Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

lively4

Unified Data Backend

Meike Baumgärtner & Jan Graichen

Software Design 2015/2016
Software Architecture Group
Supervisors Jens Lincke and Stefan Lehmann

Context

Lively Kernel 4

Next generation in-web publishing platform

- ➔ Prototype of Lively 4
- ➔ Embrace newest web technologies
- ➔ Reuse existing web elements

Software Design 2015/16

Inter-Team Collaboration

- ➔ Other teams depend on file access
- ➔ File browser based on Morphic Team
- ➔ Weekly inter-team meetings

Motivation

I am a lively4 user and I want to…

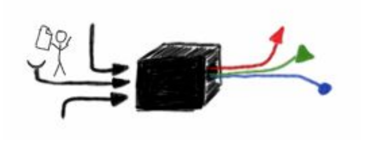
… load files and store changes…

…e.g. on GitHub, Dropbox or local…

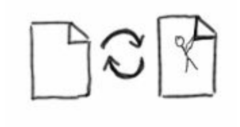
… without knowing the specific API.

Goals

➔ Abstraction layer to access files and directories

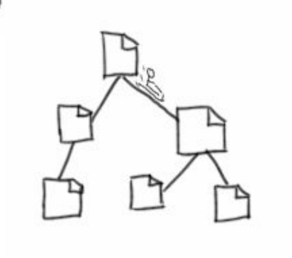


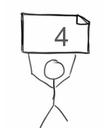
➔ Exchange file backend



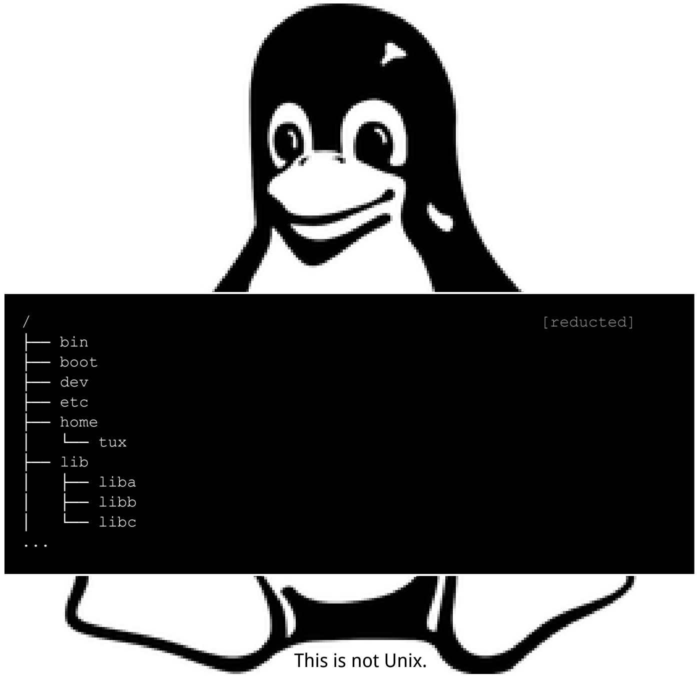
➔ Combine file backends

➔ Visually explore file backends





Background: Unix File Systems



```
/                                                    [reducted]
├── bin
├── boot
├── dev
├── etc
├── home
│   └── tux
├── lib
│   ├── liba
│   ├── libb
│   └── libc
...
```

This is not Unix.

Hierarchical tree models files and directories

- ➔ 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

A local programmable proxy 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*

Why not just a javascript framework?

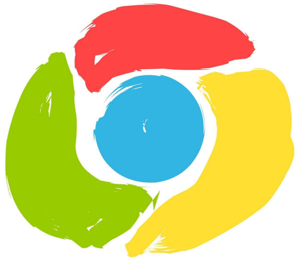
Use standard HTML tags:

```
<img src=“https://lively4/images/icon.gif” />
```

