You Can't Touch This

HCSS Conference, Annapolis, MD

UNI FREIBURG

Albert-Ludwigs-Universität Freiburg

Manuel Geffken Peter Thiemann

University of Freiburg

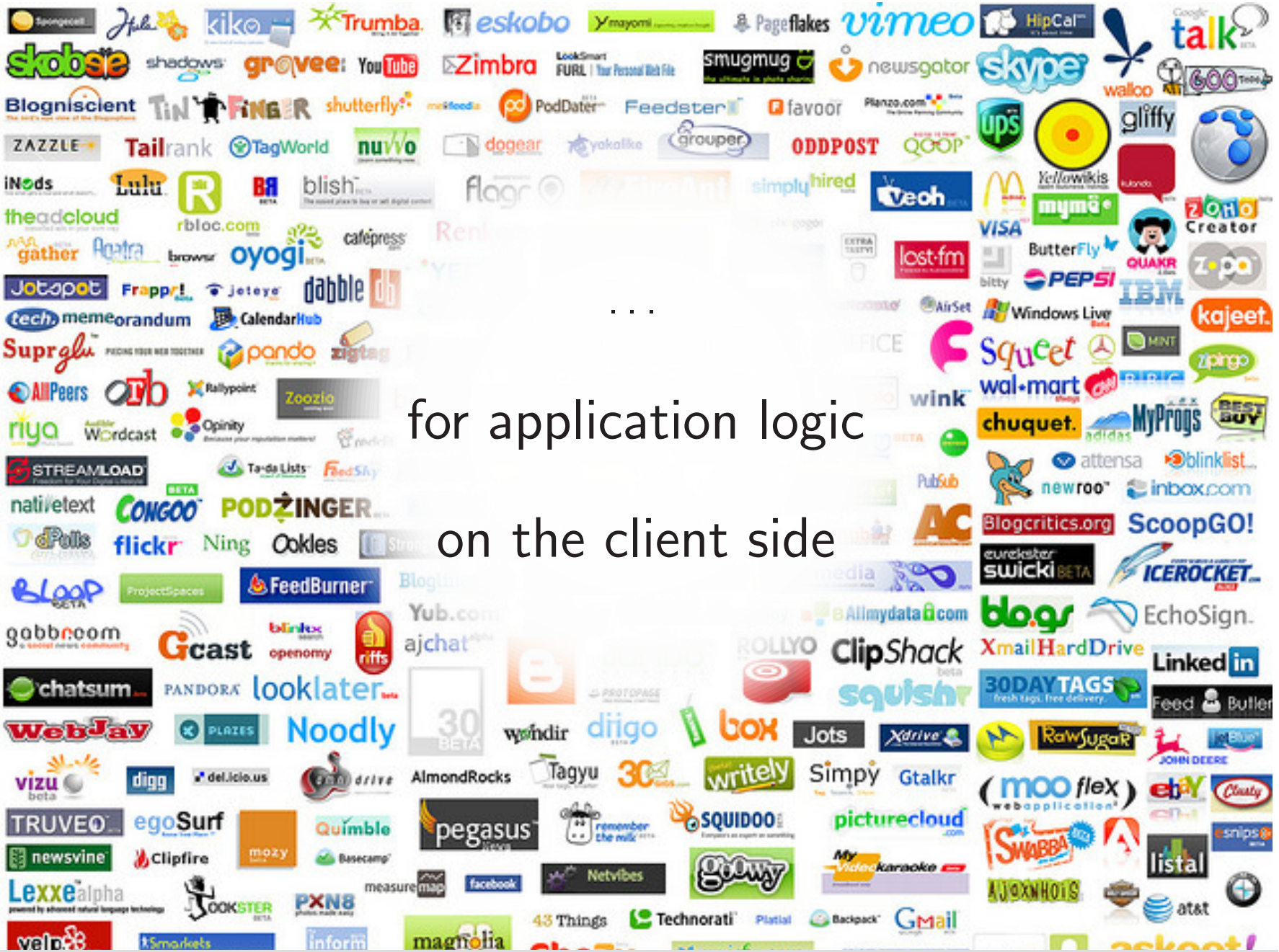
thiemann@informatik.uni-freiburg.de

07 May 2013

Motivation

according to: `http://w3techs.com/`, 30/04/13

Motivation



. . .

