
"Hello? No... I'm sorry...
I'm offline at the moment...!"

www.puremedia-online.de

- Internet-only company in Stuttgart
  - AJAX / JavaScript
  - Ruby on Rails
  - TYPO3 incl. extension development

puremedia
Reinsburgstr. 37
70178 Stuttgart

t 0700 / 0078 0079
m 0179 / 90 16 016
f 0700 / 0078 0079

info@puremedia-online.de
www.puremedia-online.de

www.puremedia-online.de

- Internet-only company in Stuttgart
  - AJAX / JavaScript
  - Ruby on Rails
  - TYPO3 incl. extension development

Customers include...

Contents

- Offline??? Browser based Applications!
- Use Cases / Examples
- Offline Tools
- Google Gears
  - Architecture
  - Components
- Dojo Offline

- Comet

"User interface and usability
in the browser are bad."

Offline??? Browser based Applications!

AJAX.

"You can only really use the browser when you're connected to the internet."

Offline??? Browser based Applications!

Offline capable Web Applications.

Offline??? Browser based Applications!

Offline closes one of the last gaps between web and desktop apps!

Use Cases / Examples

Use Cases

- Internet connection is...
  - spotty / unstable / slow
  - not existing

- on the road
  - airplane / train / bus
  - at the customer's

Use Cases / Examples

Examples

- Offline News Reader [Google Reader]
- Notepad [GearPad]
- Task list [Remember The Milk]
- Email Client
- CRM
- Projectmanagement-Tools
- Calendar
- .....

Offline Tools

Tools [in the past] - (Offline is not brand new)

- Dojo Storage
- Derby / JavaDB
- Zimbra (Derby-based)

Offline Tools

Tools [today]

- Adobe AIR (früher: Apollo)
- Joyent Slingshot (Ruby on Rails)
- Google Gears

Offline Tools

Tools [coming soon]

- Firefox 3
- Webkit (Safari)
- HTML5 Working Draft of the WHATWG ("Client-side database storage")

Google Gears - Basics

What is Google Gears?

- Browser-Plugin
  - Firefox
  - Internet Explorer
  - maybe soon: Opera and Safari

Google Gears - Basics

What does it cost? Under which license?

- License
  - OpenSource (New BSD license)

Google Gears - Differences

What distinguishes Gears from other offline tools?

- App stays completely in the browser (no extra app / start up)
- Google's AJAX background -- developers use well-known technologies (JS / CSS / HTML), extended by a few now JS-API-calls
- lightweight / hardly no installation necessary
- open framework
- "offline only" (no goodies)
- ...it's from Google!

