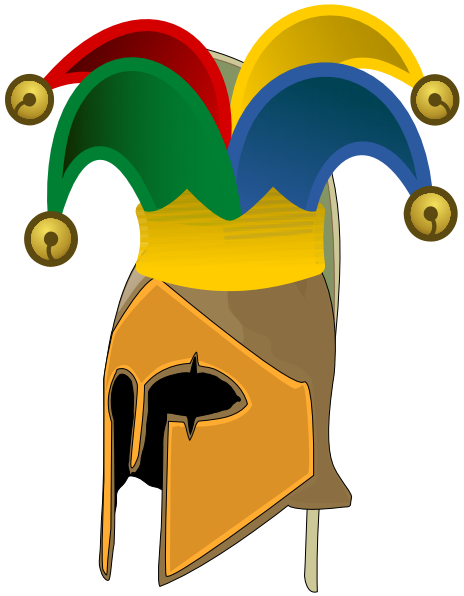


Spartan Jester

End-to-end information flow control for hybrid Android applications

Julian Sexton

MITRE

Andrey Chudnov

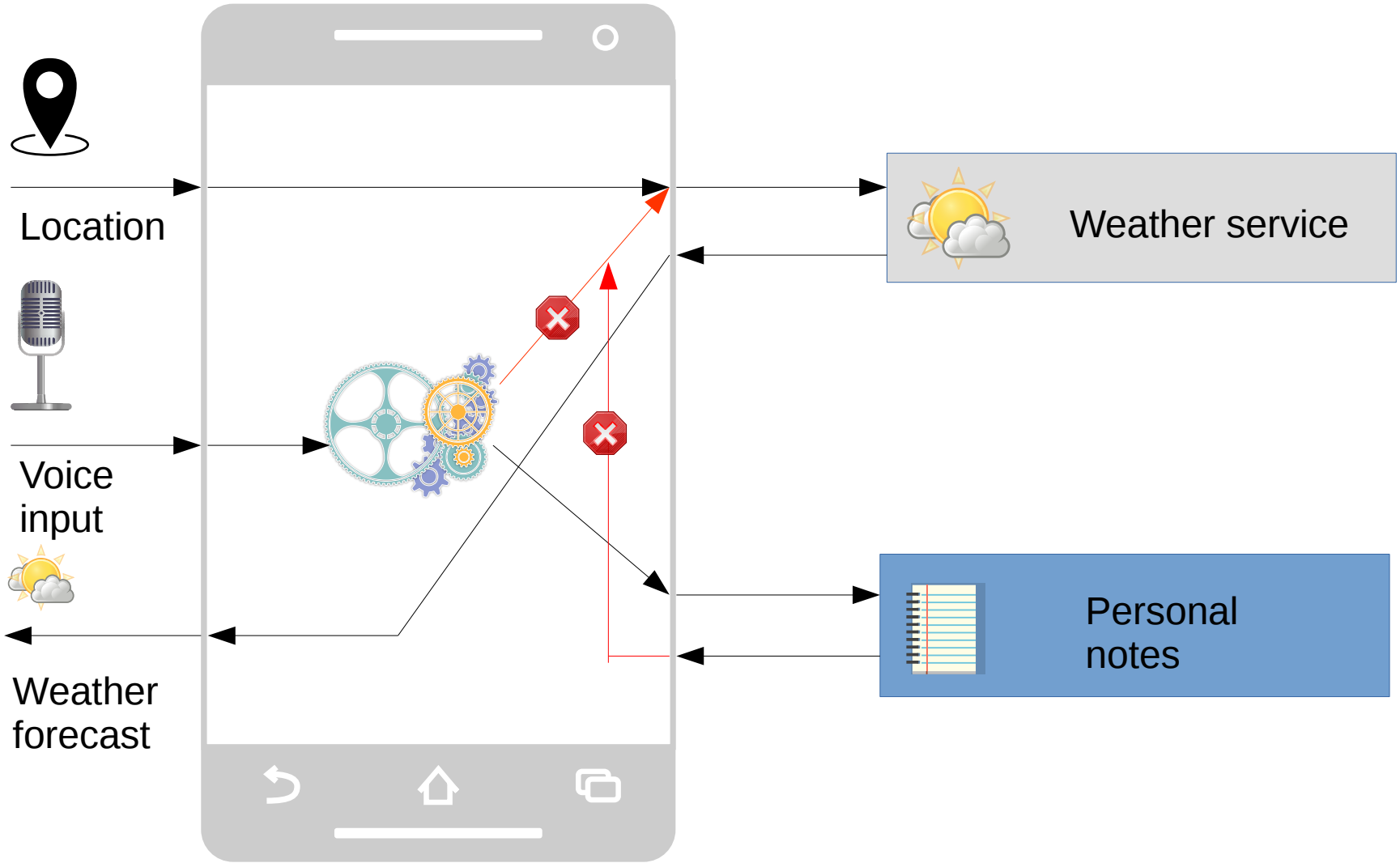
Galois, Inc

David Naumann

Stevens Institute of Technology

Partially supported by NSF award CNS-1228930

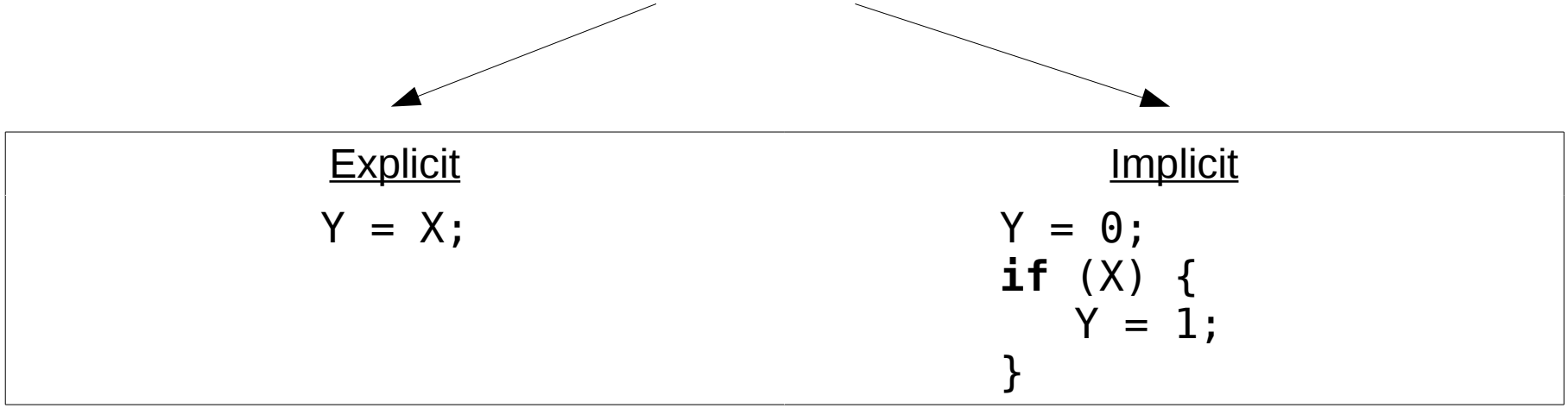
Example: personal assistant app



- Android permissions: INTERNET, RECORD_AUDIO, FILE_SYSTEM.
- Not enough to stop undesirable flows while allowing intended ones

Information flow control

X flows to Y ≡ variation in X causes variation Y



- Common abstraction: labels on sources of data, propagated to memory locations.
- Could be implemented as a type system, and types as sets of labels.
- X having a label `{A,B}` means it's value could have been influenced by sources with labels `{A}` and `{B}`

Static

- Types/labels are propagated and checked at compile time
- Best fit for statically typed languages, like Java

