

# JavaScript를 위한 새로운 코드 커버리지와 Concolic 테스트 기법

배소라 박준영 류석영 교수님

PLRG  
KAIST

2015-01-28

# JavaScript & 테스트



브라우저



Smart TV



Smartphone



Smartwatch

# 얼마나 테스트 해야 하나?

- 코드 커버리지
  - 테스트를 통해 코드가 얼마나 커버되었는지 나타내는 지표

정적으로 타입이 정해지는 언어

C, C++, Java

동적으로 타입이 정해지는 언어

JavaScript

JavaScript 코드도  
기존의 커버리지를 만족하도록  
테스트하면 충분한가?

# 아니다!

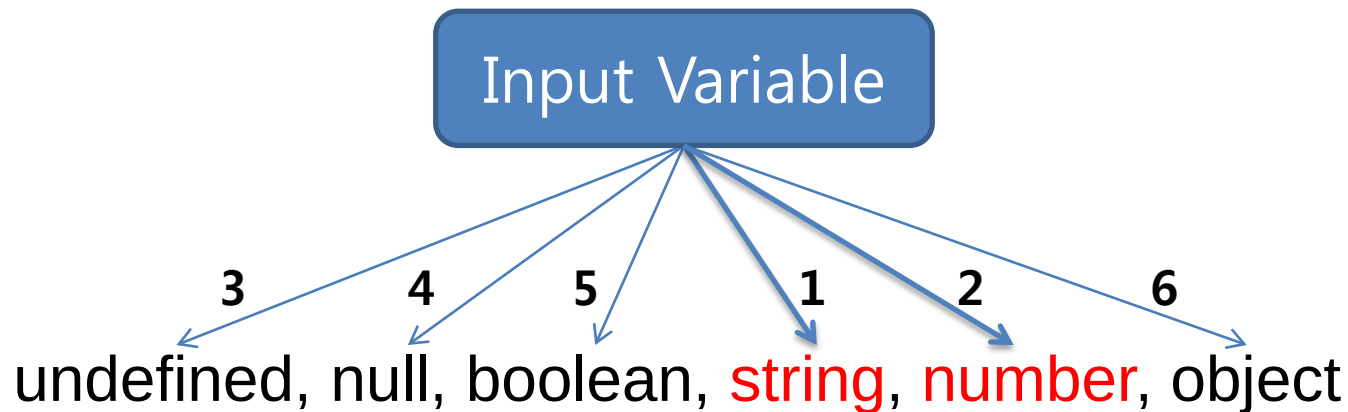
- 타입
  - 타입에 따라 프로그램의 행동이 다름
- Exception
  - C나 Java에서는 compile time에 나타나는 TypeError 등이 JavaScript는 runtime에 발생
  - 실행 환경에 따라 uncaught exception이 발생해도 프로그램이 종료되지 않을 수 있음
- 테스트를 완료해도 발견되지 않은 버그가 있을 수 있음!

# JavaScript를 위한 커버리지

- 타입을 구분
  - undefined, null, boolean, string, number, object
- Exception이 발생할 수 있는 경우까지 포함
- 다음 단계
  - 새로운 커버리지에 concolic 테스트 적용

# 새로운 Concolic 테스트 기법

- 타입 구분으로 인한 탐색 범위 증가
- 코드에 따라 버그를 찾을 가능성이 높은(효과적인) 타입이 존재!
- 효과적인 타입을 정적 분석으로 알아내어 우선적으로 탐색!



- 정적 분석 결과  
Input Variable Type = string, number

# 질의응답

## JavaScript를 위한!

### 새로운 코드 커버리지와 Concolic 테스트 기법

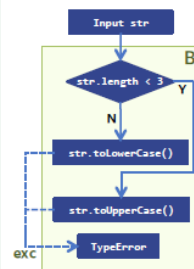
PLRG  
Programming Language  
Research Group  
배소라, 박준영

#### JavaScript 코드도 기존의 커버리지를 만족하도록 테스트하면 충분한가?

```
//str : symbolic variable
function f (str)
{
  if (str.length < 3) {
    return str.toUpperCase();
  } else {
    return str.toLowerCase();
  }
}
```

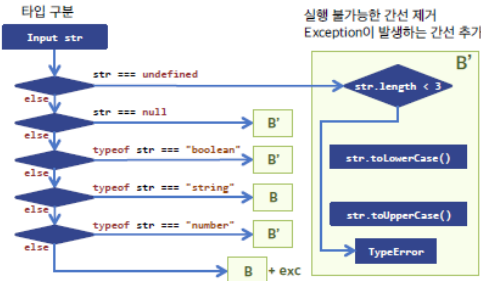
| Branch Coverage<br>모든 조건의 참과 거짓 branch를 실행 |       |        |   |                                                                                   |        |
|--------------------------------------------|-------|--------|---|-----------------------------------------------------------------------------------|--------|
| #                                          | str   | Result | # | str                                                                               | Result |
| 1                                          | ""    | ""     | 3 | {<br>length : -1,<br>toUpperCase : function () {<br>return this.length;<br>}<br>} | -1     |
|                                            | "ABC" | "abc"  |   |                                                                                   |        |
| 2                                          | "ab"  | "AB"   | 3 | {<br>toLowerCase : function () {<br>return 1;<br>}<br>}                           | 1      |
|                                            | "AAA" | "aaa"  |   |                                                                                   |        |

#### 기존 코드 커버리지



장점  
• 프로그램의 더 많은 행동을 커버  
- Type, Exception

#### 새로운 코드 커버리지



단점  
• 실행 불가능한 간선을 제거하기 어려움  
• 생성해야 할 테스트 케이스 양이 많아짐  
- 보완책: 버그를 찾기에 효과적인 테스트 케이스를 우선적으로 생성하는 Concolic 테스트 기법을 제시

#### 새로운 Concolic 테스트 기법

Information = Loc  $\mapsto$  2<sup>Type</sup>

Type ::= undefined  
null  
boolean  
string  
number  
ObjectId

ObjectId = UInt  
ObjectIdMap = ObjectId  $\mapsto$  Object  
Object = String  $\mapsto$  2<sup>Type</sup>  $\times$  Boolean

| # | 호출 코드                                                                                 | 타입 우선순위                                                                                                  |
|---|---------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| 1 | f("ab"); f("AAA");                                                                    | string ...                                                                                               |
| 2 | f({<br>length: 0,<br>toUpperCase: function() {},<br>toLowerCase: function() {}<br>}); | {<br>length: number,<br>toUpperCase: {...Call:...},<br>toLowerCase: {...Call:...}<br>}<br>> object > ... |
| 3 | f(""); f({});                                                                         | string > {} > object > ...                                                                               |