Google Gears - Components

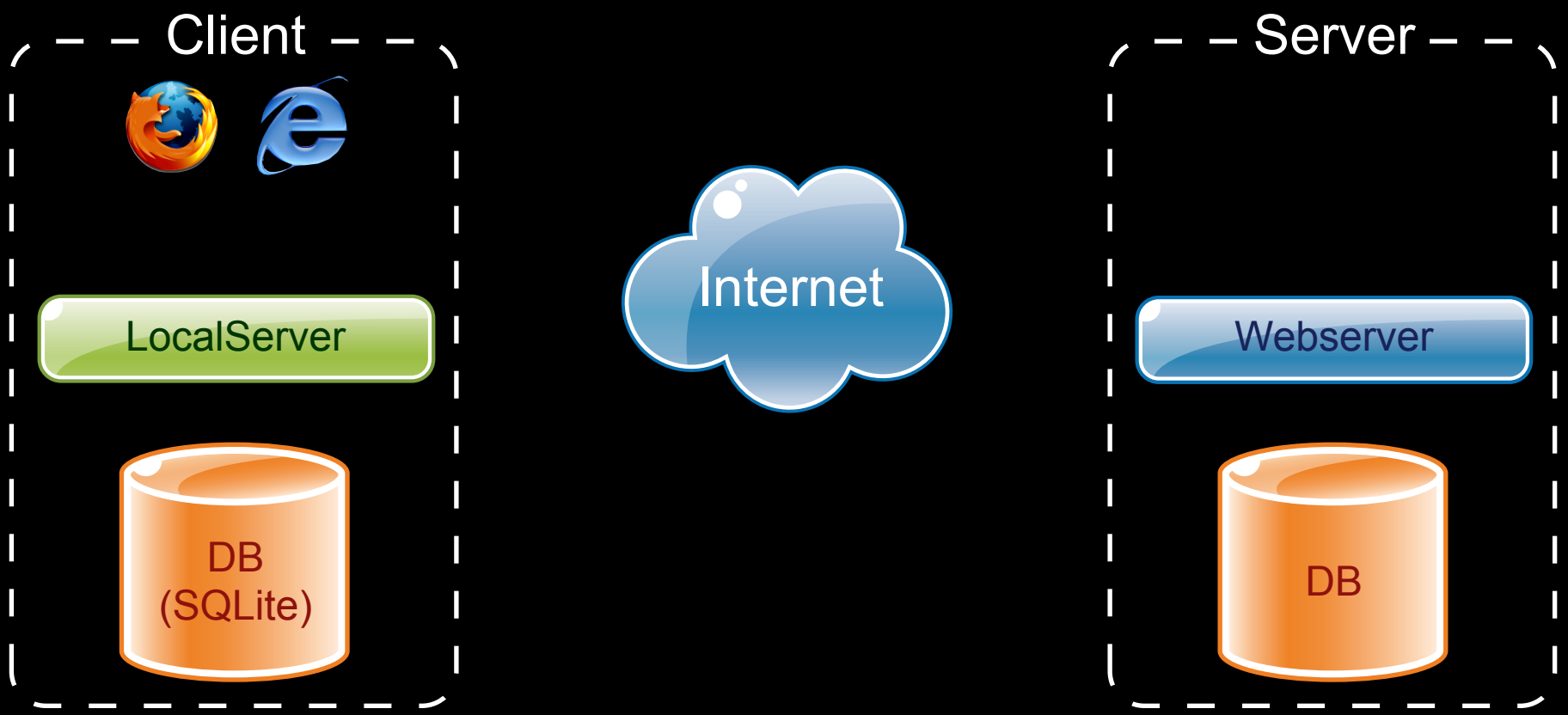
LocalServer
Local URL- / resource-cache

Database
Local relational DB (SQLite)

