Javascript in Javascipt(js.js)

sandboxing scripts

– SMV 정형검증 Proposal –

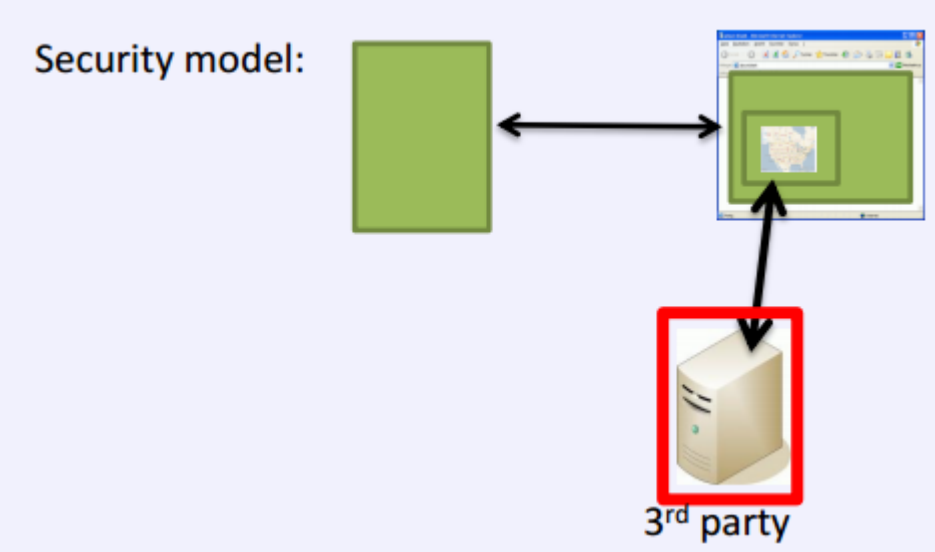
HPE LAB
남 현 우
2017.06.06

Contents

소개

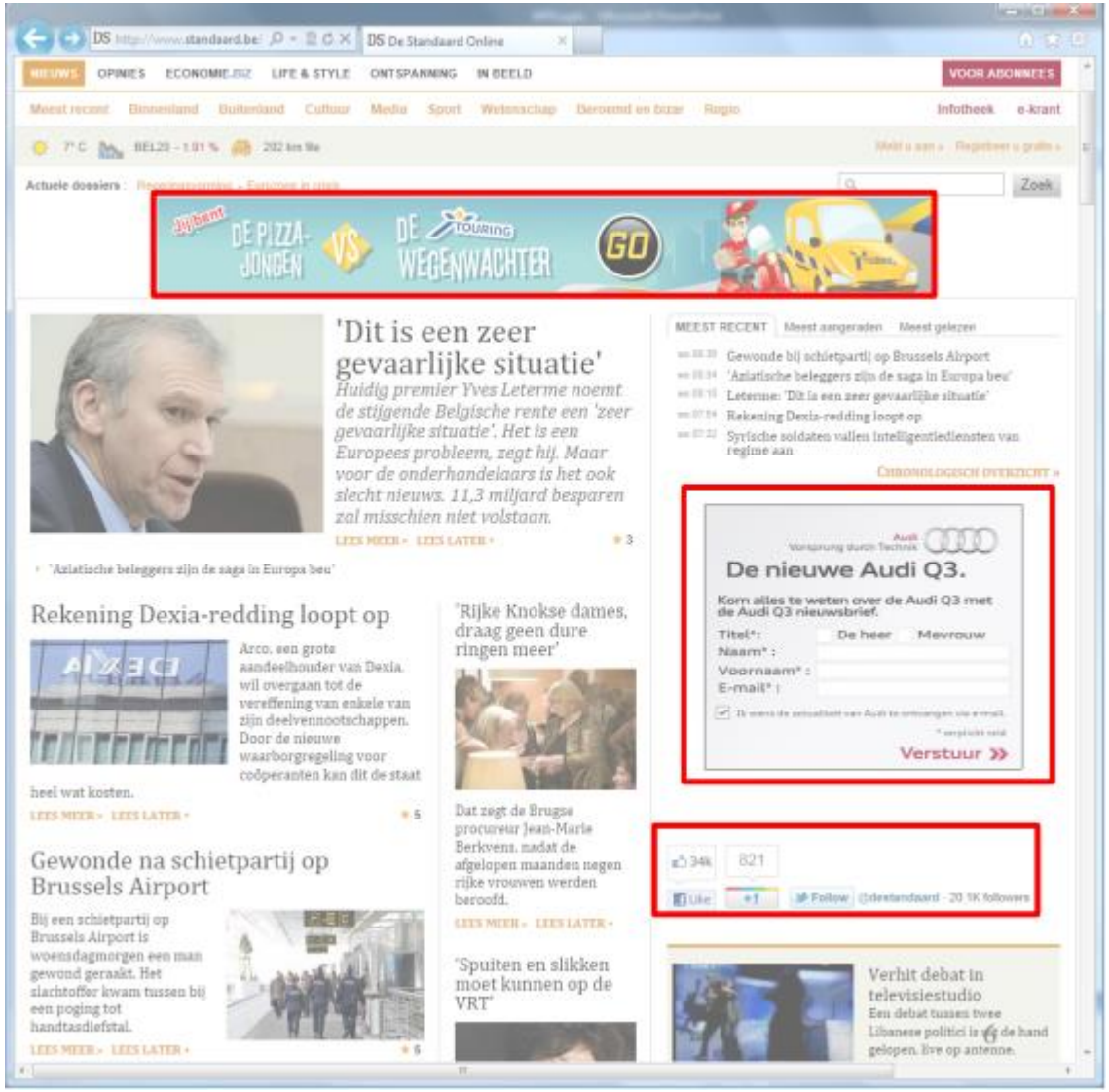
- **Javascript Sandboxing?**
  - 샌드박스(Sandbox)란 외부로부터 들어온 프로그램이 보호된 영역에서 동작해 시스템이 부정하게 조작되는 것을 막는 보안 형태.
  - 사용 예
    - 외부 라이브러리 코드의 실행 제한

```
<html><body>
…
<script src="http://3rdparty.com/script.js"></script>
…
</body></html>
```



