Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

lively4 Cache

Meike Baumgärtner & Jan Graichen

Web Development 2016
Software Architecture Group
Supervisors Jens Lincke and Stefan Lehmann

Context

Lively 4

Next generation in-web publishing platform

➔ Embrace newest web technologies

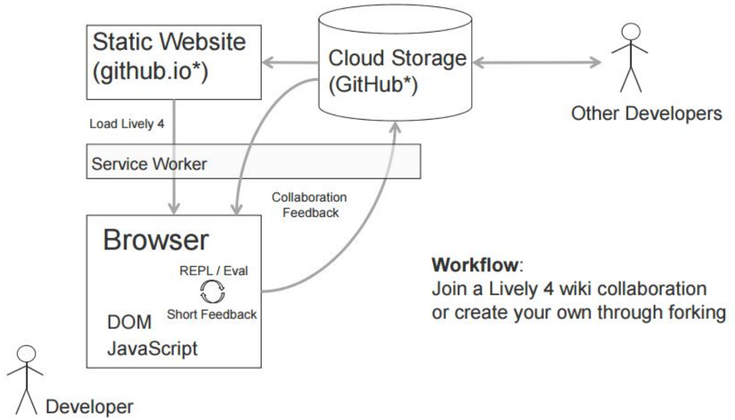
➔ Reuse existing web APIs

Web Development 2016

Inter-Team Collaboration

➔ Working on living code (don't break things)

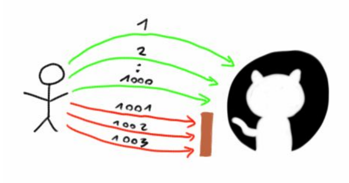
➔ Weekly inter-team meetings (if available)



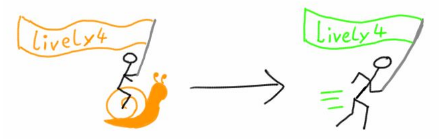
Motivation

Goals

➔ Avoid hitting API limits

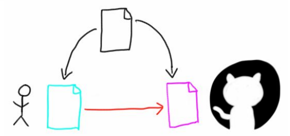


➔ Speed up loading lively environment

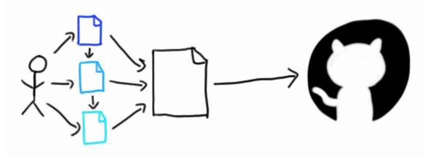


➔ Work offline

➔ Write conflict detection



➔ Local write store / bulk write



Background: Unified Data Backend

```
/                                    [reducted]
├── bin
├── node_modules
├── src
├── templates
├── test
├── sys
│   ├── mounts
│   ├── fs
│   |   ├── mount
```

This is not Unix.

Service Worker based hierarchical “file system”

- ➔ Uniform API (read, write, stat)
- ➔ Mount different filesystems (exchange)
- ➔ Mount subtree at any point (combine)

Everything is a file

- ➔ Expose internals as filesystem
- ➔ Control and configure by writing to files

Background: Cache

Read Cache

- ➔ Usually transparent to user
  - ◆ HTTP client cache
  - ◆ OS file system caches
  - ◆ RAM page cache
  - ◆ CPU instruction cache
- ➔ Widely used technique
  - ◆ In-Memory cache techniques
  - ◆ Redis, Memcached

