

(1) 1~100 사이의 난수를 5개 생성하여 가장 큰 값을 반환하는 콜백 함수 callback()을 함수 표현식으로 구현하고 callback() 콜백 함수를 매개변수로 받아 3번 호출하는 고차 함수 func_call()을 구현하시오

- 난수 생성 : `Math.floor(Math.random() * 100) + 1`
- 모든 출력은 콘솔로 한다

콜백 함수 0 번째 호출
콜백함수 실행 (1) : 55
콜백함수 실행 (2) : 78
콜백함수 실행 (3) : 34
콜백함수 실행 (4) : 3
콜백함수 실행 (5) : 58
78
콜백 함수 1 번째 호출
콜백함수 실행 (1) : 90
콜백함수 실행 (2) : 16
콜백함수 실행 (3) : 63
콜백함수 실행 (4) : 66
콜백함수 실행 (5) : 87
90
콜백 함수 2 번째 호출
콜백함수 실행 (1) : 21
콜백함수 실행 (2) : 9
콜백함수 실행 (3) : 9
콜백함수 실행 (4) : 27
콜백함수 실행 (5) : 3
27

[소스]

```
<script>
```

```
const callback = function () { //5 번 생성된 난수에서 가장 큰 값 반환
```

```
let max = 0, value;
```

```
for (let i = 1; i <= 5; i++) {
```

```
value = Math.floor(Math.random() * 100) + 1;
```

```
console.log('콜백함수 실행({i}) : ${value}');
```

```
if (value > max)
```

```
max = value;
```

```
}
```

```
return max;
```

```
}
```

```
const func_call = (f_call) => { //매개변수로 받은 함수를 3 번 호출
```

```
for (let i = 0; i < 3; i++) {
```

```
console.log('콜백 함수 {i} 번째 호출');
```

```
console.log(f_call());
```

```
}
```

```
}
```

```
func_call(callback); //매개변수로 함수를 전달
```

```
</script>
```

[실행 결과]

- (2) 다음과 같이 6자리숫자를 사용하여 369게임을 하는 프로그램을 작성하세요. 3, 6, 9 숫자일 때 '박수짝'을 출력하며 박수를 많이 받은 사람이 승자가 된다. 사용자 두 사람을 입력 받는다.

127.0.0.1:5500 내용:

사용자1

길동

확인

취소

127.0.0.1:5500 내용:

사용자2

둘리

확인

취소

길동 사용자
생성된 난수 : 197936
박수짝 1
박수짝 2
박수짝 3
박수짝 4
박수 개수 : 4

둘리 사용자
생성된 난수 : 166986
박수짝 1
박수짝 2
박수짝 3
박수짝 4
박수 개수 : 4

비겼습니다

길동 사용자
생성된 난수 : 120329
박수짝 1
박수짝 2
박수 개수 : 2

둘리 사용자
생성된 난수 : 182092
박수짝 1
박수 개수 : 1

길동 사용자가 이겼습니다

```
let user1 = prompt("사용자1");  
let user2 = prompt("사용자2");
```

```
//사용자를 매개변수로 전달하고 박수 친 횟수를 반환 받는다. - 화살표 함수  
let cnt1 = game(user1);  
let cnt2 = game(user2);
```

[소스]

```
<script>
```

```
let game = (ch) => {  
  let cnt = 0;  
  let rnd = parseInt(Math.random() * 900000) + 100000;  
  
  document.write(`${ch} 사용자 ₩<br>생성된 난수 : ${rnd}<br>`);  
  for (; rnd > 0; rnd = parseInt(rnd/10)) {  
    let mo = rnd % 10;  
    if (mo % 3 == 0 && mo != 0) {  
      document.write('박수짝 ${++cnt}<br>');  
    }  
  }  
  document.write('박수 개수 : ${cnt} <br><br>');  
  return cnt;  
}  
  
let user1 = prompt("사용자 1");  
let user2 = prompt("사용자 2");
```

```

let cnt1 = game(user1);
let cnt2 = game(user2);

if (cnt1 == cnt2) {
    document.write('비겼습니다');
}
else if (cnt1 > cnt2) {
    document.write(`${user1} 사용자가 이겼습니다`);
}
else {
    document.write(`${user2} 사용자가 이겼습니다`);
}

```

</script>

=====

```

let cnt = 0;
let random = (parseInt(Math.random() * 900000) + 100000);
let rnd = random.toString();
document.write(`>> ${ch} 사용자 <br>&nbsp;&nbsp;  생성된 난수 : ${rnd}<br>`);
for (let i = 0; i < rnd.length; i++) {
    if (parseInt(rnd[i]) % 3 == 0 && rnd[i] != '0')
        document.write(`&nbsp;&nbsp;  박수 짝 ${++cnt}회, 숫자=${rnd[i]}<br>`);
}
document.write(`&nbsp;&nbsp;  박수 개수 : ${cnt} <br><br>`);
return cnt;

```

[실행 결과]

(3) 함수 호출문과 실행 결과를 참고하여 연산 결과를 반환하는 total() 클로저 함수를 정의하시오.

```

const cfunc = total();
for (let i = 1; i < 6; i++) {
    let rnd = parseInt(Math.random() * 100) + 1;
    console.log(`result => ${cfunc(rnd)}`)
}

```

1	:	85
result =>		85
2	:	-59
result =>		26
3	:	89
result =>		115
4	:	-66
result =>		49
5	:	88
result =>		137

[소스]