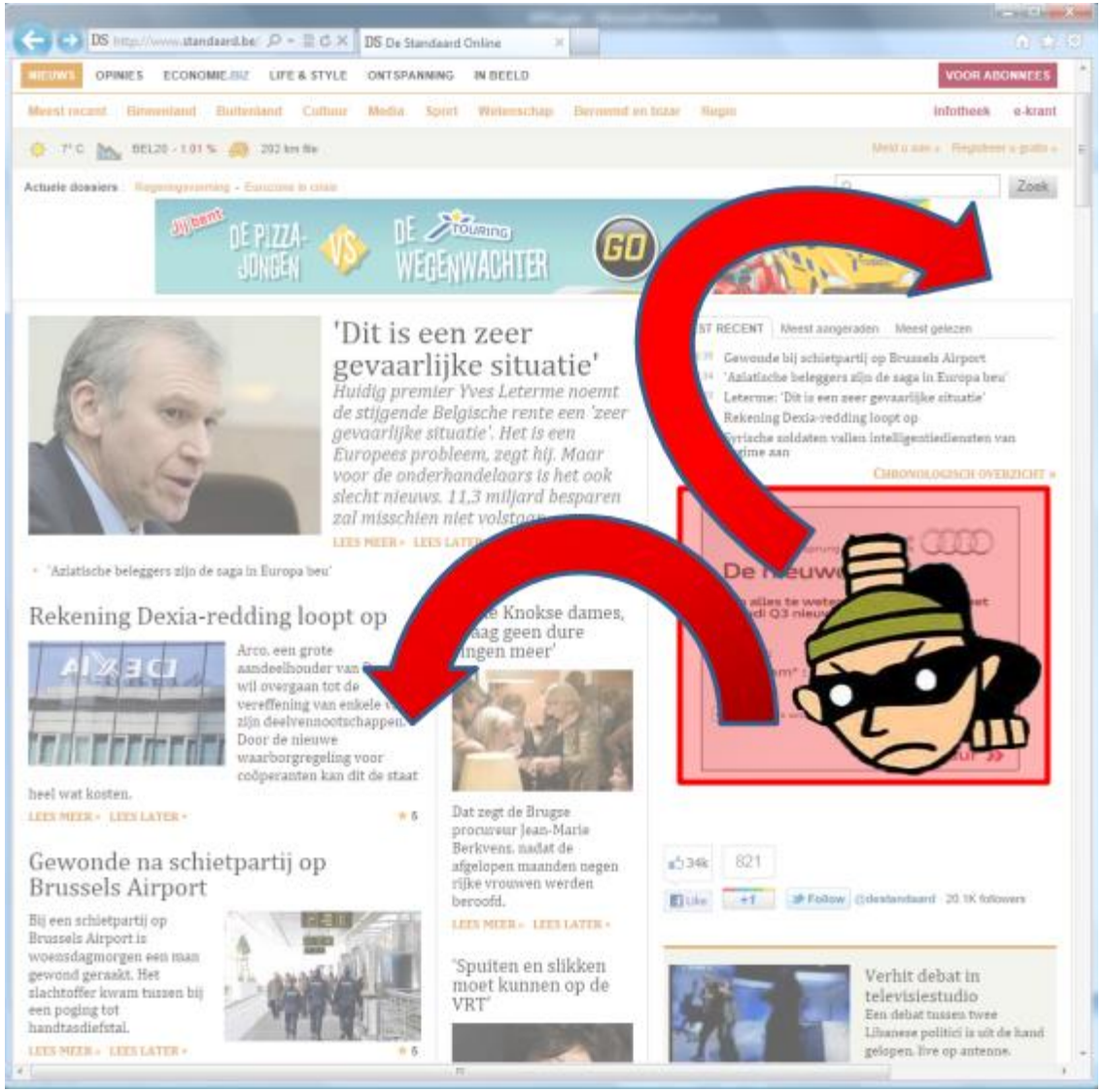
소개

- **Javascript Sandboxing?**
  - Advertisements
    - Adhese ad network
  - Social web
    - Facebook Connect
    - Google+
    - Twitter
    - Feeds burner
  - Tracking
    - Scorecard research
  - Web Analytics
    - Yahoo! Web Analytics
    - Google Analytics



소개

- **Javascript Sandboxing?**
  - 웹 애플리케이션 공격
  - 웹 페이지의 소스코드 영역은 공유하기 때문에 제3자의 공격에 노출되어 있음



시스템 구성

- **js.js 구성도**
  - Js.js Library
    - Sandboxed Script
    - Virtual DOM
  - Mediator
    - Sandbox 환경을 제어하기 위한 컨트롤러 역할
    - Sandboxing API
  - DOM
    - HTML 문서를 위한 프로그래밍 인터페이스를 말하며, 문서의 구조화된 표현양식을 트리형태로 제공.
    - 웹페이지와 프로그래밍 언어를 결합

