



Demonstrators



Tearless Demonstrators



Overview

- stip.js** ➤ tierless programming with annotations and recommender system
- CScript & SECROs** ➤ strong eventual consistency for replicated objects
- TIPC** ➤ automatically checking advanced web API constraints
- Aran & Linvail** ➤ dynamic analyses for JavaScript
- StackFul** ➤ generate test input for web applications
- NodeSentry** ➤ security framework for NodeJS
- SGX** ➤ middleware to support the use of Intel SGX
- Gavial** ➤ combine ideas from multi-tier languages with FRP

Overview

- stip.js** ➤ tierless programming with annotations and recommender system
- CScript & SECROs** ➤ strong eventual consistency for replicated objects
- TIPC** ➤ automatically checking advanced web API constraints
- Aran & Linvail** ➤ dynamic analyses for JavaScript
- StackFul** ➤ generate test input for web applications
- NodeSentry** ➤ security framework for NodeJS
- SGX** ➤ middleware to support the use of Intel SGX
- Gavial** ➤ combine ideas from multi-tier languages with FRP

Stip.js

Tierless Programming

- ✓ Automatic tier-splitting between client and server
- ✓ Customise tier-splitting process with annotations
- ✓ Feedback on offline availability

Stip.js

- ✓ Automatic tier-splitting
- ✓ Customise tier-splitting process with annotations
- ✓ Feedback on offline availability

```
1  /* @slice server */
2  {
3    function broadcast(msg) {
4      displayMessage(msg);
5    }
6  }
7
8  /* @slice client */
9  {
10   var clientId = "user" + Math.random();
11
12   function displayMessage(msg) {
13     $("#msgs").append(msg);
14   }
15
16   broadcast(clientId + " says hello!");
17
18 }
```

Stip.js

- ✓ Automatic tier-splitting
- ✓ Customise tier-splitting process with annotations
- ✓ Feedback on offline availability

```
1  /* SERVER CODE */
2
3  server.expose({
4    broadcast : function (msg, cb) {
5      server.rpc( 'displayMessage', msg);
6    }
7  })
8
9
10 /* CLIENT CODE */
11
12 var clientId = "user" + Math.random();
13 client.expose({
14   displayMessage : function (msg, cb) {
15     $( "#msgs" ).append(msg);
16   }
17 })
18
19 client.rpc( 'broadcast',
20   clientId + " says hello!");
```

Stip.js

- ✓ Automatic tier-splitting
- ✓ **Customise tier-splitting process with annotations**
- ✓ Feedback on offline availability

PLACEMENT

`@client`

`@server`

`@shared`

`@ui`

COMMUNICATION

`@remoteCall`

`@remoteFunction`

`@reply`

`@broadcast`

`@blocking`

DATA SHARING

`@local`

`@copy`

`@observable`

`@replicated`

FAILURE HANDLING

`@defineHandler`

`@useHandler`

Stip.js

- ✓ Automatic tier-splitting
- ✓ Customise tier-splitting process with annotations
- ✓ **Feedback on offline availability**

placement strategy report


client slices


- browser

server slices

- data

0% [] 100%

 = client

 = server

Application level of offline availability: 10%

Consider making following constructors replicated or observable

- Meeting

- Task

Consider making following declaration replicated or observable

- var meetings;

- var tasks;

- var courses;

- var latestUpdate;

Stip.js

STIP .JS

Home

Try it out

Learn

Publications

Links

Contact Us

Fork me on GitHub

Your JavaScript code:

Snippets ▾

Results

client

server

setup

```
1 /* @config  data:      server,
2            browser:    client,
3            getters+setters: server,
4            statistics: server
5
6 @slice  data */
7 {
8     var todos = [];
9
10    function Todo (name, priority) {
11        this.name = name;
12        this.priority = priority;
13        this.status = -1;
14    }
15 }
16 /* @slice  getters+setters */
17 {
18    function getTodos () {
19        return todos;
20    }
21
22    function addTodo (name, priority) {
23        var todo = new Todo(name, priority);
24        todos.push(todo);
```

