

# 자바스크립트 프로그램의 동시성 오류 검출 기법

---

홍신, 박용배, 김문주

KAIST 전산학과 SWTV Group

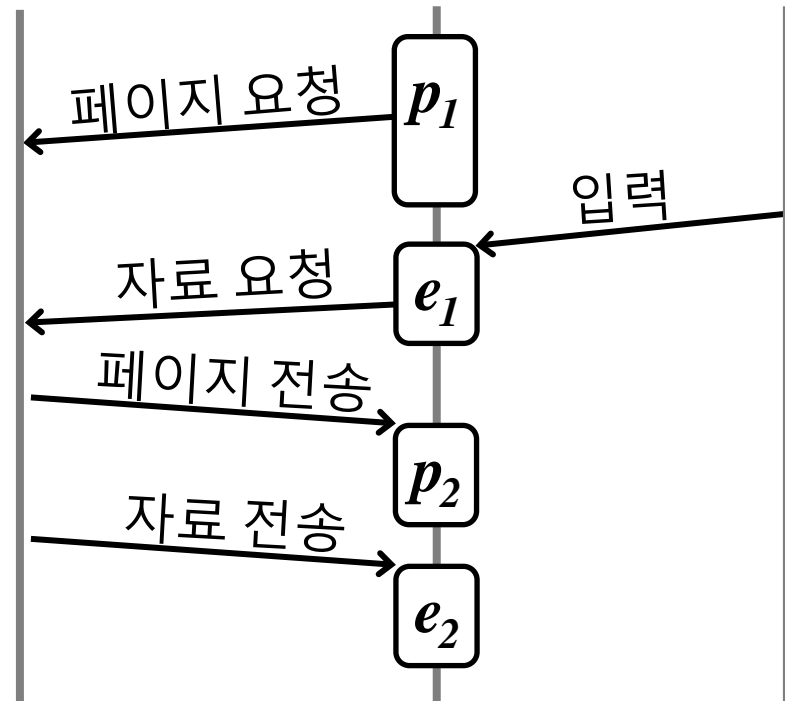
# 자바스크립트 프로그램 동시성 오류

- 대부분 웹 어플리케이션은 사용자 경험을 향상시키기 위해 동적인 내용을 포함
  - 이벤트 기반 실행을 사용하여 사용자와 상호작용
  - 서버와 통신하여 동적으로 페이지 내용을 업데이트
- 자바스크립트 실행은 순차적(sequential)이고 비선점형(non-preemptible)
- 그러나 웹 어플리케이션은 **동시성 프로그램**
  - 웹 어플리케이션에서 자바스크립트의 실행이 여러 번 발생하며, 네트워크, 사용자 입력, 타이머 등의 실행 순서에 따라 실행이 결정됨
- 웹 어플리케이션에서 자바스크립트의 동적인 실행을 통제하지 못하면 **동시성 오류 발생**
  - 동시성 오류는 발견과 재현이 어려움

# 동적인 웹 어플리케이션 실행

- 웹 어플리케이션은 비동기통신(XmlHttpRequest), 사용자 입력, 타이머에 따라서 다양한 순서로 실행
  - 실행은 파싱(parsing)과 이벤트 핸들링으로 구성됨
- 웹 어플리케이션 실행 순서는 **비결정적**(non-deterministic)

네트워크      어플리케이션      사용자

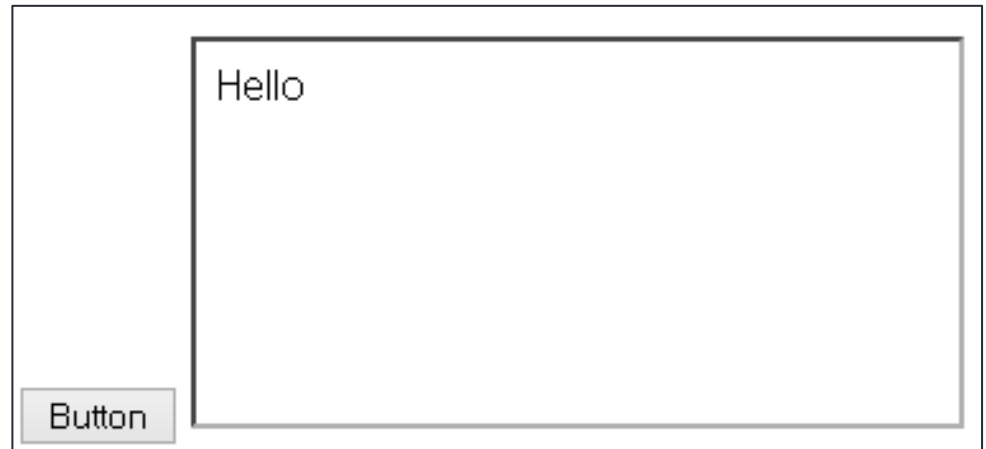


# 동시성 오류 (1/3)

```
<!-- main.html -->
01 <html> <body>
02   <button id="b1" onclick="fn()">
03     Button
04   </button>
05   <iframe src="sub.html" id="i1"/>
06   <script src="lib.js"></script>
07 </body> </html>
```

```
<!-- sub.html -->
11 <html> <body>
12   <div> Hello </div>
13 </body> </html>
```

```
<!-- lib.js -->
21 fn = function() { ... };
```