Workerpool
puts time-intensive processes in the background

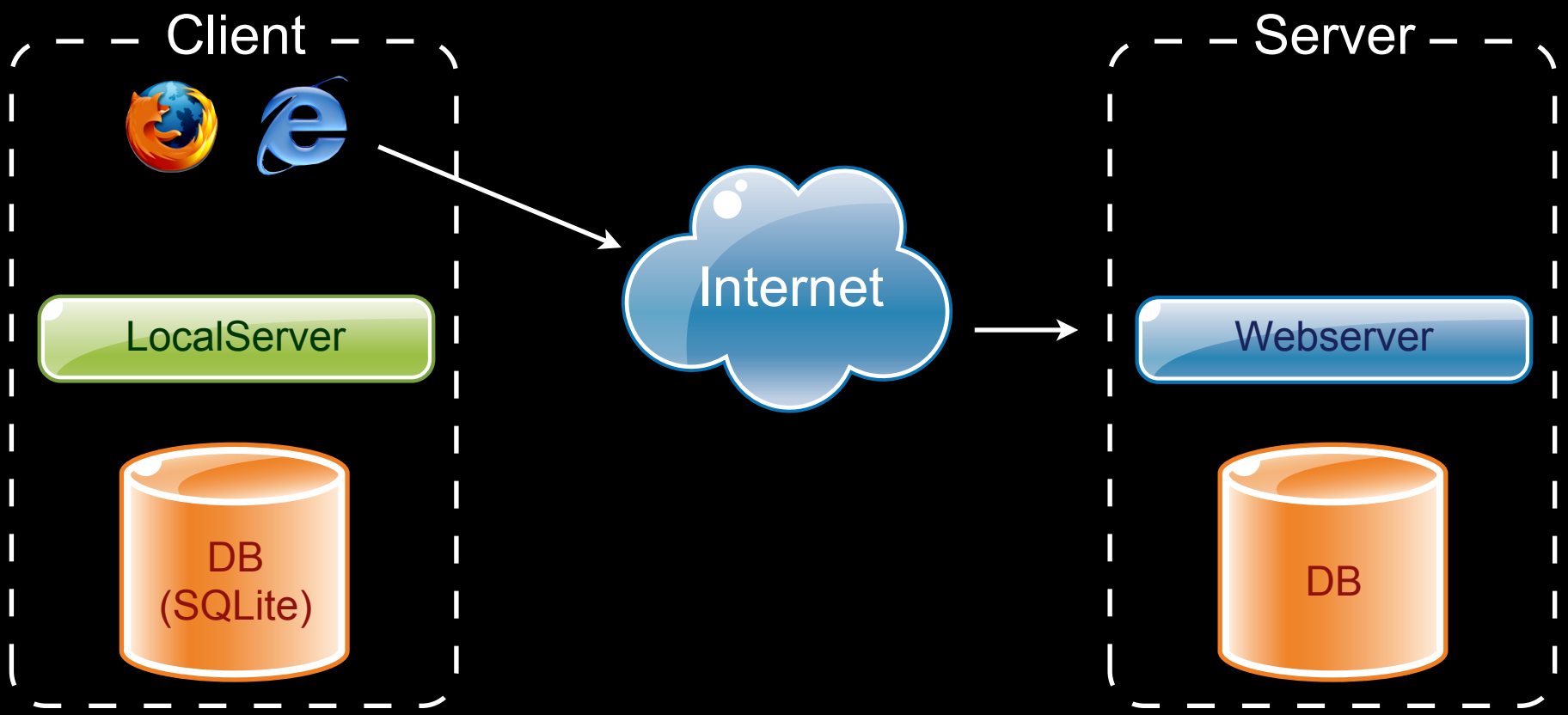
Google Gears

Flow in ONLINE mode



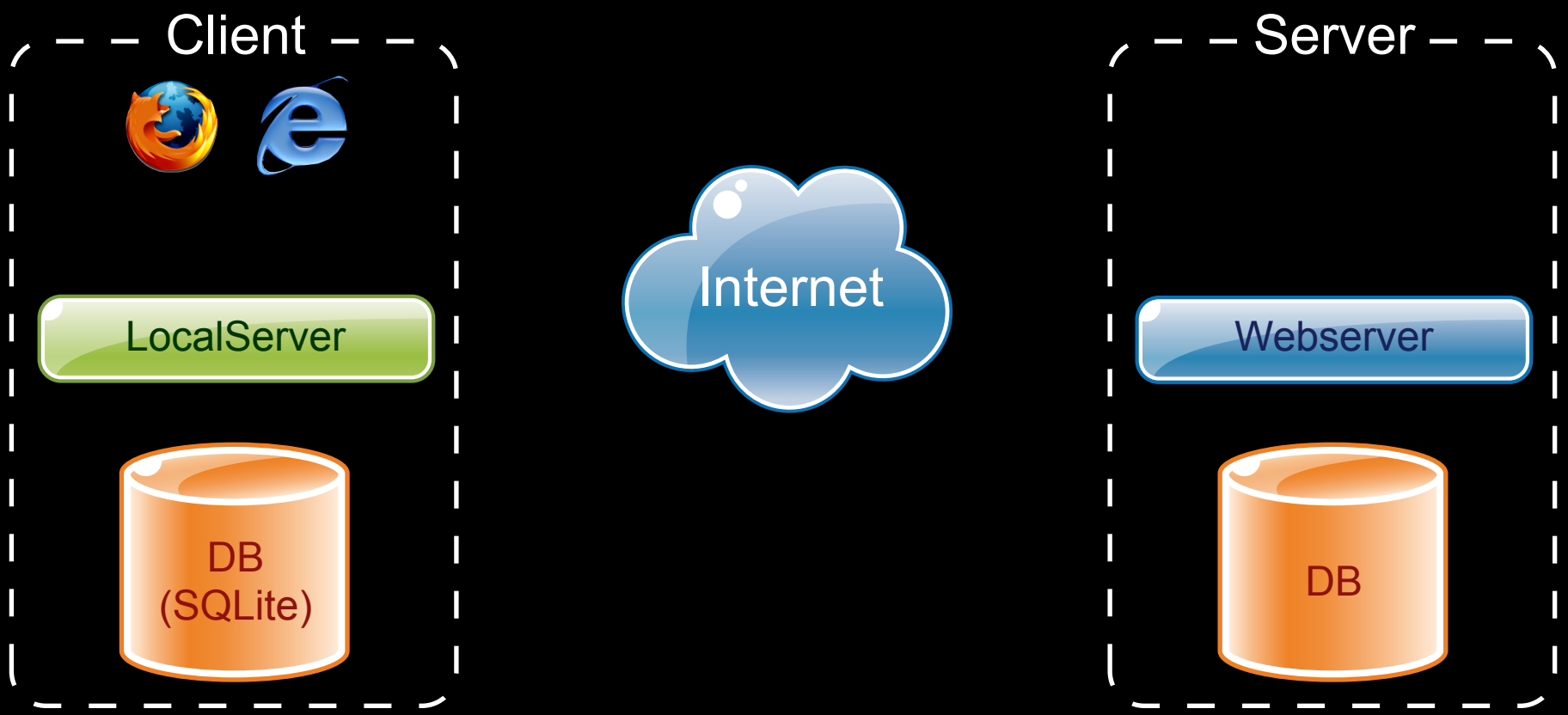
Google Gears

Flow in ONLINE mode



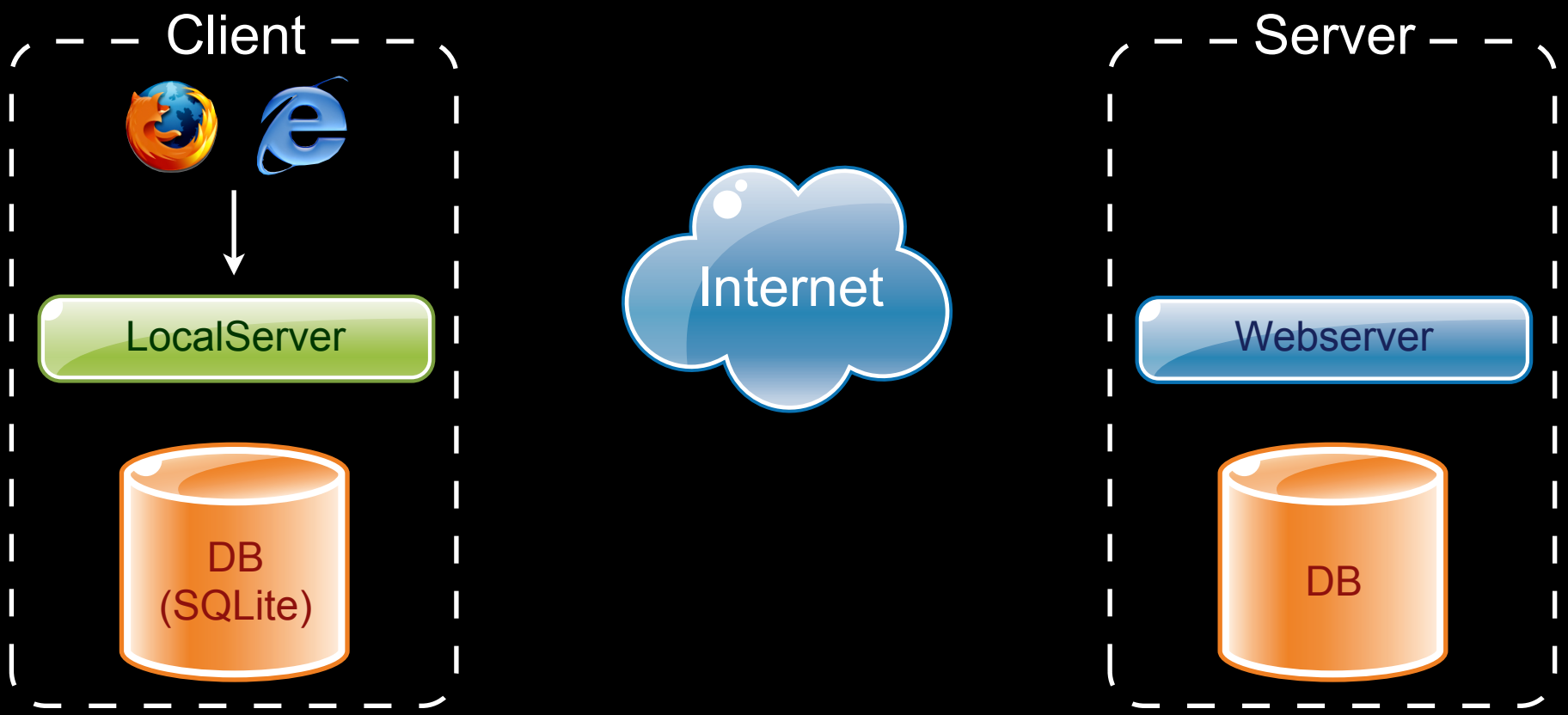
Google Gears

Flow in OFFLINE mode



Google Gears

Flow in OFFLINE mode



Google Gears - Local Server

Local Server

- Intercepts requests to previously defined resources (CSS, HTML, JS...)...
- ...and serves them from the local cache ("Resource Store")...
- ...regardless of offline/online mode.