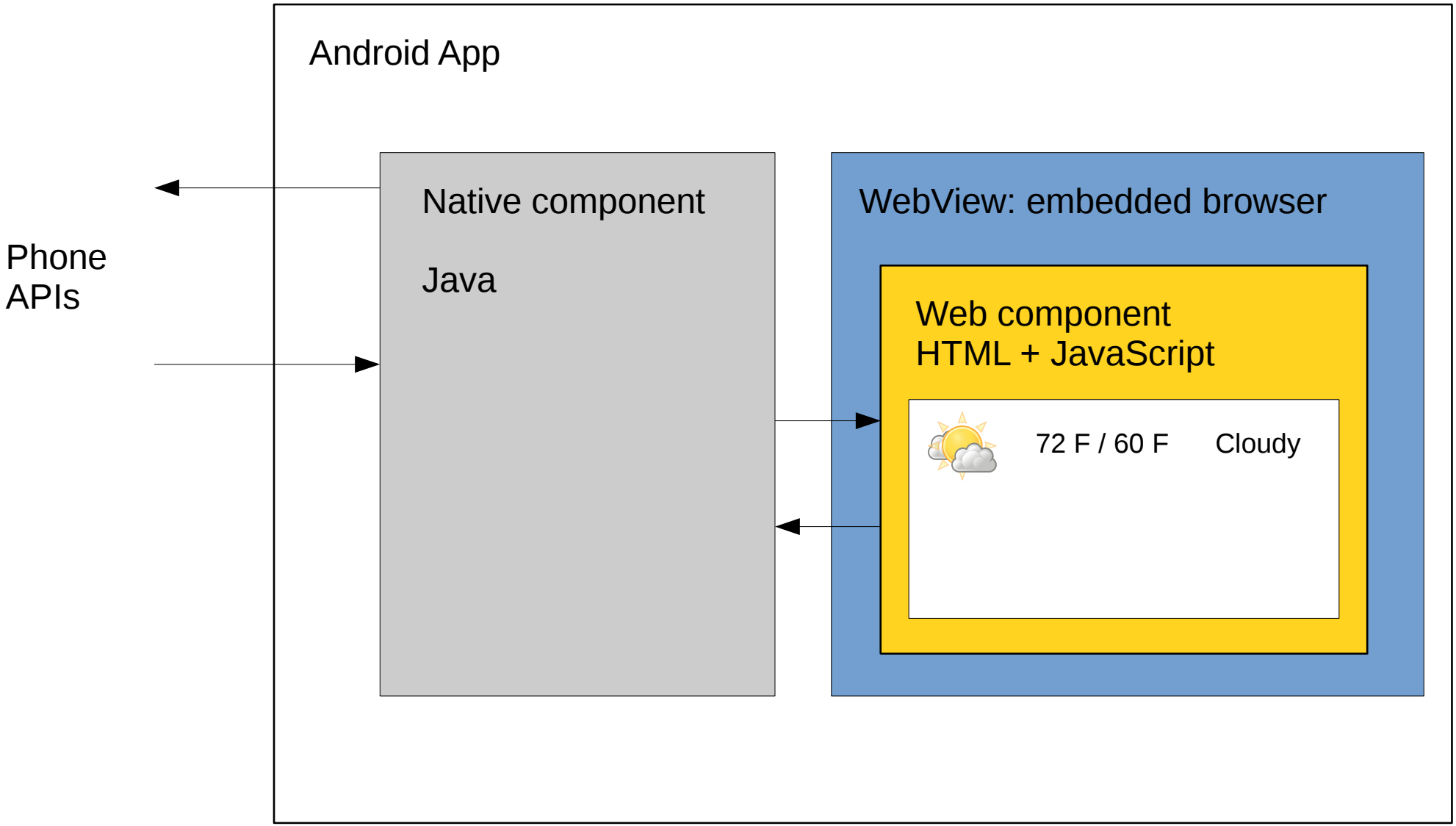
Dynamic

- Memory locations are tagged with types/labels at run-time, and updated during execution.
- Best fit for dynamically-typed languages, like JavaScript

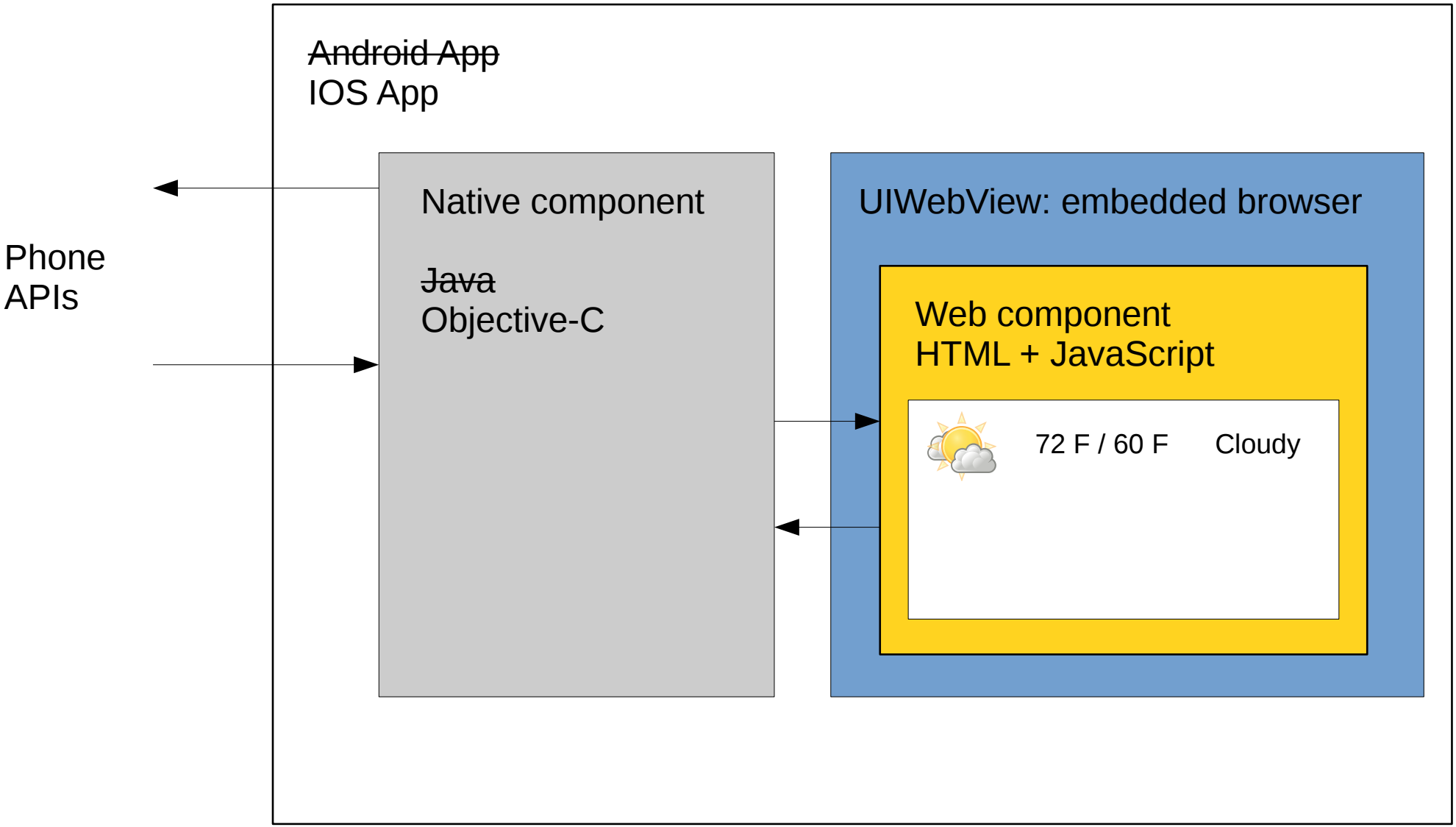
Static Information Flow Control for Android

- TaintDroid [Enck, TOCS 2014]. Dynamic taint tracking in Dalvik VM.
- FlowDroid [Arzt, PLDI 2014]. Static taint tracking for Android. Precise and comprehensive tracking of flows via APIs, intents and callbacks.
- WALA [Tripp, FASE 2013]. Multi-language static analysis framework including taint analysis. Includes good Android support.
- SPARTA [Ernst, CCS'14]. Static data flow tracking, policy-oriented taint tracking. Fits neatly with Java annotations.
- Joana [Hammer, IJIS'09] [Mohr, CEUR'15]. PDG-based sound information flow analysis for Java. Ongoing work on Android support.