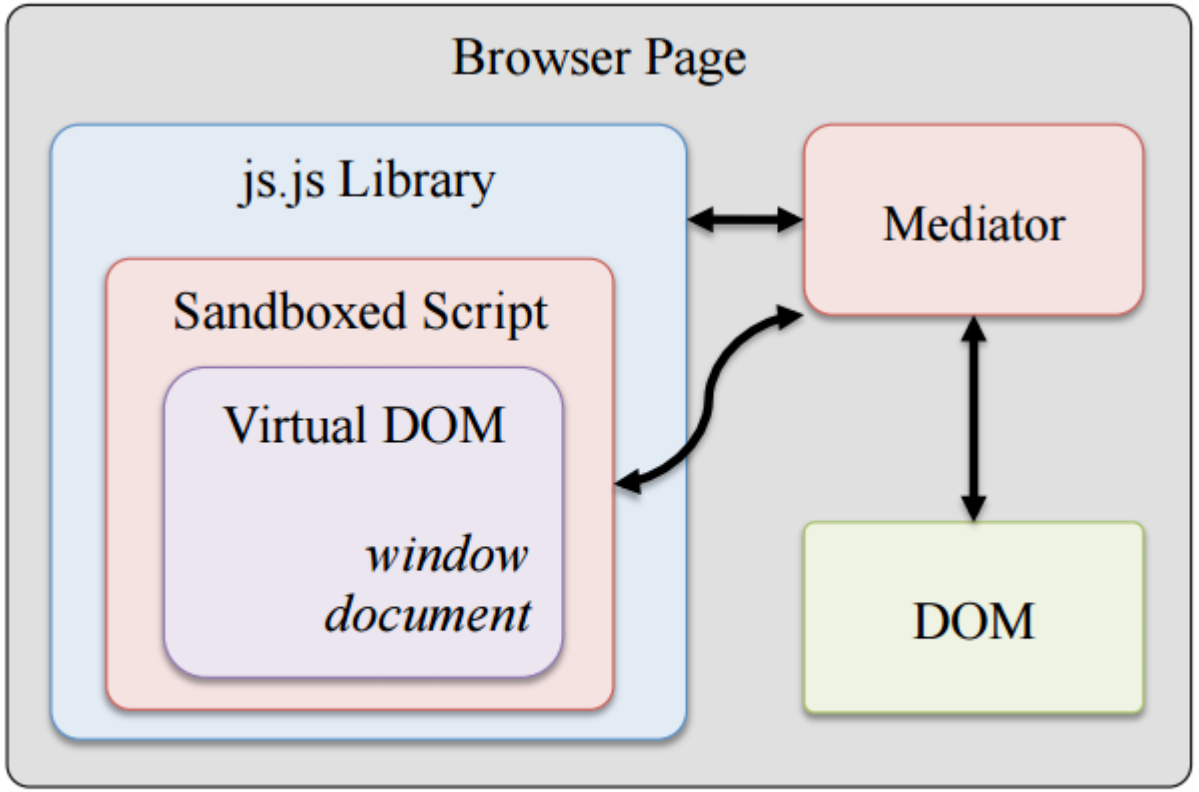


Figure 1: js.js architecture for example application

시스템 구성

- **js.js API**
  - **EvaluateScript** executes the script in sandboxing environment.
  - **ValueToNumber** converts the result of evaluating the expression to native number.

```
var src = "nativeAdd(17, 2.4);";
var jsObjs = JSJS.Init();

function nativeAdd(d1, d2) {
  return d1 + d2;
}

var dblType = JSJS.Types.double;
var wrappedNativeFunc = JSJS.wrapFunction({
  func: nativeAdd,
  args: [dblType, dblType],
  returns: dblType});

JSJS.DefineFunction(jsObjs.cx, jsObjs.glob,
    "nativeAdd", wrappedNativeFunc, 2, 0);

var rval = JSJS.EvaluateScript(jsObjs.cx,
    jsObjs.glob, src);

//Convert result to native value
var d = rval && JSJS.ValueToNumber(jsObjs.cx
    , rval);
```

