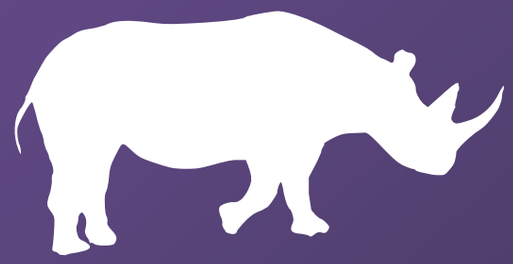


자바스크립트 코어

Core - Javascript

Core_Javascript
자바스크립트 **코어**

함수

Function

함수는 명령문의 묶음이며 재 사용 가능하다

함수이름

```
function functionIs( ) {
    with(console) {
        log('함수 선언은 변수와 비슷하죠');
        log('function 키워드 다음에 함수 이름을 넣어줍니다');
        log('함수 호출 ( )연산자 다음에 { }연산자로 명령들을 감싸죠');
        log('변수와 달리 함수는 선언과 동시에 사용되지 않아요');
        log('함수 사용을 하기 위해서는 호출이 필요합니다');
    }
}
```

함수 **이름을 호출**하면 함수가 **실행**

```
functionIs( )
```

함수를 정의하는 2가지 방법은...
함수 선언 방법과 함수 표현(참조) 방법이 있다.

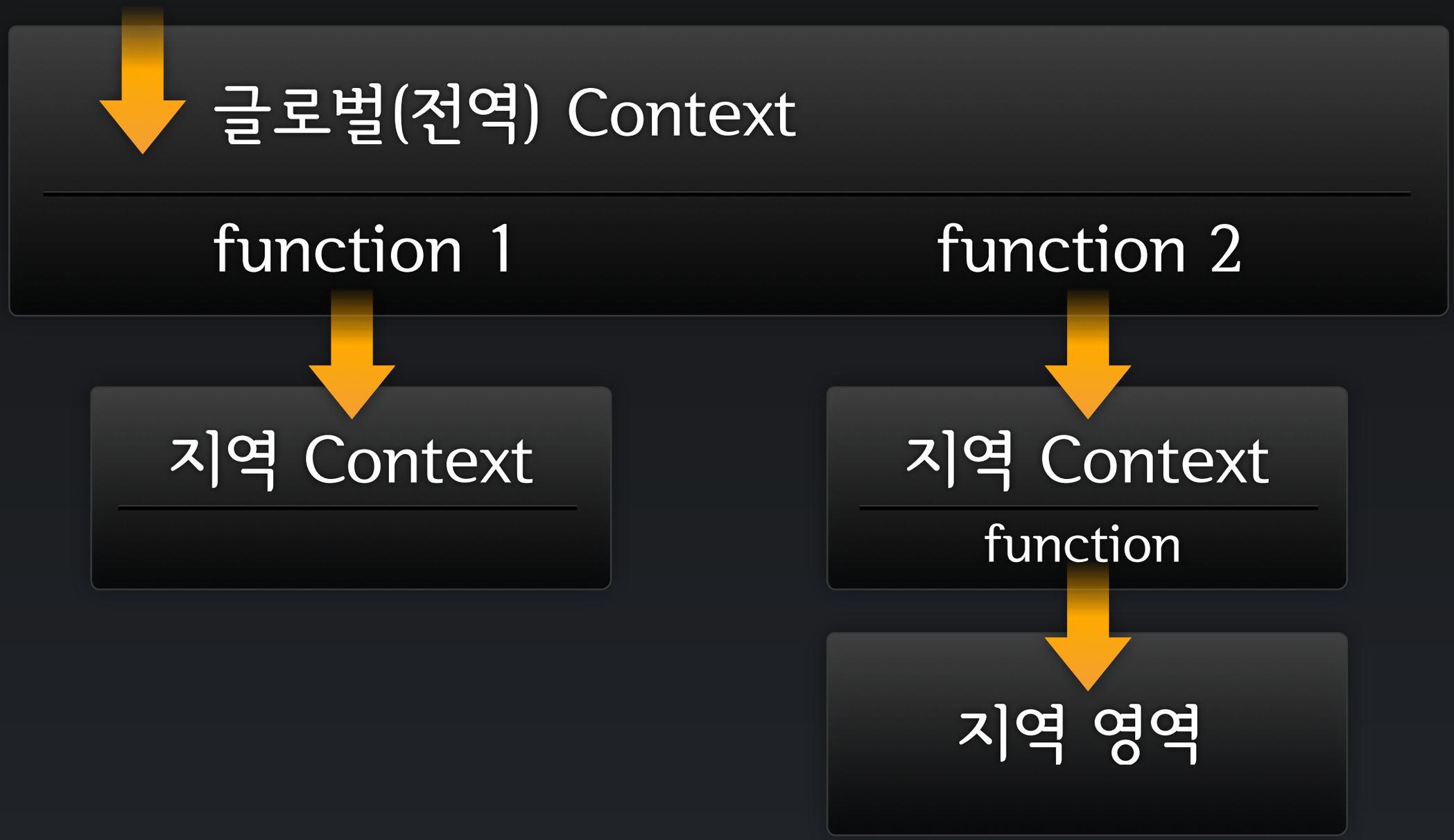
```
function fnName1() { ... }          // 선언: declarations

var fnName2 = function() { ... }    // 표현: expressions
```

