구문

구문 : 어떤 일을 하기 위해 실행

1. 표현문

가장 간단한 형태 : 부수효과가 있는 표현식
할당문 , 함수호출 등이 있음

2. 복합문과 빈 구문

하나의 표현식 안에 여러 표현식을 합칠 때에는 쉼표(,)연산자를 사용

2.1 구문블록

구문블록은 여러 구분을 중괄호로 감싼 것
- 세미콜론으로 끝나지 않음
- 들여쓰기를 해주는 것이 좋다 (가독성, 이해도)
- 안에서 선언된 변수는 지역변수가 아닌 전역변수이다

2.2 빈 구문

자바스크립트 인터프리터는 빈 구문을 만나면 아무것도 실행하지 않는다
- 종종 몸체가 비어있는 루프를 만들 때 유용
- 임의로 빈 구문 사용시 코드에 고의로 사용했다는 설명을 주석으로 표시

3. 선언문

식별자를 정의, 변수나 함수를 생성하는 중요한 역할을 담당
var, function 은 삭제할 수 없는 변수를 만듬

3.1 var

변수를 정의하는 데 쓰임
쉼표로 구분되어 여러 개를 한번에 선언 가능
초기화 하지 않으면 undefined 값을 가짐

3.2 function

함수를 정의하는 데 쓰임
유일한 이름을 가짐
선언문 문법

```
function 함수이름(전달인자1, 전달인자2, ... ,전달인자n) {
        구문
}
```

4. 조건문

특정 표현식의 값에 따라 구문을 실행시키거나 건너뛴다

4.1 if

단순히 어떤 결정을 내리거나, 조건에 따라 좀더 정교하게 구문들을 실행할 수 있도록 한다
형태 (표현식이 true 이면 실행)

```
- if (표현식)
        구문
- if (표현식)
        구문1
  else
        구문2
```

4.2 else if

조건문을 평가한 결과에 따라 여러 개 중 하나의 코드를 실행하게 해준다

4.3 switch

```
switch(구문) {
        표현식
}
표현식의 모양
        case 1:
                //코드블록 1실행
                break;
        case 2:
                //코드블록 2실행
                break;
        case 3:
                //코드블록 3실행
                break;
        default:
                //코드블록 4실행
                break;
```

5. 루프

while, do/while, for, for/in 4개의 루프문이 있다
표현식이 true 이면 루프 실행

5.1 while

```
while(표현식)
        구문
```

5.2 do/while

적어도 한번은 루프 몸체가 실행
끝에 세미콜론이 붙음
do
구문
while (표현식);

5.3 for

세미콜론으로 구분된 세가지 표현식을 가짐
for(초기화; 테스트; 증가)
구문
for(;;) == while(true)

5.4 for/in

for키워드를 사용하지만 일반적인 for 루프와는 다른 종류
자마에서 확장 for 문과 비슷
어느 한 객체에 있는 속성을 차례대로 한번씩 꺼내오는 역할 (배열 등에 이용)
for(변수 in 객체)

예) a 배열의 모든 프로퍼티 값을 b가 참조하는 Array 로 복사

```
var a= {x:1, y:2, z:3};
var b= new Array();
var I = 0;
for ( b[i++] in a);
```

6. 점프문

자바스크립트 인터프리터가 점프문을 만나면 특정 위치로 건너뜀

6.1 레이블

프로세스의 흐름을 관리하는데 사용하기 위해 사용하는 인식표
어떤 구문에라도 그 앞에 식별자 이름과 콜론을 넣음으로 레이블을 붙일 수 있다
식별자: 구문

6.2 break

반복하던 행위를 끝마치고 구문 밖으로 나온다
루프나 switch 문 내부에서만 적법하다
break 키워드와 레이블 이름 사이에는 줄바꿈이 허용되지 않음
(자바스크립트가 생략된 세미콜론을 자동으로 넣어주기 때문)
break; break 레이블이름;

6.3 continue

break 문과 유사하지만 루프를 빠져 나오지 않고 새로운 반복을 시작한다
항상 루프의 몸체 내부에서 사용해야 한다.
continue; continue 레이블이름;

6.4 return

함수호출 표현식의 값, 즉 함수에서 반환되는 값을 지정하는 데 쓰인다
오직 함수 몸체 내부에서만 나타날 수 있음
return 키워드와 표현식 사이에는 줄바꿈이 허용되지 않음
(자바스크립트가 생략된 세미콜론을 자동으로 넣어주기 때문)
return 표현식;

