Creating LOCAL-FIRST collaboration software with AUTOMERGE

it's better than the cloud!

MARTIN KLEPPMANN TU MUNICH

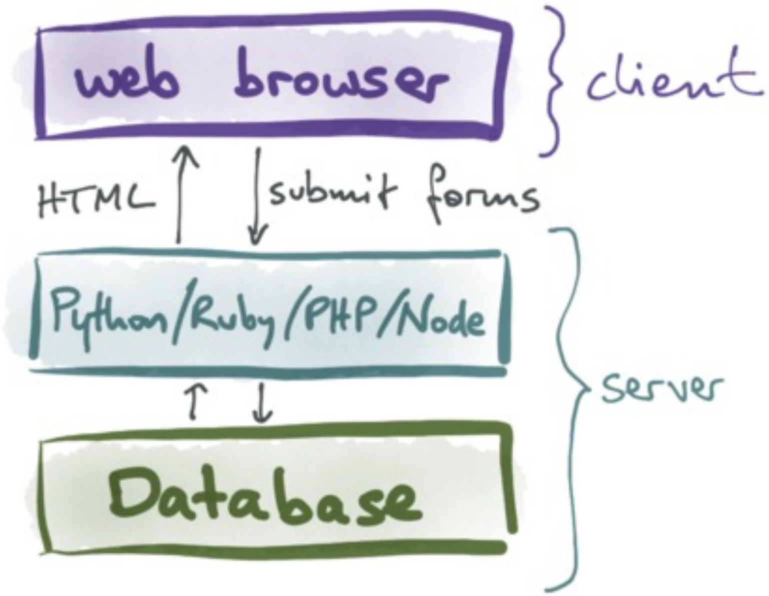
Bluesky: @martinkl.com Twitter: @martinkl