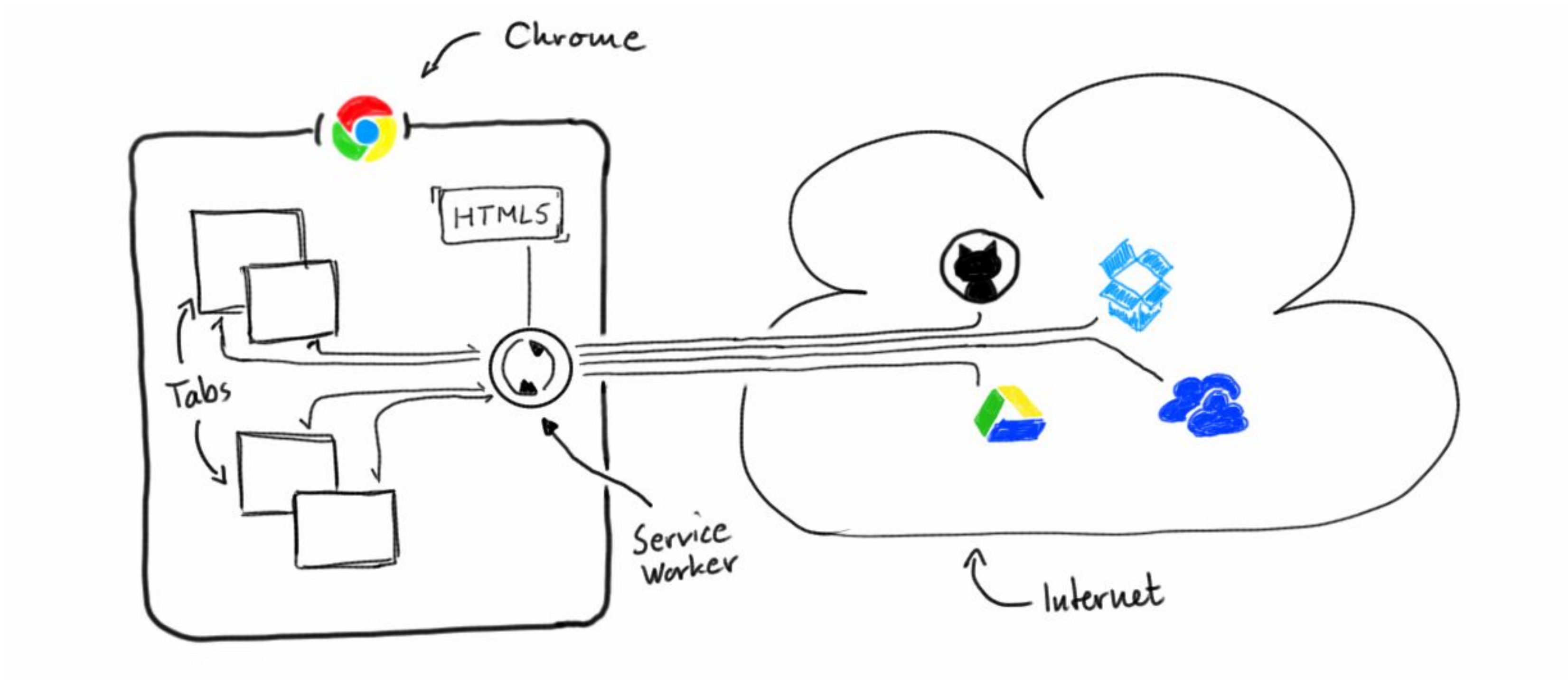
Write Cache

- ➔ Often explicit controlled
  - ◆ Git Index
  - ◆ Offline google docs
- ➔ Less used
  - ◆ Requires application specific handling code (conflicts)