<script>

```

const total = function () {
  let result = 0;
  let cnt = 0
  const add = function (value) {
    cnt++;
    if (cnt % 2 == 0) {
      value *= -1
    }
    console.log(`${cnt} : ${value}`);
    result += value;
    return result;
  }
  return add;
}

const cfunc = total();
for (let i = 1; i < 6; i++) {
  let rnd = parseInt(Math.random() * 100) + 1;
  console.log(`result =>  ${cfunc(rnd)}`)
}
</script>

```

[실행 결과]

- (4) 다음과 같은 속성과 메소드로 구성되는 리터럴 객체를 정의 하시오. 단, 메소드는 축약하여 구현하고 결과는 웹브라우저로 출력한다. 단, 출력 시 템플릿 문자열 사용
- 속성 : 소유주, 차량번호, 주행거리
 - 메소드 : 주행거리를 dist 만큼 증가시키는 addDistance(dist) 메소드, 반환값 없음
차량번호와 주행거리를 문자열로 반환하는 toString()
소유주와 차량번호를 변경하는 접근자 함수 fullCar()



```

소유자 : hallym   차량번호 : 40마3456   주행거리 : 1000
차량번호 40마3456의 주행거리를 30 추가
소유자 : hallym   차량번호 : 40마3456   주행거리 : 1030
car 객체의 소유자와 차량번호를 모두 변경합니다.
소유자 : software   차량번호 : 456서2950   주행거리 : 1030

```

[소스]

```

<script>
const car = {
  cnum: '40 마 3456',
  owner : 'hallym',
  distance: 1000,

```


현재 상태 입니다
예금주 : 스크립트
연락처 : 010-8765-1386
현재 잔액 : 50000

50000 예금 후 상태 입니다
예금주 : 스크립트
연락처 : 010-8765-1386
현재 잔액 : 100000

1000000을 인출하려고 합니다
잔액 부족 : 900000

변경 전 연락처 010-8765-1386
변경 후 연락처 010-3456-9743

```
let acc1 = new Account('스크립트', 50000, '010-8765-1386');  
document.write(`현재 상태 입니다 <br>`)  
acc1.display();
```

```
document.write(`<br>50000 예금 후 상태 입니다 <br>`)  
acc1.deposit(50000);  
acc1.display();
```

```
document.write(`<br>1000000 을 인출하려고 합니다<br>`)  
acc1.withdraw(1000000);
```

```
document.write(`<br>변경 전 연락처 ${acc1.Tel}`);  
acc1.Tel='010-3456-9743';  
document.write(`<br>변경 후 연락처 ${acc1.Tel}`);
```

[소스]

<script>

```
class Account {  
  constructor(name, balance, tel) {  
    this.name = name;  
    this.balance = balance;  
    this.tel = tel;  
  }  
  get Tel() {  
    return this.tel;  
  }  
  set Tel(tel) {  
    this.tel = tel;  
  }  
  get Balance() {  
    return balance;  
  }  
  deposit(money) {  
    this.balance += money;  
  }  
}
```

```

withdraw(money) {
  if (this.balance > money) {
    this.balance -= money;
  }
  else {
    document.write(`잔액 부족 : ${money - this.balance} <br>`);
  }
}

display() {
  document.write(`예금주 : ${this.name} <br>`);
  document.write(`연락처 : ${this.tel} <br>`);
  document.write(`현재 잔액 : ${this.balance} <br>`);
}
}

let acc1 = new Account('스크립트', 50000, '010-8765-1386');
document.write(`현재 상태 입니다 <br>`)
acc1.display();
acc1.deposit(50000);
document.write(`<br>50000 예금 후 상태 입니다 <br>`)
acc1.display();
document.write(`<br>1000000 을 인출하려고 합니다<br>`)
acc1.withdraw(1000000);

document.write(`<br>변경 전 연락처 ${acc1.Tel}`);

acc1.Tel='010-3456-9743';
document.write(`<br>변경 후 연락처 ${acc1.Tel}`);
</script>

```

[실행 결과]

(6) 다음과 같이 성적을 처리하는 스크립트를 제시된 조건대로 작성하시오.

-person() 화살표 함수

✓ 성적과 이름은 prompt()로 입력 받고, 성적과 이름을 리터럴 객체로 생성하여 반환한다

-grade() 화살표 함수

✓ person() 함수 실행 후 반환 받은 객체를 매개변수로 받아 등급을 계산한 후 반환한다

127.0.0.1:5500 내용:

이름을 입력하세요

127.0.0.1:5500 내용:

자바 성적을 입력하세요

| Elements | | Console | Sources |
|----------|--|---------------------------|---------|
| | | top ▼ | Filter |
| | | name : hallym | |
| | | java : 89 | |
| | | hallym 학생의 java 성적은 B 입니다 | |

```

const std = person();
for(let key in std){
  console.log(`${key} : ${std[key]}`);
}

```

```
}  
const gradeJava = grade(std);  
console.log(`${std.name} 학생의 java 성적은 ${gradeJava} 입니다`);
```

[소스]

<script>

```
const person = () => {  
  const name = prompt('이름을 입력하세요');  
  const java = prompt('자바 성적을 입력하세요');  
  return {name:name, java:java};  
  //return {name, java}; //프로퍼티 축약  
}
```

```
const grade = (student) =>{  
  if(student.java >= 90){  
    return 'A';  
  }  
  else if (student.java >=80){  
    return 'B';  
  }  
  else if (student.java >=70){  
    return 'C';  
  }  
  else if (student.java >=60){  
    return 'D';  
  }  
  else{  
    return 'F';  
  }  
}
```

```
const std = person();  
for(let key in std){  
  console.log(`${key} : ${std[key]}`);  
}  
const gradeJava = grade(std);  
console.log(`${std.name} 학생의 java 성적은 ${gradeJava} 입니다`);
```

</script>

[실행 결과]