for application logic

on the client side

Concerns and Objective

UNI FREIBURG

Security & integrity

Concerns and Objective

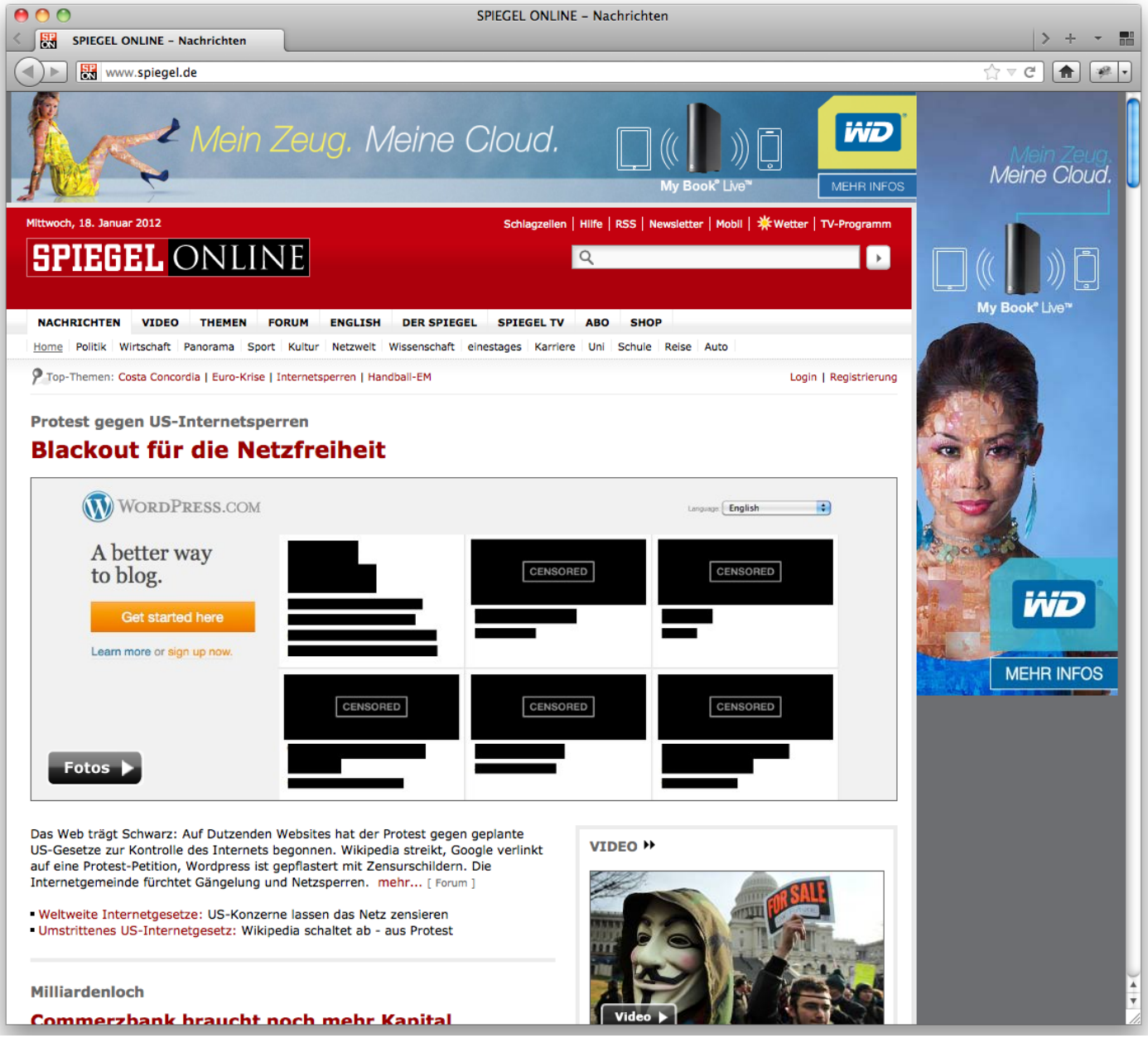
UNI FREIBURG

Security & integrity

Access control for JavaScript objects