Offline report

Code

Slice Dependence Graph

<https://soft.vub.ac.be/~lphilips/jspdg/stip/stip-web/>

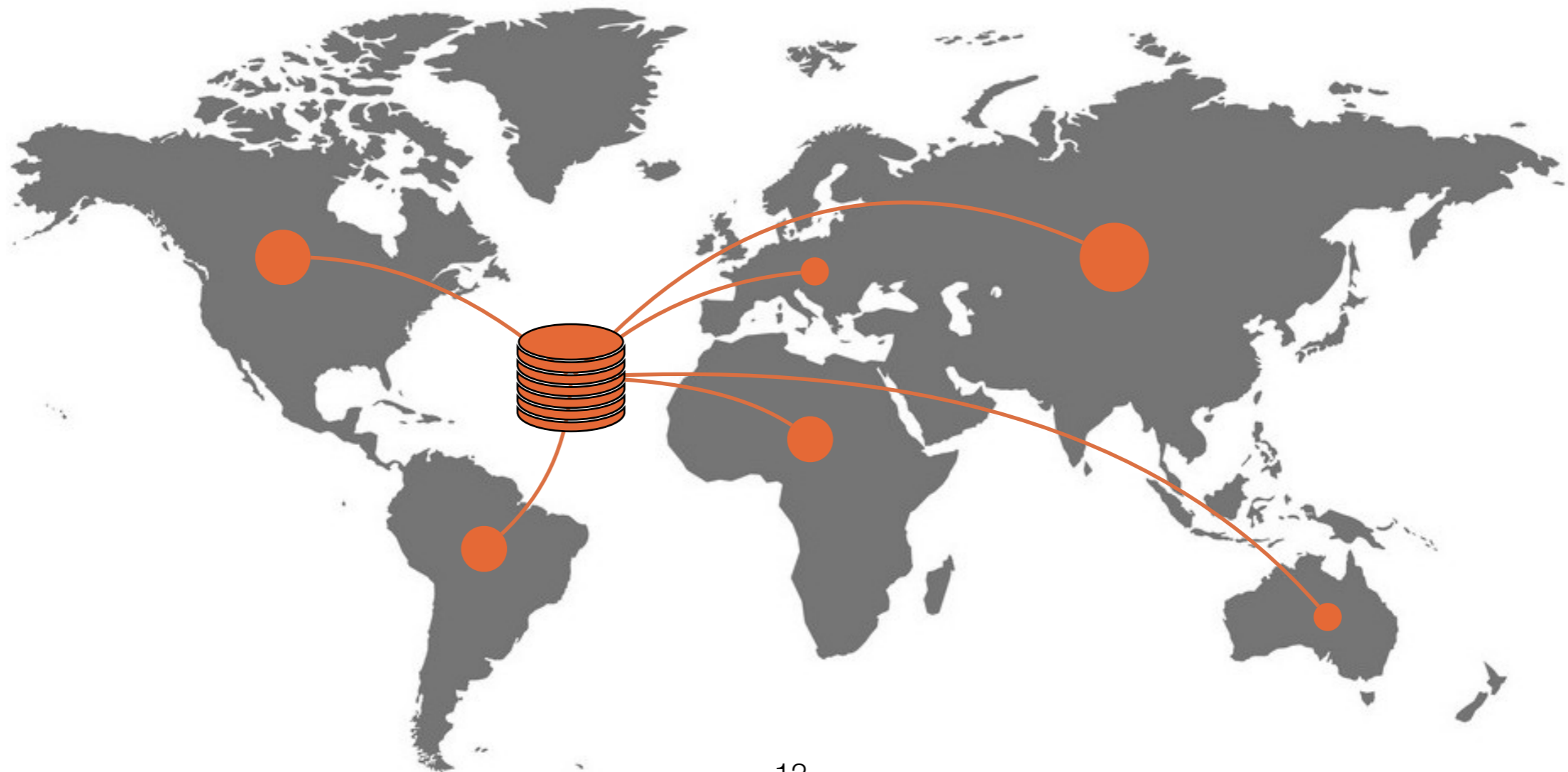
Overview

- stip.js** ➤ tierless programming with annotations and recommender system
- CScript & SECROs** ➤ **strong eventual consistency for replicated objects**
- TIPC** ➤ automatically checking advanced web API constraints
- Aran & Linvail** ➤ dynamic analyses for JavaScript
- StackFul** ➤ generate test input for web applications
- NodeSentry** ➤ security framework for NodeJS
- SGX** ➤ middleware to support the use of Intel SGX
- Gavial** ➤ combine ideas from multi-tier languages with FRP

CScript

Replicated Data Types in JavaScript

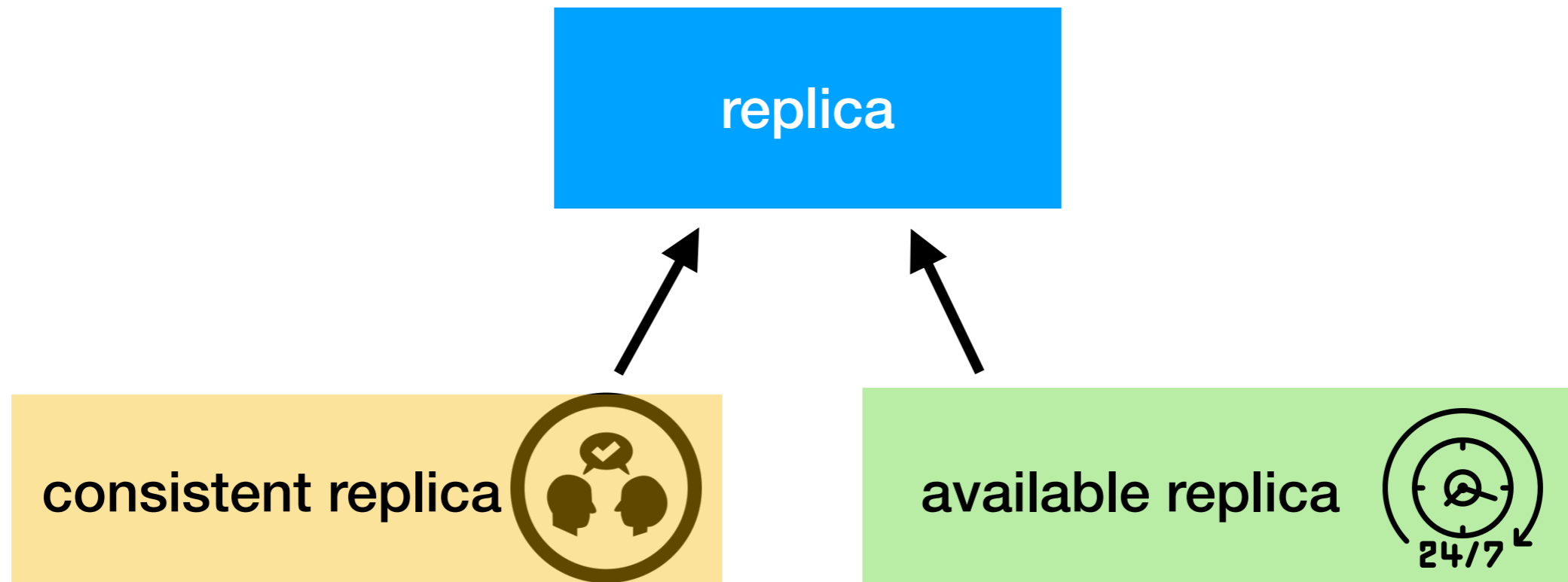
- ✓ **Facilitate replication in JavaScript**
- ✓ Choose between highly available or strongly consistent replicated objects



