Towards Analysis and Bug Finding of JavaScript Web Applications in the Wild

Sukyoung Ryu
KAIST

Jihyeok Park
KAIST

Joonyoung Park
KAIST

28 Sept. 2018

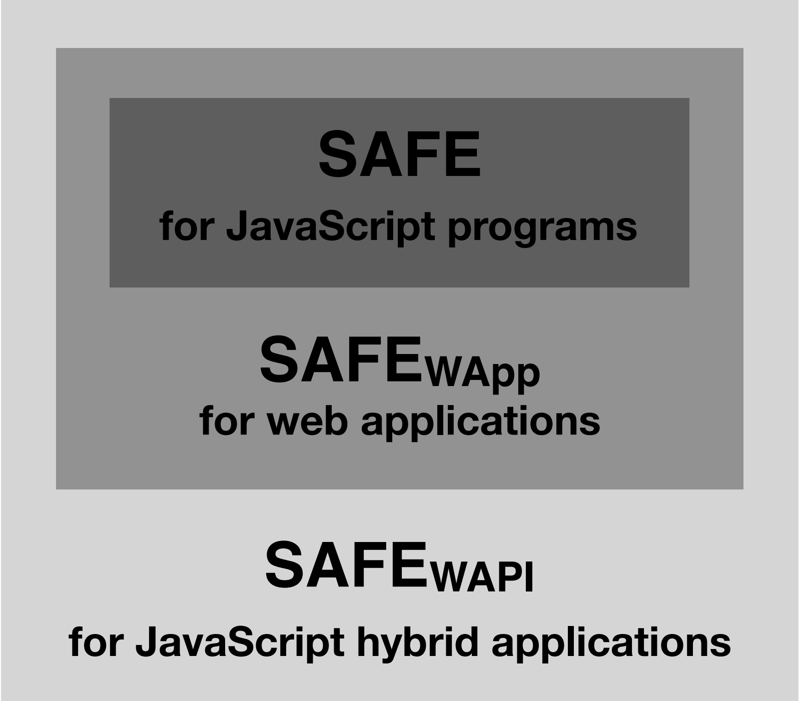
JavaScript

- **Expressivity**
  - First-class functions
  - Dynamic code generations
- **Portability**
  - Web browsers
  - Smart devices

Bugs in JavaScript

- **Loosely typed language**
  - Type-related run-time errors
- **Third-party libraries**
  - Vulnerable to security attacks

SAFE Family



SAFE

for JavaScript programs

SAFE$_{\text{WApp}}$

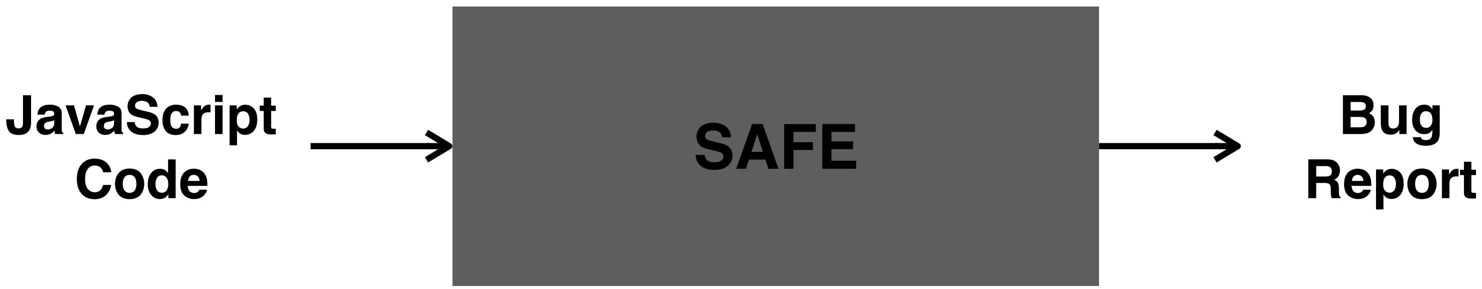
for web applications

SAFE$_{\text{WAPI}}$

for JavaScript hybrid applications

To develop a tool that can
analyze and detect bugs
in *real-world JavaScript web applications*

Analysis of JavaScript Program



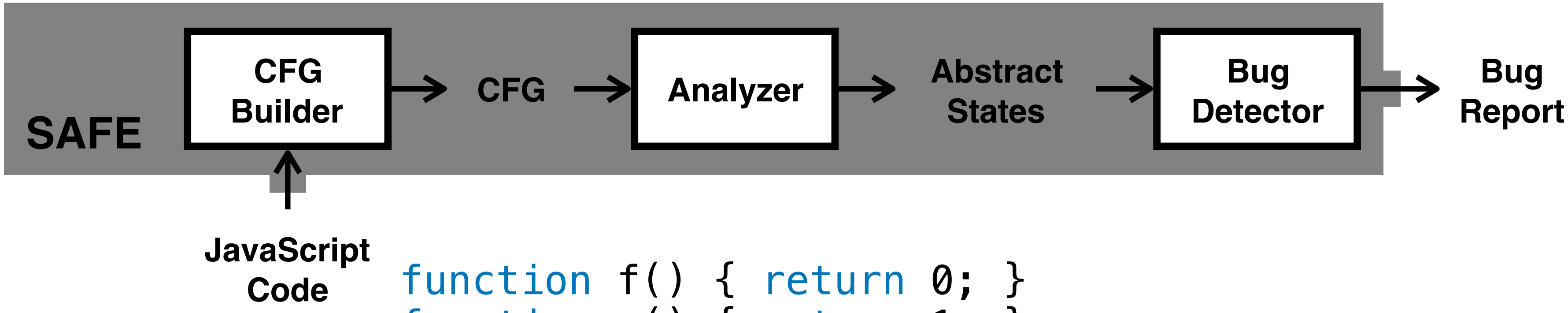
JavaScript Code

SAFE

Bug Report

* H. Lee, S. Won, J. Jin, J. Cho, and S. Ryu, SAFE: Formal specification and implementation of a scalable analysis framework for ECMAScript
* C. Park, H. Lee, and S. Ryu, All about the with statement in JavaScript: Removing with statements in JavaScript applications
* C. Park and S. Ryu, Scalable and precise static analysis of JavaScript applications via loop-sensitivity
* C. Park, H. Im, and S. Ryu, Precise and scalable static analysis of jQuery using a regular expression domain

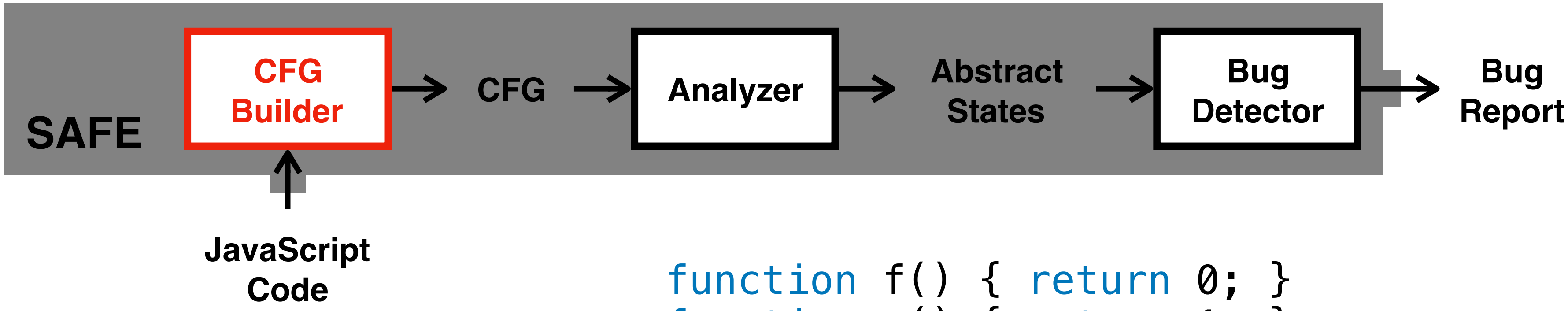
Analysis of JavaScript Program



```
function f() { return 0; }
function g() { return 1; }
function h() { return 2; }

var o = { a : 0, b : 1, c : 2 };

with(o) {
  a = f; b = g; c = h;
};

for (name in o) {
  eval("o[name] = o[name]();");
}
```

