Ch.8 제어문

▼ 🌟제어문

제어문을 사용하면 코드의 흐름을 인위적으로 제어할 수 있다.

제어문에는 조건문과 반복문이 있다.

forEach, map, filter, reduce같은 고차 함수를 사용한 함수형 프로그래밍 기법에서는 제어문의 사용을 억제하여 복잡성을 해결하려고 노력한다.

▼ 조건문

- if... else 문
  - if문의 조건식이 불리언 값이 아닌 값으로 평가되면 자바스크립트 엔진에 의해 암묵적으로 불리언 값으로 강제변환되어 실행할 코드 블록을 결정한다.
  - 코드 블록내의 문이 하나뿐이라면 중괄호를 생략할 수 있다.
  - 대부분의 if...else문은 삼항 조건 연산자로 바꿔 쓸 수 있다.
- switch 문

▼ 폴스루

문을 탈출하지 않고 문이 끝날때 까지 이후의 모든 문을 실행. p98(e132)

▼ 🌟🌟반복문

- for 문

- for 문의 변수 선언문, 조건식, 증감식은 모두 옵션이므로 반드시 사용할 필요는 없다. 단, 어떤 식도 선언하지 않으면 무한루프가 된다.

```
for (;;) { } // 무한루프
```

- 중첩 for 문 p102(e136)

- while 문
  - for 문은 반복횟수가 명확할때 주로 사용하고 while문은 반복횟수가 불명확할 때 주로 사용한다.
  - 조건식의 평가 결과가 불리언 값이 아니면 불리언 값으로 강제 변환한다.
  - 무한루프에서 탈출하기 위해서는 코드 블록 내에 if문으로 탈출 조건을 만들고 break문으로 코드 블록을 탈출한다. 예제 08-16

```
while(true) { } // 무한루프

----------------------------------------------------

var count = 0;

while(true) {
    console.log(count);
    count++;

    if(count === 3) break;
}

// 0 1 2
```

- do...while 문
  - 코드블록을 먼저 실행하고 조건식을 평가한다. 따라서 코드 블록은 무조건 한 번 이상 실행된다.

```
var count = 0;

do {
    console.log(count); // 0 1 2
```

```
    count++;
    } while (count < 3);
```

자바스크립트는 반복문을 대체할 수 있는 다양한 기능을 제공한다.

- 배열을 순회할 때 사용하는 forEach 메서드,
- 객체의 프로퍼티를 열거할 때 사용하는 for...in 문,
- 이터러블을 순회할 수 있는 for...of문

▼ 🌟break 문

레이블문, 반복문, switch문의 코드 블록을 탈출하는 문

레이블문, 반복문, switch문 외에 break문을 사용하면 문법에러 발생

▼ 레이블 문

식별자가 붙은 문. 프로그램의 실행 순서를 제어하는데 사용한다.

switch문의 case문과 default문도 레이블 문이다.

레이블 문을 탈출하려면 break문에 레이블 식별자를 지정한다. 반복문, switch문에서 사용할 때는 레이블 문에 식별자를 지정하지 않는다.

레이블 문은 중첩된 for문 외부로 탈출할때 유용하지만 그 밖의 경우에는 일반적으로 권장하지 않는다. p 104-105(e138-139)

▼ 🌟continue문

반복문의 코드 블록 실행을 현 지점에서 중단하고 반복문의 증감식으로 실행 흐름을 이동시킨다.

continue문이 실행되면 continue문 이후의 코드는 실행되지 않는다.

▼ 🌟🌟문자열에서 특정 문자의 인덱스를 검색하는 예

p105(e139) 예제 08-22 문자열에서 특정 문자의 인덱스를 검색하는 예

문자열은 유사 배열이므로 for문으로 순회할 수 있다.

```
var string = 'Hello World.';    //length: 12
var search = 'l';
var index;

for(var i=0; i<string.length; i++) {
    if(string[i] === search) {
        index = i;
        break;
    }
}

console.log(index); // 2
```

String.prototype.indexOf 메서드를 사용해도 같은 동작을 한다.

```
console.log(string.indexOf(search));  // 2
```

▼ 🌟🌟문자열에서 특정 문자의 개수를 세는 예

p106-107(e140-141) 예제 08-23. 08-24

if문 내에서 실행해야 할 코드가 길다면 들여쓰기가 한 단계 더 깊어지므로 continue문을 사용하는 편이 가독성이 더 좋다.

```
var string = 'Hello World.';    //length: 12
var search = 'l';
var index;

for(var i=0; i<string.length; i++) {
    if(string[i] !== search) continue;
    count++; // continue문이 실행되면 이 문은 실행되지 않는다.
}

console.log(count); // 3
```

```
----------------------------------------------------------------

for(var i=0; i<string.length; i++) {
    if(string[i] === search) count++;
}
```

String.prototype.match 메서드를 사용해도 같은 동작을 한다.

```
const regexp = new RegExp(search, 'g');

console.log(string.match(regexp).length);  // 2
```

`const regexp = new RegExp(search, 'g');`

search 변수에 저장된 문자열 또는 패턴을 사용하여 새로운 정규 표현식 객체를 생성합니다. 여기서 'g' 옵션은 전역 검색을 의미하며, 문자열 전체에서 일치하는 모든 부분을 찾습니다.

`console.log(string.match(regexp).length);`

string 변수에('Hello World.') 저장된 문자열에서 regexp에 정의된 패턴에 일치하는 모든 부분을 찾습니다 (match 함수 사용).

일치하는 결과는 배열로 반환되고, length 속성을 사용하여 이 배열의 크기(즉, 일치하는 요소의 수)를 계산합니다.