Function 오브젝트

프로퍼티 리스트

이름	개요
Function	
new Function()	인스턴스 생성
Function()	인스턴스 생성
Function 프로퍼티	
length	함수의 파라미터 수
Function.prototype	
constructor	생성자
call()	함수 호출
apply()	함수 호출: 배열을 파라미터로 사용
toString()	함수를 문자열로 반환
bind()	새로운 오브젝트를 생성하고 함수 실행

new Function()

구분	데이터(값)
파라미터	파라미터opt
	실행 가능한 JS 코드opt
반환	생성한 Function 인스턴스

- Function 인스턴스 생성
- 파라미터에 문자열로
  함수의 파라미터와 함수 코드 작성
  - var obj = new Function("book", "return book;");
  - obj("JS 책");
- 파라미터 수에 따라 인스턴스 생성 기준이 다름

new Function()

- 파라미터 2개 이상 작성
  - 마지막 파라미터에
    함수에서 실행할 함수 코드 작성
  - 마지막을 제외한 파라미터에 이름 작성 코드 1
- 파라미터 하나 작성
  - 함수에서 실행할 함수 코드 작성
  - 파라미터가 없을 때 사용 코드 2
- 파라미터를 작성하지 않으면
  - 함수 코드가 없는 Function 인스턴스 생성

Function()

구분	데이터(값)
파라미터	파라미터opt
	실행 가능한 JS 코드opt
반환	생성한 Function 인스턴스

- Function 인스턴스 생성
- 처리 방법과 파라미터 작성이 new Function()과 같음
- 단지 new 연산자를 사용하지 않은 것

함수 생명 주기

함수 분류

- function 분류
  - 빌트인 Function 오브젝트
  - function 오브젝트
  - function 인스턴스(new 연산자 사용)
- function 오브젝트 생성 방법
  - function 키워드 사용
  - function getBook(title){return title};
- JS 엔진이 function 키워드를 만나면
  - 이름이 getBook인 function 오브젝트 생성

함수 생명 주기

- 함수 호출 코드 1
  - getBook("JS북");
  - 함수를 호출하면서 파라미터 값을 넘겨 줌
- 함수 코드 실행
  - JS 엔진 컨트롤이 함수의 처음으로 이동
  - 파라미터 이름에 넘겨 받은 파라미터 값 매핑
  - 함수 코드 실행
  - return 작성에 관계없이 반환 값을 갖고 함수를 호출한 곳으로 돌아 감

length 프로퍼티

- 함수의 파라미터 수가
  - 생성되는 function 오브젝트에 설정됨 코드 1
  - 함수를 호출한 곳에서 보낸
    파라미터 수가 아님 코드 2
- JS 엔진이 자동으로 설정

함수 형태

함수 형태

- 함수 선언문
  - Function Declaration
  - function getBook(book){코드}
- 함수 표현식
  - Function Expression
  - var getBook = function(book){코드}

함수 선언문

구분	데이터(값)
function	function 키워드
식별자	함수 이름
파라미터	파라미터 리스트opt
함수 블록	{실행 가능한 코드opt}
반환	생성한 function 오브젝트

- function getBook(title){함수 코드} 형태 코드 1
  - function 키워드, 함수 이름, 블록{}은 작성 필수
  - 파라미터, 함수 코드는 선택
- 함수 이름을 생성한 function 오브젝트의 이름으로 사용

함수 표현식

구분	데이터(값)
function	function 키워드
식별자	함수 이름opt
파라미터	파라미터 리스트opt
함수 블록	{실행 가능한 코드opt}
반환	생성한 function 오브젝트

- var getBook = function(title){코드} 코드 1
  - function 오브젝트를 생성하여 변수에 할당
  - 변수 이름이 function 오브젝트 이름이 됨
- 식별자 위치의 함수 이름은 생략 가능
  - var name = function abc(){}에서
    abc가 식별자 위치의 함수 이름 코드 2

함수 호출

call()

구분	데이터(값)
object	호출할 함수 이름
파라미터	this로 참조할 오브젝트
	호출된 함수로 넘겨줄 파라미터opt
반환	호출된 함수에서 반환한 값

- getTotal.call(this, 10, 20); 코드 1
- 첫 번째 파라미터
  - 호출된 함수에서 this로 참조할 오브젝트
  - 일반적으로 this 사용
    다른 오브젝트 작성 가능 코드 2

apply()

구분	데이터(값)
object	호출할 함수 이름
파라미터	this로 참조할 오브젝트
	[호출된 함수로 넘겨줄 파라미터opt]
반환	호출된 함수에서 반환한 값

- getTotal.apply(this, [10, 20]);
- 파라미터 수가 유동적일 때 사용
  두 번째 파라미터에 배열 사용 코드 1
- call(), apply() 부가적인 목적?
  - 첫 번째 파라미터에 호출된 함수에서 this로 참조할 오브젝트 사용
  - 논리 전개는 단계적 설명이 필요하므로 중고급 강좌에서 다룹니다.

toString()

구분	데이터(값)
object	function
파라미터	사용하지 않음
반환	변환한 값

- 모든 빌트인 오브젝트에
  - toString()이 있지만
  - 오브젝트마다 반환되는 형태가 다름
- function 오브젝트의 toString()은
  함수 코드를 문자열로 반환 코드 1

Argument 오브젝트

아규먼트 오브젝트

- 함수가 호출되어 함수 안으로 이동했을 때
  arguments 이름으로 생성되는 오브젝트
- 함수를 호출한 곳에서 넘겨 준 값을 순서대로 저장
- 호출된 함수에 파라미터를 작성한 경우
  - 호출된 함수의 파라미터에도 값을 설정하고
    아규먼트 오브젝트에도 저장 코드 1
  - apply()와 아규먼트 오브젝트 코드 2
- 파라미터라고 부른 것은
  아규먼트 오브젝트와 구분하기 위한 것