* H. Lee, S. Won, J. Jin, J. Cho, and S. Ryu, SAFE: Formal specification and implementation of a scalable analysis framework for ECMAScript

Analysis of JavaScript Program

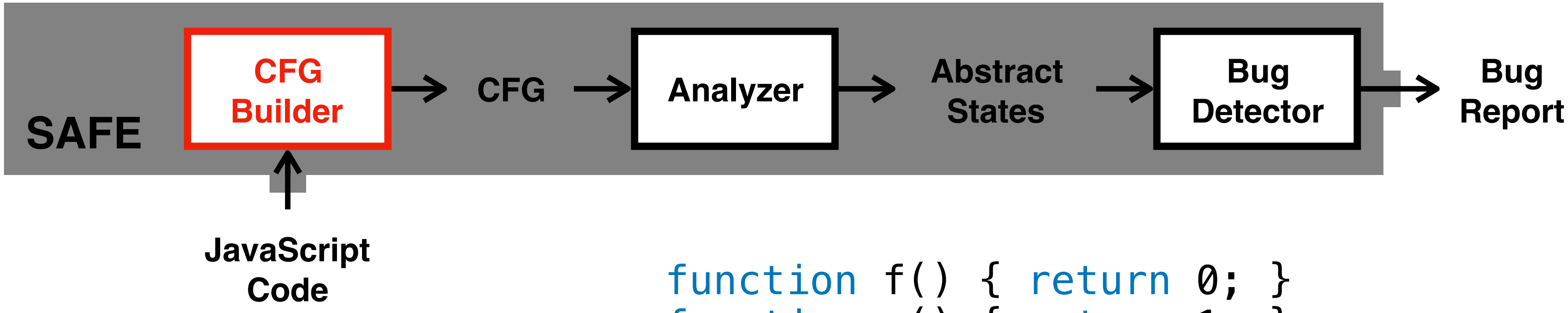


1. Dynamic code generation
2. Dynamic scoping via `with` statements
3. Join of analysis results for loops
4. First-class property names

```
function f() { return 0; }
function g() { return 1; }
function h() { return 2; }

var o = { a : 0, b : 1, c : 2 };

with(o) {
  a = f; b = g; c = h;
};

for (name in o) {
  eval("o[name] = o[name]();");
}
```

* H. Lee, S. Won, J. Jin, J. Cho, and S. Ryu, SAFE: Formal specification and implementation of a scalable analysis framework for ECMAScript

Analysis of JavaScript Program

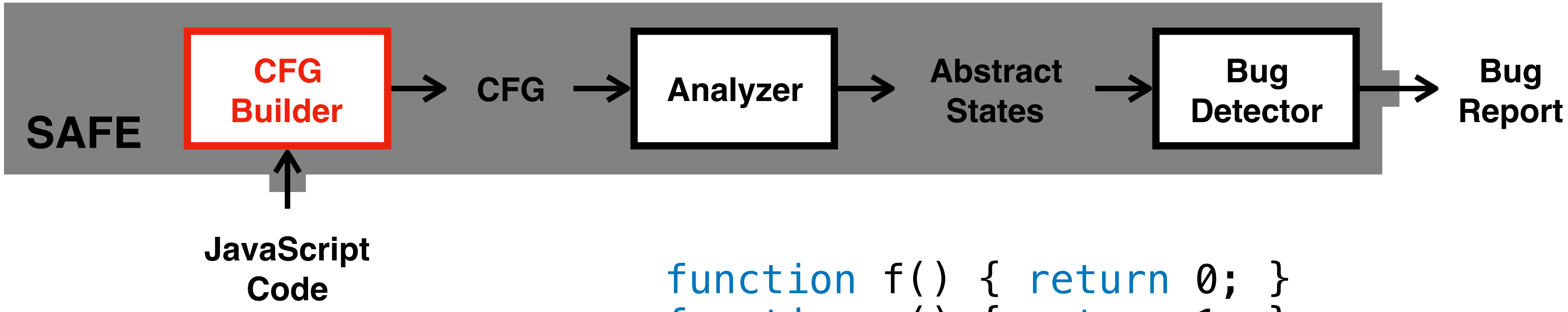


1. Dynamic code generation
2. Dynamic scoping via `with` statements
3. Join of analysis results for loops
4. First-class property names

```
function f() { return 0; }
function g() { return 1; }
function h() { return 2; }

var o = { a : 0, b : 1, c : 2 };

with(o) {
  a = f; b = g; c = h;
};

for (name in o) {
  o[name] = o[name]();
}
```

* H. Lee, S. Won, J. Jin, J. Cho, and S. Ryu, SAFE: Formal specification and implementation of a scalable analysis framework for ECMAScript

Analysis of JavaScript Program

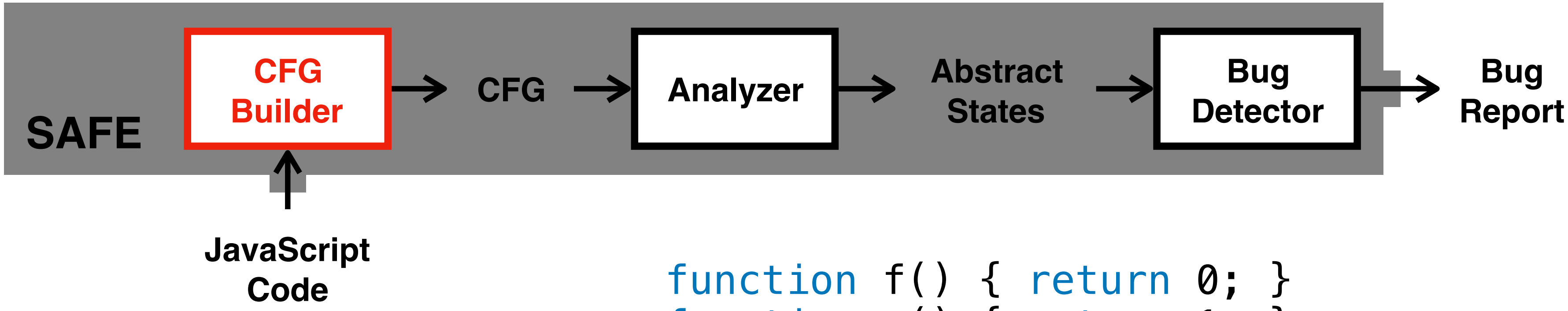


1. Dynamic code generation
2. Dynamic scoping via `with` statements
3. Join of analysis results for loops
4. First-class property names

```
function f() { return 0; }
function g() { return 1; }
function h() { return 2; }

var o = { a : 0, b : 1, c : 2 };

with(o) {
  a = f; b = g; c = h;
};

for (name in o) {
  o[name] = o[name]();
}
```

* C. Park, H. Lee, and S. Ryu, All about the with statement in JavaScript: Removing with statements in JavaScript applications