- Resources are kept in a "Resource Store" ("URL-Container")
  - Resource Store (manual update)
  - Managed Resource Store (automatic update)

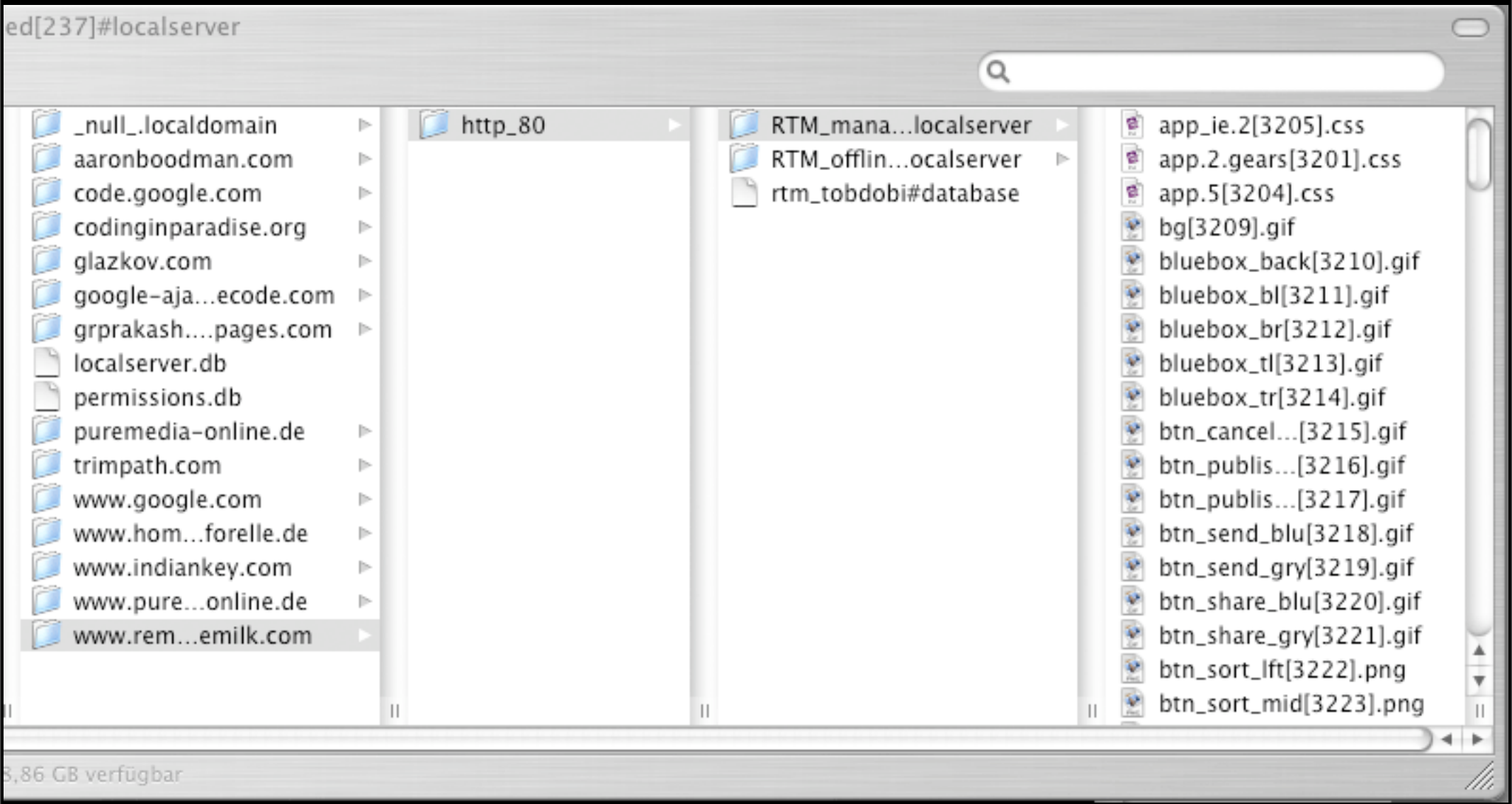
Google Gears - Database

SQLite

- complete relational DB (Transactions, Primary keys, Views, Full-text search...)
- multiple DBs per App possible
- sandbox / same origin policy
  - ATTACH / DETACH not available
- bind parameter
  - execute() with bind parameters protects from SQL Injections

Google Gears

Local data storage



Google Gears - Workerpool

JavaScript in the background

- time-intensive processes can be run in the background
  - UI stays responsive / browser doesn't block
  - no warning dialog "Unresponsive Script"
- Restrictions:
  - no access to the DOM
  - no access to XMLHttpRequest (see Gears 0.2)

- Use case: Synchronization

Google Gears - Security

Security in Gears

- same-origin policy
  - access only your own DB
  - LocalServer can only cache resources in its own origin
- User opt-in before Gears can run
- Data is stored in user profile of OS
- Bind parameter protects from SQL-Injection attacks

Google Gears - Architecture

Architecture - modal

- online? Communication with the server
- offline? Communication with local Gears components

- Pros
  - easy to implement

- Cons
  - User must explicitly and manually change modes (Button "Go offline")
  - If user forgets to click "take me offline"...