# 동시성 오류 (2/3)

```

<!-- main.html -->
01 <html> <body>
02   <button id="b1" onclick="fn()">
03     Button
04   </button>
05   <iframe src="sub.html" id="i1"/>
06   <script src="lib.js"></script>
07 </body> </html>

```

```

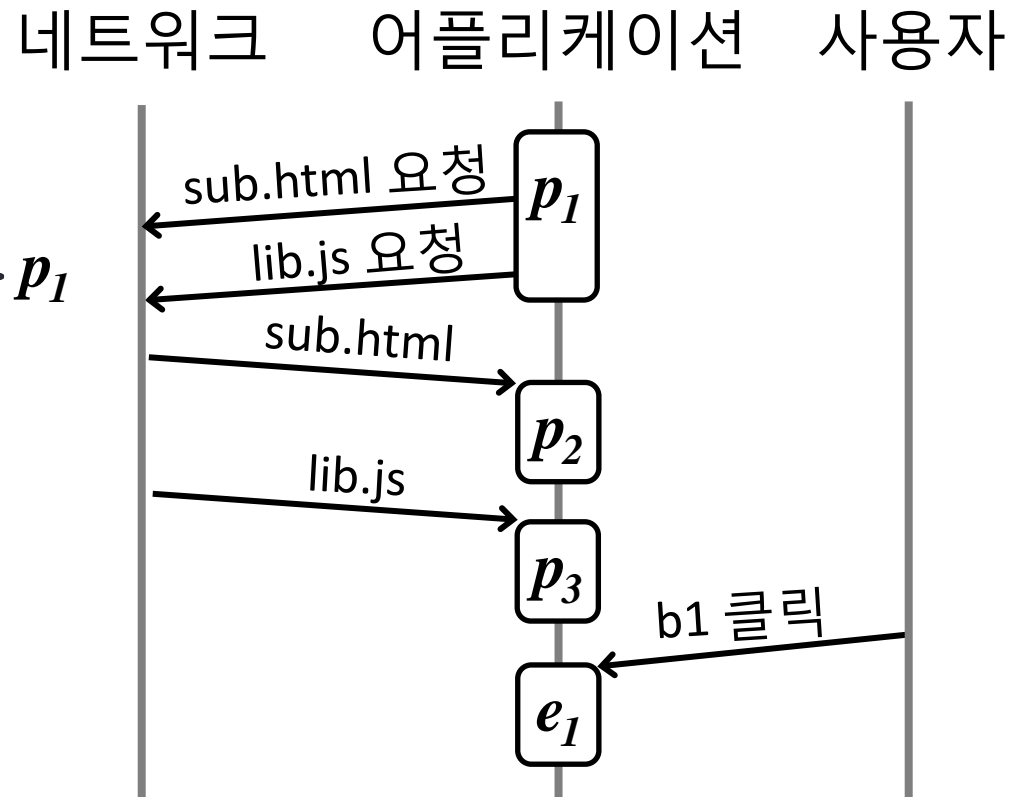
<!-- sub.html -->
11 <html> <body>
12   <div> Hello </div>
13 </body> </html>

```

```

<!-- lib.js -->
21 fn = function() { ... };

```



- $p_1$  실행 후,  $p_2, p_3, e_1$ 이 실행됨
- $p_2, p_3, e_1$ 간의 순서는 변경 가능

# 동시성 오류 (3/3)

```

<!-- main.html -->
01 <html> <body>
02   <button id="b1" onclick="fn()">
03     Button
04   </button>
05   <iframe src="sub.html" id="i1"/>
06   <script src="lib.js"></script>
07 </body> </html>