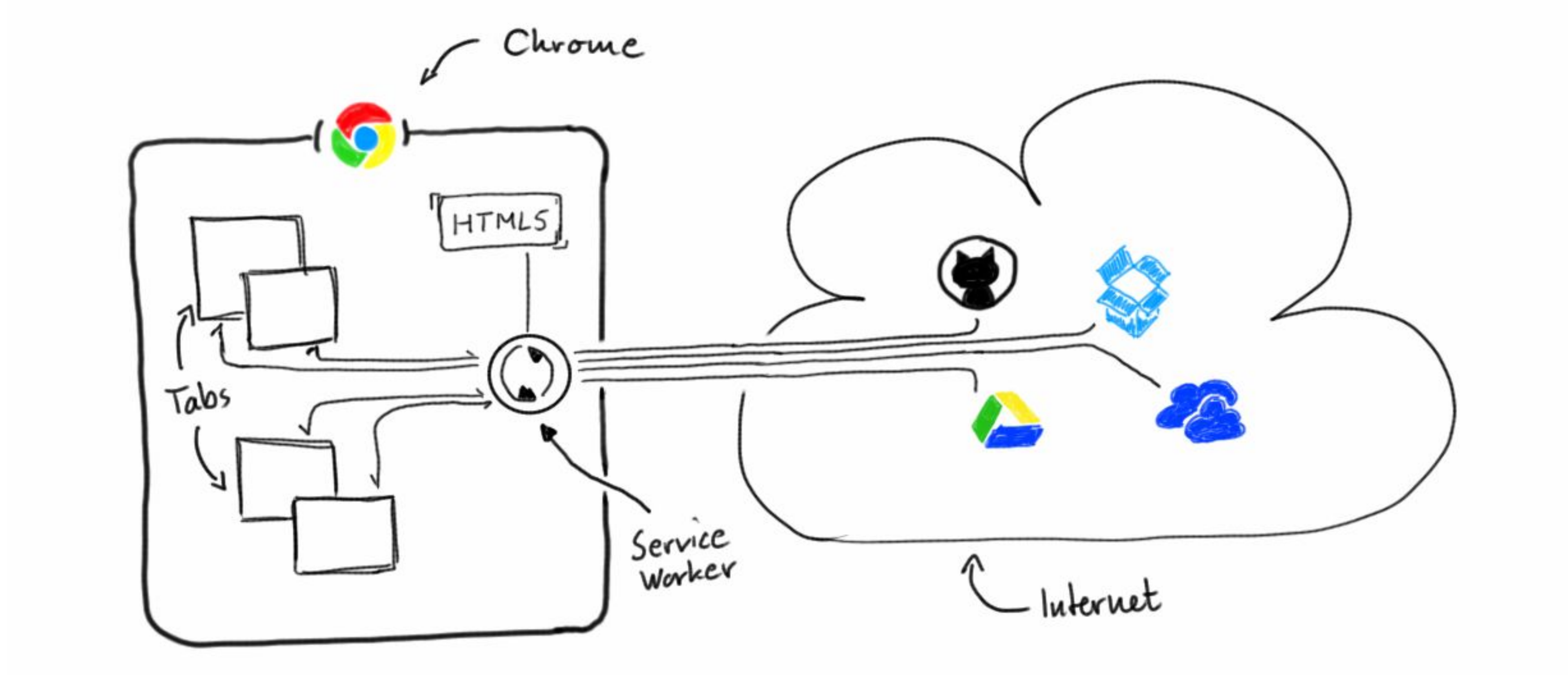
Use vendor code libraries and frameworks:

```
$.ajax(“https://lively4/my/resource.json”)
```