Google Gears - Architecture

Architecture - modeless

- online? Almost only for syncing (continuously)
- offline? Always being assumed

- Pros
  - User doesn't have to switch modes by hand
  - Data is always available
  - Performance is better

- Cons
  - more difficult to implement...
  - ...mostly because of sync in background

"How do I know if the user
is currently online or offline?"

"How can I do easy synchronization without reinventing the wheel?"

Dojo Offline

JavaScript toolbox for Gears

- "Offline Widget" (on/offline detection)
- slurp() finds used resources
- Data storage +
  - encryption
  - alternative storage
- Sync framework

Dojo Offline

Data storage

- Dojo SQL
  - sits on top of Gears' SQLite-implementation
  - pro: query-results directly available als JS-objects!
  - encryption

- Dojo Storage
  - simple Hash-Table (key/value-pairs)
  - pros: quick & easy

Dojo Offline

Synchronization

- automatic syncing when...
  - ...app starts / reloads
  - ...when going back online
- manual sync possible

Offline

Resume

- Offline closes one of the last gaps between web and desktop apps
- Use cases are limited - but in some cases very interesting
- Offline capability is not a byproduct

C: "What's the news?" - S: "Nothing."
C: "What's the news?" - S: "Nothing!"
C: "What's the news?" - S: "Nothing!!!"
C: "What's the news?" - S: "NOTHING!!!"
C: "What's the news?" - S: "Data. Listen!"

Comet

Polling

- Using polling in AJAX means:
  - "The client (JavaScript / AJAX) asks...
  - the server...
  - in certain intervals...
  - if there's new data."

Comet

The problem with polling

a) intervals are too short
- most of the request are useless
- wasted bandwidth / high server load

b) intervals are too long
- UI gets out of date
- displayed data is out-dated

Comet

The bottom line

- why not let the server inform the client [PUSH]
  instead of
- having the client permanently ask [PULL]

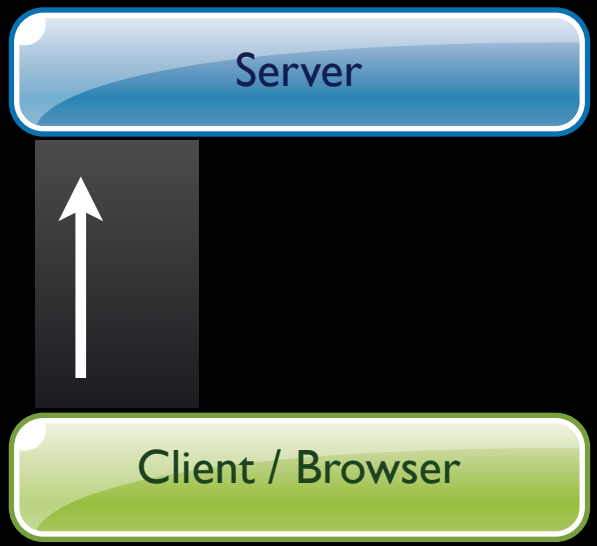
Comet

How does a traditional request work? [simplified...]

Comet

How does a traditional request work?

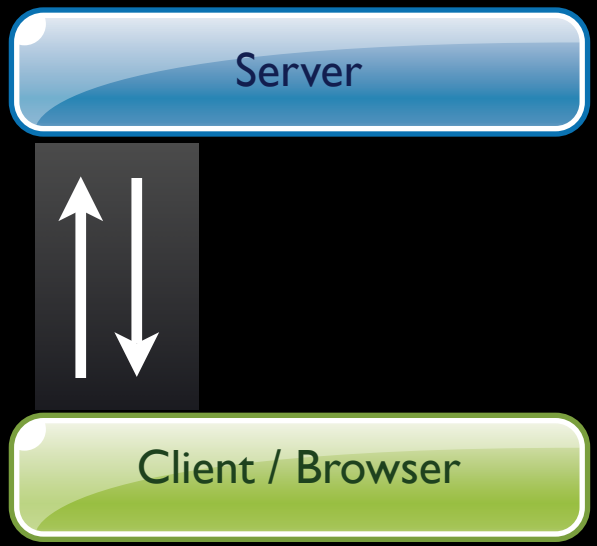
1. Client makes a request to the server. A connection is opened.