```

```

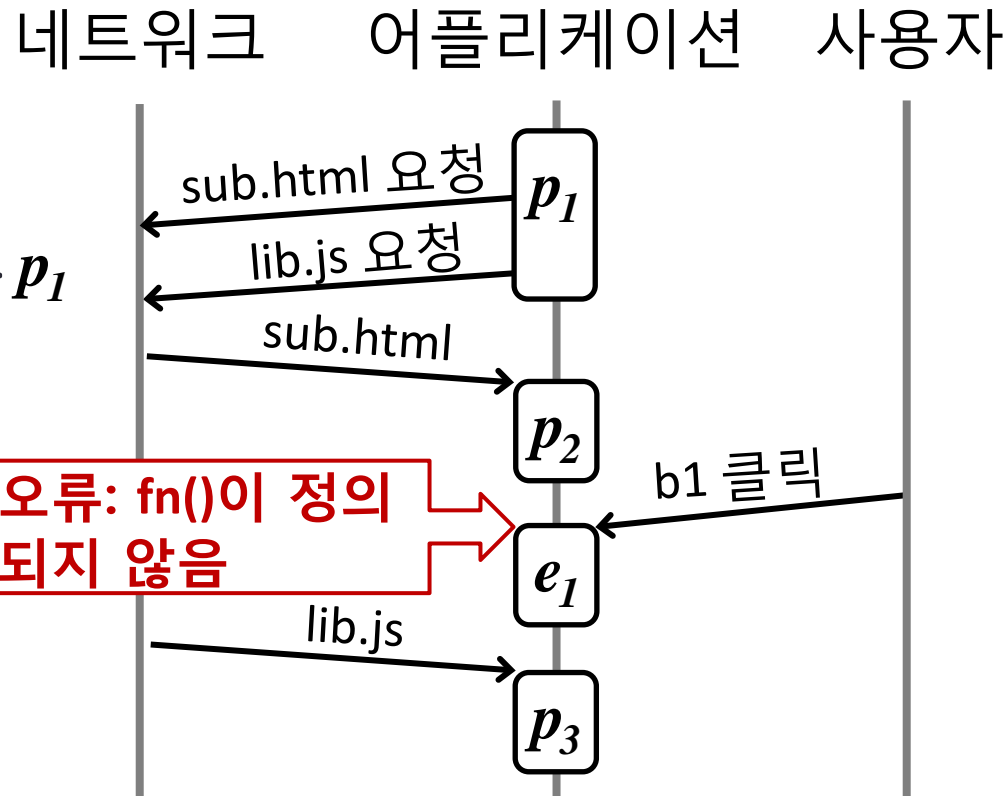
<!-- sub.html -->
11 <html> <body>
12   <div> Hello </div>
13 </body> </html>

```

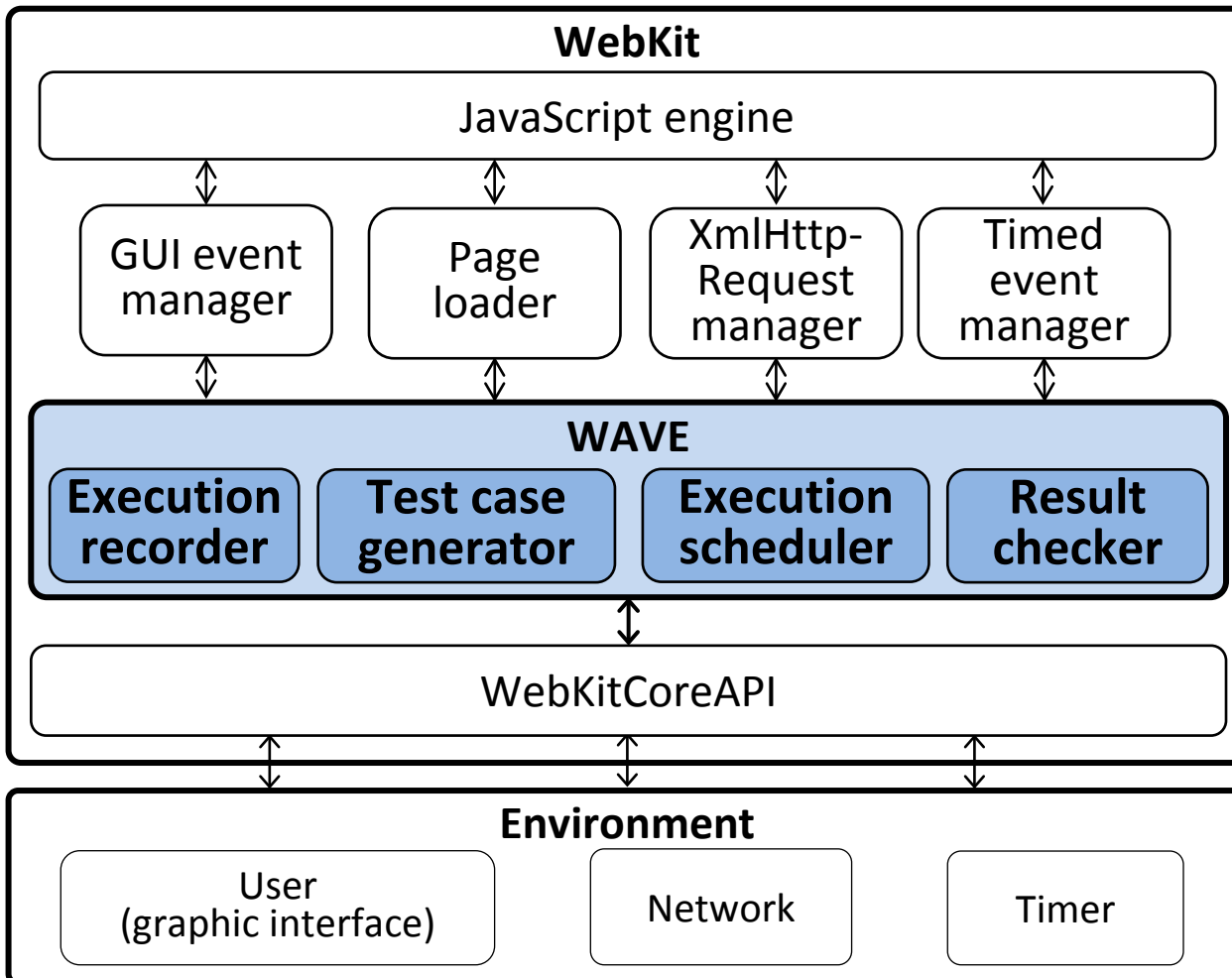
```

<!-- lib.js -->
21 fn = function() { ... };