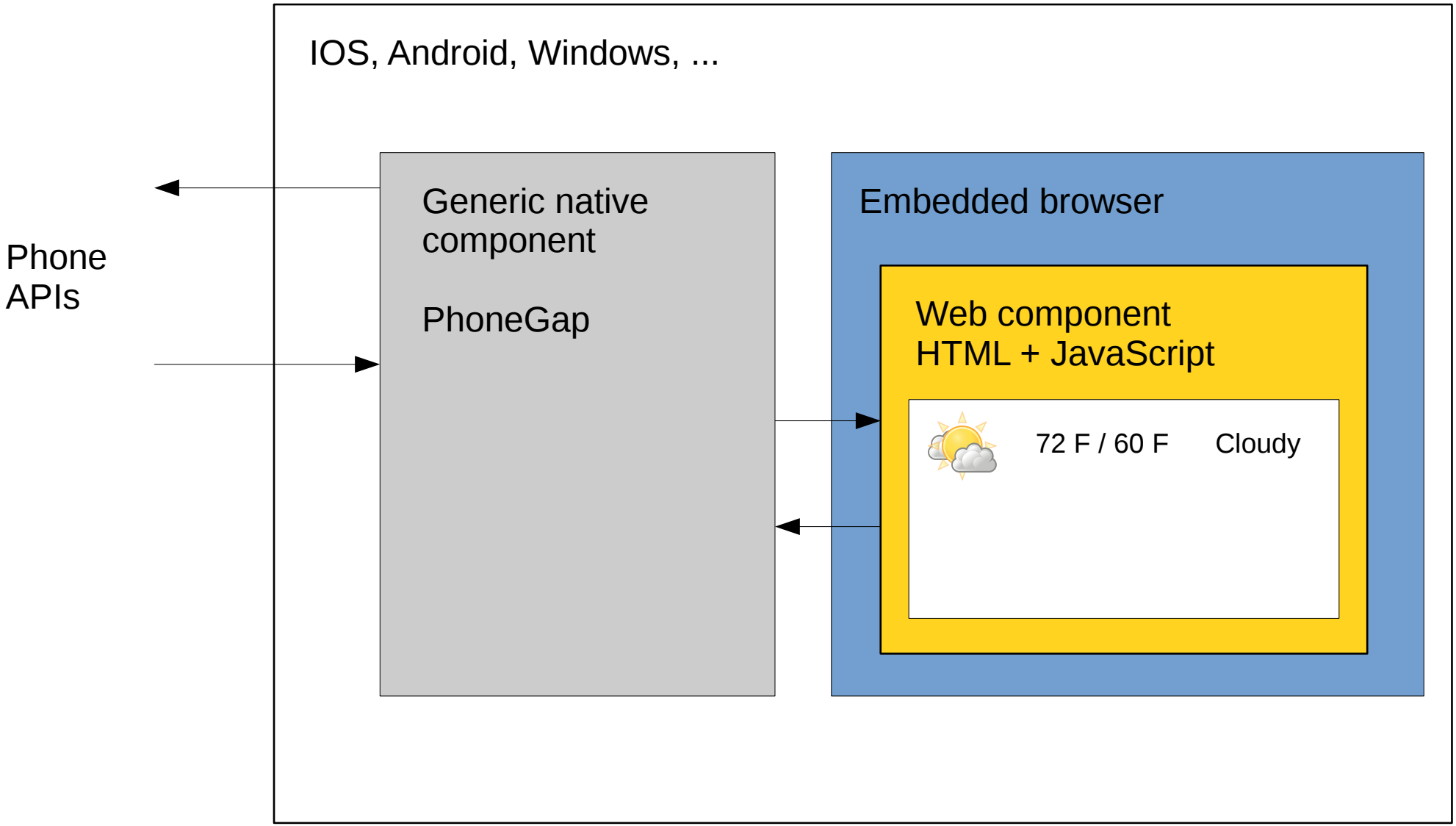
Hybrid mobile apps



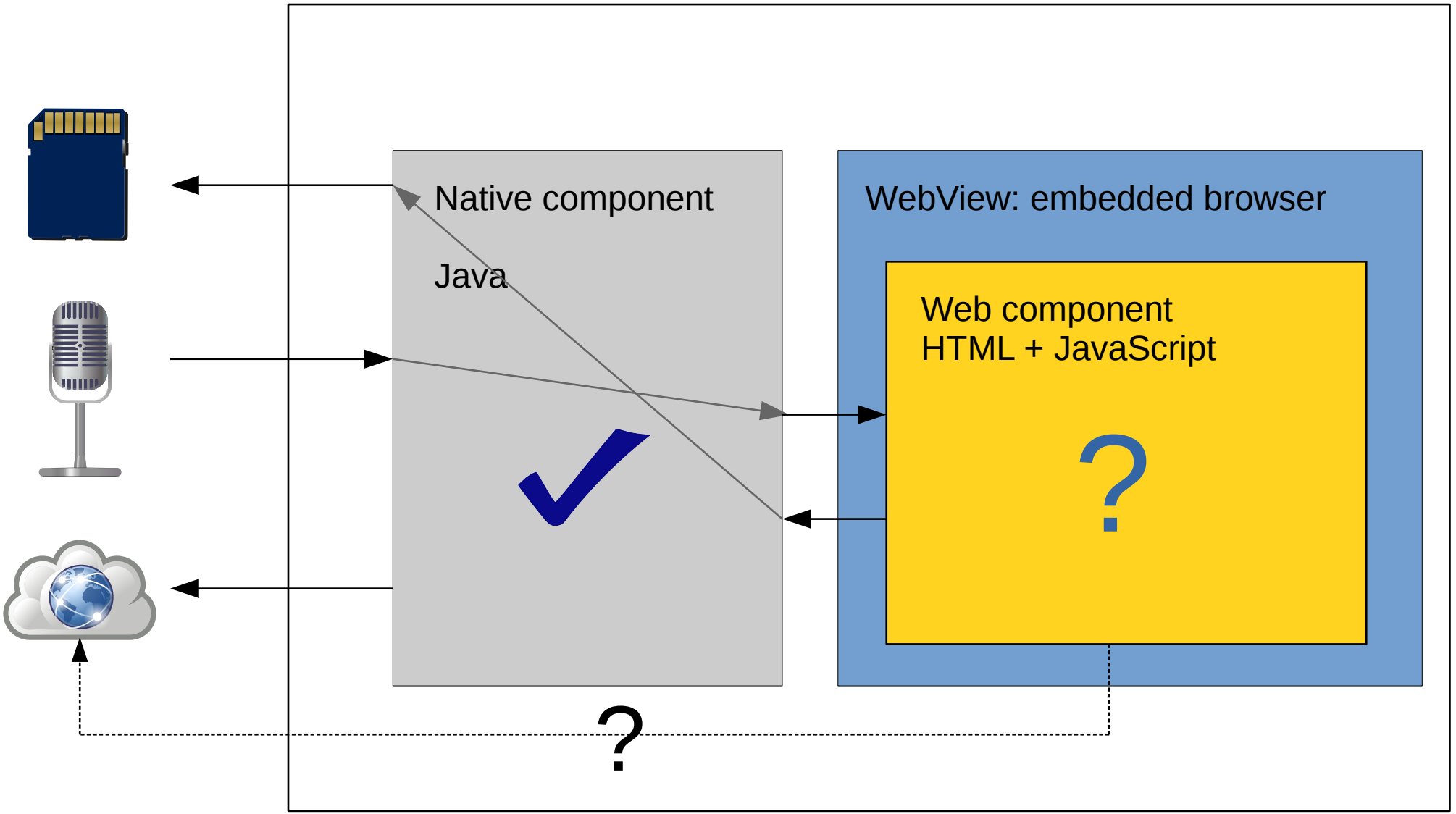
Hybrid mobile apps are portable



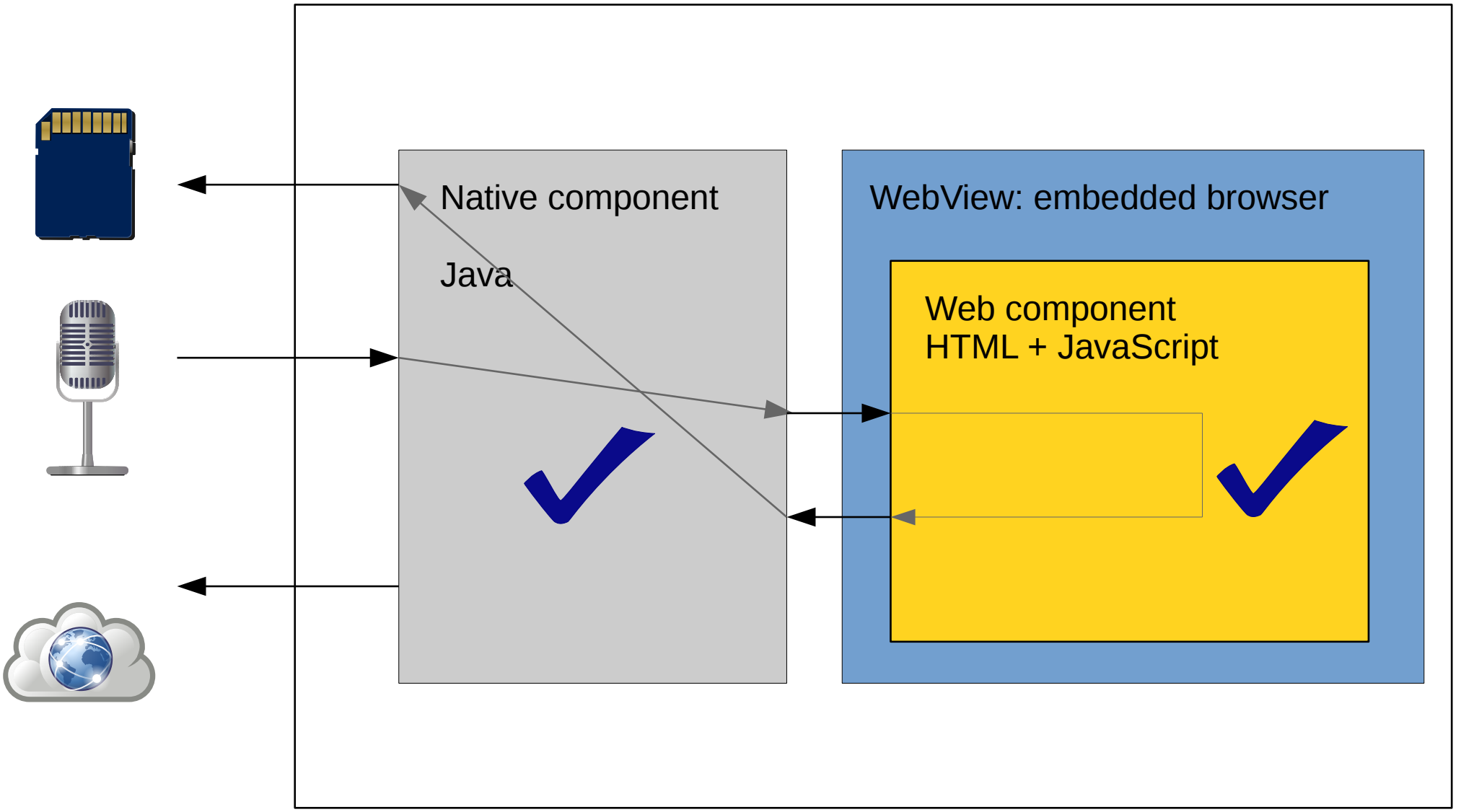
Hybrid mobile apps are portable



Hybrid mobile apps: IFC challenges



Goal: strong end-to-end IFC for hybrid apps



Previous work on IFC for hybrid apps

Code injection attacks in HTML5 mobile apps

[Jin, CCS'14]

HybriDroid (WALA)

[Lee, ASE'16]

Cordova/PhoneGap only

Accurate modeling of APIs

Focused on detecting Cross-Site Scripting attacks

Taint analysis (no implicit flow)

Use Actarus [Guarnieri, ISSTA'11] for JavaScript analysis:

Limitations in handling DOM, with and eval

Previous work on IFC for JavaScript

Label tracking ◄► Secure Multi-Execution

Inlined ◄► Browser modification

IFlowSigs [Santos 2014]	JEST [Chudnov 2015]	JSFlow [Hedin 2014]	WebKitIFC [Bichawat 2014] [Rajani 2015]	ZaphodFacets [Austin 2012]	FlowFox [DeGroef 2012]
Compiler	Compiler	Interpreter	Browser	Interpreter	Browser
⊂ES3	ES5\strict	ES5\strict	ES5	⊂ES3	ES5
Partial DOM Support	Significant DOM support	Bare-bones DOM support	Significant DOM support	No DOM support	Black box DOM
No UCF	Full UCF	Coarse UCF	Full UCF	No UCF	N/A

UCF = Unstructured Control Flow

Previous work on IFC for JavaScript