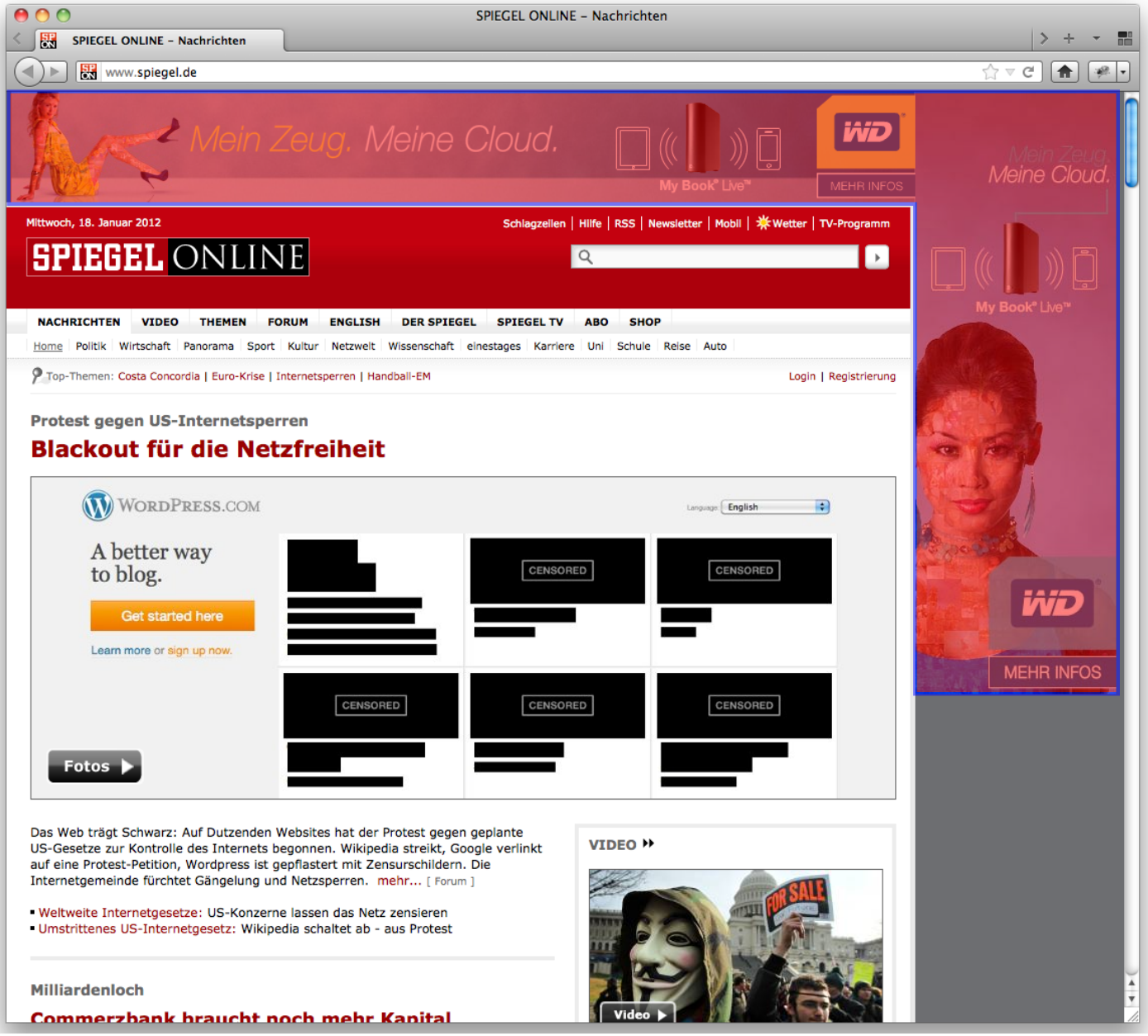
- Domain specific language for specifying AC
- Dynamic analysis:
  Enforcement of AC at run time
- Implementation as JavaScript library
  + extension of SpiderMonkey
- Planned: integration with static AC analysis