CScript

Replicated Data Types in JavaScript

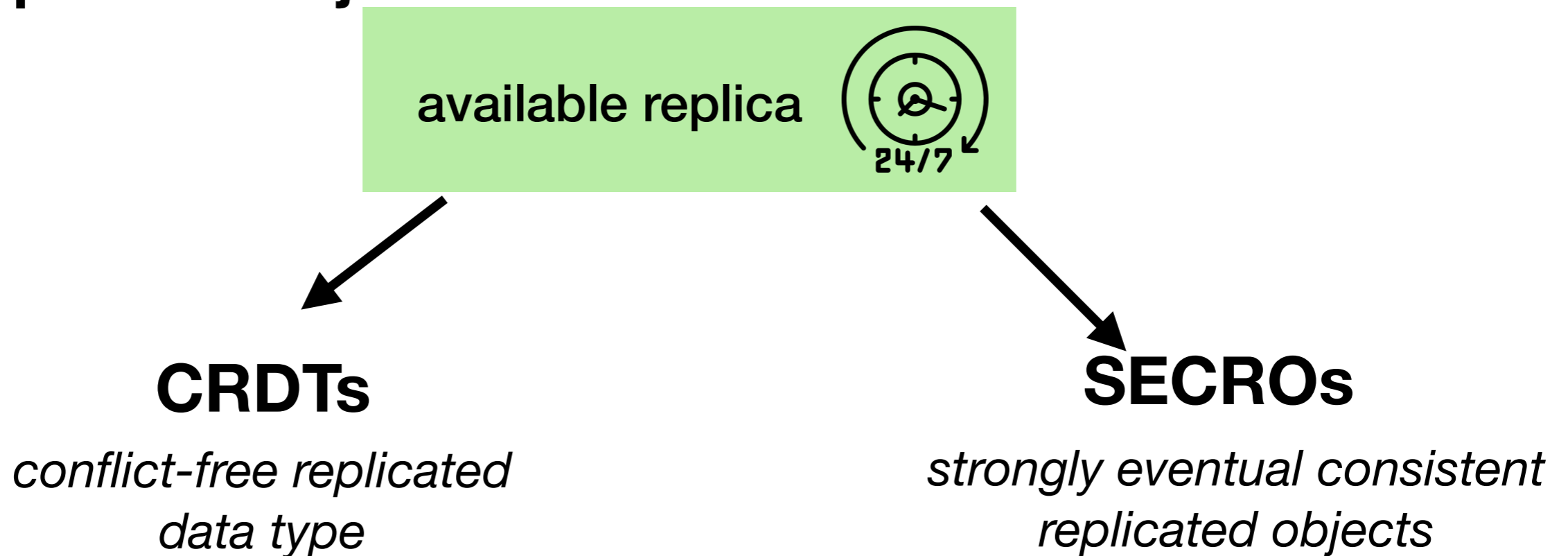
- ✓ Facilitate replication in JavaScript
- ✓ **Choose between highly available or strongly consistent replicated objects**



CScript

Replicated Data Types in JavaScript

- ✓ Facilitate replication in JavaScript
- ✓ **Choose between highly available or strongly consistent replicated objects**



- ✗ all operations must be commutative
- ✗ limited set of CRDTs
- ✗ not generally applicable

- ✓ employ application invariants
- ✓ modular + composable objects
- ✓ general-purpose data types

CScript

Replicated Data Types in JavaScript

- ✓ Facilitate replication in JavaScript
- ✓ **Choose between highly available or strongly consistent replicated objects**



Grocery Lists

New

Create

Lists

Shopping List by Kevin	+		
Mangos (3/5)	+		🛒 ✕
Lasagna (0/2)	+		🛒 ✕
Pizza (0/1)	+		🛒 ✕

CScript

Replicated Data Types in JavaScript

```

service GroceryService {
  rep list = new GroceryList();
  rep inventory = new Inventory();

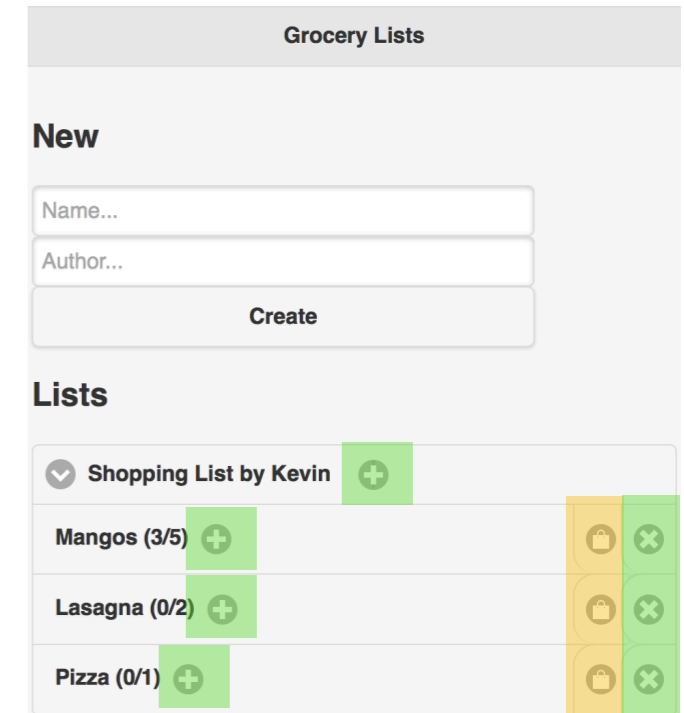
  constructor(name, author){
    this.name = name;
    this.author = author;
  }

  add(item, qty) {
    return this.list.add(item, qty);
  }

  delete (itemName) {
    return this.list.delete(itemName);
  }

  buy(itemName , qty ) { /* ... */ }
}