```



# WAVE: Web Application's Virtual Env.



1. 웹 어플리케이션의 실행을 분석
2. 새로운 테스트 케이스 생성
3. 가상환경하에서 테스트 케이스 실행

# 결과 및 결론

- WAVE [ICST 2014]는 8개의 벤치마크 어플리케이션과 5개의 실제 웹 어플리케이션에 적용
  - AjaXplorer, Feng Office, Gallery 3, TYPO3, WordPress
- 실제 웹 어플리케이션에서 **새로운 동시성 오류 탐지**
- WAVE가 웹 어플리케이션의 동시성 오류를 효과적이고 효율적으로 탐지

### Detecting Concurrency Errors in Client-side JavaScript Web Applications

Shin Hong, Yongbae Park, Prof. Moonzoo Kim @ SWTV Group  
Computer Science Department, KAIST

**Overall research goal**

- We developed an **automated web application test generation technique** to detect **actual concurrency bugs** effectively and efficiently

**Challenge**

- Conventional in-house testing is *not* effective to detect race bugs due to
  - Difficulty to feed various execution scenarios with diverse combination of network comm. and user interactions
- Bug detectors report many false alarms

**Approach**

- Define **race bug patterns** in client-side web app.
- Develop an **automated testing framework** that controls user interactions, network comm., and timer for testing purpose
- Develop **test case prioritization algorithms** to maximize performance in bug detection

**Contribution**

- Developed an **automated testing framework** to detect concurrency bugs in web applications
  - **More effective & efficient** than random testing
  - **More accurate** than bug detectors (no false pos.)
- Detected **5 new bugs in real-world web apps**

**Concurrency bug patterns in web application**

- **Order violation**: unintended race condition of two operations that must be executed one order only, but allow to be in the reverse order
- **Atomicity violation**: unintended race condition that permits an operation to be executed between two operations that should be executed consecutively

**WAVE: Auto. testing framework w/ virtual env.**

1. Monitor web app behavior
2. Predict race bugs
3. Generate test cases
4. Execute test case

**Features**

1. Program analysis for generating only valid test cases
2. Test case prioritization for fast race bug detection
3. Explicit control of user, network, and timed events

**Evaluation**

- Experiment setup
  - Target 5 real apps (AjaXplorer, Feng Office, Gallery3, TYPO3, WordPress) + 8 benchmark apps
  - Compare with 6 random testing techniques (with 3 random delays and 2 web browsers) and EventRacer
- Results
  - WAVE detects **5 new bugs in real web applications**
  - Effectiveness: WAVE **detected race bugs in all 13 apps**, while the random techniques and EventRacer did in 11 apps
  - Efficiency: WAVE **detected race bugs 10x ~ 30x faster** than the random techniques on average
  - Ex. **New bug** detected in WordPress
    - Symptom: failed reaction to user input
    - Bug pattern: order violation

**Application in telecomm. & multimedia**

- WAVE can detect concurrency bugs of web-based multimedia apps written in JavaScript, HTML5, Flash, Silverlight, etc.
- WAVE's virtual environment can be extended to control multiple clients to **generate systematic workload to test a networked system**

Visit <http://swtv.kaist.ac.kr/data/webapp-race> for more information on this project

포스터 세션에서 자세하게 설명해 드립니다