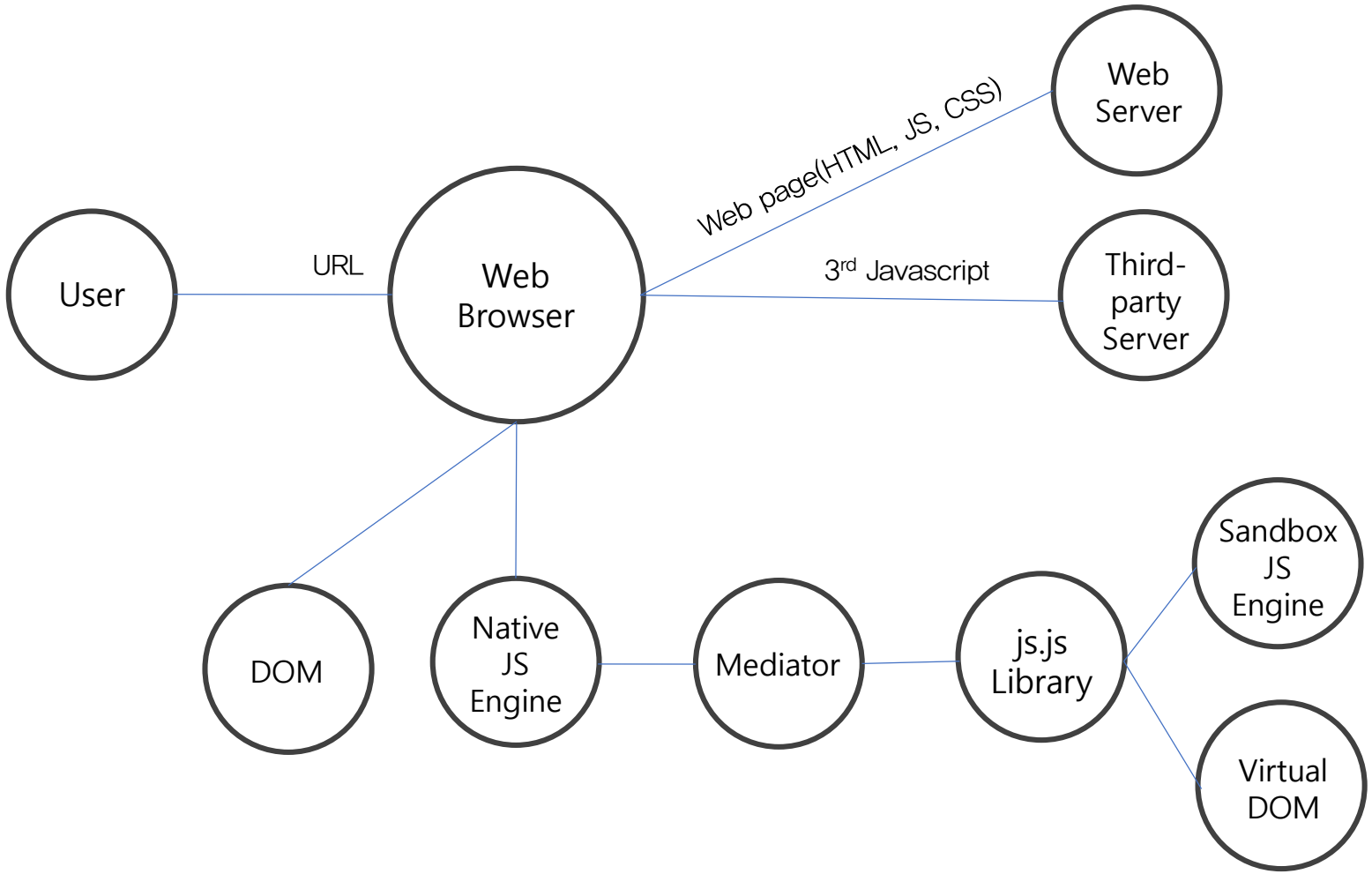
Figure 2: Example of binding a native function to the global object space of a sandboxed script.

검증 요구사항

- Sandbox 내부의 코드는 메인 JS 엔진 메모리 영역(run, read, write)에 접근할 수 없다.
- Sandbox 내부의 코드는 모든 Javascript 스펙을 모두 동일하게 지원한다.
- Page redirection, Spin loop, Memory exhaustion과 같은 공격에 대처 가능하다.