```



CScript

Replicated Data Types in JavaScript

```

service GroceryService {
  rep list = new GroceryList();
  rep inventory = new Inventory();

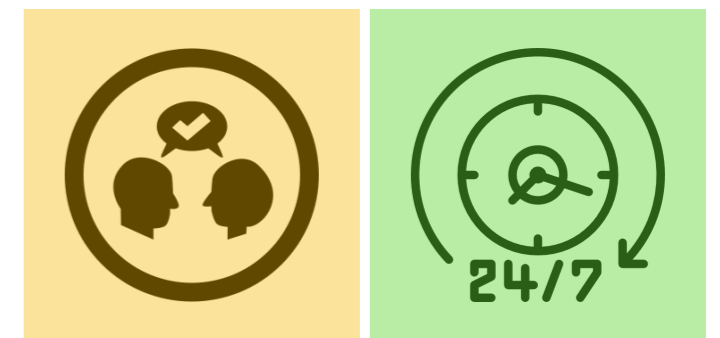
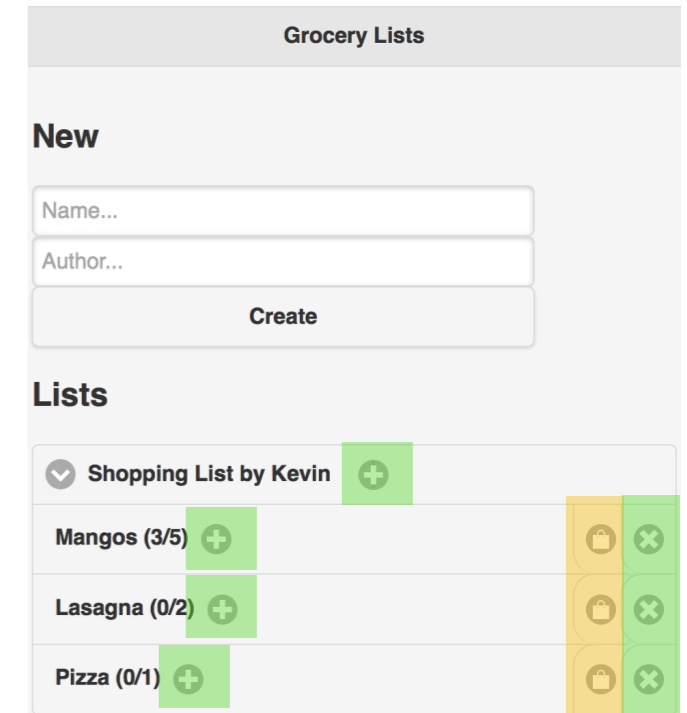
  constructor(name, author){
    this.name = name;
    this.author = author;
  }

  add(item, qty) {
    return this.list.add(item, qty);
  }

  delete (itemName) {
    return this.list.delete(itemName);
  }

  buy(itemName , qty ) { /* ... */ }
}

```



CScript

Replicated Data Types in JavaScript

```
class GroceryList extends SECRO {
  constructor() {
    super();
    this.items = new Map();
  }

  add(item, qty) {
    const description =
      this.items.getOrElse(item.name, {requested: 0, bought: 0});
    description.requested += qty;
    this.items.set(item.name, description);
  }

  post add(oState, state, args, res) {
    const [item] = args,
          addedQty = item.requested,
          resQty = state.items.getOrElse(item.name, 0).requested;
    return resQty >= addedQty;
  }

  delete(itemName) {
    this.items.delete(itemName);
  }
}
```

Overview

- stip.js** ➤ tierless programming with annotations and recommender system
- CScript & SECROs** ➤ strong eventual consistency for replicated objects
- TIPC** ➤ **automatically checking advanced web API constraints**
- Aran & Linvail** ➤ dynamic analyses for JavaScript
- StackFul** ➤ generate test input for web applications
- NodeSentry** ➤ security framework for NodeJS
- SGX** ➤ middleware to support the use of Intel SGX
- Gavial** ➤ combine ideas from multi-tier languages with FRP

TIPC

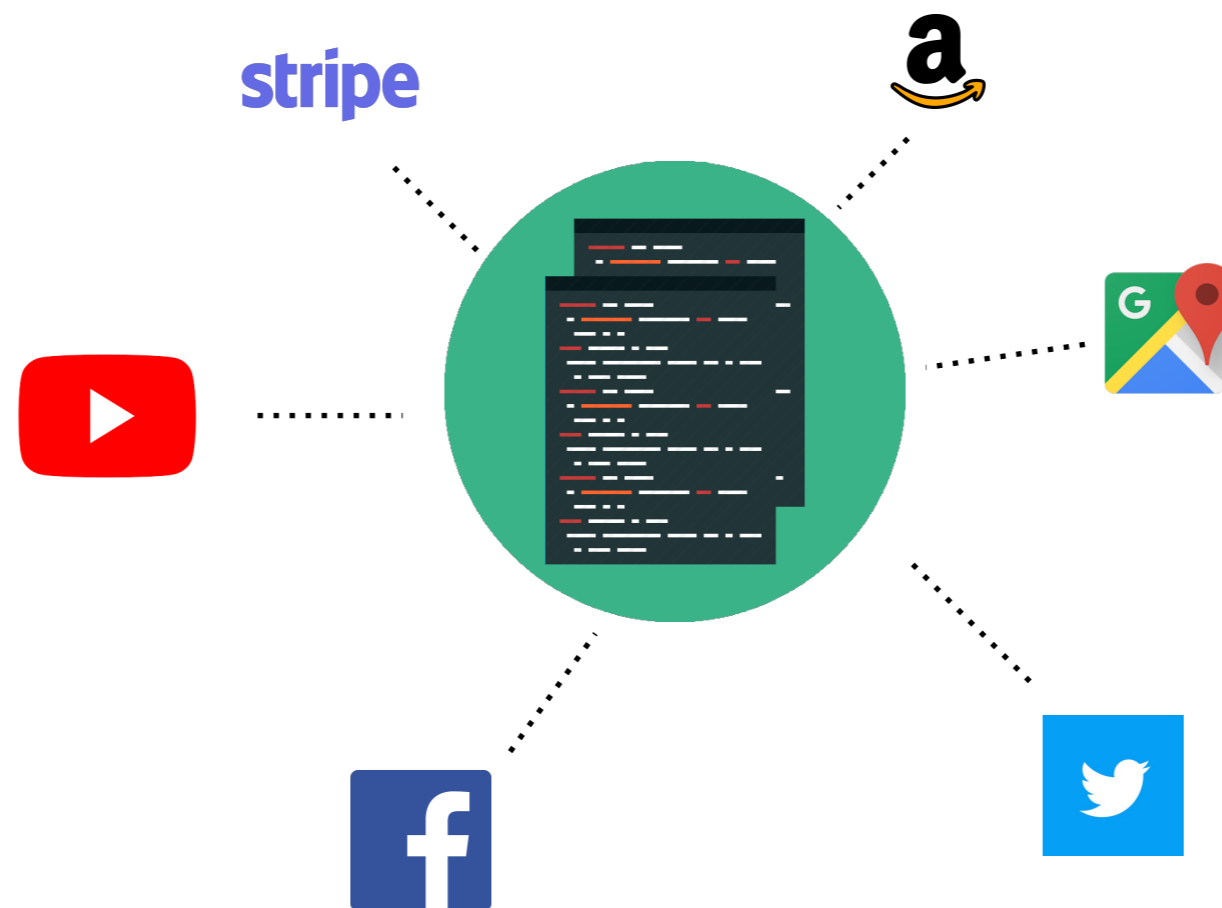
Static type checking for web applications

