Towards Improving the Deprecation Process of Web Features through Progressive Web Security

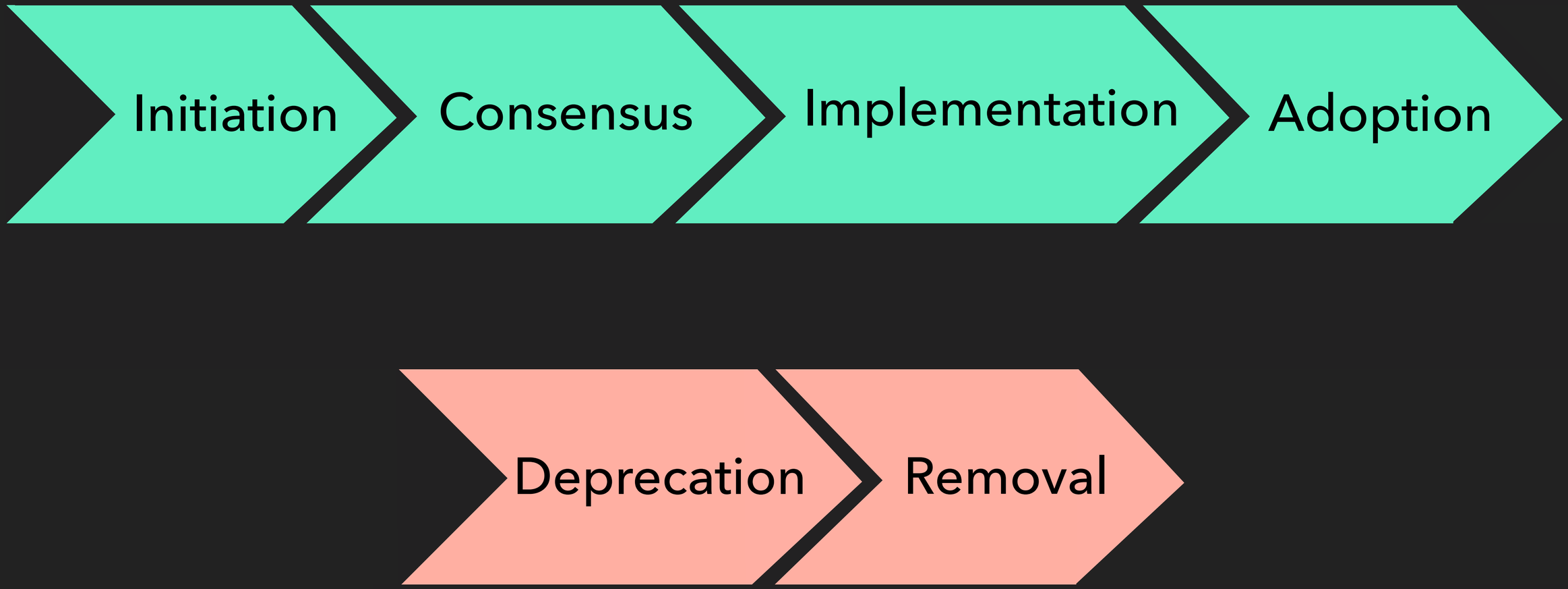
by Tom Van Goethem & Wouter Joosen

DistriNet KU LEUVEN

Browser features

- Browsers are constantly evolving
  - Tens of new mechanisms are added yearly
- Web APIs, CSS features, HTTP header, HTML elements, JavaScript features, and XML markup (MathML & SVG)
- Currently: ~527 browser mechanisms (CanIUse)
  - Each mechanism has a number of different features
  - MDN tracks **11,912** features