Label tracking ◄► Secure Multi-Execution

Inlined ◄► Browser modification

IFlowSigs [Santos 2014]	JEST [Chudnov 2015]	JSFlow [Hedin 2014]	WebKitIFC [Bichawat 2014] [Rajani 2015]	ZaphodFacets [Austin 2012]	FlowFox [DeGroef 2012]
Compiler	Compiler	Interpreter	Browser	Interpreter	Browser
⊂ES3	ES5\strict	ES5\strict	ES5	⊂ES3	ES5
Partial DOM Support	Significant DOM support	Bare-bones DOM support	Significant DOM support	No DOM support	Black box DOM
No UCF	Full UCF	Coarse UCF	Full UCF	No UCF	N/A

UCF = Unstructured Control Flow

Previous work on IFC for JavaScript

Label tracking ◄► Secure Multi-Execution

Inlined ◄► Browser modification

IFlowSigs [Santos 2014]	JEST [Chudnov 2015]	JSFlow [Hedin 2014]	WebKitIFC [Bichawat 2014] [Rajani 2015]	ZaphodFacets [Austin 2012]	FlowFox [DeGroef 2012]
Compiler	Compiler	Interpreter	Browser	Interpreter	Browser
⊂ES3	ES5\strict	ES5\strict	ES5	⊂ES3	ES5
Partial DOM Support	Significant DOM support	Bare-bones DOM support	Significant DOM support	No DOM support	Black box DOM
No UCF	Full UCF	Coarse UCF	Full UCF	No UCF	N/A

UCF = Unstructured Control Flow

Bridging SPARTA and JEST

Host App

Analysed with SPARTA against an information-flow policy

WebView

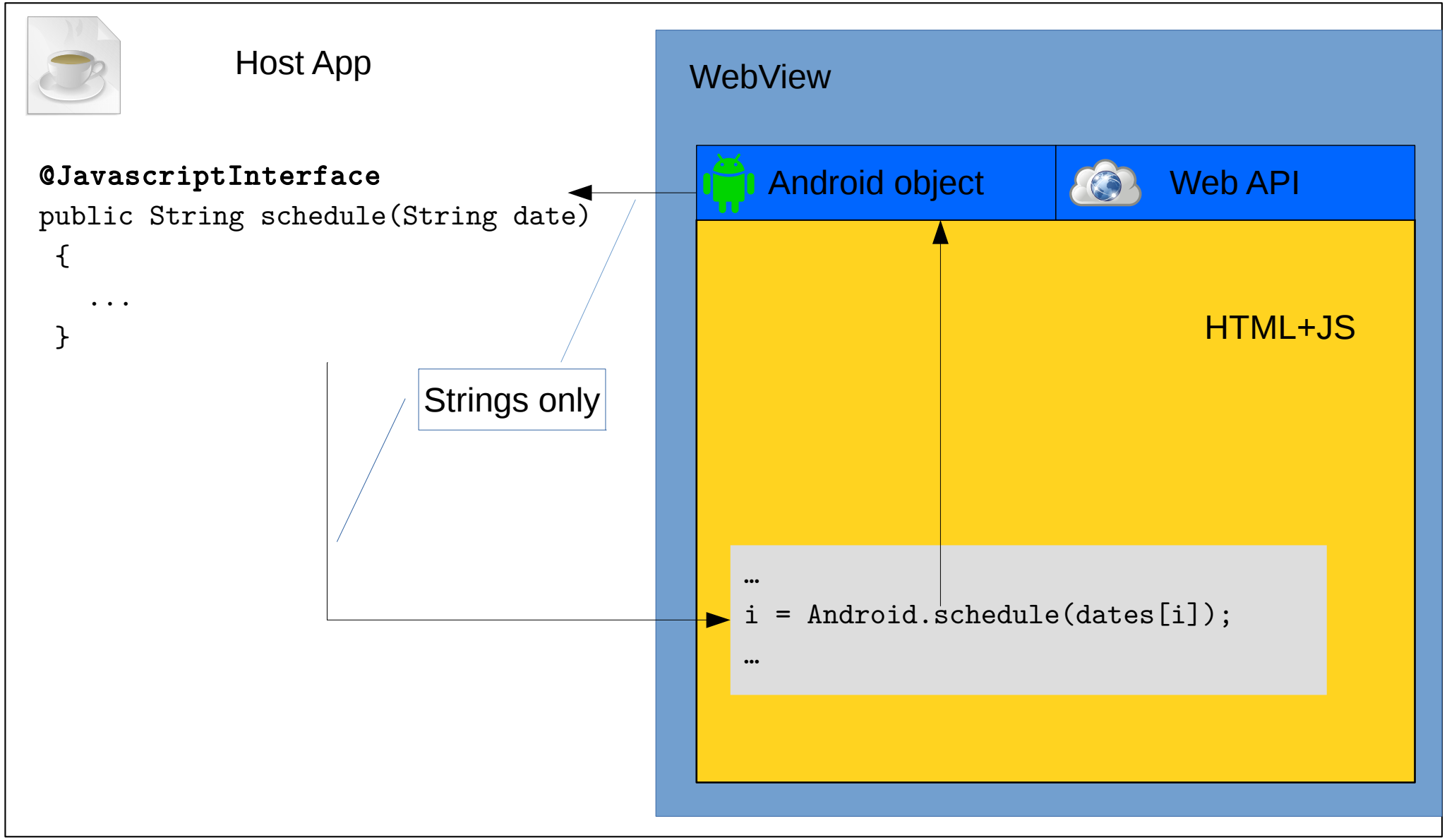
Android object	Web API
Android facade	Web Facade