Analysis of JavaScript Program

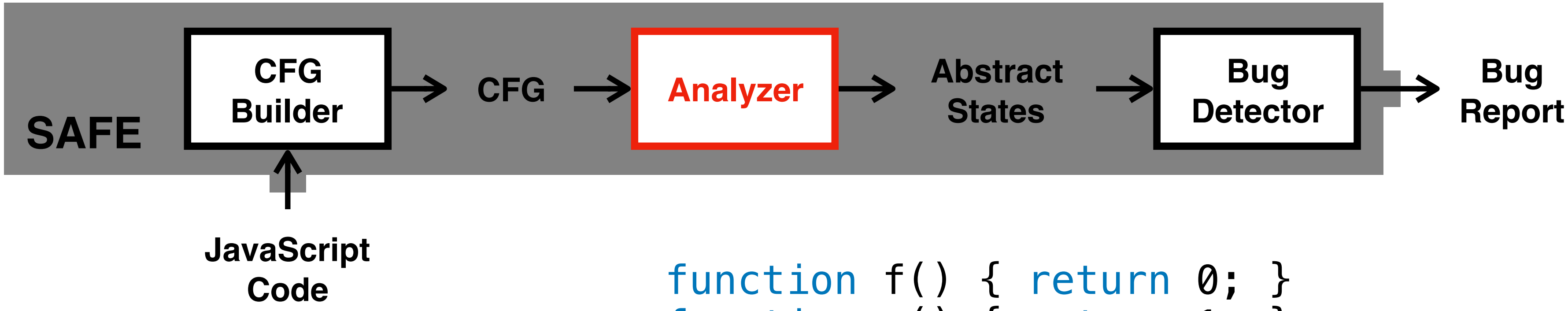


1. Dynamic code generation
2. Dynamic scoping via `with` statements
3. Join of analysis results for loops
4. First-class property names

```
function f() { return 0; }
function g() { return 1; }
function h() { return 2; }

var o = { a : 0, b : 1, c : 2 };

o.a = f;
o.b = g;
o.c = h;

for (name in o) {
  o[name] = o[name]();
}
```

* C. Park, H. Lee, and S. Ryu, All about the with statement in JavaScript: Removing with statements in JavaScript applications

Analysis of JavaScript Program

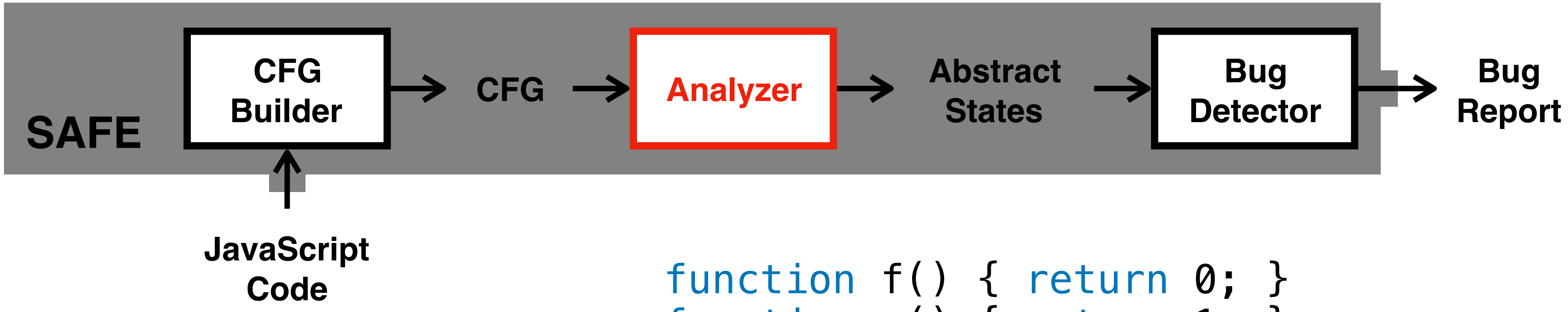


```
function f() { return 0; }
function g() { return 1; }
function h() { return 2; }

var o = { a : 0, b : 1, c : 2 };

o.a = f;
o.b = g;
o.c = h;

for (name in o) {
  o[name] = o[name]();
}
```

1. Dynamic code generation
2. Dynamic scoping via `with` statements
3. Join of analysis results for loops
4. First-class property names

* C. Park and S. Ryu, Scalable and precise static analysis of JavaScript applications via loop-sensitivity

Analysis of JavaScript Program



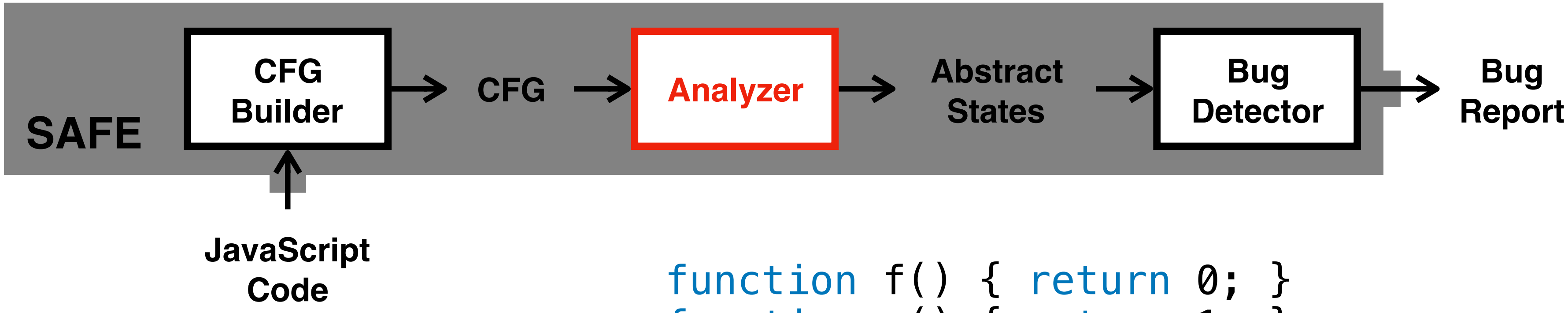
```
function f() { return 0; }
function g() { return 1; }
function h() { return 2; }

var o = { a : 0, b : 1, c : 2 };

o.a = f;
o.b = g;
o.c = h;

for (name in o) {
  o[name] = o[name]();
}
```

loop sensitive analysis (LSA)

1. Dynamic code generation
2. Dynamic scoping via `with` statements
3. Join of analysis results for loops
4. First-class property names

* C. Park and S. Ryu, Scalable and precise static analysis of JavaScript applications via loop-sensitivity

Analysis of JavaScript Program

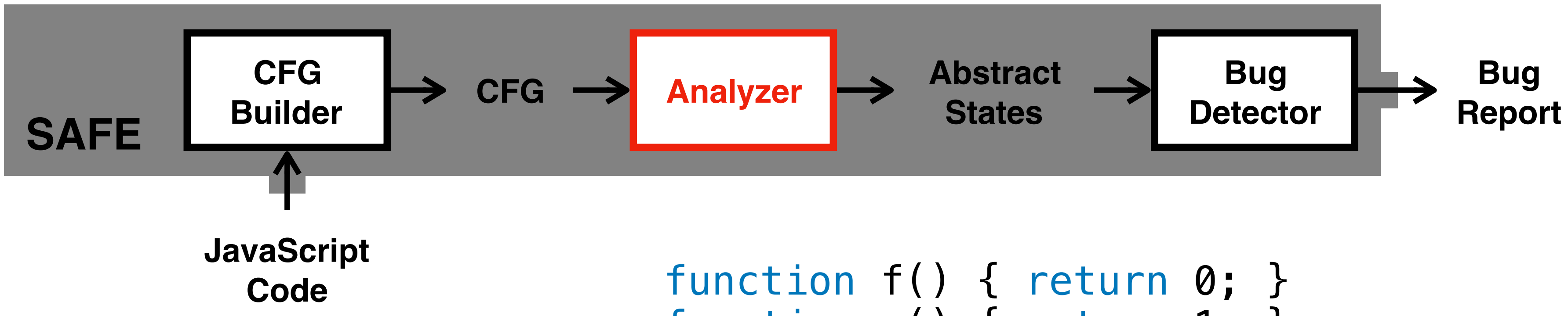


1. Dynamic code generation
2. Dynamic scoping via `with` statements
3. Join of analysis results for loops
4. First-class property names

```
function f() { return 0; }
function g() { return 1; }
function h() { return 2; }

var o = { a : 0, b : 1, c : 2 };

o.a = f;
o.b = g;
o.c = h;

for (name in o) {
  o[name] = o[name]();
}
```

* C. Park, H. Im, and S. Ryu, Precise and scalable static analysis of jQuery using a regular expression domain