Composition of a Web Page

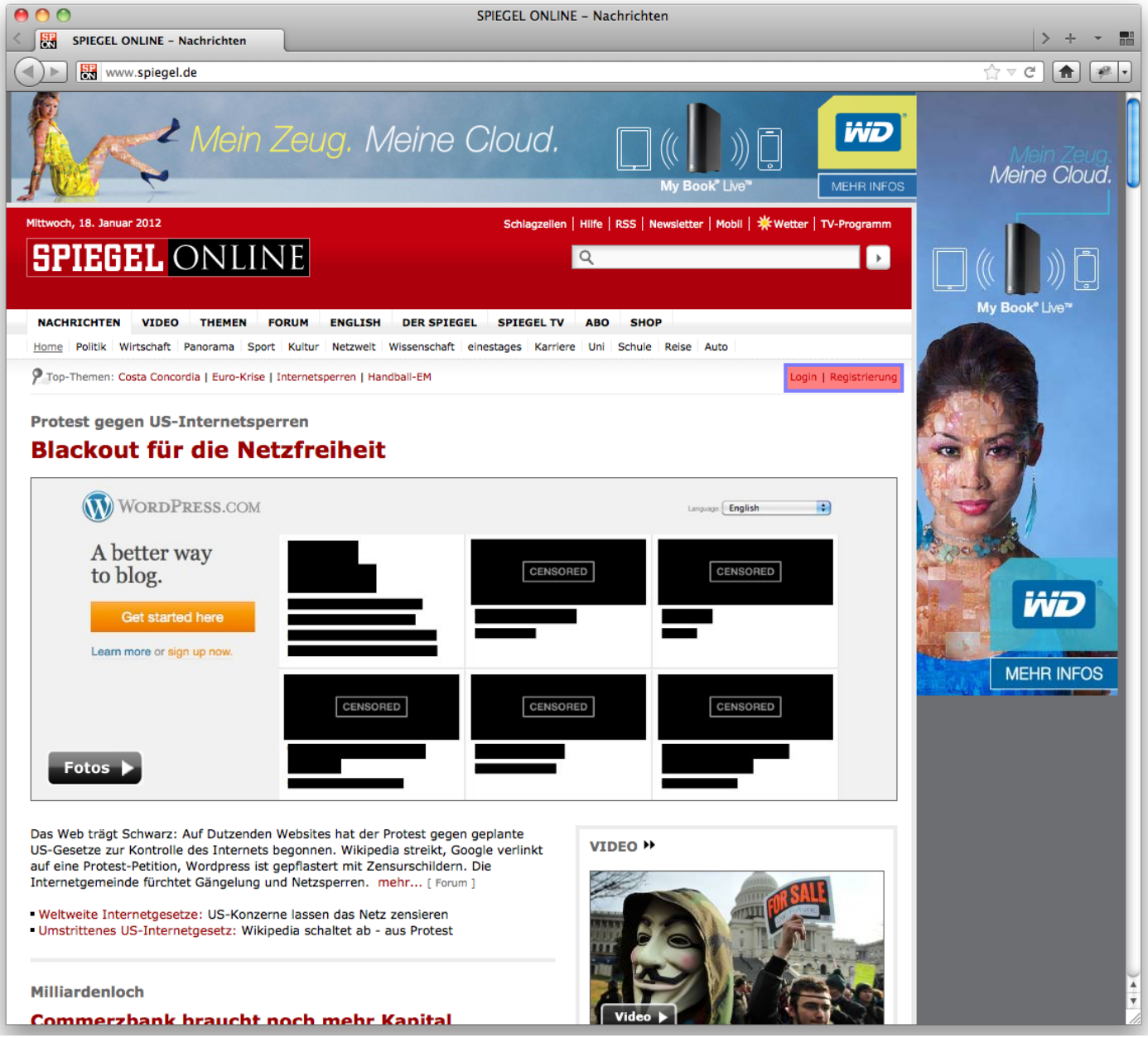


Composition of a Web Page

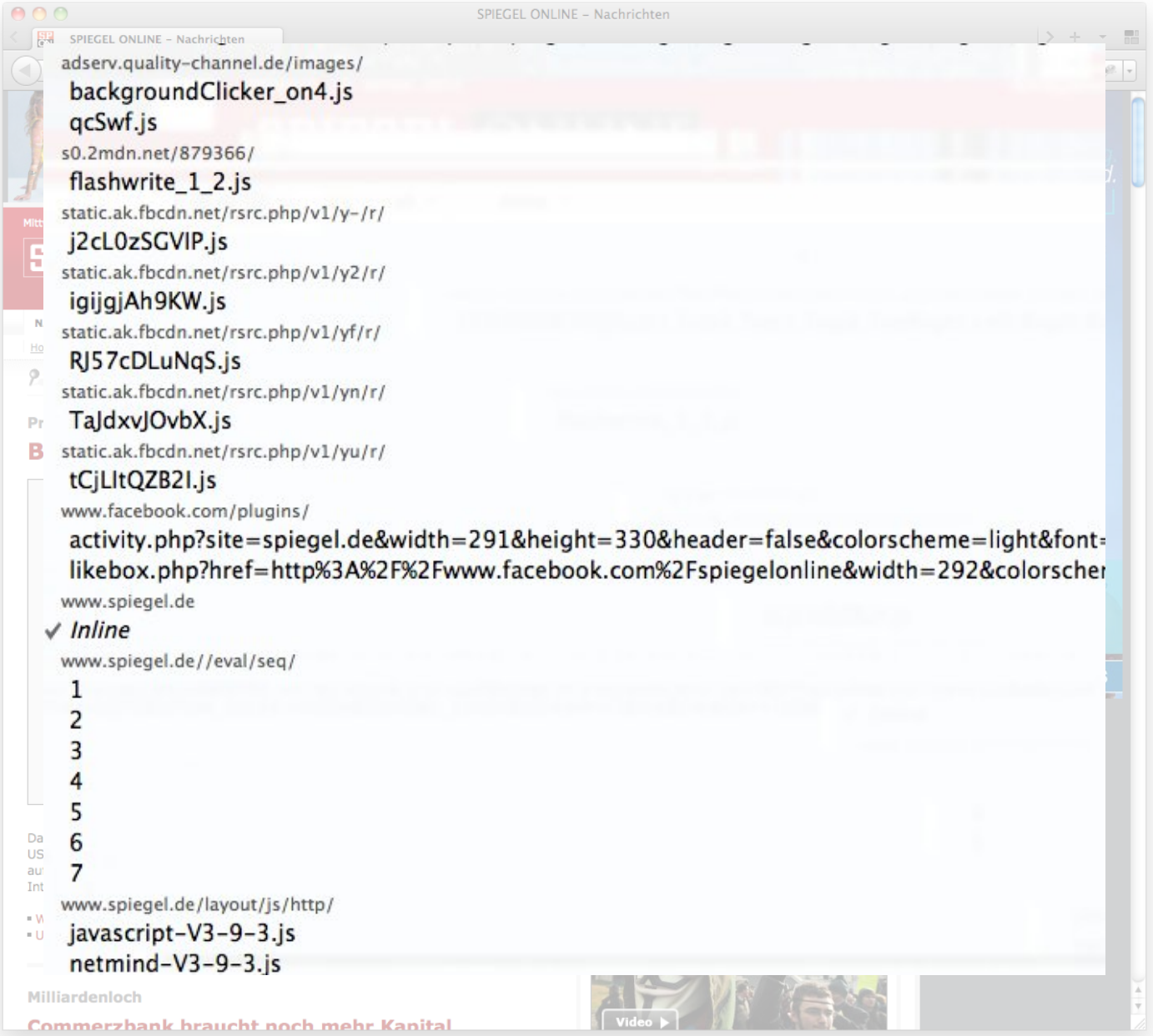
Composition of a Web Page



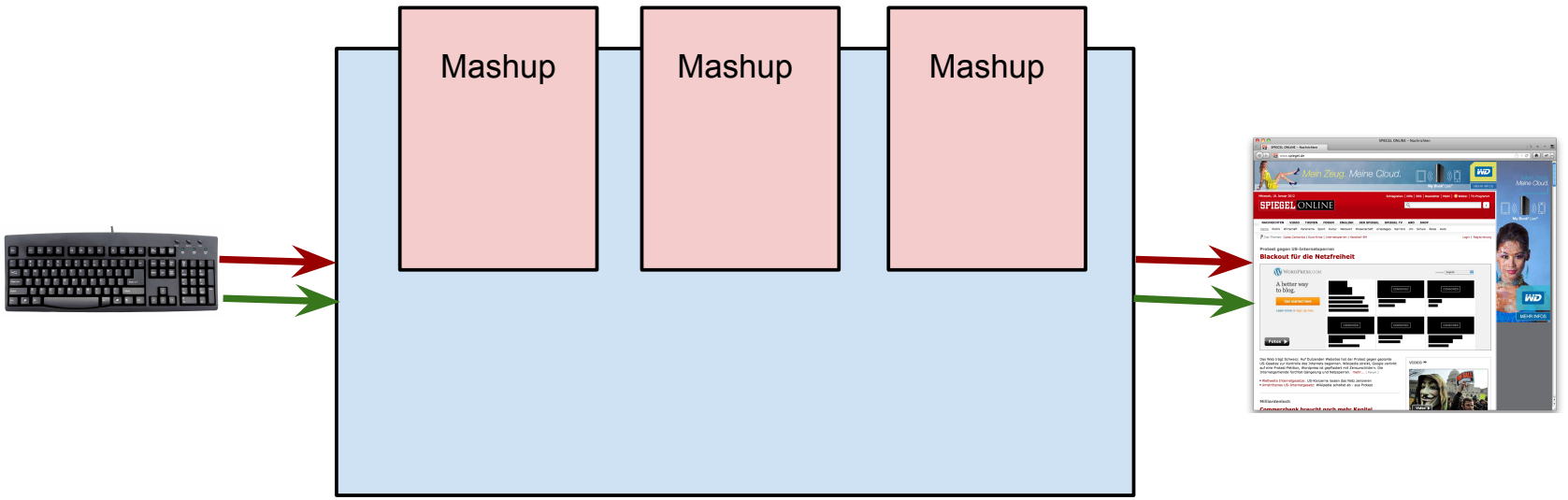
Composition of a Web Page



Composition of a Web Page

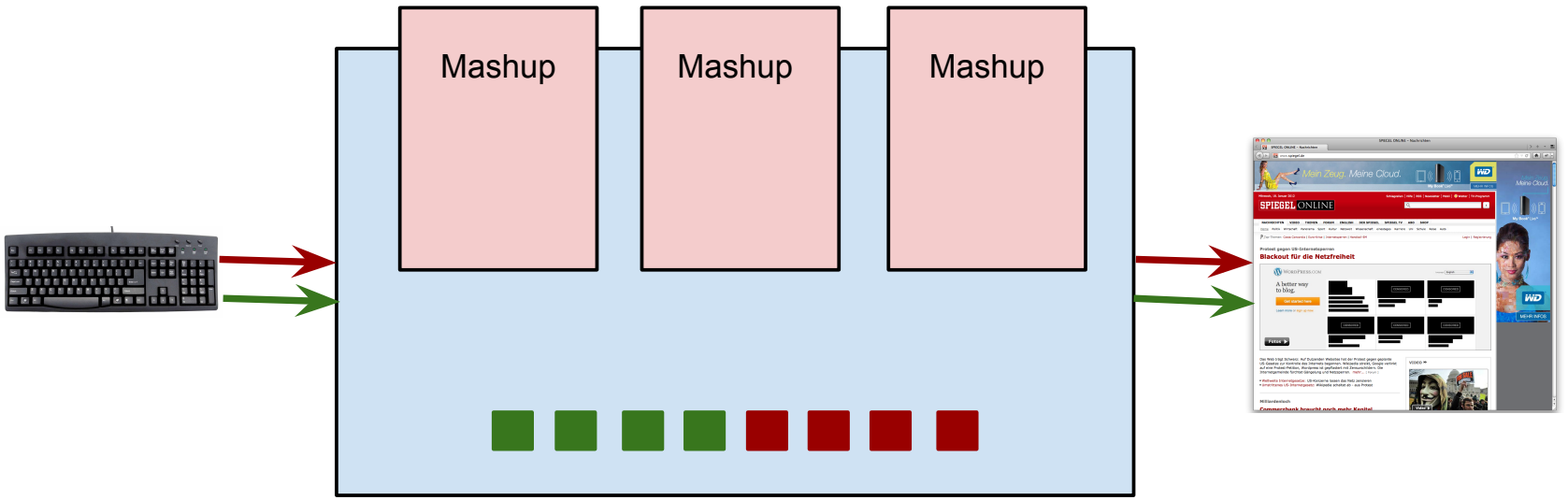


Visualization of the Code



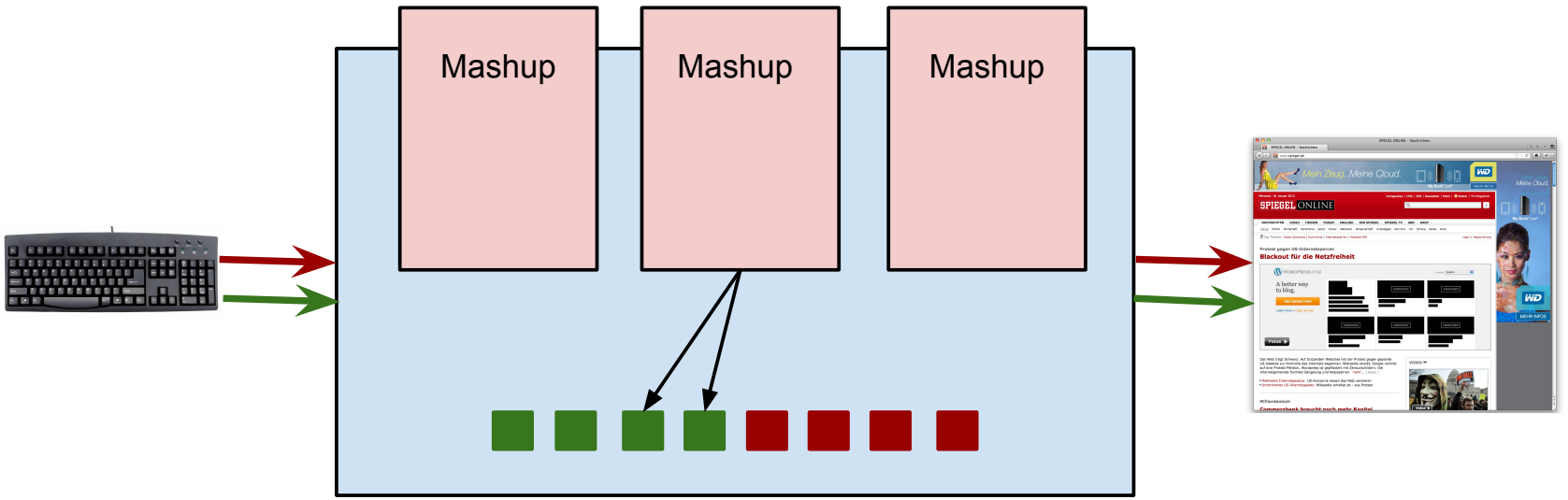
Base Application

Visualization of the Code

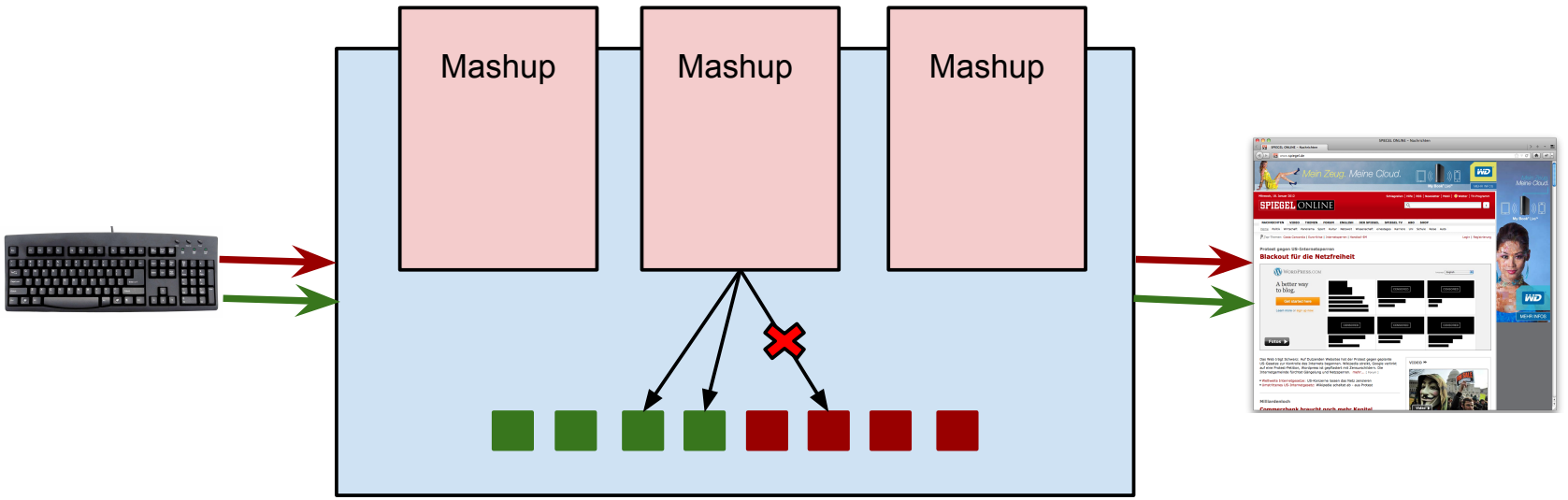


Base Application

Visualization of the Code



Visualization of the Code



Base Application

Problem

UNI FREIBURG

(Mandatory) Access Control for Mashups

- No access to private data of the client
- No access to sensitive resources

Problem

(Mandatory) Access Control for Mashups

- No access to private data of the client