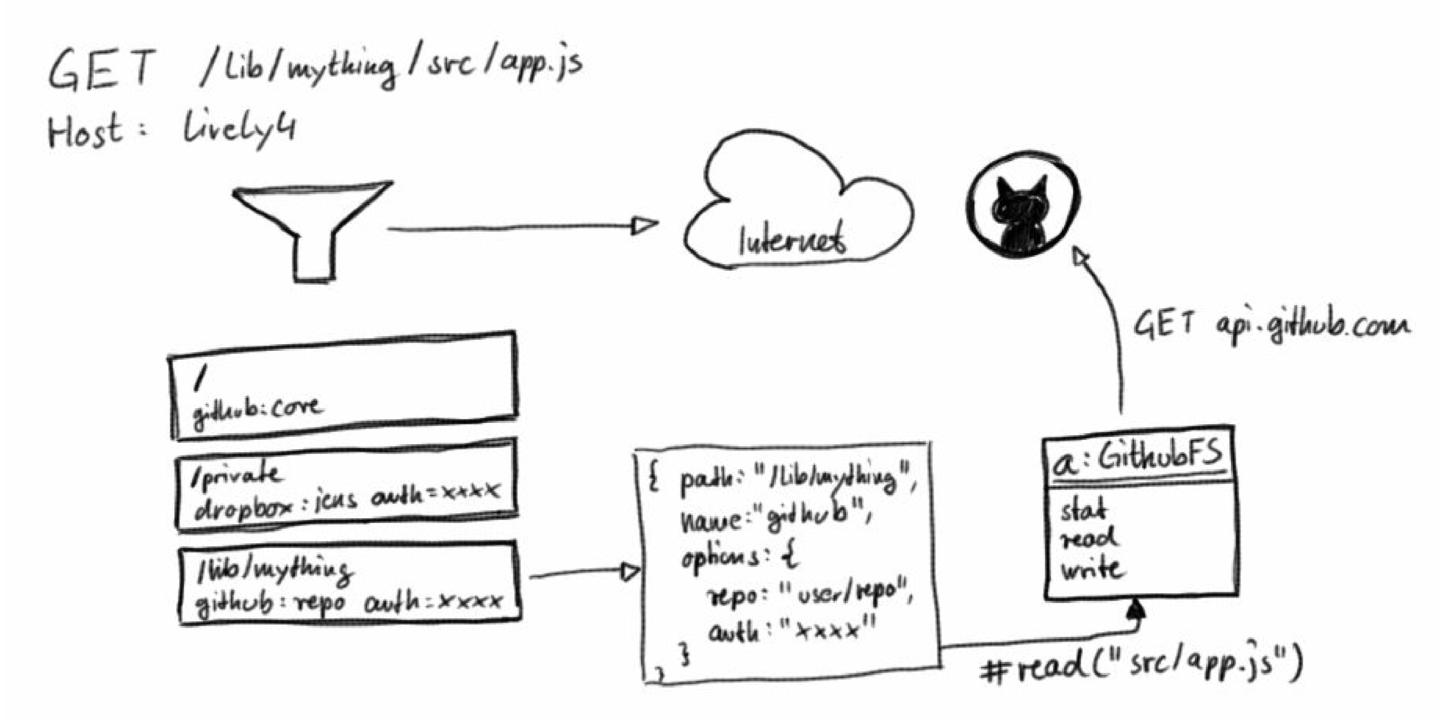
Embrace newest web technologies :3



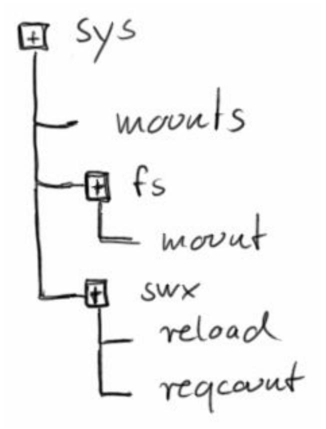
Concept



Service Worker Internals



Controlling Service Worker

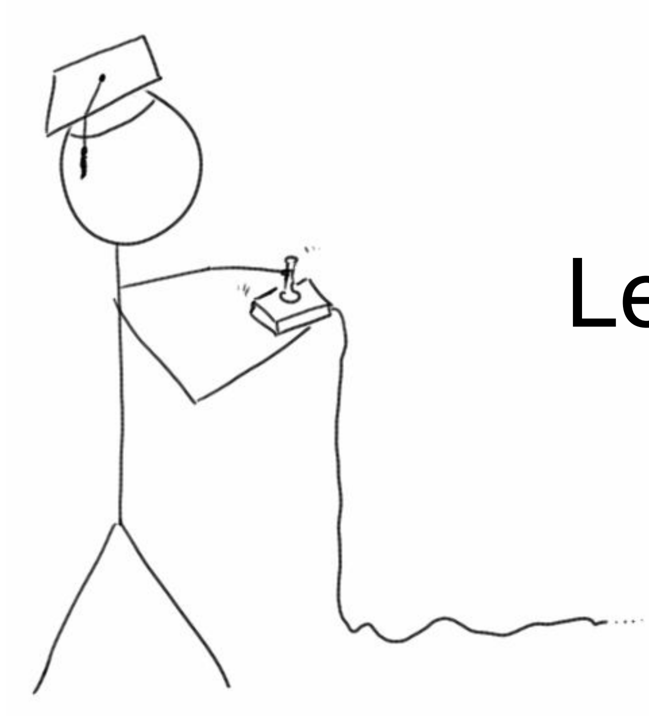


Control by reading and writing to files

- ❑ **GET** `https://lively4`**`/sys/mounts`**
  Returns list of all mounted file systems and their mount options

- ❑ **PUT** `https://lively4`**`/sys/fs/mount`**
  Mounts a new file system using given JSON content

- Read meta data about service worker
  - e.g. request count
- Invoke actions etc.
  - e.g. reload



Let's Play

Discussion

- Concept Advantages
  - Well-known (REST API and Unix FS semantics)
  - Extendable with new filesystems
  - Client library independent
- Technology Advantages
  - Modern browser APIs
  - Client library independent

- Serviceworker Limitations
  - Can always be suspended by browser
    - ⇨ Program state lost
  - Can only use real async browser APIs
  - Can't use tab-related APIs
    - ⇨ No HTML5 file system API
- Filesystem Limitations
  - Can only write single file per request
  - Can't delete files or folders
  - Can't create folders

Future Work

- Filesystems for more backends (e.g. Google Drive, Owncloud, …)
- Collect multiple changes and write as bulk
- Self-contained pre-compiled minimal kernel for distribution
- Reload kernel “modules” from mounted userland file system
- Add FS API for delete and mkdir
- Merge mount points with directory content
- VT100, sh.js, exec call convention (env.fd[], argv)

Conclusion

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: XKCD

Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

lively4

Unified Data Backend

Meike Baumgärtner & Jan Graichen

Software Design 2015/2016
Software Architecture Group
Supervisors Jens Lincke and Stefan Lehmann