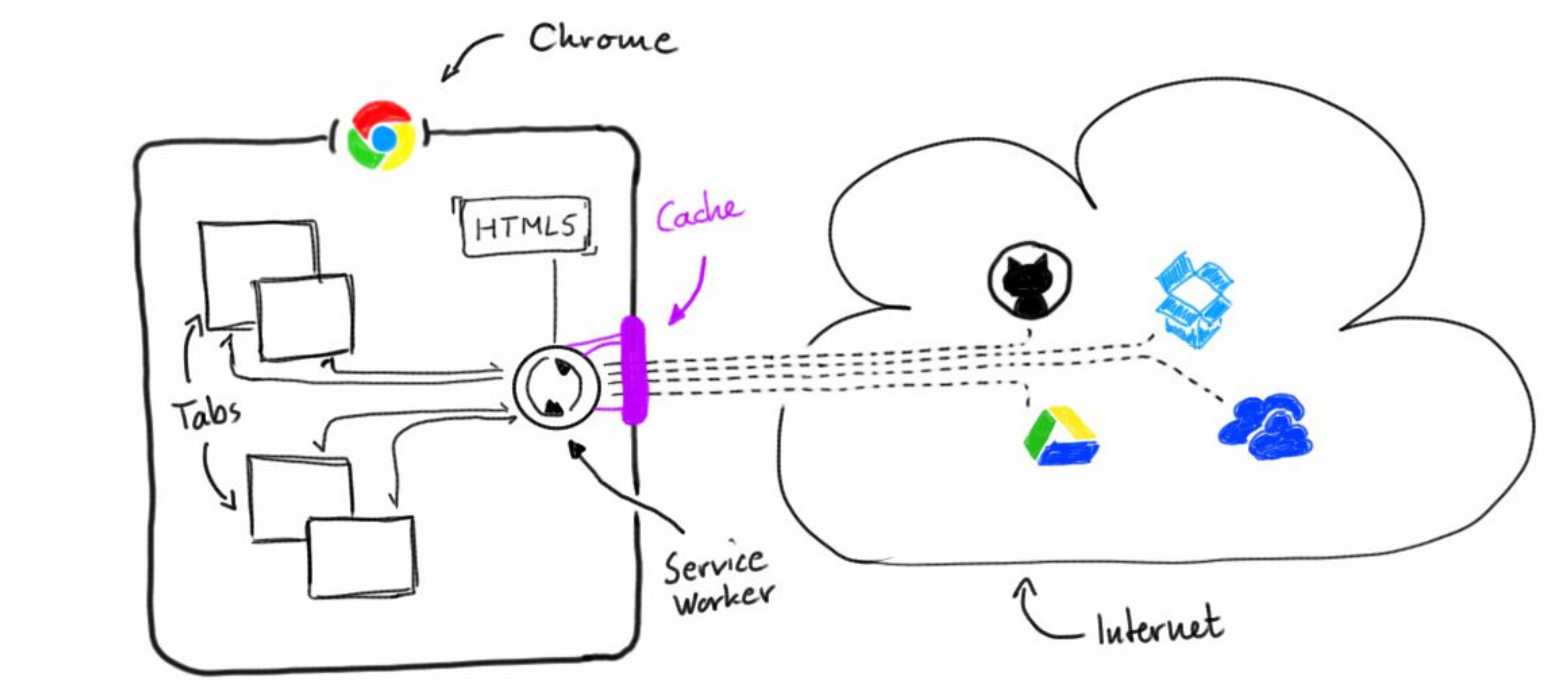
A local programmable caching server?

> **Service workers** essentially act as **proxy servers** that sit **between web applications**, and the browser **and network** (when available.)
>
> They are intended to (amongst other things) enable the creation of effective **offline experiences**, intercepting network requests and taking appropriate action based on whether the network is available and **updated assets reside on the server**. They will also allow access to push notifications and background sync APIs.
>
> *- Mozilla Developer Network*