Lifecycle of browser features



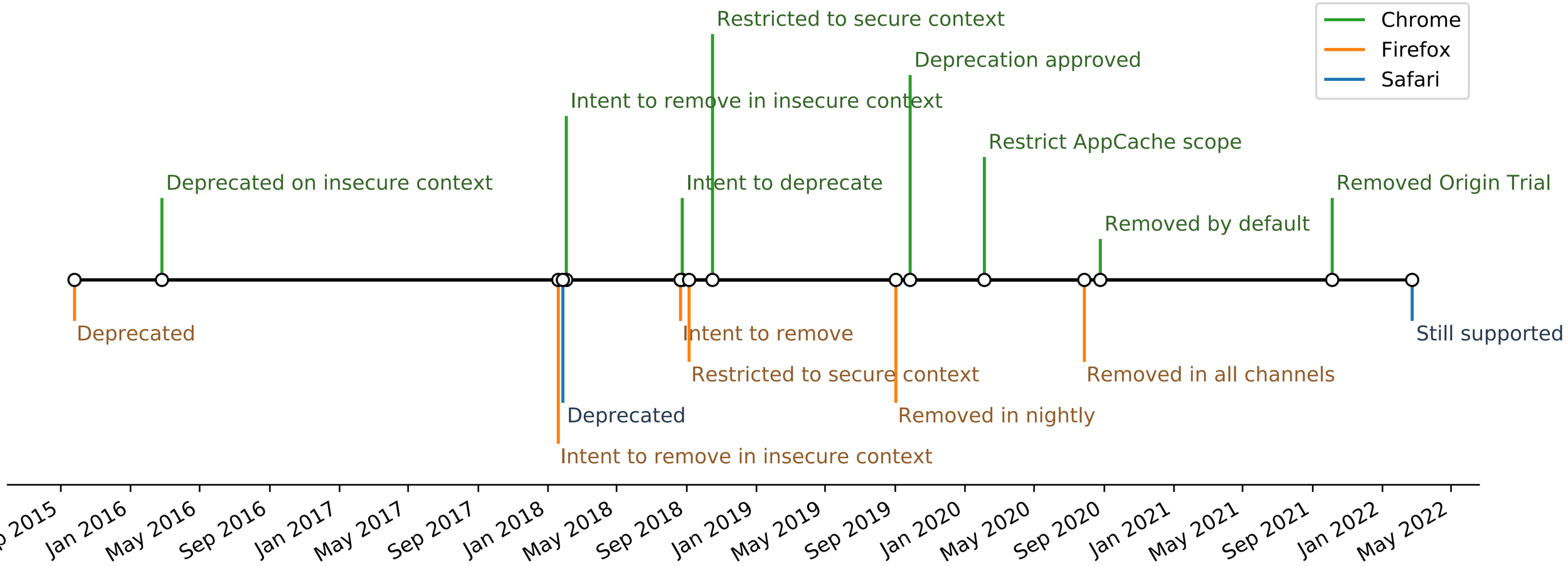
Deprecation and removal

- Features in all stages of development may be removed
- After initiation/consensus
  - Fairly easy: no shipped implementation yet
- After implementation
  - Mainly requires an effort from browser vendors
- After adoption
  - Not that trivial…

Challenges of feature removal

- Many websites might depend on the feature
- Removal may cause breakage in websites
- **Uncoordinated** removal: website breaks in browser A, but not (yet) in browser B
  - Might drive users to another browser…
- Long and painstaking process
  - Deprecation -> removal might take several years
- Security impact: deprecated features can cause vulnerabilities