Analysis of JavaScript Program



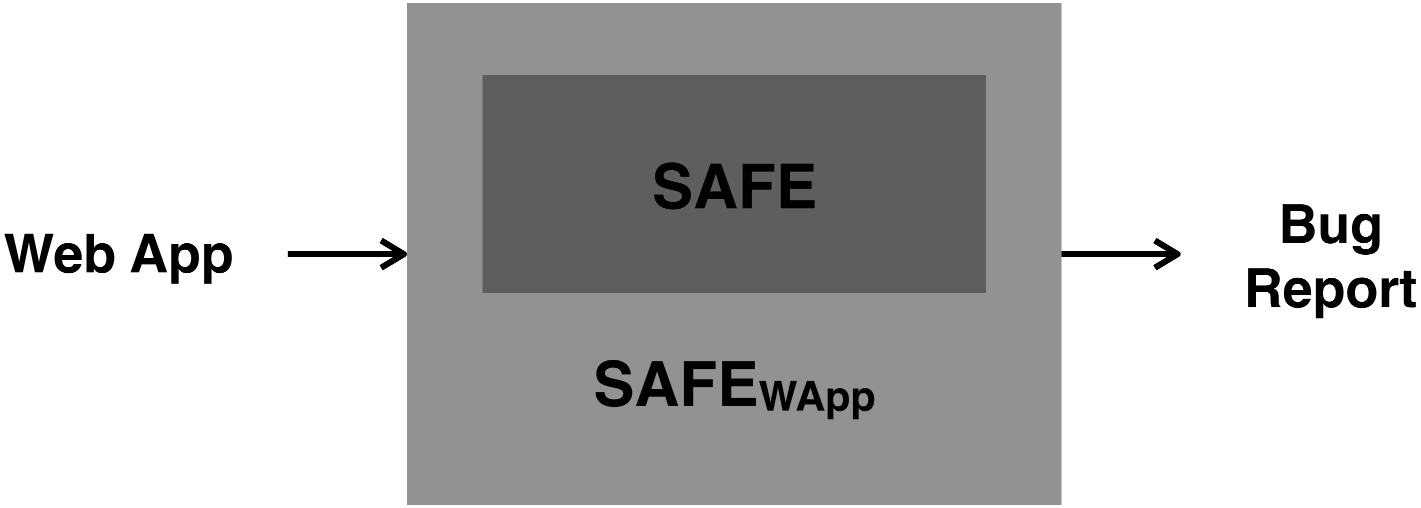
1. Dynamic code generation
2. Dynamic scoping via `with` statements
3. Join of analysis results for loops
4. First-class property names

```
function f() { return 0; }
function g() { return 1; }
function h() { return 2; }

var o = { a : 0, b : 1, c : 2 };

o.a = f;
o.b = g;
o.c = h;

for (name in o) {
  o[name] = o[name]();
}
```

regular expression domain

* C. Park, H. Im, and S. Ryu, Precise and scalable static analysis of jQuery using a regular expression domain

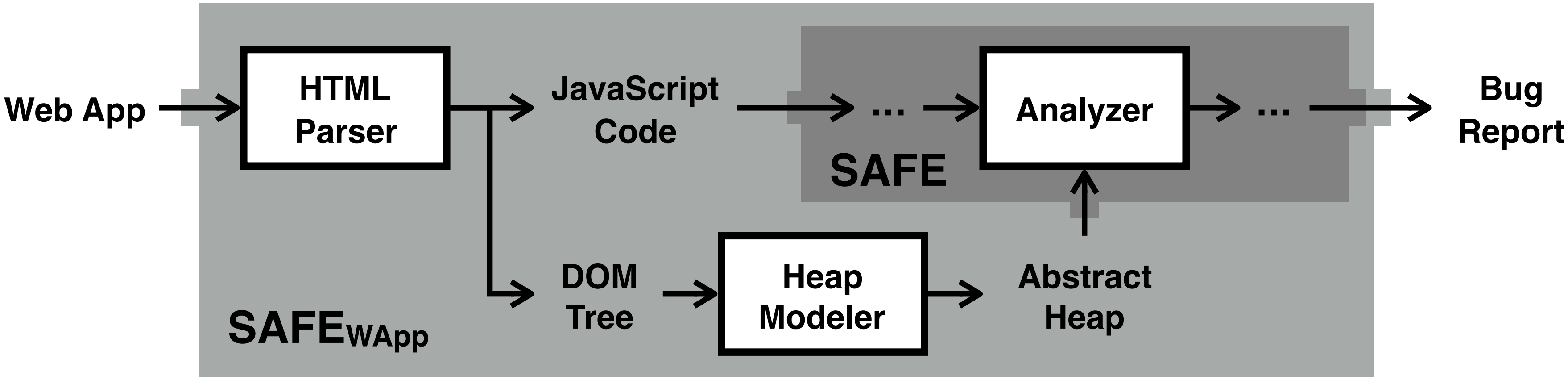
Analysis of Web Applications



* C. Park, S. Won, J. Jin, and S. Ryu, Static analysis of JavaScript web applications in the wild via practical DOM modeling

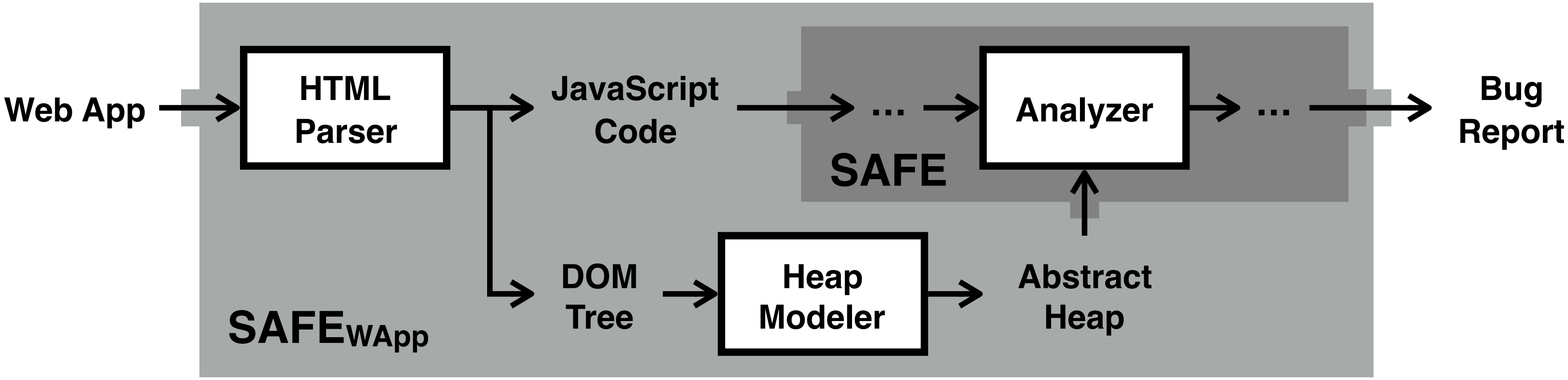
* J. Park, I. Lim, and S. Ryu, Battles with false positives in static analysis of JavaScript web applications in the wild

Analysis of Web Applications



```
<script>
var app = chrome.app;
function isDiv(elem) {
  var t = elem.tagName;
  return t.match(/^\w+/) === "DIV";
}
</script>

<div onclick="isDiv(this)">foo</div>
<p onclick="isDiv(this)">bar</p>
```