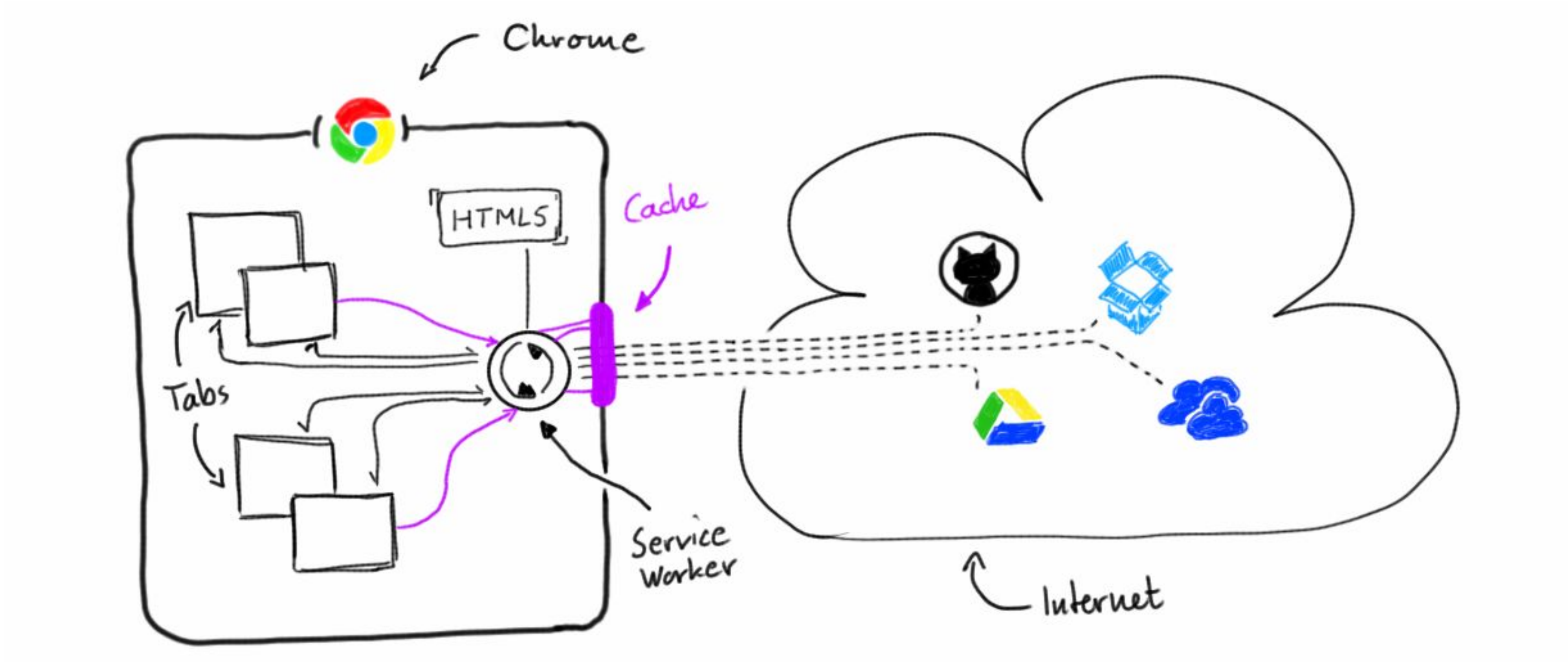
Unified Data Backend Concept



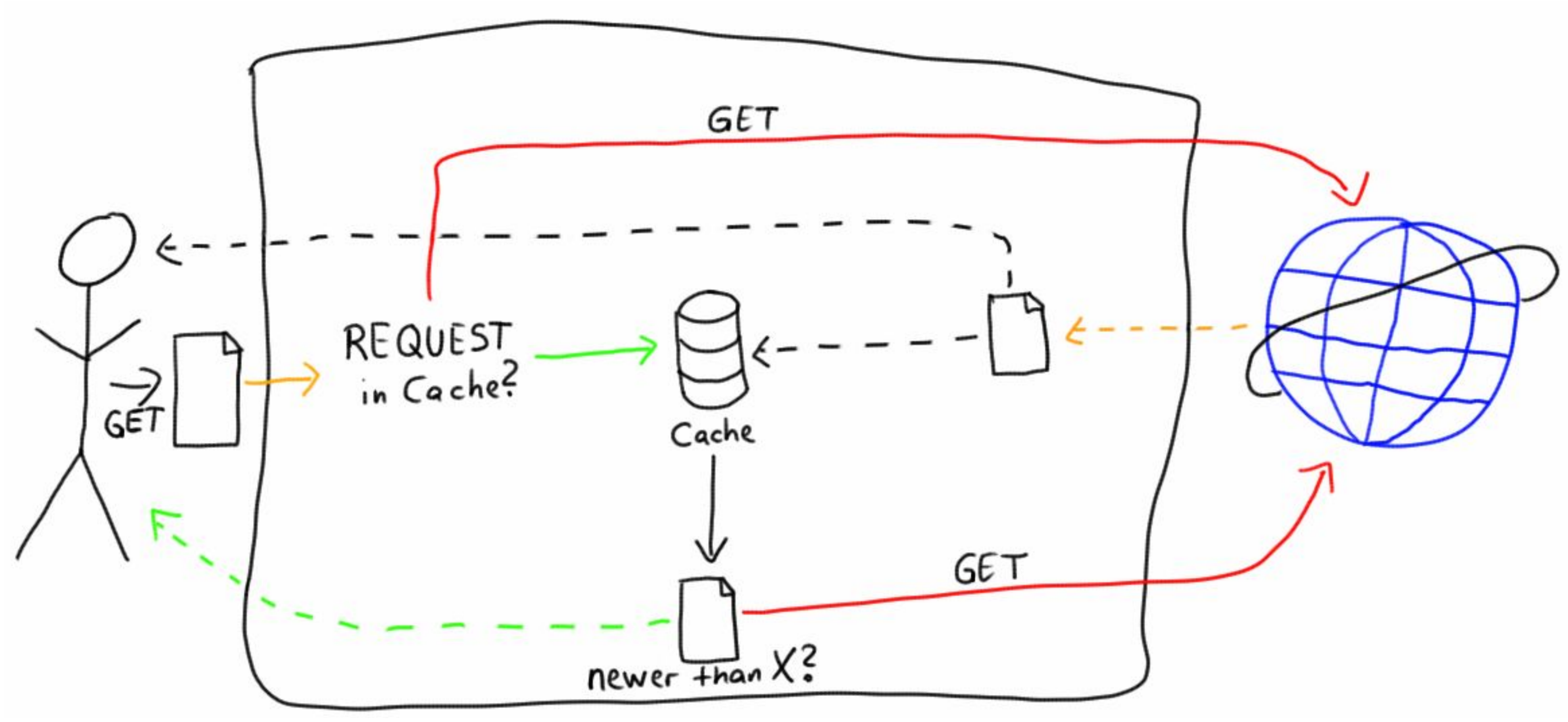
Cache Concept



Cache Interaction Concept



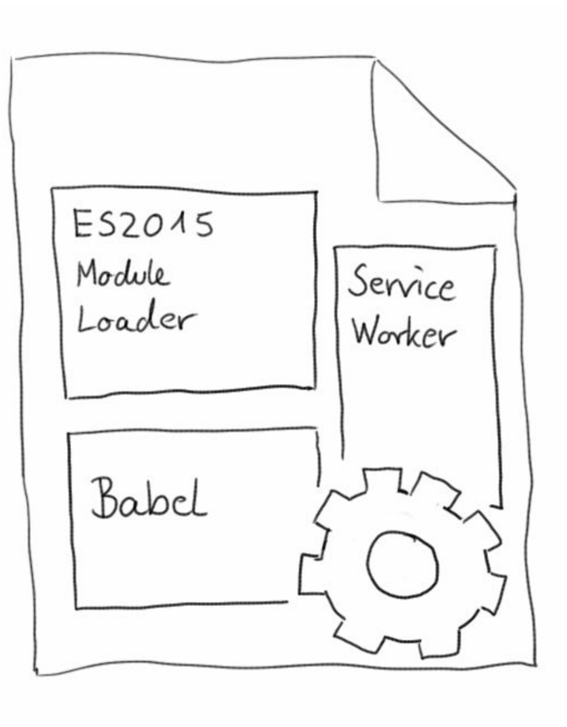
Cache Internals



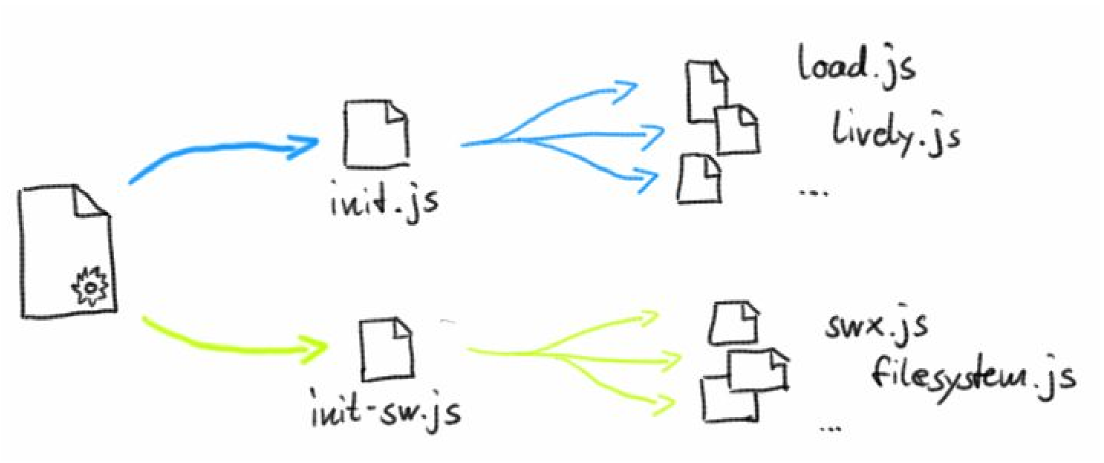
Fast Boot-Up: lively4-loader

Single precompiled loader:

- Handling ES2015 modules
  - Custom ES6 module micro loader
- Transpiling ES6/7+ code
  - Babel 6 w/ plugin support
  - No transpile to ES3 but latest Chrome (already has 98% ES6 cov)
- Initialize service worker (SW)