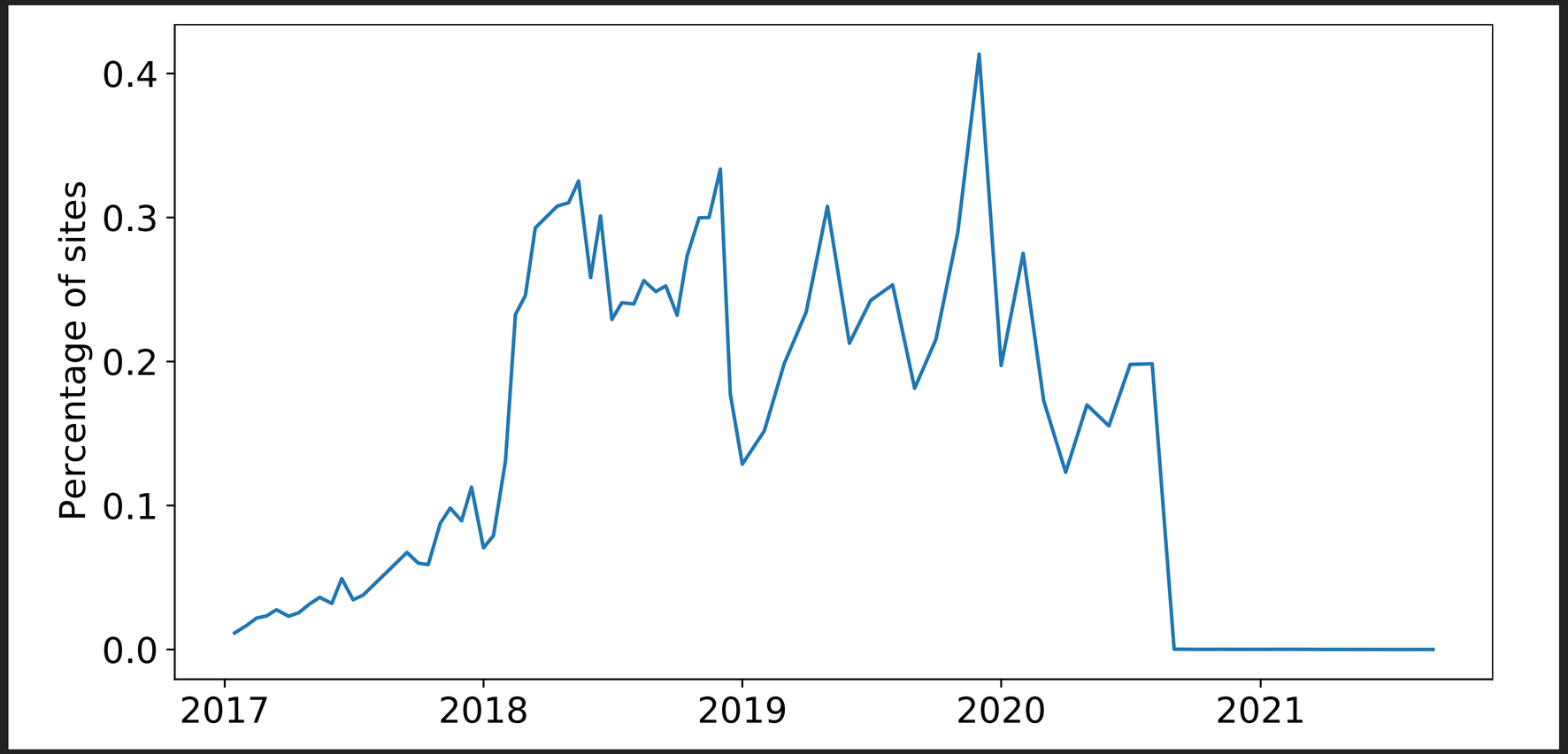
Example: Application Cache



“Application Cache is a douchebag”

- Several design issues with AppCache
  - Superseded by programable Cache API
- First mechanism to allow offline access
  - Non-negligible adoption rate
  - => Took several years to remove from web platform
- Was the main cause of several **security issues**
  - Leaks cross-origin status code (Lee et al. - NDSS’15)
  - Determine URL of redirect (Luan Herrara)
    - User identity, OAuth tokens, ...

Deprecation, not very effective...



Firefox: deprecated Sept 2015

Safari: deprecated Jan 2018

Chrome: deprecated Sept 2018

Chrome & Firefox: removed in Sept 2020

Not just an AppCache problem!

Browser	Deprecated & enabled	Deprecated & removed	Total removed
Chrome	1.093	152	234
Firefox	1.036	209	306
Safari	1.191	54	68

Data from Mozilla Developer Network (MDN)


KUMBAYA....LET'S MAKE THE WORLD
A BETTER PLACE !!!!

Towards improving security

- Synchronized across different browsers
  - Same timeframe for removal across browsers
- Gradual approach: all-at-once would be infeasible
- Tailored mechanism: different features require different approaches for removal
- Eventually move towards security-by-default
- Easy to implement

A proposal: progressive security

- System with incremental versions
  - With every new version: strict security improvements
- Browsers have a default/minimal required version
  - Websites can opt in to a newer version
  - Communicated via response header
- Distinguished between features that may harm other sites vs. features that may harm site itself
  - AppCache: leak information about cross-site responses
  - X-XSS-Protection: enables other sites to leak information

It’s all in your header

```
Progressive-Security:  version ;
                       [unsafe-opt-out=(feature list) ;]
                       [unsafe-opt-in=(feature list)]
```

Versioned mechanism

- Each version has list of:
  - Deprecated features
  - Unsupported features: disabled by default - allows opt-in
    - Requires user approval when feature could be abused
  - Removed features: cannot be re-enabled
  - Enabled by default features: security mechanisms (can opt out)
  - Enabled by default without opt out
- New versions gradually improve security of the web platform

AppCache

-- version 1 --
 # deprecated
+ appcache-insecure

-- version 2 --
 # deprecated
+ appcache

-- version 3 --
 # removed
+ appcache-insecure

-- version 4 --
 # unsupported
+ appcache

-- version 5 --
 # unsupported
- appcache
 # removed
+ appcache

Origin isolation

-- version 1 --
 # deprecated
+ document-domain
+ cross-origin-wasm

-- version 2 --
 # enabled-by-default
+ origin-agent-cluster
 # unsupported
+ document-domain
+ cross-origin-wasm

-- version 3 --
 # unsupported
- document-domain
- cross-origin-wasm
 # removed
+ document-domain
+ cross-origin-wasm
 # enabled-by-default
- origin-agent-cluster
 # enabled-by-default-no-opt-out
+ origin-agent-cluster

Conclusion

- Many features are introduced to the Web
- Difficult to remove features when they start getting used
- Currently no synchronization between browser vendors on feature removal
  - May delay actual removal
- We proposed “progressive security”
  - Requires synchronization between browsers
  - Supports current feature removal patterns
  - Versioned system that gradually improves security

@tomvangoethem

Would love to hear what you think!

Is synchronization between browsers feasible?

Should we offload decisions to opt-in to dangerous functionality to users?

Are there alternative/better approaches?

DistriNet KU LEUVEN