6.5 throw

예외의 값을 평가해서 보내 알린다
예외가 발생하면 자바스크립트 인터프리터는 정상적인 프로그램 실행을 즉시 중단하고 가장 가까운 예외처리기로 넘어간다
예외처리기를 찾을 수 없을 경우 해당 예외는 에러로 취급되고 사용자에게 보고된다
throw 표현식;

6.6 try / catch / finally

예외처리 기법
try : 단순히 예외가 발생할지도 모르는 코드 블록을 정의
catch : try 블록 내부에서 예외가 발생할 경우 호출
finally : try 블록에 일어난 상황과 관계없이 항상 실행이 보장 되어야 할 뒷정리용 코드가 포함
try 블록은 catch, finally 둘 중 하나이상의 블록과 함께 사용되어야 한다

```
try{
}
catch (e) {
    //try 블록에서 예외가 발생되어야만 실행 되는 부분
    //throw 를 사용해서 예외를 다시 발생시킬 수도 있다
}
finally {
    // try 블록에 일어난 일과 관계없이 try 블록이 종료되면 실행
}
```

7. 기타 구문

with, debugger, use strict

7.1 with

유효범위 체인을 임시로 변경하려 할 때 쓰임
유효범위 체인의 첫 번째에 '객체'를 추가 한 후 '구문'을 실행한 다음 유효범위체인을 '객체' 추가하기 전으로 되돌림
엄격한 모드에서 사용 할 수 없고 with 문 사용시 현저하게 느려지기 때문에 사용을 자제해야 한다
주로 깊이 중첩된 객체 계층 구조를 좀더 쉽게 다루기 위해 사용
with (객체)
구문

7.2 debugger

디버거가 실행 중일 때 자바스크립트 구현체는 해당위치에서 정의된 코드 디버깅을 수행
중단점과 같이 동작

7.3 "use strict"

ECMAScript5 에서 처음 소개된 지시어
지시어지만 구문에 가깝다
일반적인 구문과의 차이점
- 키워드 목록에 포함되지 않음
- 스크립트의 시작 부분이나 함수 몸체의 시작 부분에만 올 수 있다
  (하지만 반드시 스크립트나 함수의 시작 부분이 될 필요는 없다)

사용이유는 지시어 다음에 오는 코드들이 엄격한 모드를 따르게 하기 위해서다
엄격한 모드와 일반모드의 차이점 (중요3가지)
- with 문은 엄격한 모드에서 사용할 수 없다
- 모든 변수는 반드시 선언 되어야 한다
- 함수가 메서드가 아닌 함수로 호출 될 때 this 의 값은 undefined가 된다.

8. 구문 요약

구문	문법	용도
break	break 레이블;	가장안쪽의 루프, switch문 또는 '레이블'로 명명된 구문에서 빠져나온다
case	case 표현식;	switch문 내부의 구문에 레이블을 붙인다
continue	continue 레이블;	가장 안쪽의 루프, 또는 '레이블'로 명명된 루프를 재시작한다
debugger	debugger;	디버거 중단점
default	default;	switch문에서 디폴트 구문에 레이블을 붙인다
do/while	do 구문 while(표현식);	while 루프를 만드는 다른 방법
empty	;	아무 일도 안함
for	for(초기화; 테스트; 증가)구문	편리하게 쓸 수 있는 루프
for/in	for(변수 in 객체)구문	객체에 속한 프로퍼티들을 열거한다
function	function 이름(전달인자,...){ 구문}	'이름'이라는 함수를 선언한다
if/else	if(표현식){구문1} else{구문2}	구문1 또는 구문2를 실행한다
label	레이블 : 구문	'구문'에 '레이블'이라는 이름을 붙인다
return	return 표현식;	함수에서 값을 반환한다
switch	switch(표현식){구문}	case 또는 default: 레이블이 붙은 구문들로 다중 분기
throw	throw 표현식;	예외를 발생시킨다
try	try {구문} catch(식별자){구문} finally{구문}	예외를 잡아낸다
use strict	"use strict";	스크립트나 함수를 엄격한 모드로 제한 시킨다
var	var 이름 =값 , ... ;	하나 이상 변수의 선언과 초기화
while	while (표현식) 구문	기본적인 루프 생성문
with	with (객체) 구문	유효범위 체인의 확장 (엄격한 모드에서 사용 불가)