HTML+JS

Web Component

Automatically rewritten using JEST against an information-flow policy

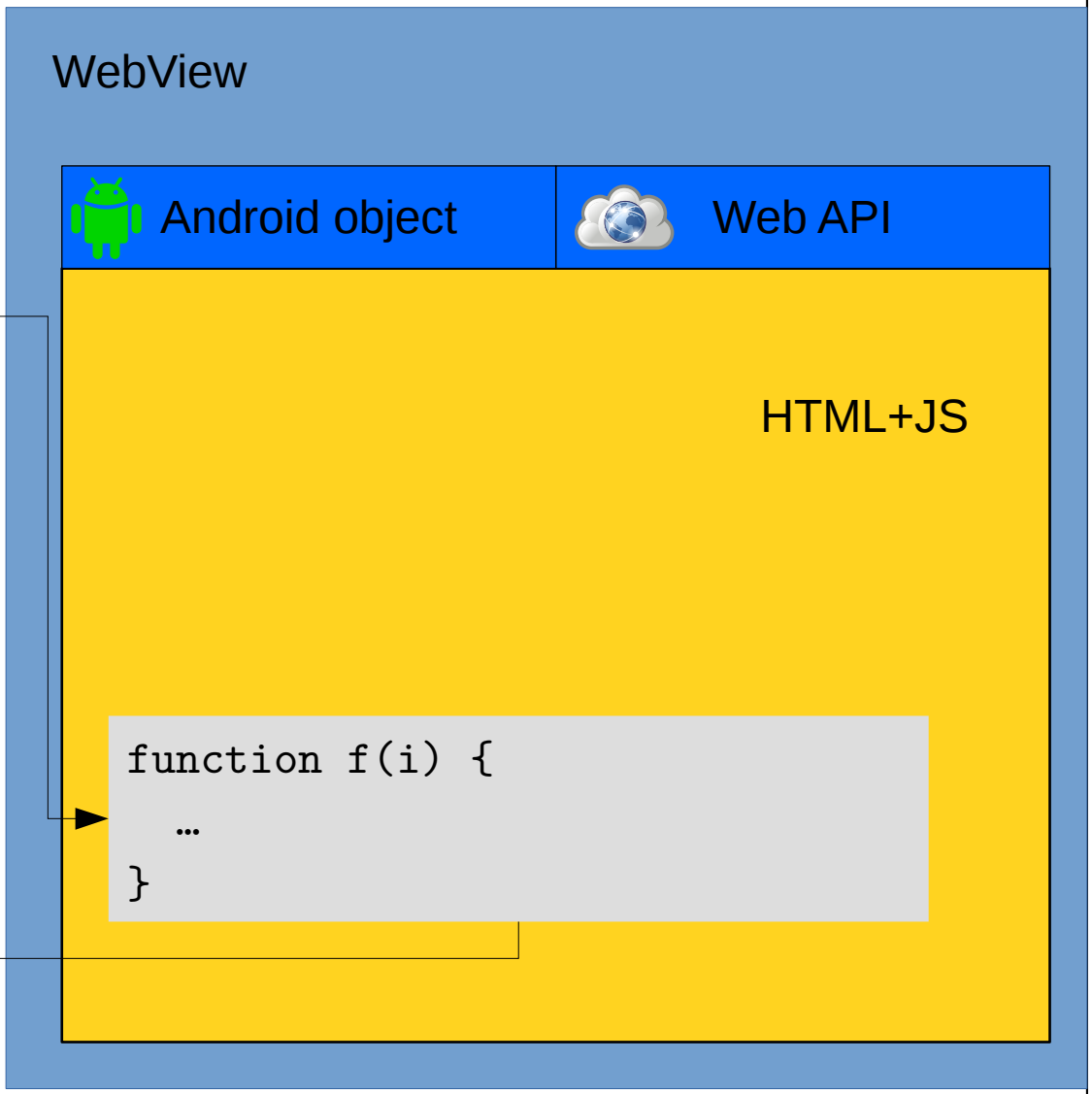
Android WebView



Android WebView

Host App

```
WebView wv = WebView(this);
int arg = 3;
String invoke = "f("+arg+")";
wv.evaluateJavaScript(invoke, callb);
```

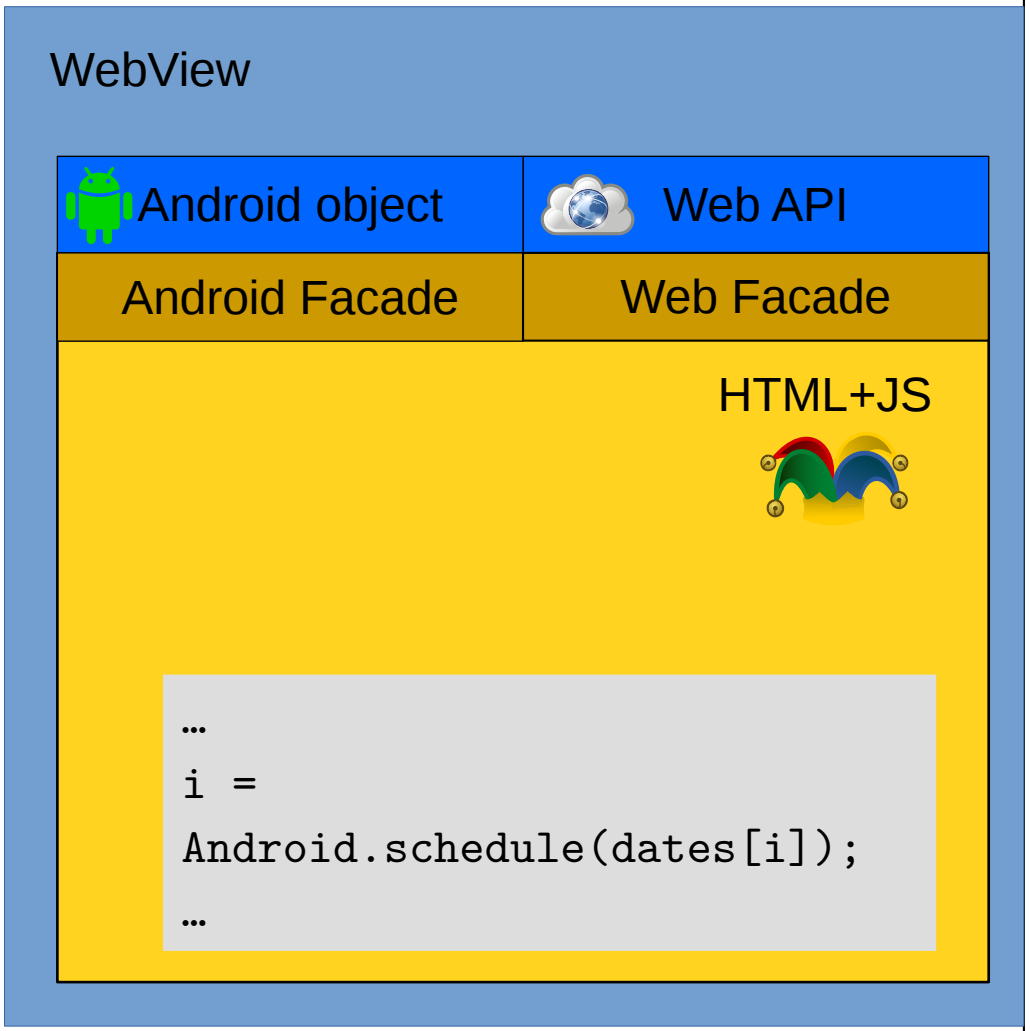


Bridging SPARTA and JEST

SPARTA policy file ◄——————————► JEST policy file

Host App

```
@JavascriptInterface
public void scheduleDate
 @Source({"USER_INPUT"}) @Sink({}) String

 (@Source({"INTERNET","FILESYSTEM"})
  @Sink({"WRITE_CALENDAR"}) String date)

 {
   ...
 }
```

