(var i = 0; i < count; i++) {
        console.log(func());
    }
    // 4. 함수의 반환값으로 사용할 수 있다.
    return function () {
        console.log(`${count}번 반복 완료`);
    }
}

var returnFunc = repeat(obj.hello, 5);
returnFunc();
```

- 함수가 일급 객체라는 것은 함수를 객체와 동일하게 사용할 수 있다는 의미이다.
- 객체는 값이므로 함수는 값과 동일하게 취급할 수 있다.
- 변수 할당문, 객체의 프로퍼티 값, 배열의 요소, 함수 호출의 인수, 함수 반환문 등에서 사용할 수 있다.