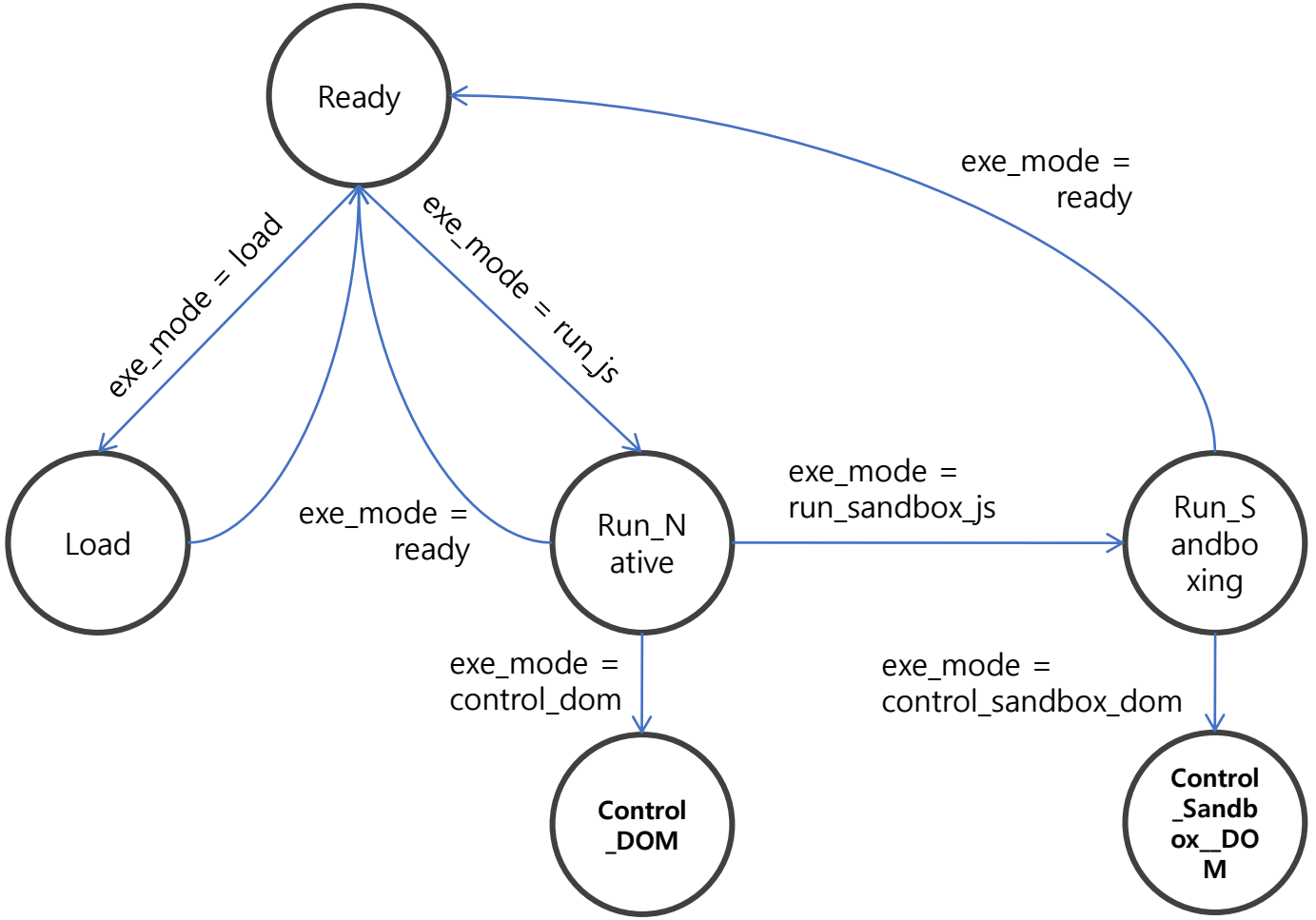
시스템 구성

- 전체 구성도


Web Server
Web page(HTML, JS, CSS)
User
URL
Web Browser
3rd Javascript
Third-party Server
DOM
Native JS Engine
Mediator
js.js Library
Sandbox JS Engine
Virtual DOM