- ✓ **Improve success rate of API requests in web applications**
- ✓ **Automatic checking of constraints imposed by API provider**
- ✓ **Errors at compile time**

TIPC

Static type checking for web applications

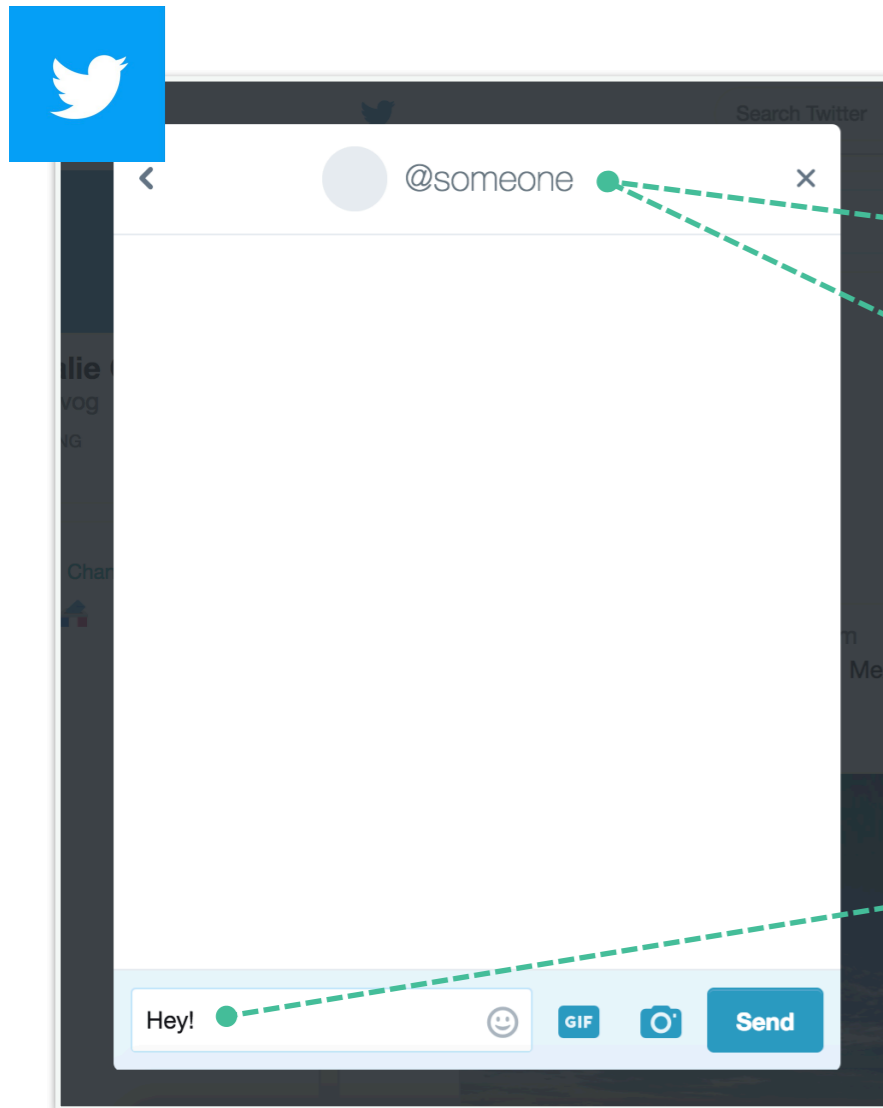
- ✓ **Improve success rate of API requests in web applications**
- ✓ Automatic checking of constraints imposed by API provider
- ✓ Errors at compile time



TIPC

Static type checking for web applications

- ✓ Improve success rate of API requests in web applications
- ✓ Automatic checking of constraints imposed by API provider
- ✓ Errors at compile time