Comet

How does a traditional request work?

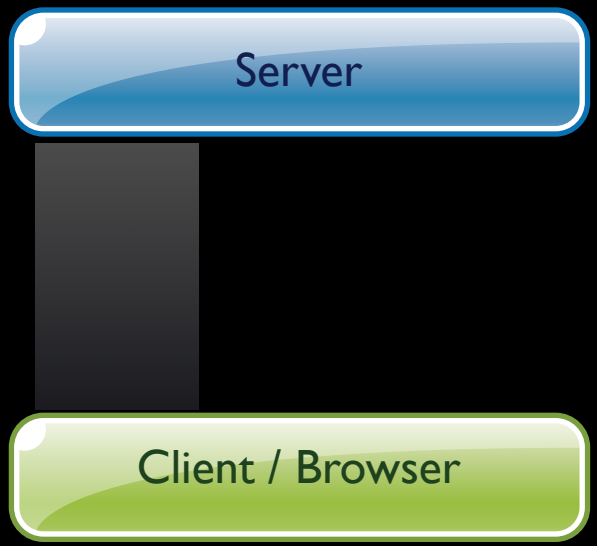
1. Client makes a request to the server. A connection is opened.
2. Server answers.



Comet

How does a traditional request work?

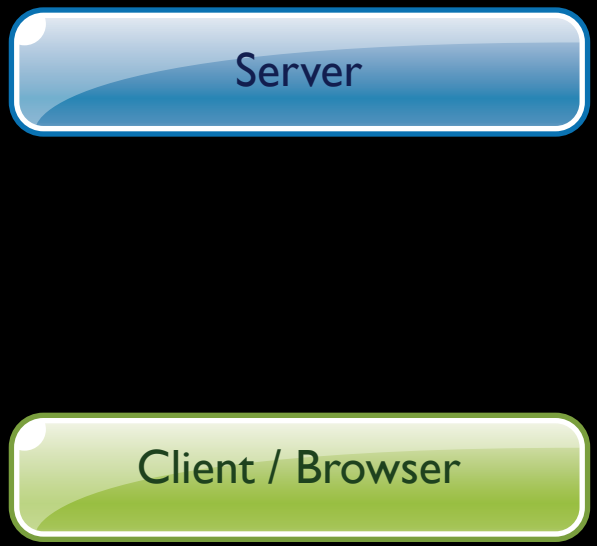
1. Client makes a request to the server. A connection is opened.
2. Server answers.
3. Connection is closed.



Comet

How does a traditional request work?

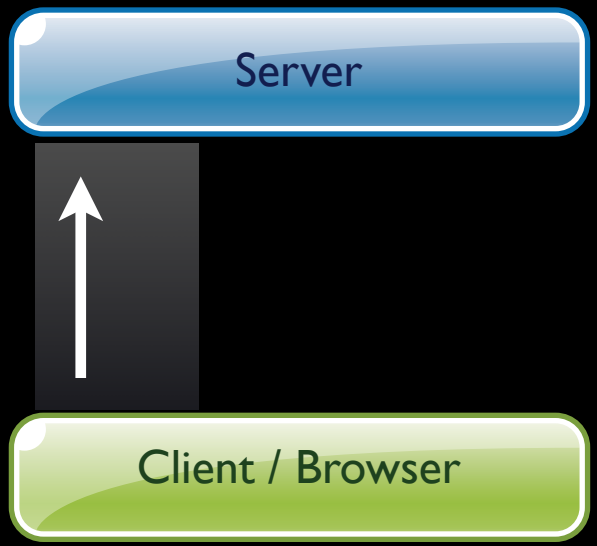
1. Client makes a request to the server. A connection is opened.
2. Server answers.
3. Connection is closed.



Comet

How does a Comet request work?

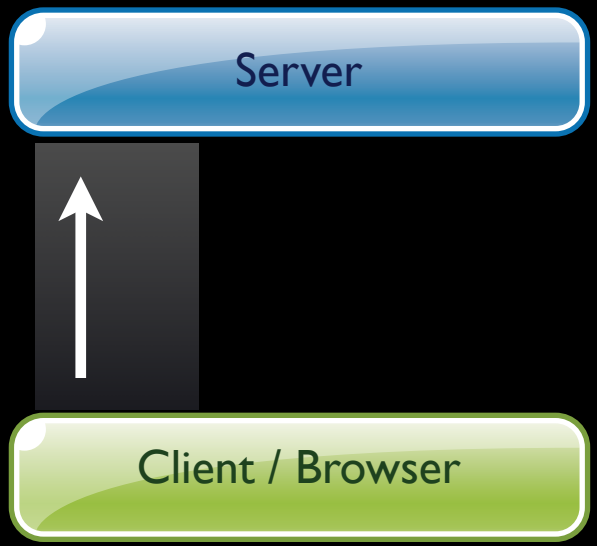
1. Client makes a request to the server. A connection is opened.