Analysis of Web Applications

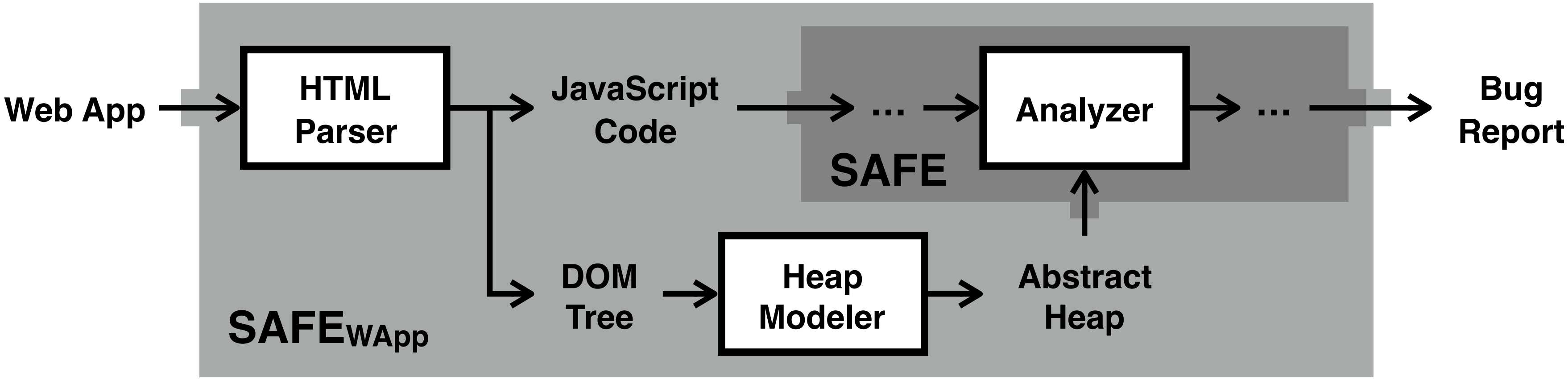


1. DOM structures
2. Interactions with JS
3. Browser-specific APIs
4. Dynamic file loading

```
<script>
var app = chrome.app;
function isDiv(elem) {
  var t = elem.tagName;
  return t.match(/^\w+/) === "DIV";
}
</script>

<div onclick="isDiv(this)">foo</div>
<p onclick="isDiv(this)">bar</p>
```

* C. Park, S. Won, J. Jin, and S. Ryu, Static analysis of JavaScript web applications in the wild via practical DOM modeling

Analysis of Web Applications

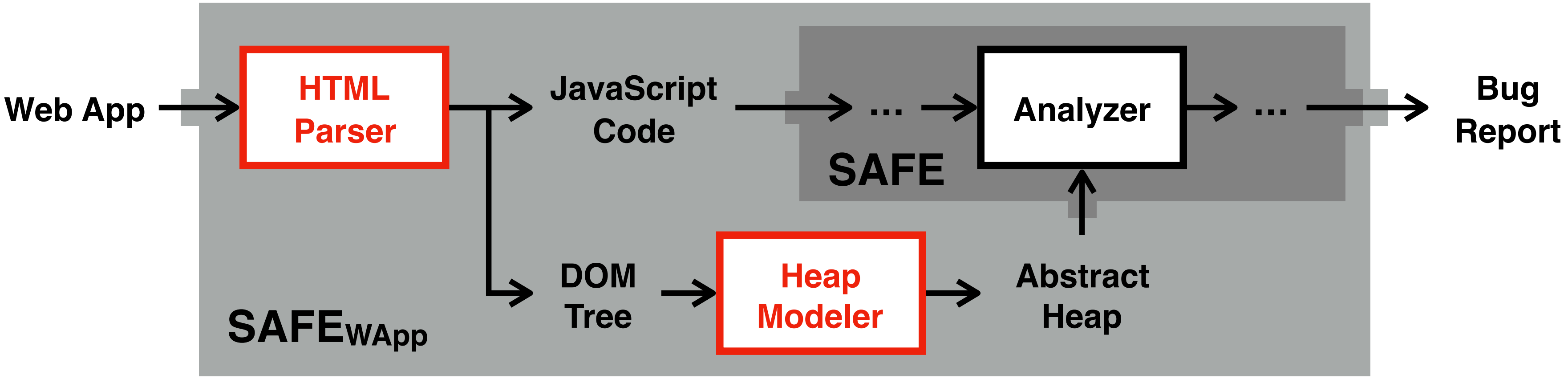


1. DOM structures
2. Interactions with JS
3. Browser-specific APIs
4. Dynamic file loading

```
<script>
var app = chrome.app;
function isDiv(elem) {
  var t = elem.tagName;
  return t.match(/^\w+/) === "DIV";
}
</script>

<div onclick="isDiv(this)">foo</div>
<p onclick="isDiv(this)">bar</p>
```

* C. Park, S. Won, J. Jin, and S. Ryu, Static analysis of JavaScript web applications in the wild via practical DOM modeling

Analysis of Web Applications



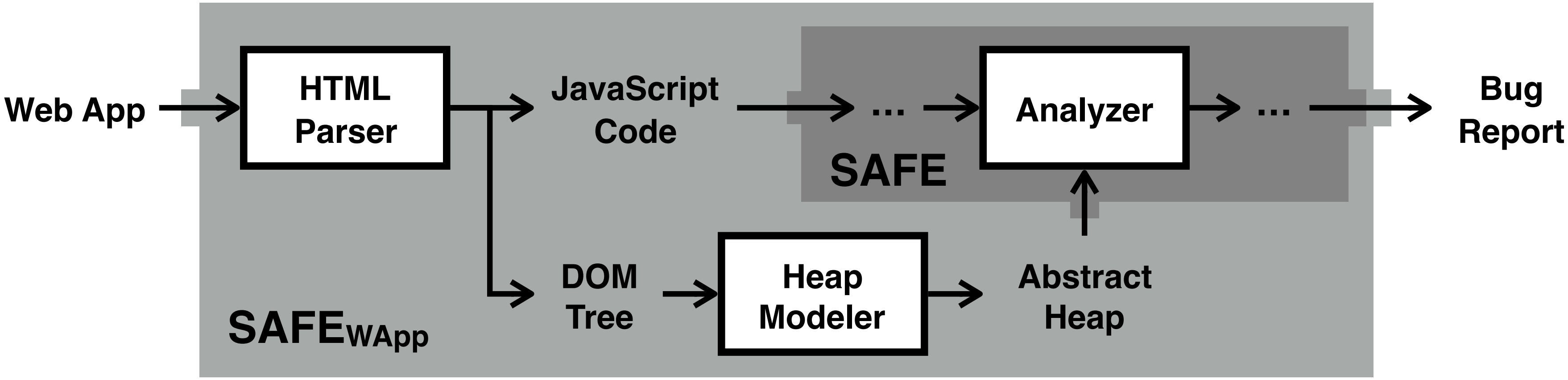
modeling DOM objects / APIs

1. DOM structures
2. Interactions with JS
3. Browser-specific APIs
4. Dynamic file loading

```
<script>
var app = chrome.app;
function isDiv(elem) {
  var t = elem.tagName;
  return t.match(/^\w+/) === "DIV";
}
</script>

<div onclick="isDiv(this)">foo</div>
<p onclick="isDiv(this)">bar</p>
```

* C. Park, S. Won, J. Jin, and S. Ryu, Static analysis of JavaScript web applications in the wild via practical DOM modeling