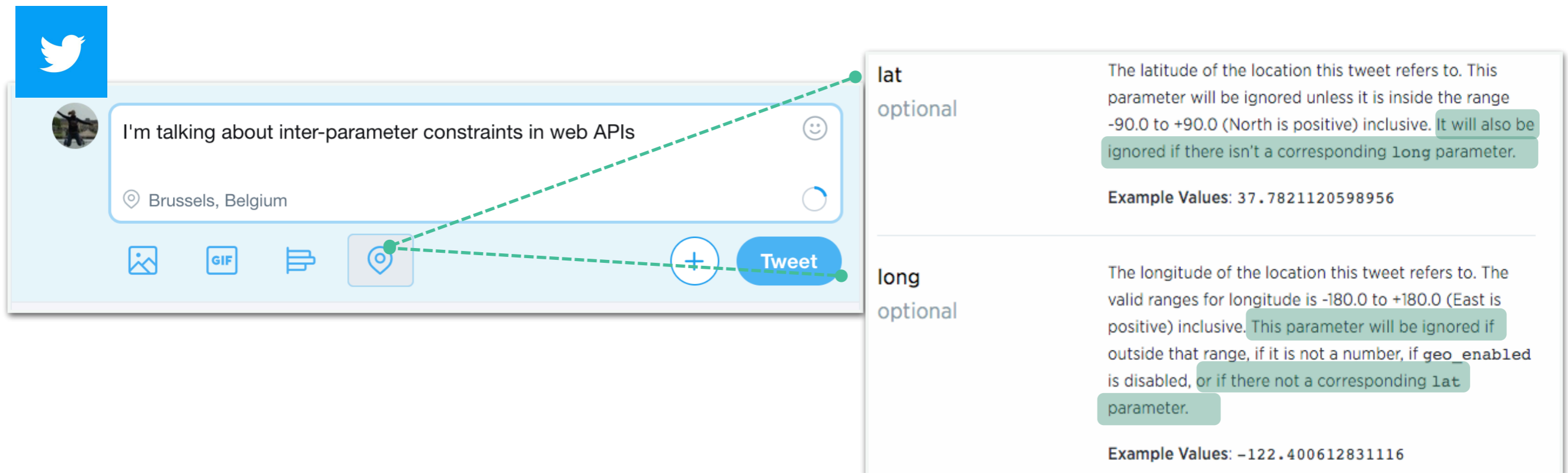


Parameters	
One of <code>user_id</code> or <code>screen_name</code> are required.	
<code>user_id</code> optional	The ID of the user who should receive the direct message. Helpful for disambiguating when a valid user ID is also a valid screen name. Example Values: 12345
<code>screen_name</code> optional	The screen name of the user who should receive the direct message. Helpful for disambiguating when a valid screen name is also a user ID. Example Values: noradio
<code>text</code> required	The text of your direct message. Be sure to URL encode as necessary, and keep the message within the character count limit (available in the help/configuration endpoint) Example Values: Meet me behind the cafeteria after school

TIPC

Static type checking for web applications

- ✓ Improve success rate of API requests in web applications
- ✓ Automatic checking of constraints imposed by API provider
- ✓ Errors at compile time



lat optional

The latitude of the location this tweet refers to. This parameter will be ignored unless it is inside the range -90.0 to +90.0 (North is positive) inclusive. It will also be ignored if there isn't a corresponding long parameter.

Example Values: 37.7821120598956

long optional

The longitude of the location this tweet refers to. The valid ranges for longitude is -180.0 to +180.0 (East is positive) inclusive. This parameter will be ignored if outside that range, if it is not a number, if geo_enabled is disabled, or if there not a corresponding lat parameter.

Example Values: -122.400612831116

TIPC

Static type checking for web applications

- ✓ Improve success rate of API requests in web applications
- ✓ **Automatic checking of constraints imposed by API provider**
- ✓ Errors at compile time



user_id
optional

screen_name
optional

text
required

```
interface Message {  
    user_id:      number;  
    screen_name:  string;  
    text:         string;  
} constraining {  
    present(text);  
    present(user_id) XOR present(screen_name);  
}
```


TIPC

Static type checking for web applications

- ✓ Improve success rate of API requests in web applications
- ✓ Automatic checking of constraints imposed by API provider
- ✓ **Errors at compile time**

```
request.post(  
{ url: "api.twitter.com/1.1/direct_messages/new.json",  
  form: { user_id: 42,  
         screen_name: "Alice",  
         text: "Hello" }},  
(error, r, result) => { console.log(result) });
```



Request to "api.twitter.com/1.1/direct_messages/new.json" is incorrect:
the exclusive constraint on `user_id` and `screen_name` is not satisfied.

TIPC

Static type checking for web applications

demo.ts

```
1 interface PrivateMessage {  
2   · text?: string;  
3   · userid?: number;  
4   · screenname?: string;  
5 } constrains {  
6   · present(text);  
7   · or(and(present(userid), not(present(screenname))),  
8     · and(not(present(userid)), present(screenname)));  
9 }  
10
```

I

Overview

- stip.js** ➤ tierless programming with annotations and recommender system
- CScript & SECROs** ➤ strong eventual consistency for replicated objects
- TIPC** ➤ automatically checking advanced web API constraints
- Aran & Linvail** ➤ **dynamic analyses for JavaScript**
- StackFul** ➤ generate test input for web applications
- NodeSentry** ➤ security framework for NodeJS
- SGX** ➤ middleware to support the use of Intel SGX
- Gavial** ➤ combine ideas from multi-tier languages with FRP

Aran & Linvail

Dynamic analysis platform for web applications

- ✓ **Develop custom dynamic analyses for web applications**
- ✓ **Tailored to JavaScript**
- ✓ **Built-in support for external libraries**

Aran & Linvail

Dynamic analysis platform for web applications

- ✓ **Develop custom dynamic analyses for web applications**
- ✓ Tailored to JavaScript
- ✓ Built-in support for external libraries

program
profiling

data flow
analysis

control-flow
analysis

type
checking

whitebox
testing

model
checking

information
flow analysis

program
slicing

Aran & Linvail

Dynamic analysis platform for web applications

- ✓ **Develop custom dynamic analyses for web applications**
- ✓ Tailored to JavaScript
- ✓ Built-in support for external libraries

program
profiling

data flow
analysis

control-flow
analysis

measure frequency of
function calls

type
checking

whitebox
testing

model
checking

information
flow analysis

program
slicing

Aran & Linvail

Dynamic analysis platform for web applications

- ✓ **Develop custom dynamic analyses for web applications**
- ✓ Tailored to JavaScript
- ✓ Built-in support for external libraries

program
profiling

data flow
analysis

control-flow
analysis

dead code elimination

type
checking

whitebox
testing

model
checking

information
flow analysis

program
slicing

Aran & Linvail

Dynamic analysis platform for web applications

- ✓ **Develop custom dynamic analyses for web applications**
- ✓ Tailored to JavaScript
- ✓ Built-in support for external libraries

program
profiling

data flow
analysis

control-flow
analysis

type
checking

whitebox
testing

model
checking

information
flow analysis

program
slicing

taint analysis

Aran & Linvail

Dynamic analysis platform for web applications

- ✓ **Develop custom dynamic analyses for web applications**
- ✓ Tailored to JavaScript
- ✓ Built-in support for external libraries