- No access to sensitive resources

What is Needed?

- Demarcation between trusted and untrusted code
- Mashup-specific access-control policies
- Enforcement of these policies

Observation

UNI FREIBURG

In JavaScript, every resource is controlled by reading or writing a property in scope.

Examples

- `document.location`, `document.cookie`, ...
- `window.onload`, `window.onkeypress`, ...
- `node.data`, `node.innerHtml`, ...
- `myData.contacts.JohnDoe.email`, ...

Domain Specific Language for AC

UNI FREIBURG

Basic permissions — sets of object references

```
Read (document, "location|cookie");
Read (window, "onload|onkeypress");
Write(document.documentElement, "*.(data|innerHtml)");
Read (myData, "*.email");
```

Domain Specific Language for AC

UNI FREIBURG

Basic permissions — sets of object references

```
Read (document, "location|cookie");
Read (window, "onload|onkeypress");
Write(document.documentElement, "*.(data|innerHtml)");
Read (myData, "*.email");
```

Building blocks – path sets

a	$::=$	$\texttt{Read}(e, path) \mid \texttt{Write}(e, path)$	anchored path set
	$\mid$	$\texttt{Join}(a, a) \mid \texttt{Meet}(a, a) \mid \texttt{Not}(a)$	boolean operations
	$\mid$	$\texttt{All}$	universal permission

Semantics of AP expressions

UNI FREIBURG

- M : AP expression $\to \mathbb{P}(Loc \times Prop) \times \mathbb{P}(Loc \times Prop)$
  read and write components
- $M(\texttt{Read}(\ell, p)) = (\{(\ell, p)\}, \{\})$
  singleton read set
- $M(\texttt{Write}(\ell, p)) = (\{\}, \{(\ell, p)\})$
  singleton write set
- $M(\texttt{Join}(a_1, a_2)) = M(a_1)(\cup \times \cup)M(a_2)$
  componentwise union
- $M(\texttt{Meet}(a_1, a_2)) = M(a_1)(\cap \times \cap)M(a_2)$
  componentwise intersection
- $M(\texttt{Not}(a)) = (Loc \times Prop, Loc \times Prop) \setminus M(a)$
  componentwise negation
- $M(\texttt{All}) = (Loc \times Prop, Loc \times Prop)$

Enforcing Access Permissions

UNI FREIBURG

DSL: Enforcement of APs

`ENFORCE` takes two parameters

- AP expression describing read set R and write set W
- thunk executed under dynamic monitoring of R and W

Enforcing Access Permissions

UNI FREIBURG

DSL: Enforcement of APs

`ENFORCE` takes two parameters

- AP expression describing read set R and write set W
- thunk executed under dynamic monitoring of R and W

Example: Withdrawing Access Permissions

```
ENFORCE( Not (Join ( Read (...), Write (...))),
  function () {
    // scope of enforcement
  });
```