  - Avoid SW scoping issues
- Load "init process" files
  - Expose small kernel API as module



Fast Boot-Up: lively4-loader



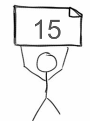
Fast Boot-Up: lively4-loader

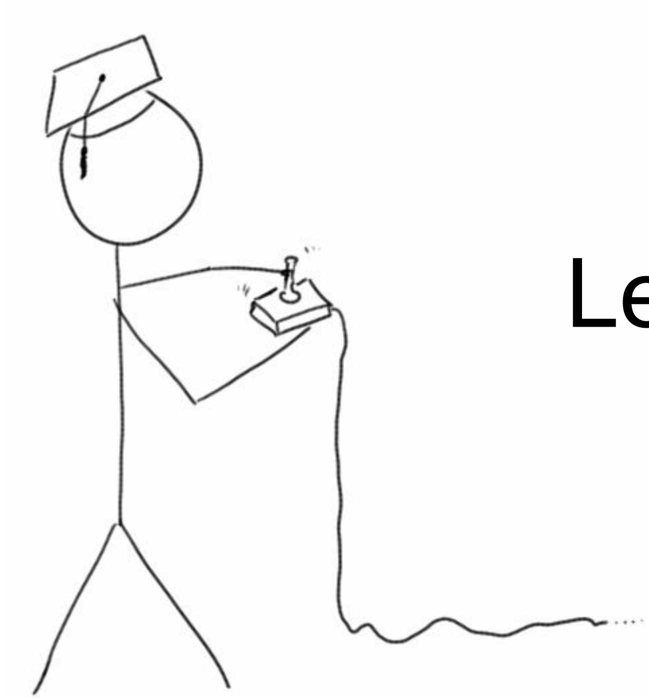
```
// kernel.conf.l4.js
module.exports = {
  LOADER_TRANSPILE: true,
  WORKER_BASE: "https://raw.githubusercontent.com/LivelyKernel/lively4-serviceworker/master/src/",
  WORKER_ENABLED: true,
  WORKER_INIT: "/swx.js",
  WORKER_EMBED: false,
  CLIENT_ENABLED: true,
  CLIENT_BASE: "./",
  CLIENT_INIT: "/src/init.js",
}
```

```
$ # Compile loader once
$ KERNEL_CONFIG=./kernel.conf.l4.js  npm start  -- --output-file ../lively4-core/swx-loader.js
Hash: de9f1d334fb3a11c988f
Version: webpack 1.13.1
Time: 2605ms
                            Asset     Size  Chunks             Chunk Names
    ../lively4-core/swx-loader.js  1.43 MB       0  [emitted]  kernel
../lively4-core/swx-loader.js.map  1.76 MB       0  [emitted]  kernel
  [0] multi kernel 28 bytes {0} [built]
   + 550 hidden modules
```