program
profiling

data flow
analysis

control-flow
analysis

type
checking

whitebox
testing

model
checking

information
flow analysis

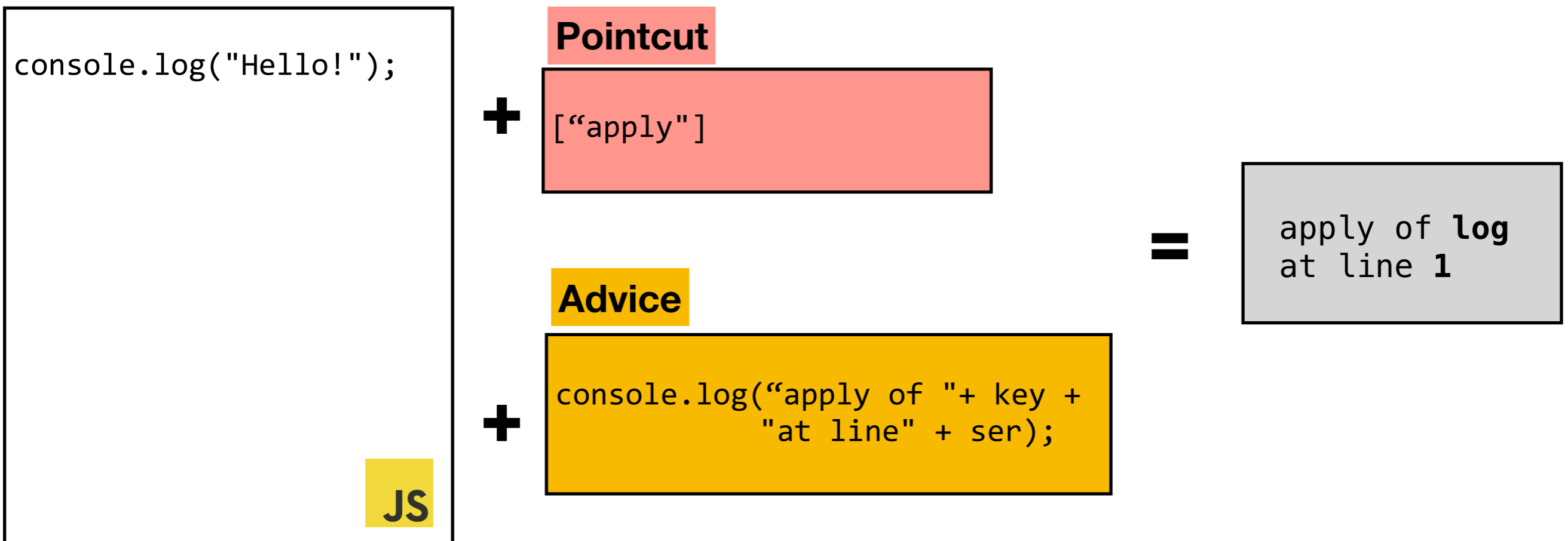
program
slicing

Aran & Linvail

Aran & Linvail

Dynamic analysis platform for web applications

- ✓ **Develop custom dynamic analyses for web applications**
- ✓ Tailored to JavaScript
- ✓ Built-in support for external libraries



Aran & Linvail

Dynamic analysis platform for web applications

- ✓ **Develop custom dynamic analyses for web applications**
- ✓ Tailored to JavaScript
- ✓ Built-in support for external libraries

Pointcut

apply
unary
binary
eval
...

Advice

```
console.log("apply of "+ key +  
           "at line" + ser);  
  
advice.closure = (inner, serial) => {  
  const key = ...  
  return function callee () {  
    counters.set(key, (counters.get(key) || 0) + 1);  
    if (new.target)  
      return Reflect.construct(inner, arguments, new.target)  
    return Reflect.apply(inner, this, arguments);  
  };  
};
```

Aran & Linvail

Dynamic analysis platform for web applications

/live/linvail-taint.js [commonjs]

```

1
2 const Acorn = require("acorn");
3 const Aran = require("aran");
4 const Linvail = require("linvail");
5
6 const advice = {};
7 const pointcut = (name, node) => name in advice;
8 const aran = Aran({format:"script"});
9 const internals = new WeakSet();
10 const callstack = [];
11 const push = (tag, name, serial) => {
12   callstack.push({tag, name, inputs:[], serial});
13   // console.log(Array(callstack.length + 1).join("."), tag, name, serial);
14 };
15 const peek = () => callstack[callstack.length - 1];
16 const pop = () => {
17   const {tag, name, inputs, serial} = callstack.pop();
18   // console.log(Array(callstack.length + 2).join("."), "pop", tag, name, s
19 };
20 global[aran.namespace] = advice;
21 global.eval(aran.setup());
22 const istainted = ($$value) => $$value.meta;
23 const membrane = {
24   taint: (value, reason) => f

```

/target/taint.js

argv...

```

1 const _ARAN_SOURCE_ = "p@ssw0rd";
2 const a = _ARAN_SOURCE_.split("");
3 a.forEach((c) => {
4   const n1 = c.charCodeAt(0);
5   const n2 = 2 * n1;
6   const o1 = new Number(n2);
7   const o2 = {foo:o1};
8   const s = JSON.stringify(o2);
9   const _ARAN_SINK_ = s;
10 });