Comet

How does a Comet request work?

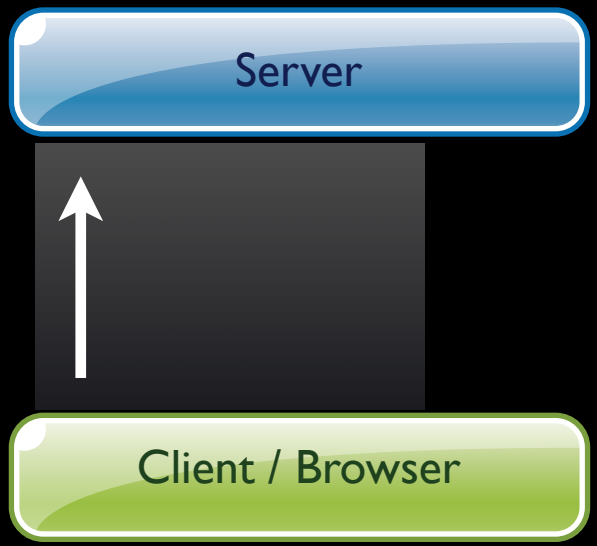
1. Client makes a request to the server. A connection is opened.
2. Maybe there's no new data yet, but the connection is kept open!



Comet

How does a Comet request work?

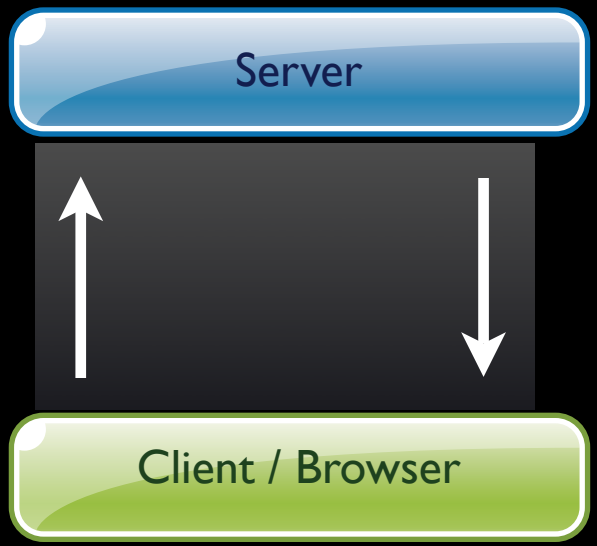
1. Client makes a request to the server. A connection is opened.
2. Maybe there's no new data yet, but the connection is kept open!
3. The connection is kept open...



Comet

How does a Comet request work?

1. Client makes a request to the server. A connection is opened.
2. Maybe there's no new data yet, but the connection is kept open!
3. The connection is kept open...
4. News on the server! The server answers! The connection is closed or is even kept open!



Comet

Central idea with Comet

- "long-lived HTTP-Connections"

Comet

Comet

- PUSH instead of PULL
- Low Latency
- Low Bandwidth
- UI not getting out-dated / always up-to-date


"Is Comet »Web 3.0«??"

Comet

1. Problems with Comet

- one thread on the server per connection...
  long-lived connection = many threads
  many threads = many resources (limited number of threads!)
  no more resources / threads allowed = no more new connections
- Firewalls: close connections that are open for "too long"
- Proxies: can buffer pushed data - and therefore deliver late
- Browser: one browser can only open 2 connections to the same server

Comet

1. Problems with Comet

Solutions are on the way:

- Jetty Continuations
- Cometd
- different Hostnames

Comet

2. Is Comet necessary???

- For the majority of use cases AJAX / polling is sufficient!

- Use Cases for Comet
  - multi-user / collaborative applications
    - e.g. collaboratively working on a document; chatting
  - applications with real-time data
    - e.g. stock prices, chats

Offline Web Applications with Dojo Offline & Google Gears

puremedia
Reinsburgstr. 37
70178 Stuttgart

t 0700 / 0078 0079
m 0179 / 90 16 016
f 0700 / 0078 0079

info@puremedia-online.de
www.puremedia-online.de