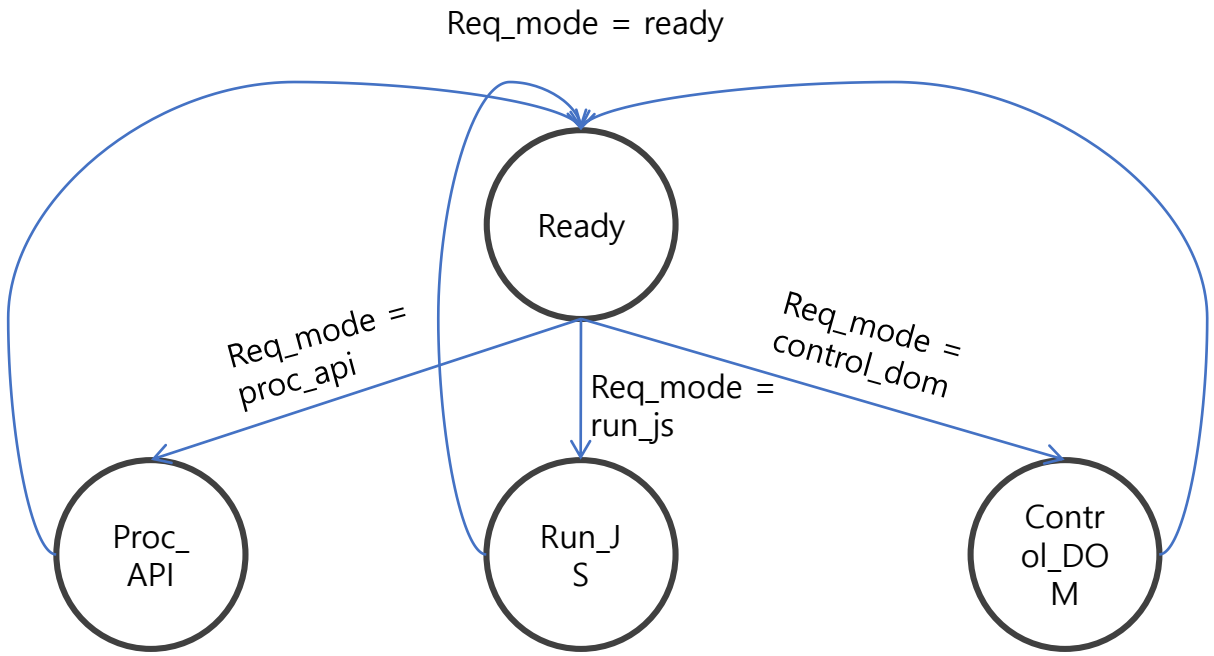
Automata

- Web Browser



Automata

- Mediator – js.js Library



Property

-- 1. Sandbox 내부의 코드는 메인 JS 엔진 메모리 영역(run, read, write)에 접근할 수 없다.

SPEC AG !(nativeJsEngine.reqExcuteBuffer=sandboxJsEngine.src)

SPEC AG !(nativeJsEngine.reqReadBuffer=sandboxJsEngine.reqReadAddr)

SPEC AG !(nativeJsEngine.reqWriteBuffer=sandboxJsEngine.reqWriteAddr)

-- 2. Sandbox Javascript 엔진도 모든 Javascript 스펙을 모두 동일하게 지원한다

SPEC AG (nativeJsEngine. ????)

-- 3. Page redirection, Spin loop, Memory exhaustion과 같은 공격에 대처 가능하다.

SPEC AG((sandboxJsEngine.req_func=check_pageredirection))

SPEC AG((sandboxJsEngine.req_func=check_spinloop))

SPEC AG((sandboxJsEngine.req_func=check_memory_exhaustion))

감사합니다.