Example: Granting Access Permissions

UNI FREIBURG

```
function Person(nick, pass, mail) { /* constructor */
  this.nickname = nick;
  this.password = pass;
  this.email    = mail;
}

function base_functionality() {
  var p = new Person("honda", "t243v3r", "mh@t2.com");
  ...
  ENFORCE( Read (p, "nickname"),
    function() { mashup1 (p); });
  ...
  var out = document.getElementById("for_mashup");
  ENFORCE( Join (Read (out, "*"), Write (out, "*"))),
    function() { mashup2 (out, ...); });
}
```

Discussion: Scope of Enforcement

UNI FREIBURG

```
function mash(x, my) {
  ... my.secret ...
}

var r = ENFORCE( Not(
  Read(my, "secret")),
  function() {
    mash(x, my);
  });
```

Discussion: Scope of Enforcement

```
function mash(x, my) {
  ... my.secret ...
}

var r = ENFORCE( Not(
  Read(my, "secret")),
  function() {
    mash(x, my);
  });
```

Lexical Scope

- Restriction applies only to subphrases of `mash(x, my)`
- **Does not impose proper demarcation**:
  untrusted body of `mash` runs without restriction.

Discussion: Scope of Enforcement

```
function mash(x, my) {
  ... my.secret ...
}

var r = ENFORCE( Not(
  Read(my, "secret")),
  function() {
    mash(x, my);
  });
```

Dynamic Scope

- Restriction applies during execution of `mash`.
- Semantics of access permission contracts [POPL2012]

Discussion: Scope of Enforcement

UNI FREIBURG

```
function mash(x, my) {
  return function() {
    ... my.secret ...
  }
}

var r = ENFORCE( Not(
  Read(my, "secret")),
  function() {
    mash(x, my);
  });

r();// may access my.secret
```

Dynamic Scope

- Restriction applies during execution of `mash`.
- Semantics of access permission contracts [POPL2012]
- **Does not impose proper demarcation**:
  If the untrusted `mash` returned a function, then `r()`, i.e., code produced by `mash`, would run without restriction.

Discussion: Scope of Enforcement

UNI FREIBURG

```
function mash(x, my) {
  return function() {
  ... my.secret ...
  }
}

var r = ENFORCE( Not(
    Read(my, "secret")),
    function() {
        mash(x, my);
    });

r();
// no access to my.secret
```

Wrapper Semantics

- The restriction applies to the execution of `mash(x, y)` and to all functions and objects produced by it, recursively.
- If `mash(x, y)` returns a function, then the function call `r()` runs with (at least) the same restriction as `mash`.
- **Fits the requirements.**

Discussion: Scope of Enforcement

```
function mash(x, my) {
  ...  x()  ...
}

var r = ENFORCE( Not(
  Read(my, "secret")),
  function() {
    mash(x, my);
  });

// @syscall
function x() {
  ... my.secret ...
}
```

Wrapper Semantics for Higher-Order Functions

- Suppose `x` is a function called in `mash`'s body.
- Which restriction applies to the execution of `x(...)`?
- Choice#1 (system call): `x`'s creation-time restriction

Discussion: Scope of Enforcement

```
function mash(x, my) {
  ... x()...
}

var r = ENFORCE( Not(
  Read(my, "secret")),
  function() {
    mash(x, my);
  });

// @callback
function x() {
  ... my.secret ...
}
```

Wrapper Semantics for Higher-Order Functions

- Suppose `x` is a function called in `mash`'s body.
- Which restriction applies to the execution of `x(...)`?
- Choice#1 (system call): `x`'s creation-time restriction
- Choice#2 (callback): same plus the call-site's restriction

Customers for Access Permissions

UNI FREIBURG

- Implementer of base application wants to restrict mashups to guarantee confidentiality of the end user's data.
  - Explicit.
  - Instrumenting script tags.
- End user wants to restrict applications.
  - Global restriction.
  - Mapping: URL $\rightarrow$ restrictions.
  - Mapping prepared by third party; might be too complicated / tedious for end user.
- Implementer of mashup provides access restrictions: run time can check compatibility before executing

Implementation

UNI FREIBURG

- Integration in Spidermonkey / Firefox
  - Security application requires total interposition
  - Only achievable in the JS engine (Thank you, eval & friends!)
- DSL implementation and object traversal in JavaScript
- Result: set of objects with permissions
- Interception of read and write operations in Spidermonkey
- Problems
  - set must be a weak set hashed with object identities
  - only available in the latest version of Spidermonkey
  - interface JavaScript vs. implementation language C++

Project Status

UNI FREIBURG

- Mechanized formal semantics
  - Properties of the semantics
  - Correctness of implementation
- Ongoing implementation in Spidermonkey / Firefox
- Corresponding gradual type system
  - in development
  - integrates statically typed and dynamically checked code

The End

UNI FREIBURG

Questions?