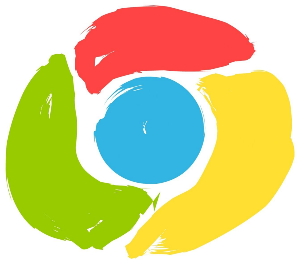
Fast Boot-Up: lively4-loader

```
<html>
  <head>
    <title>Lively 4 Kernel Example Page</title>
    <script src="../dist-kernel-loader.js"
            type="text/javascript"
            data-lively-kernel>
    </script>
  </head>
  <body>
    <script>
      System.import('/examples/test.js').then((test) => {
        if (Notification.permission !== "granted")
          Notification.requestPermission();
        var notification = new Notification(test.message);
      })
    </script>
  </body>
</html>
```





Let’s Play



Discussion

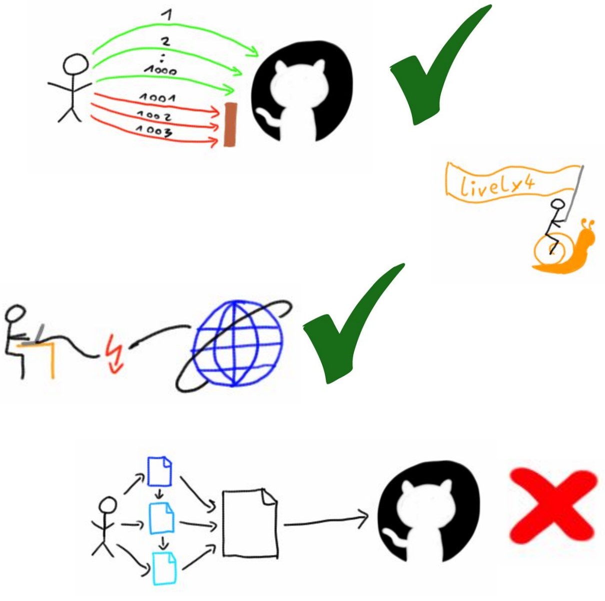
- Concept Advantages
    - Based on HTTP caching
    - Read cache transparent to user
    - Integration into filesystems
    - Client library independent
- Technology Advantages
    - Use browser cache API
    - Build on client library independent code

- Serviceworker Limitations
    - No caching of early boot files without hacks
    - Cannot/does not cache its own files
- Filesystem Limitations
    - File systems need to be adjusted
    - Does not quickly detect changes (no backend support)
    - Limited sysfs controlling

Future Work

- Collect multiple changes and write as bulk
  - FS-specific bulk writes: git commit multiple files [more]
- Self-contained pre-compiled minimal loader for distribution
  - Use it! Boot lively4 core from loader
- Cache loader files itself
  - start.html, swx-loader, SW files
- Handle conflicts in editor, file browser, diff tool [more]
- Use file systems (infrastructure) on lively4-server

Conclusion



Yes, I can load and (save) files fast.

Sources

https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API

http://uxrepo.com/static/icon-sets/font-awesome/png32/256/000000/linux-256-000000.png

Inspiration: The Greatest™ Seminar Software Design 15/16 Team 4 Final Presentation

lively4 Cache

Meike Baumgärtner & Jan Graichen

Web Development 2016
Software Architecture Group
Supervisors Jens Lincke and Stefan Lehmann

Future Work

- Collect multiple changes and write as bulk
  - FS-specific bulk writes: git commit multiple files [more]
- Self-contained pre-compiled minimal loader for distribution
  - Use it! Boot lively4 core from loader
- Cache loader files itself
  - start.html, swx-loader, SW files
- Handle conflicts in editor, file browser, diff tool [more]
- Use file systems (infrastructure) on lively4-server

Explicit commit on GitHubFS

```
PUT /src/file1.js
    ...
PUT /src/file2.js
    ...

GET /sys/fs/0/changes
// => {files: {435: “/src/files1.js”, 264: “/src/files2.js”}, diff: ...}

PUT /sys/fs/0/commit
  {message: “Change whitespace formatting”, ...}
// => {sha: “43f45...”, url: “https://github.com/...”}


// PS.: Could be mirrored on lively4-server for remote usage (gitfs)
```

Conflict detection

```
GET /file.js
  ETag: 63fee7436

PUT /file.js
  If-Unmodified: 63fee7436
  //=> 412 Precondition failed

GET /file.js
  //=> Returns new file content

// Do local diff in editor and save with new ETag
```