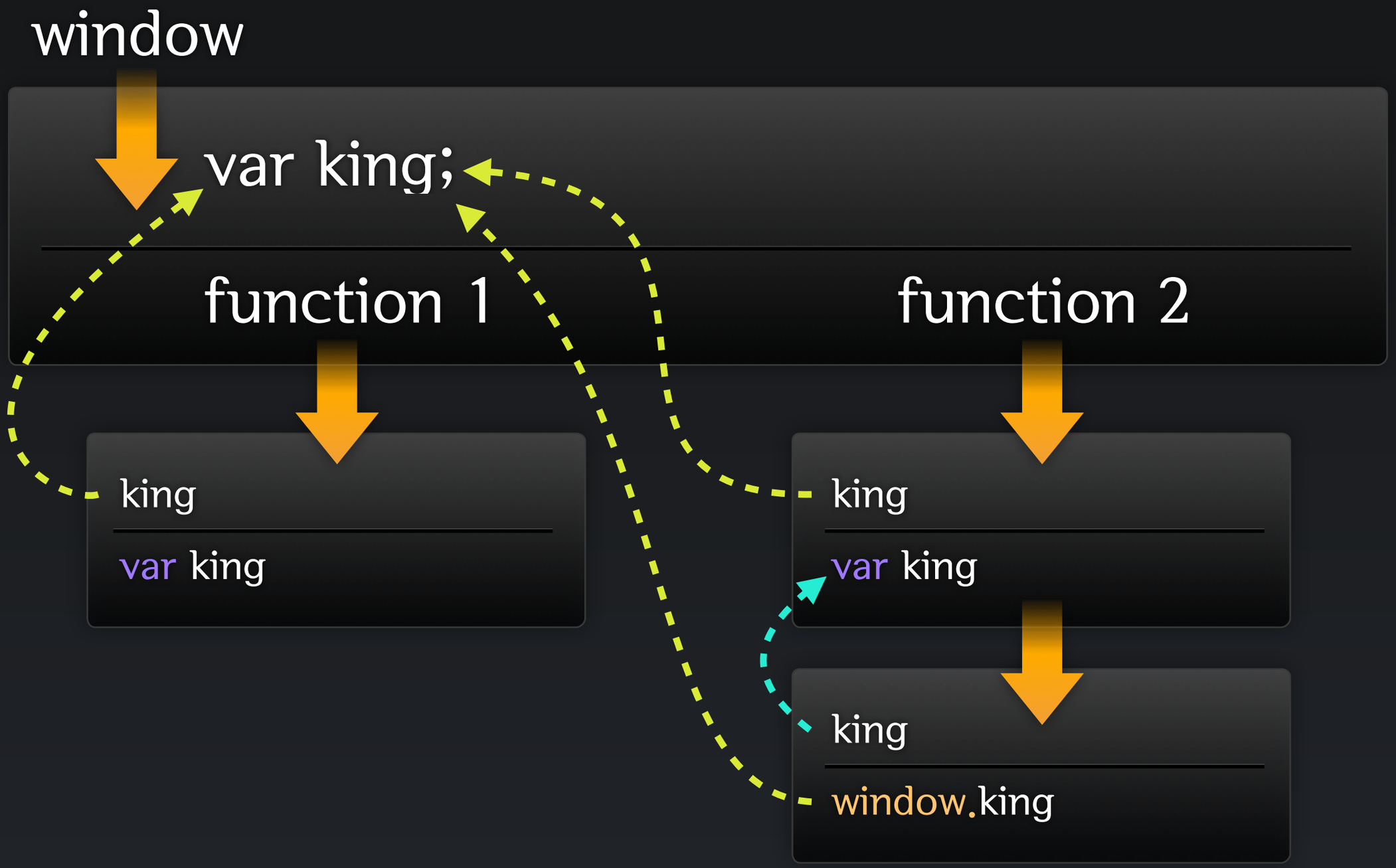
함수를 정의하는 2가지 방식의 차이점은...