Analysis of Web Applications

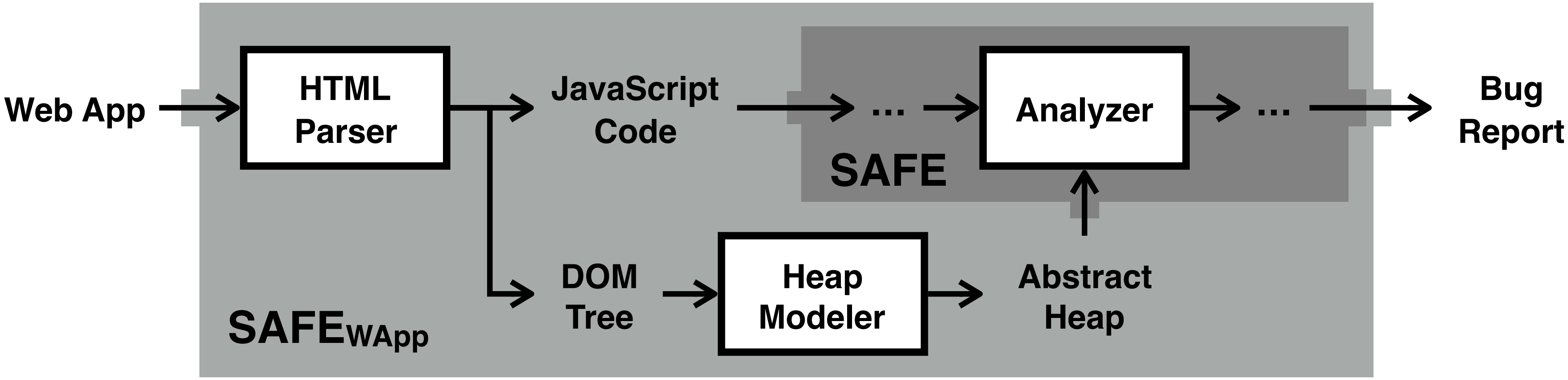


1. DOM structures
2. Interactions with JS
3. Browser-specific APIs
4. Dynamic file loading

```
<script>
var app = chrome.app;
function isDiv(elem) {
  var t = elem.tagName;
  return t.match(/^\w+/) === "DIV";
}
</script>

<div onclick="isDiv(this)">foo</div>
<p onclick="isDiv(this)">bar</p>
```

* J. Park, I. Lim, and S. Ryu, Battles with false positives in static analysis of JavaScript web applications in the wild

Analysis of Web Applications

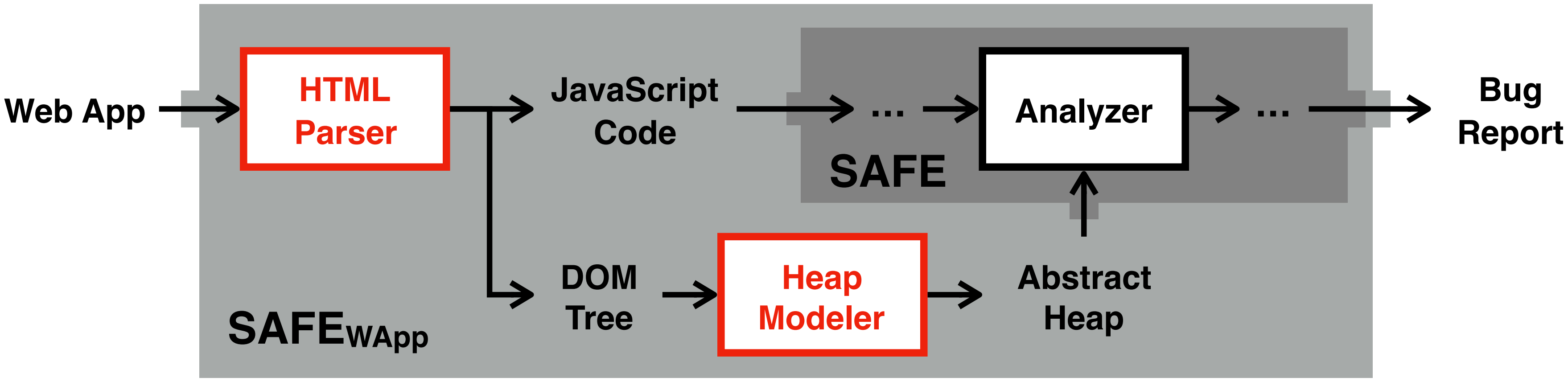


1. DOM structures
2. Interactions with JS
3. Browser-specific APIs
4. Dynamic file loading

```
<script>
var app = chrome.app;
function isDiv(elem) {
  var t = elem.tagName;
  return t.match(/^\w+/) === "DIV";
}
</script>

<div onclick="isDiv(this)">foo</div>
<p onclick="isDiv(this)">bar</p>
```

* J. Park, I. Lim, and S. Ryu, Battles with false positives in static analysis of JavaScript web applications in the wild

Analysis of Web Applications



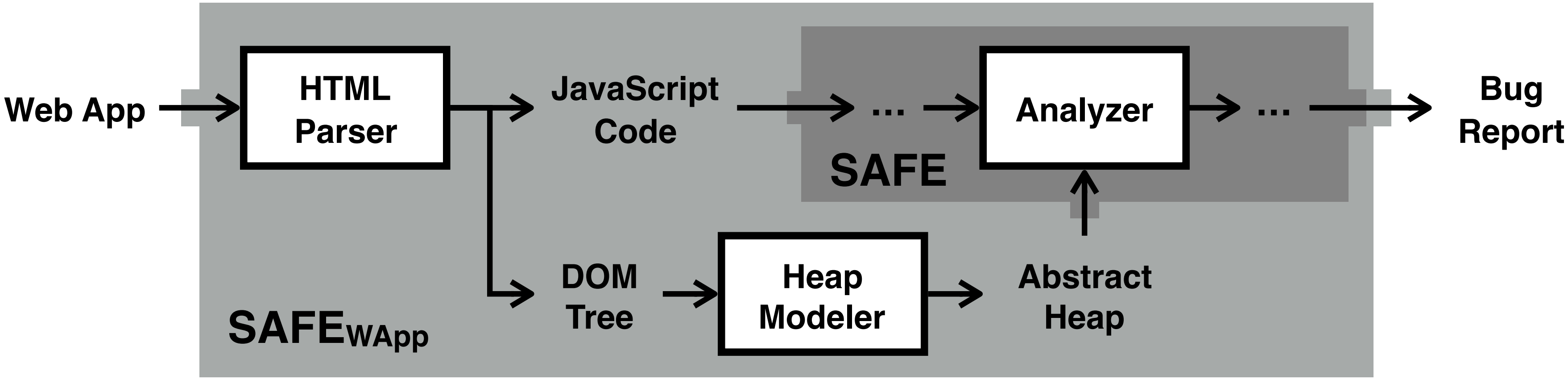
1. DOM structures
2. Interactions with JS
3. Browser-specific APIs
4. Dynamic file loading

dynamic information

```
<script>
var app = chrome.app;
function isDiv(elem) {
  var t = elem.tagName;
  return t.match(/^\w+/) === "DIV";
}
</script>

<div onclick="isDiv(this)">foo</div>
<p onclick="isDiv(this)">bar</p>
```

* J. Park, I. Lim, and S. Ryu, Battles with false positives in static analysis of JavaScript web applications in the wild