```

>

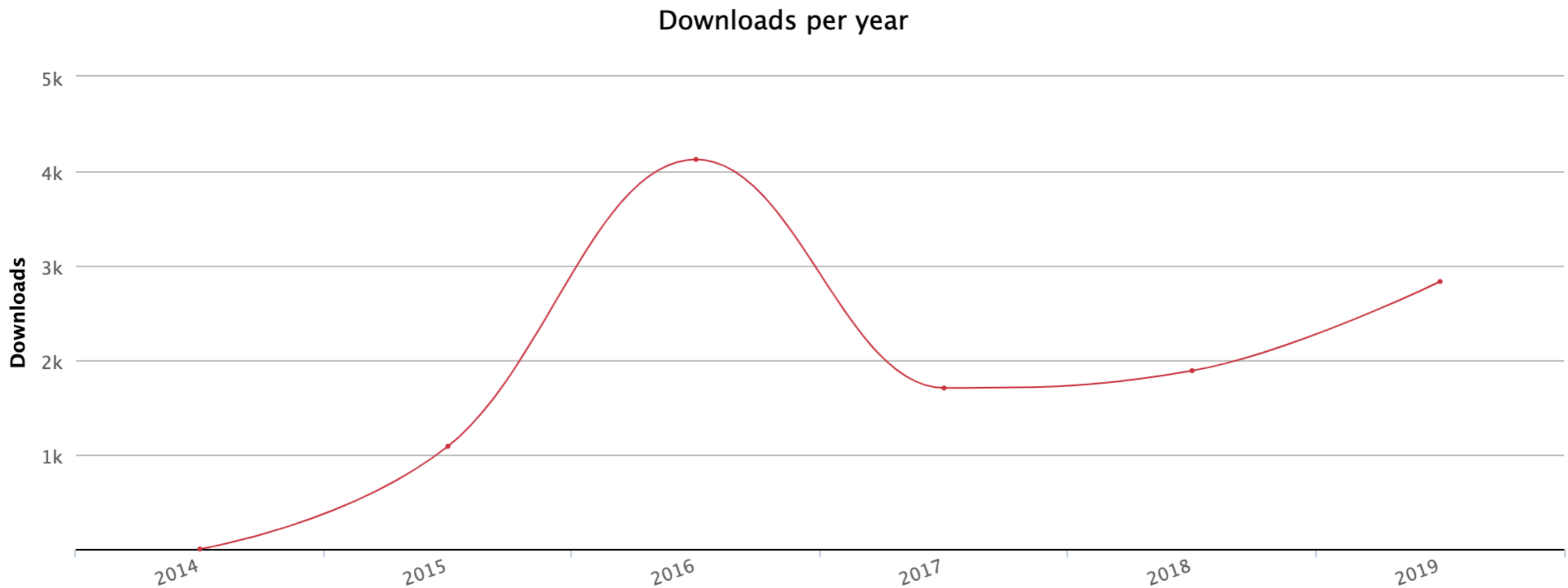
Clear

Aran & Linvail

Dynamic analysis platform for web applications

NPM STAT T

- ✓ Develop custom dynamic analyses for web applications
- ✓ **Tailored to JavaScript**
- ✓ **Built-in support for external libraries**



Overview

- stip.js** ➤ tierless programming with annotations and recommender system
- CScript & SECROs** ➤ strong eventual consistency for replicated objects
- TIPC** ➤ automatically checking advanced web API constraints
- Aran & Linvail** ➤ dynamic analyses for JavaScript
- StackFul** ➤ **generate test input for web applications**
- NodeSentry** ➤ security framework for NodeJS
- SGX** ➤ middleware to support the use of Intel SGX
- Gavial** ➤ combine ideas from multi-tier languages with FRP

StackFul

Concolic Tester for full-stack web applications

- ✓ **Automatically generate test input for web applications**
- ✓ **Support for event-handlers and client-server interactions**
- ✓ **Improves reproducibility of bugs**

https://gitlab.soft.vub.ac.be/mvdcamme/JS_Concolic

Overview

- stip.js** ➤ tierless programming with annotations and recommender system
- CScript & SECROs** ➤ strong eventual consistency for replicated objects
- TIPC** ➤ automatically checking advanced web API constraints
- Aran & Linvail** ➤ dynamic analyses for JavaScript
- StackFul** ➤ generate test input for web applications
- NodeSentry** ➤ **security framework for NodeJS**
- SGX** ➤ middleware to support the use of Intel SGX
- Gavial** ➤ combine ideas from multi-tier languages with FRP

NodeSentry

- Security framework for Node.js
 - Relies on the membrane pattern to regulate the interactions with third-party libraries
 - Upper-bound policies: policies on the interaction of the application with the library API
 - Lower-bound policies: policies on the interaction of the library with other libraries
- Evaluation has shown
 - Policies can “compensate” for library vulnerabilities
 - Performance impact depends mainly on policy
 - The current prototype implementations is fragile

Overview

- stip.js** ➤ tierless programming with annotations and recommender system
- CScript & SECROs** ➤ strong eventual consistency for replicated objects
- TIPC** ➤ automatically checking advanced web API constraints
- Aran & Linvail** ➤ dynamic analyses for JavaScript
- StackFul** ➤ generate test input for web applications
- NodeSentry** ➤ security frameworks for NodeJS
- SGX** ➤ **middleware to support the use of Intel SGX**
- Gavial** ➤ combine ideas from multi-tier languages with FRP

Infrastructural support for Intel SGX

- Intel Software Guard Extensions is a new ISA extension that enables the execution of security-critical code with a very small Trusted Computing Base
- The project developed a variety of middleware to support the use of Intel SGX
 - Automatically generating secure wrappers for SGX enclaves from separation logic specifications
 - Securely deploying distributed computation systems on peer-to-peer networks
- The project also developed attack tools

Overview

- stip.js** ➤ tierless programming with annotations and recommender system
- CScript & SECROs** ➤ strong eventual consistency for replicated objects
- TIPC** ➤ automatically checking advanced web API constraints
- Aran & Linvail** ➤ dynamic analyses for JavaScript
- StackFul** ➤ generate test input for web applications
- NodeSentry** ➤ security framework for NodeJS
- SGX** ➤ middleware to support the use of Intel SGX
- Gavial** ➤ **combine ideas from multi-tier languages with FRP**

Gavial

- Gavial is an embedded DSL for web programming in Scala
 - Combines ideas from multi-tier languages with Functional Reactive Programming
 - Integrates with Scala and Scala.js ecosystems
 - Built-in support for Client/Session/Application tiers
 - Realistic handling of distribution (e.g. incremental behaviors)
- Prototype implementation available on github

Overview

- stip.js** ➤ tierless programming with annotations and recommender system
- CScript & SECROs** ➤ strong eventual consistency for replicated objects
- TIPC** ➤ automatically checking advanced web API constraints
- Aran & Linvail** ➤ dynamic analyses for JavaScript
- StackFul** ➤ generate test input for web applications
- NodeSentry** ➤ security framework for NodeJS
- SGX** ➤ middleware to support the use of Intel SGX
- Gavial** ➤ combine ideas from multi-tier languages with FRP