스코프 호이스트(Scope Hoist)로 인한 결과 차이

스코프(Scope)란?


window
글로벌(전역) Context
function 1
function 2
지역 Context
지역 Context
function
지역 영역

전역 변수(Global) vs 지역 변수(Local)



호이스트(Hoist)

Scope

```
goKingdom();
var king = '왕';
var goKingdom = function() { ... };
awayKingdom();
function awayKingdom() { ... }
```

호이스트(Hoist)

Scope

```
function awayKingdom() { ... }
var king, goKingdom;
goKingdom(); // undefined, Type Error
king = '왕';
goKingdom = function() { ... };
awayKingdom();
```

정보 할당은 런타임 중에 실행
브라우저 문서 실행 중에 대입된다.

호이스트란? 코드를 실행하기 전에...

var, function 선언을 해당 Scope의 상위로 이동

함수를 보다 유용하게 사용하려면 확장이 필요하다

```
function showMessage( ) {
    console.log('함수가 호출되면...');
}

showMessage( ) // '함수가 호출되면 ... '
showMessage( ) // '함수가 호출되면 ... '
showMessage( ) // '함수가 호출되면 ... '
```

함수**선언**

함수**호출**

함수**호출**

함수**호출**

전달인자를 넘겨주는 함수는 재사용 가능한 함수

함수**확장**

```
function showMessage(msg) {
  console.log(msg); // 전달받은 msg를 출력한다.
}

showMessage( '전달하는 변수를 전달인자라고 부른다' );
showMessage( '전달인자(argument)또는 매개변수(parameter)' );
showMessage( '전달인자를 넘겨주지 않으면 undefined 된다' );
```

전달받은 인자를 검증하지 않으면 원하는 결과가 나오지 않을 수도 있다

```
function sum(A, B) {
  console.log(A + B); // A + B 의 합을 출력한다.
}

sum(24, 67); // 24 + 67
sum(-31, 24); // -31 + 24
sum('정글', 9); // '정글' + 9
sum('정글', '북'); // '정글' + '북'
```

숫자 값 기대

숫자 값 결과

숫자 값 결과

문자 값 결과

문자 값 결과

return을 만나면, 함수는 바로 종료된다

```
function sum(A, B) {
    if(typeof A !== 'number' && typeof B !== 'number') return;
    console.log(A + B); // A + B 의 합을 출력한다.
}
sum(24, 67); // 24 + 67
sum(-31, 24); // -31 + 24
sum('정글', 9); // '정글' + 9
sum('정글', '북'); // '정글' + '북'
```

숫자 값 결과

숫자 값 결과

실행되지 않는다

실행되지 않는다

console.log()는 연산된 결과 값을 돌려주지 못한다

```
function sum(A, B) {
    if(typeof A != 'number' && typeof B != 'number') return;
    console.log(A + B); // 합의 값을 출력하지만, 값은 못 전한다.
}

var Result = sum(24, 67); // undefined
```

결과반환 X

return은 함수종료이며, 연산된 결과 값을 돌려준다

```
function sum(A, B) {
    if(typeof A != 'number' && typeof B != 'number') return;
    return A + B; // A + B 의 합의 값을 반환한다.
    console.log(A-B); // 이 코드는 실행되지 않는다.
}
var Result = sum(24, 67); // 24 + 67 계산된 값을 대입
```

결과반환

function statement	description
function FnName (){...}	함수선언
function FnName (arg1, arg2){...}	함수 확장
return	결과값 반환
var FnName = function (){...}	함수리터럴(값), 익명함수(이름이 없는 함수)