함수

함수

- 함수 : 동작 실행 코드
- 함수 선언

```
function sayHello() {
    console.log('hello');
}
```

- 함수 실행

```
sayHello();
```

함수 식

- 함수식

```
var hello = function() {
    console.log('Hi~');
}
```

- 호출

```
hello();
```

함수

- 함수에 데이타 전달
- 결과값 : return

```
function thankYou() {
    return 'and you?';
}
```

- 사용

```
var ret = thankYou();
```

함수 파라미터

- 함수 파라미터

```
function say(what) {
    console.log('say ' + what);
}
```

- 사용

```
say('How are you');
```

함수 파라미터

- 함수식

```
var sayHelloTo = function(who) {
    console.log('Hello ' + who);
}
```

- 함수식

```
sayHelloTo('IU');
```

함수 파라미터

- 함수 파라미터

```
function func(a, b) {
    console.log(a, b);
}

func(2, 3);
func(4); // b의 값은?
func(5, 6, 7); // a, b 의 값은?
```

오버로딩

- 오버로딩

```
function add(i) { console.log(‘add(i)'); }
function add(i, j) { console.log('add(i, j)’); }
function add(i, j, k) { console.log('add(i, j, k)’); }
```

- 뭐가 호출되는가?

```
add(1);
add(1, 2);
add(1, 2, 3);
```

실습

- 04.Function
  - function
  - parameter
  - overloading

가변 길이 파라미터

- 가변 길이 함수

```
function sayHelloTo(who) {
   for ( var key in arguments ) {
      console.log('For-in Loop ', arguments[key]);
   }
}
sayHelloTo('IU', 'YuInna', 'Taeyon');
```

- ES5
- arguments : 객체로 전달
- who : 첫번째 파라미터 -IU

가변 길이 파라미터

- 가변 길이 함수

```
function wantToBuy(...things) {
   for(var i = 0 ; i < things.length ; i++) {
      console.log('I want to buy ' + things[i]);
   }
}
sayHelloTo('IU', 'YuInna', 'Taeyon');
```

- ES6
- 별도의 예약어 없이 파라미터 사용. 배열로 전달

실습

- 04.Function/variadic_parameters

함수 내 함수

- 함수 내부에 함수 정의

```
function circle(radius) {
    var pi = 3.14;
    function area(r) {
        return r * r * pi;
    }
    return area(radius);
}
circle(3)
```

- 예제 코드 : innerFunction.js

IIFE

- IIFE - Immediately Invoked Function Expression

```
(function(){
   console.log('IIFE Test1');
})();

// with Parameter
(function(arg){
   console.log('IIFE Test2 – argumanet : ', arg.name);
})({‘name':'value'});
```

- 예제 코드 : JavaScript/Function/iife.js

클로저

클로저

- 클로저 : 함수 + 객체
- 클로저 콘텍스트

함수 객체

- 함수

```
var hello ='Hello';
function sayHello() {
   return hello;
}
```

함수 객체

- 함수

```
var hello = function() {
   console.log('Hello');
}
```

- 함수와 함수

```
function sayHello() {
   return hello;
}
```

- 호출과 결과

```
var ret = sayHello()
ret();
```

함수 객체

- nested function

```
function sayHi() {
   function hi() {
      console.log('hi');
   }
   return hi;
}

var ret2 = sayHi();
ret2();
```

함수 객체

- Inline 방식

```
function sayThankyou() {
   return function() {
      console.log('Thank you');
   };
}

var ret3 = sayThankyou();
ret3();
```

클로저

- 클로저와 변수 Scope

```
function makeId() {
    var lastId = 0;
    return function(){
        return ++lastId
    };
}

var idFunc = makeId();
idFunc();
idFunc();
```

함수 파라미터

- 함수와 파라미터

```
var hello = 'Hello';


function sayWhat(msg) {
  console.log(msg);
}

sayWhat(hello);
```

함수와 파라미터

- 함수 객체, 함수 파라미터

```
var sayHello = function() {
  console.log('Hello');
}
function say(what) {
  what();
}

say(sayHello)
```

함수와 파라미터

- 함수 파라미터로 클로저, Inilne

```
function doIt(what) {
  what();
}
```

- Inline

```
doIt( function() {
  console.log('just do it!');
});
```

함수와 파라미터

- 파라미터가 있는 클로저

```
function add(i, j, handler) {
  var ret = i + j;
  handler(ret);
}

function handleAddResult(sum) {
  console.log('Handling Result : ', sum);
}

add(1, 2, handleAddResult);
```

함수와 파라미터

- 파라미터가 있는 클로저(Inline)

```
add(1, 2, function(ret) {
  console.log('1 + 2 = ', ret);
});
```

실습

- 05.Closure
  - closure
  - closure_context
  - closure_parameter

자주 사용하는 형태

- node.js 에서 많이 사용하는 코드 패턴

```
http.createServer(function(req, res) {
    // HTTP 서버 코드
});
```

Arrow Function

- from ES6

```
const helloFn = () => {
    console.log('Hello JavaScript');
}
doIt( () => {
  console.log('Arraw Function')
} );
function getRandom(num) {
    return () => {
        return Math.floor( Math.random() * num )
    }
}
```