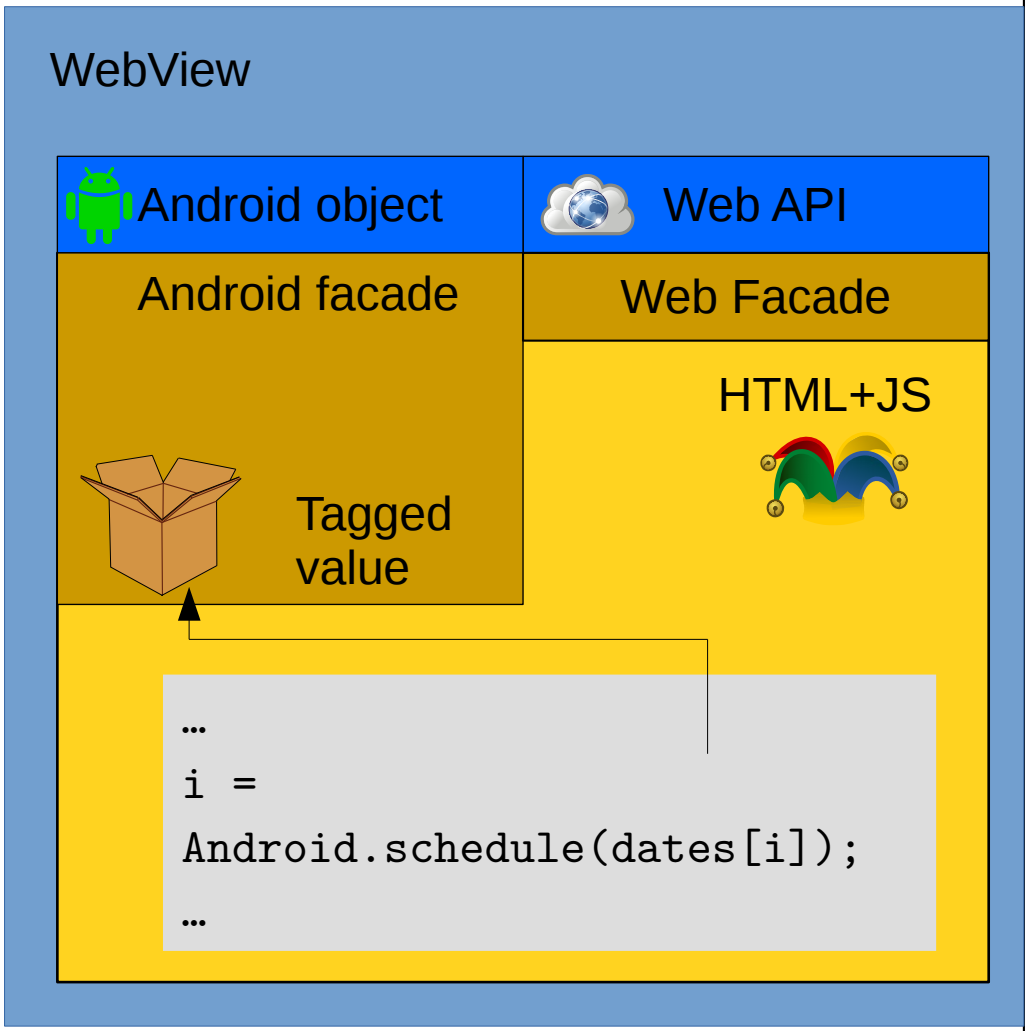


Bridging SPARTA and JEST

SPARTA policy file ⟷ JEST policy file

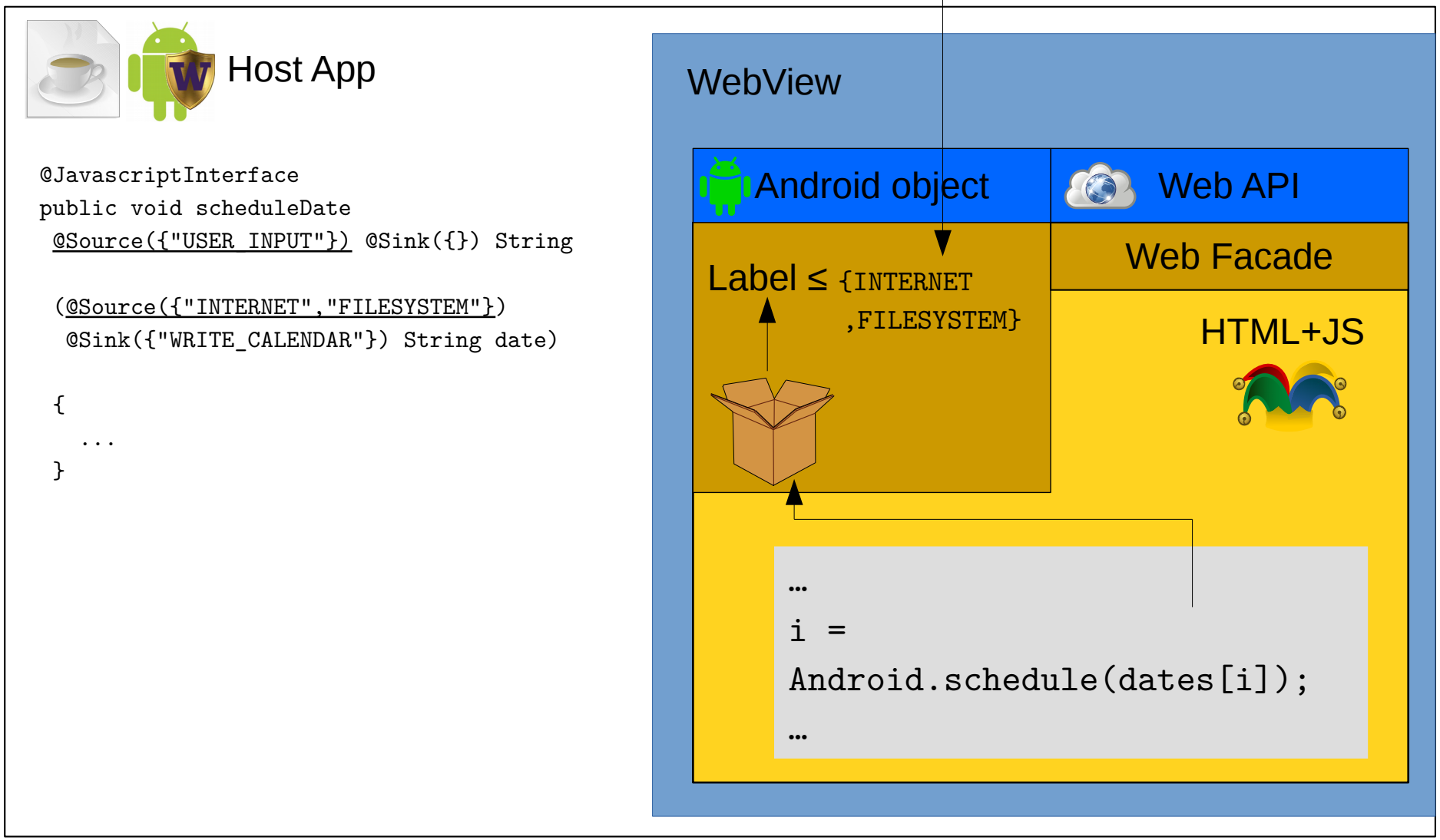
Host App

```
@JavascriptInterface
public void scheduleDate
 @Source({"USER_INPUT"}) @Sink({}) String

 (@Source({"INTERNET","FILESYSTEM"})
  @Sink({"WRITE_CALENDAR"}) String date)

 {
   ...
 }
```