Analysis of Web Applications

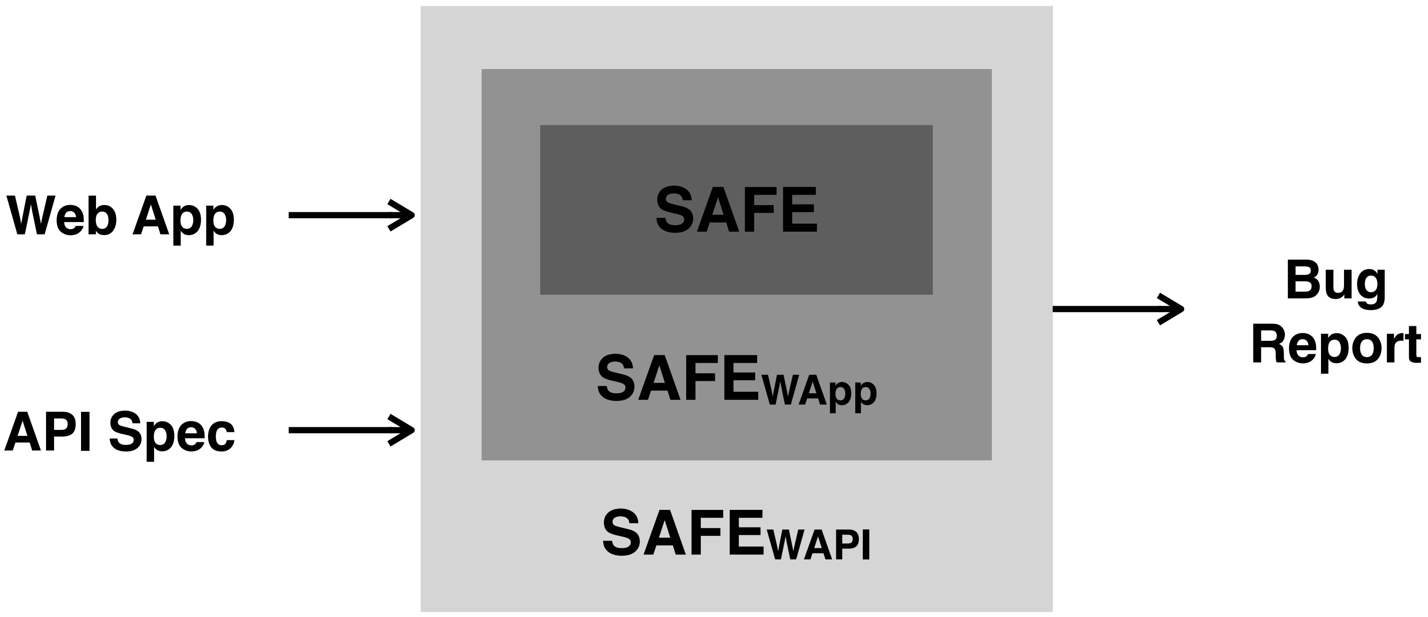


WIKIPEDIA
The Free Encyclopedia

```
<script>
var app = chrome.app;
function isDiv(elem) {
  var t = elem.tagName;
  return t.match(/^\w+/) === "DIV";
}
</script>

<div onclick="isDiv(this)">foo</div>
<p onclick="isDiv(this)">bar</p>
```

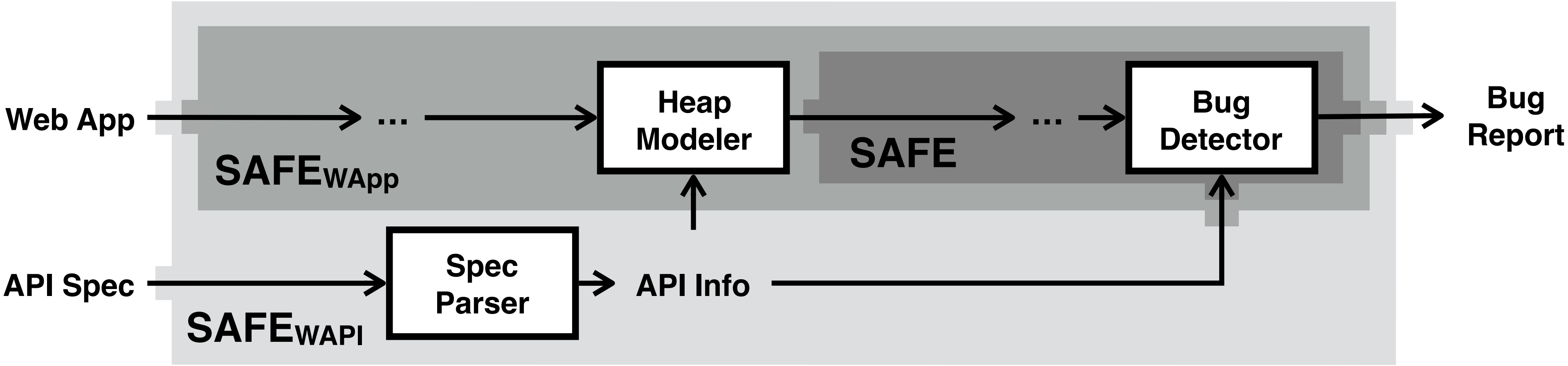
Always false!!

Analysis of JS Hybrid Applications



* S. Bae, H. Cho, I. Lim, and S. Ryu, SAFEWAPI: Web API misuse detector for web applications

Analysis of JS Hybrid Applications

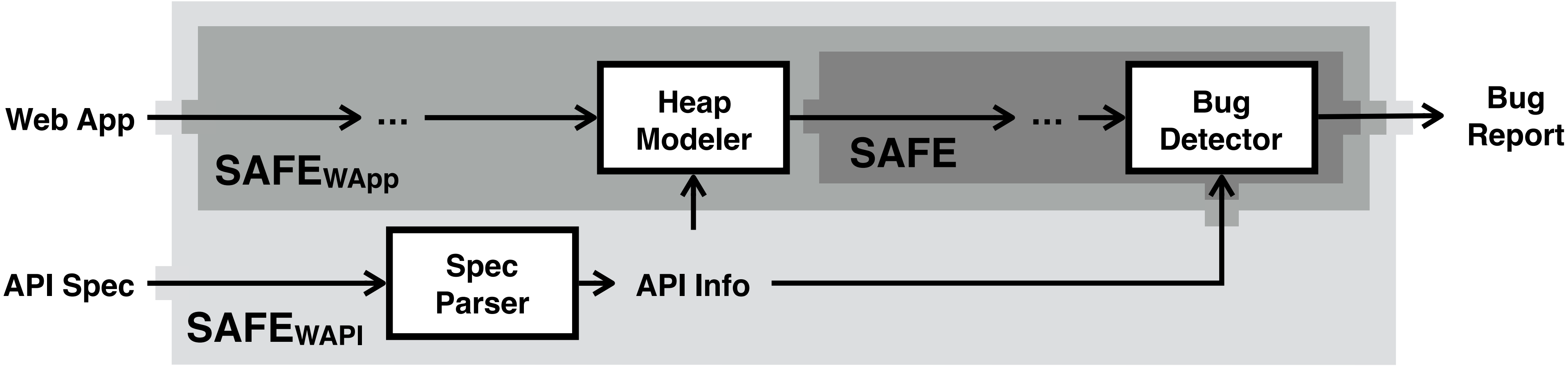


```
function f(cs) { cs.map(c => c.type); }
var c = webapis.calendar;
c.getCalendars("EVENT", f);
```

```
interface CalendarManager {
  void getCalendars(
    CalendarType type,
    Callback callback
  )
}
```

* S. Bae, H. Cho, I. Lim, and S. Ryu, SAFEWAPI: Web API misuse detector for web applications

Analysis of JS Hybrid Applications



```
function f(cs) { cs.map(c => c.type); }
var c = webapis.calendar;
c.getCalendars("EVENT", f);
```

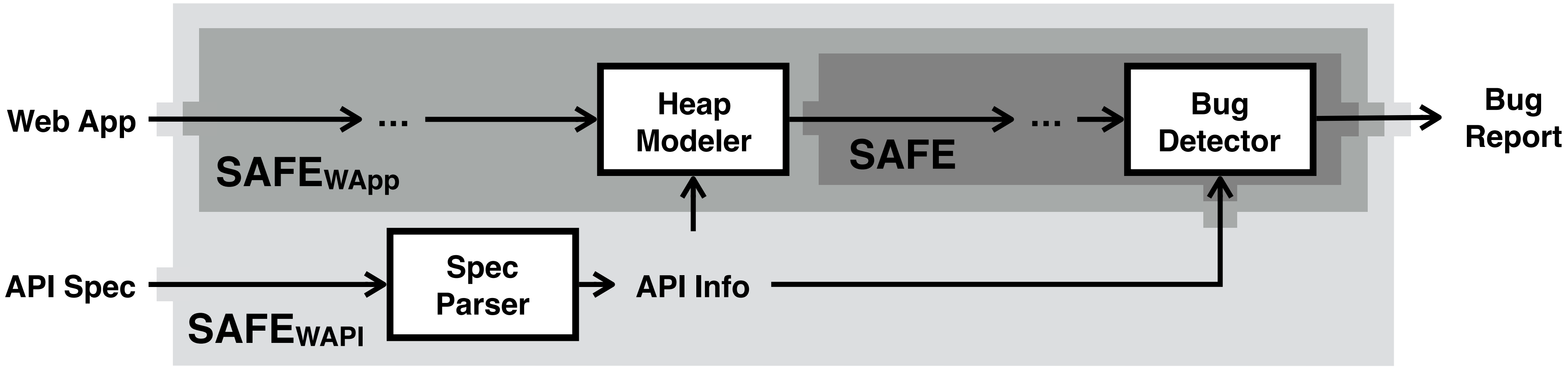
1. Platform APIs

2. Implicit callback functions

```
interface CalendarManager {
  void getCalendars(
    CalendarType type,
    Callback callback
  )
}
```