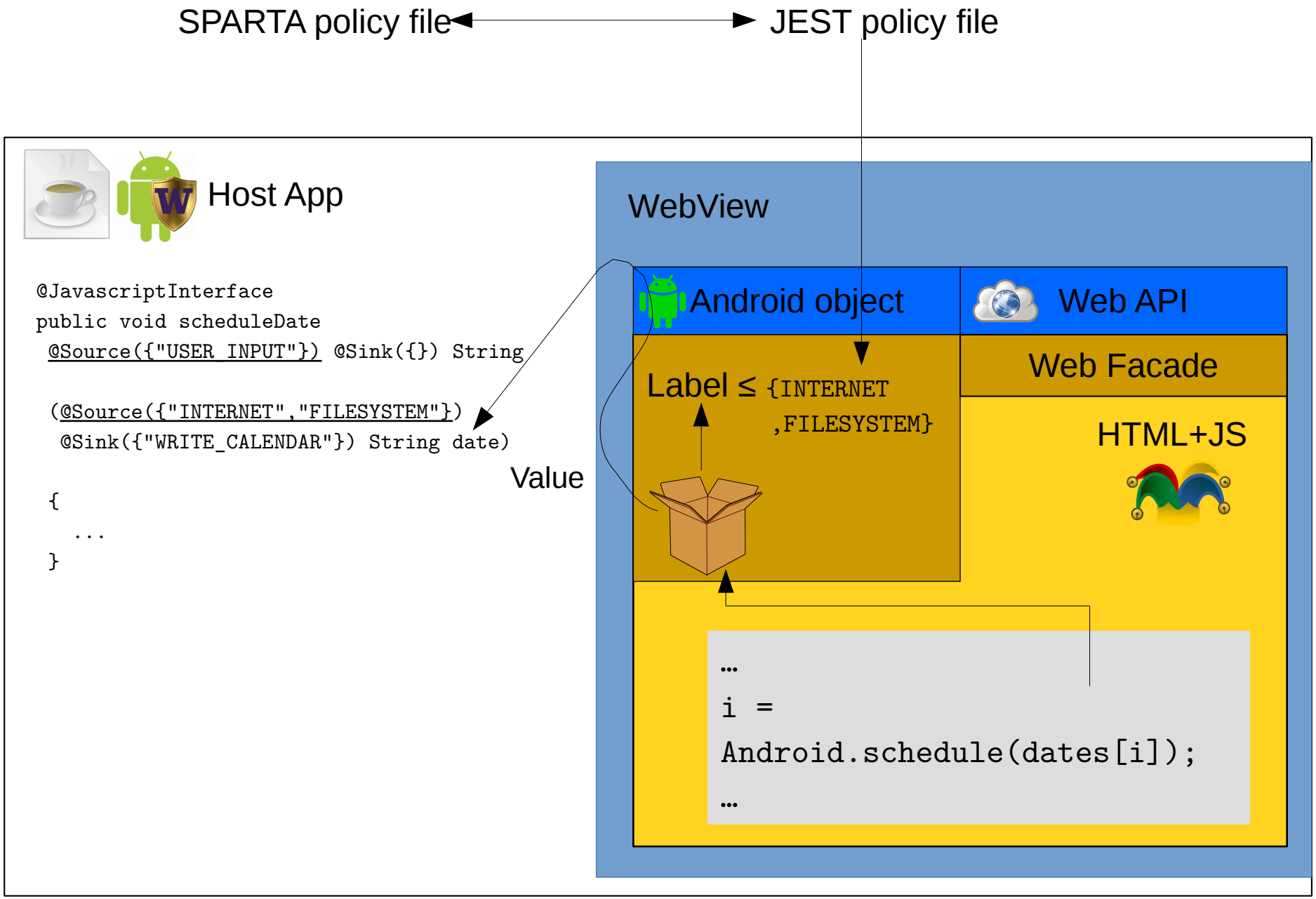


Bridging SPARTA and JEST

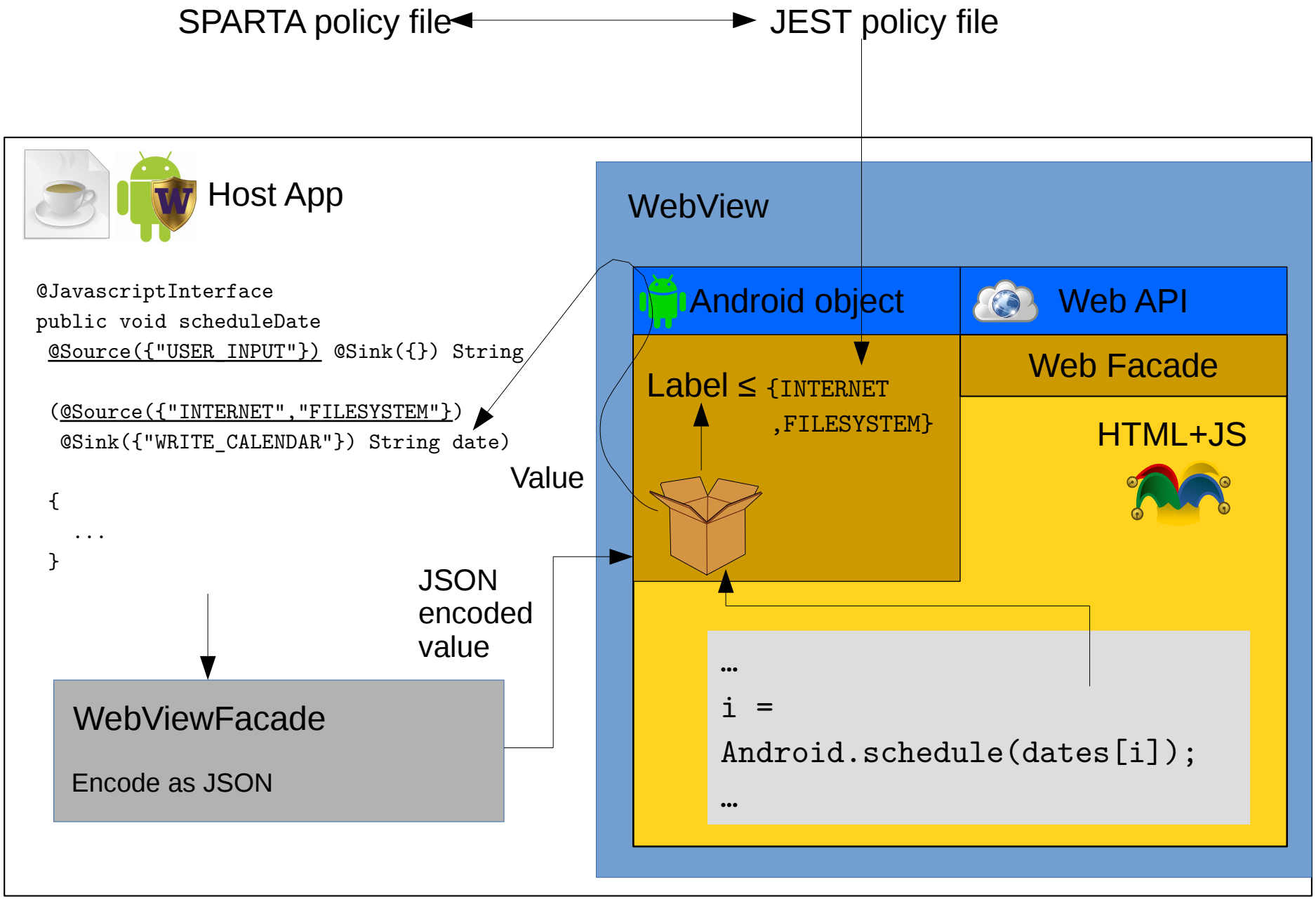
SPARTA policy file ⟷ JEST policy file



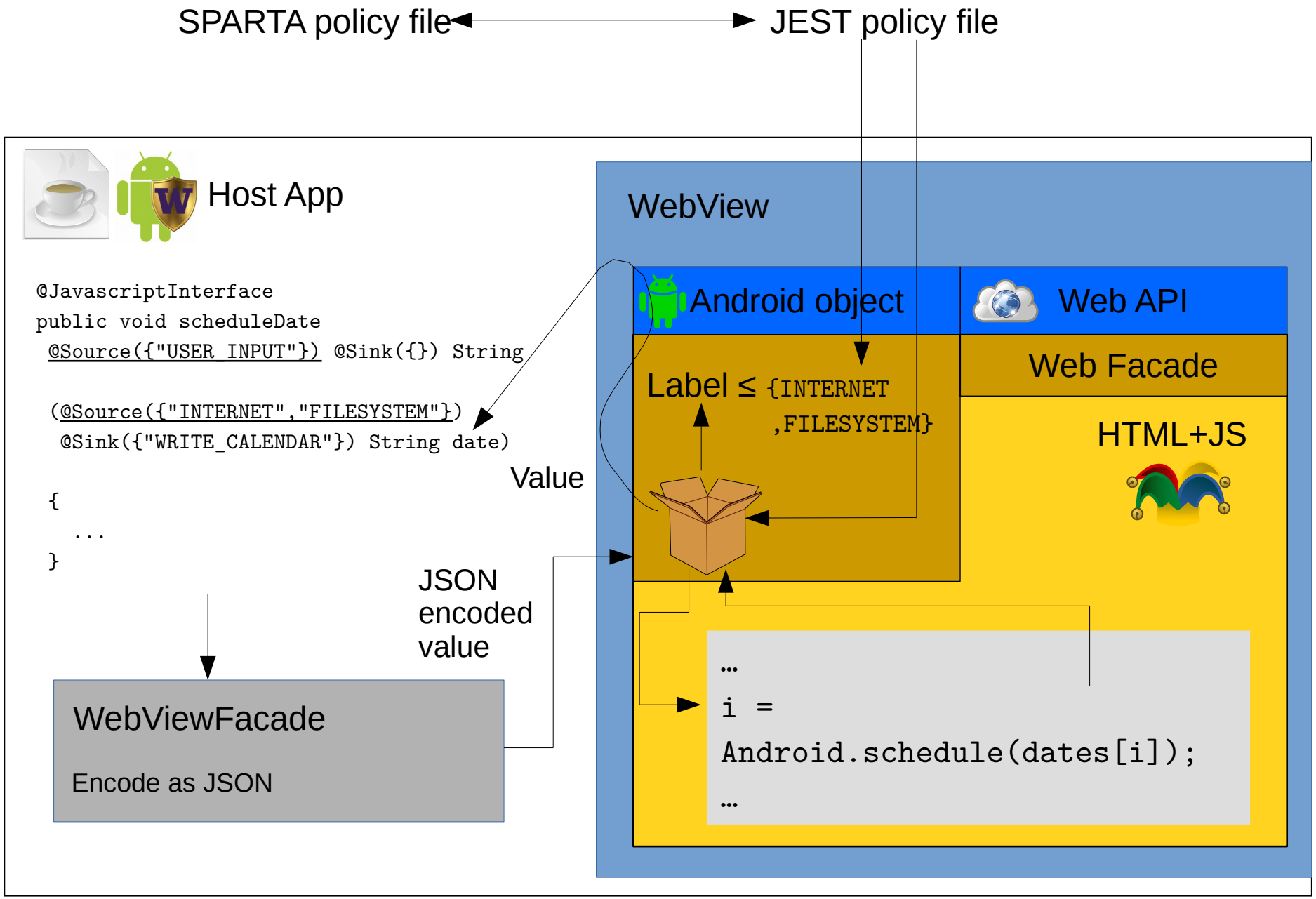
Bridging SPARTA and JEST



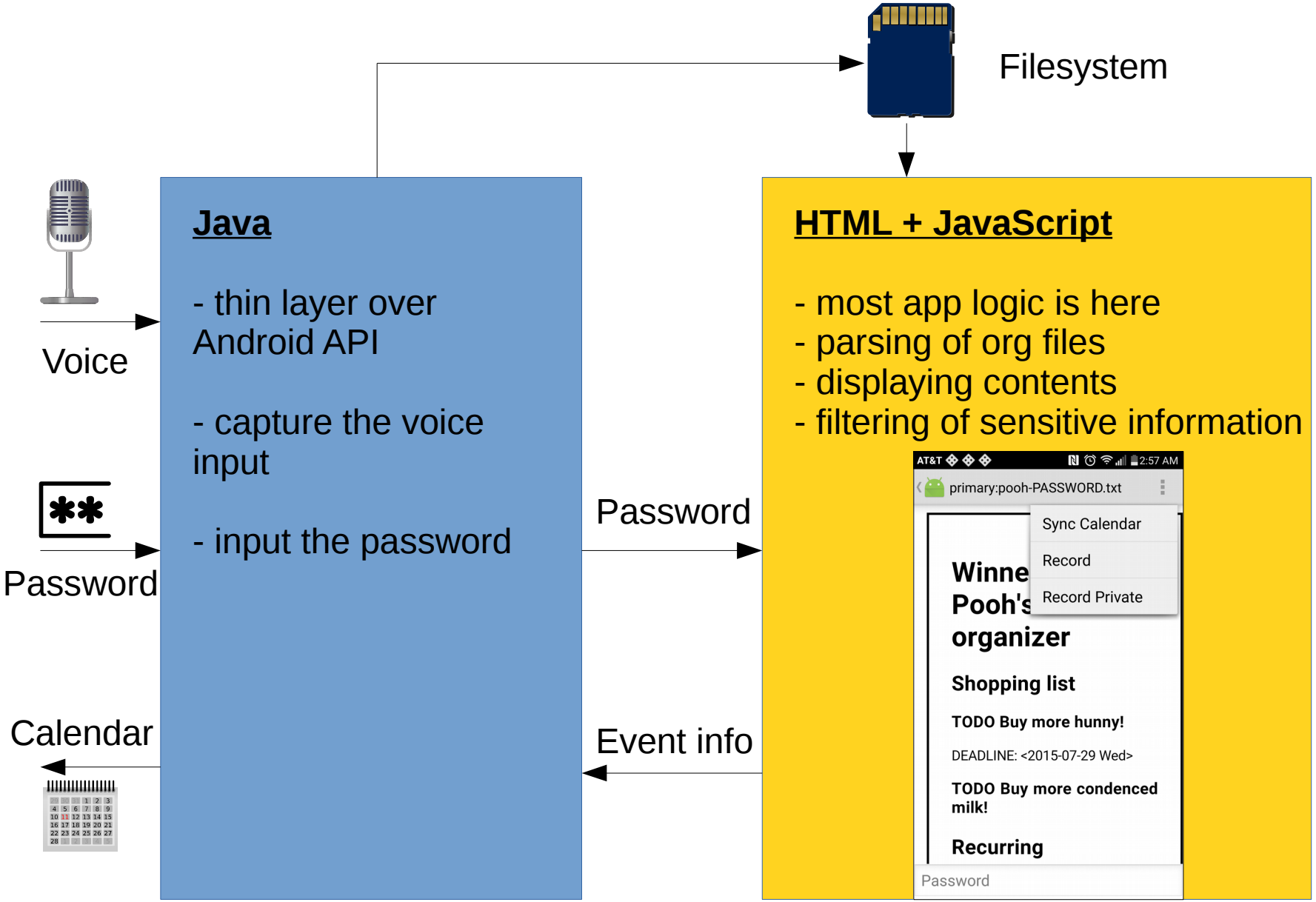
Bridging SPARTA and JEST



Bridging SPARTA and JEST



Case study: mobile org-mode



Analyst workflow

1) Express the policy for Java code in terms of a SPARTA policy file and SPARTA source annotations

2) Derive the JEST policy: principals and channel labels based on the Java policy for functions marked as `@JavascriptInterface`.

3) Run SPARTA to check the Java code.

4) Run JEST on the embedded page

- at build time, or
- via a proxy server

Experience report

- IFC still non-trivial...
- SPARTA
  - Explicit flows only
  - Type inference is intraprocedural and requires annotating every function
  - The labeling of WebView interactions is coarse-grained. Labels it with INTERNET, but our policy prevents flows there.
- JEST
  - Multiple implicit flows due to exceptions in the web component
  - This is due to the use of string operations that are methods
  - All this causes run-time failures (“No-sensitive-upgrade”)
  - We convert implicit flow to explicit and use upgrade annotations

Conclusions. Q&A.

- Investigated an approach to track information flow in hybrid Android apps
- First combination of static (Java) and dynamic (JavaScript) analysis for this goal
- This is a promising direction, but more work is required on improving the usability of tools and the policy specification.
- A single policy language that can talk about the IO channels of both native and web components is necessary to avoid duplication and policy mismatch