* S. Bae, H. Cho, I. Lim, and S. Ryu, SAFEWAPI: Web API misuse detector for web applications

Analysis of JS Hybrid Applications



```
function f(cs) { cs.map(c => c.type); }
var c = webapis.calendar;
c.getCalendars("EVENT", f);
```

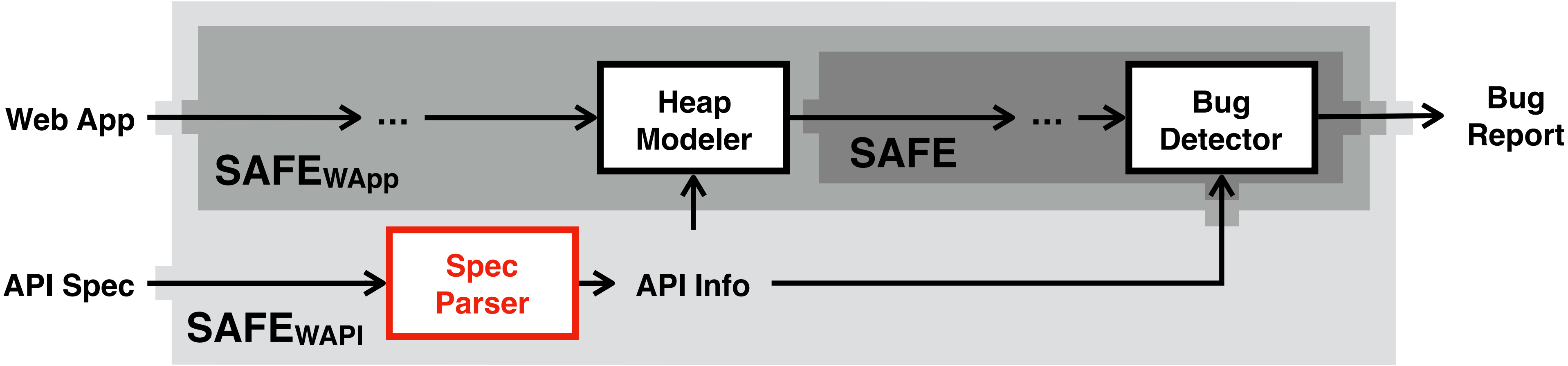
1. Platform APIs

2. Implicit callback functions

```
interface CalendarManager {
  void getCalendars(
    CalendarType type,
    Callback callback
  )
}
```

* S. Bae, H. Cho, I. Lim, and S. Ryu, SAFEWAPI: Web API misuse detector for web applications

Analysis of JS Hybrid Applications



```
function f(cs) { cs.map(c => c.type); }
var c = webapis.calendar;
c.getCalendars("EVENT", f);
```

```
interface CalendarManager {
  void getCalendars(
    CalendarType type,
    Callback callback
  )
}
```

1. Platform APIs

2. Implicit callback functions

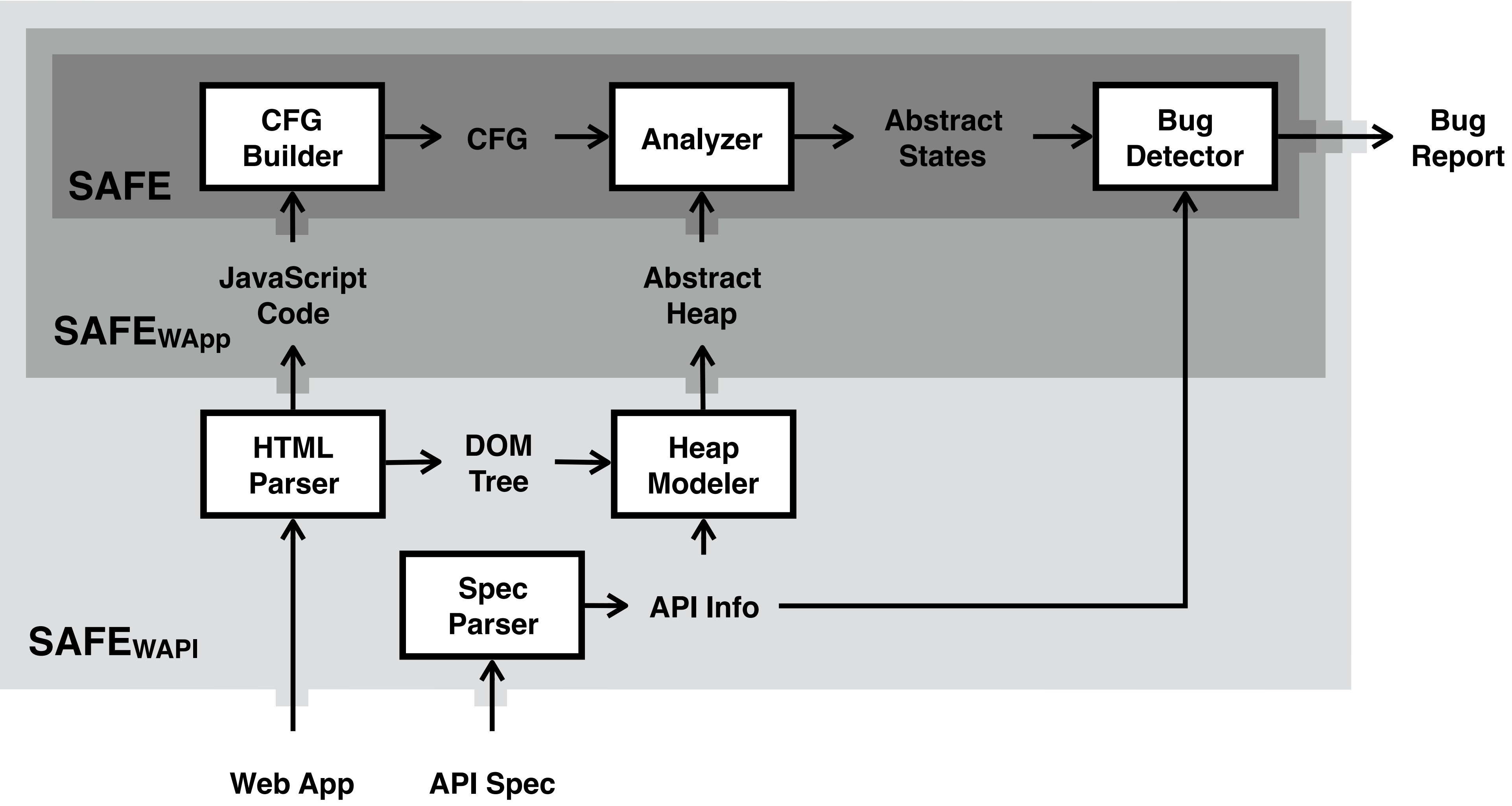
automatic modeling based on API spec

* S. Bae, H. Cho, I. Lim, and S. Ryu, SAFEWAPI: Web API misuse detector for web applications

Moving Forward

- **Dynamic code generation / Event loops**
  - More dynamic information
- **APIs implemented in platform specific languages**
  - Advanced automatic modeling
- **Finding the best analysis configuration**
  - Automatic suggestion for the best configuration

Question?


SAFE
CFG Builder
CFG
Analyzer
Abstract States
Bug Detector
Bug Report
SAFEWApp
JavaScript Code
Abstract Heap
HTML Parser
DOM Tree
Heap Modeler
Spec Parser
API Info
SAFEWAPI
Web App
API Spec