Mastodon: @martin@nondeterministic.computer

TUM VolkswagenStiftung Ink & Switch

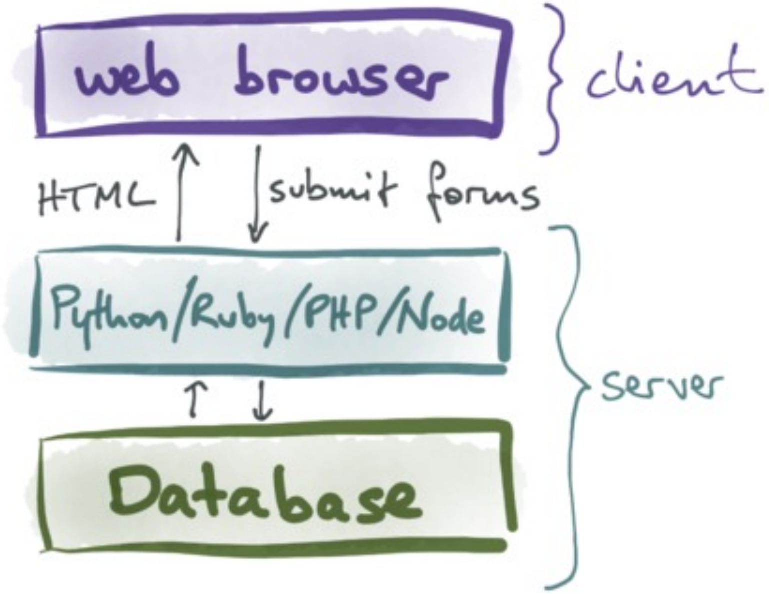
WEB APP ARCHITECTURE THROUGH THE AGES

ca. 2000–2010

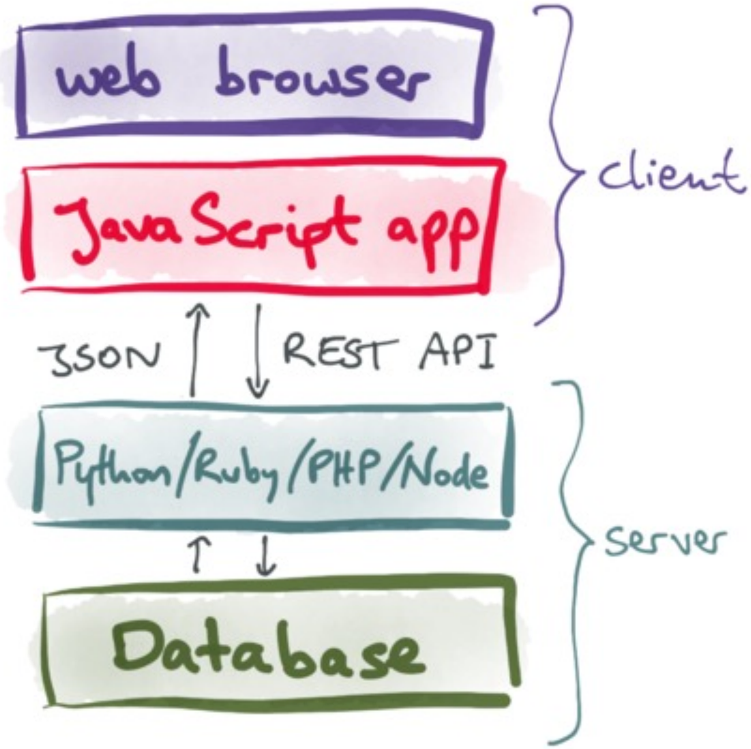


WEB APP ARCHITECTURE THROUGH THE AGES

ca. 2000–2010



ca. 2010–2020

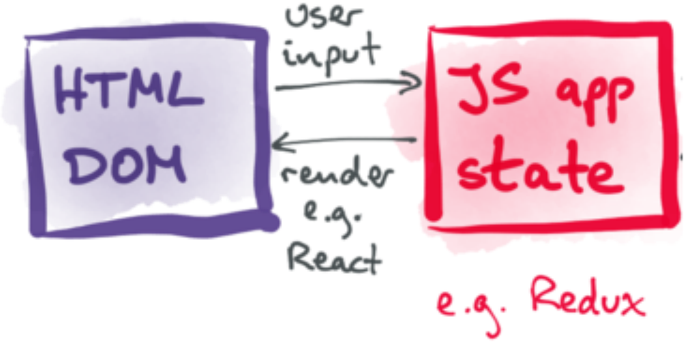


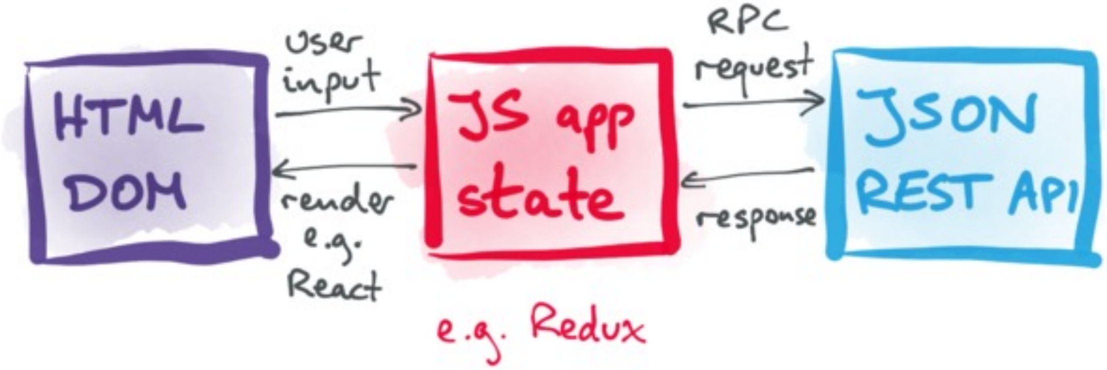
Google Docs

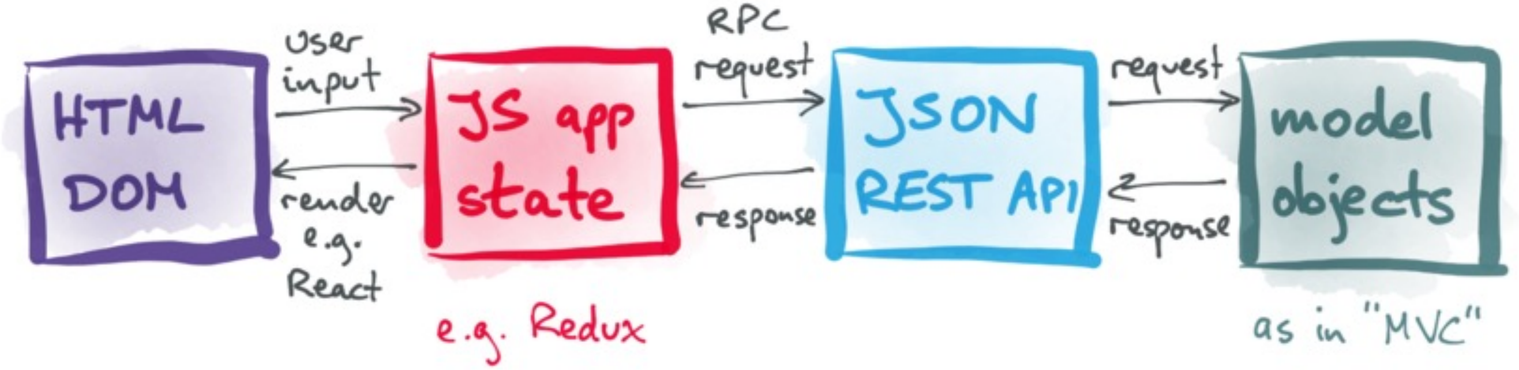
Office 365

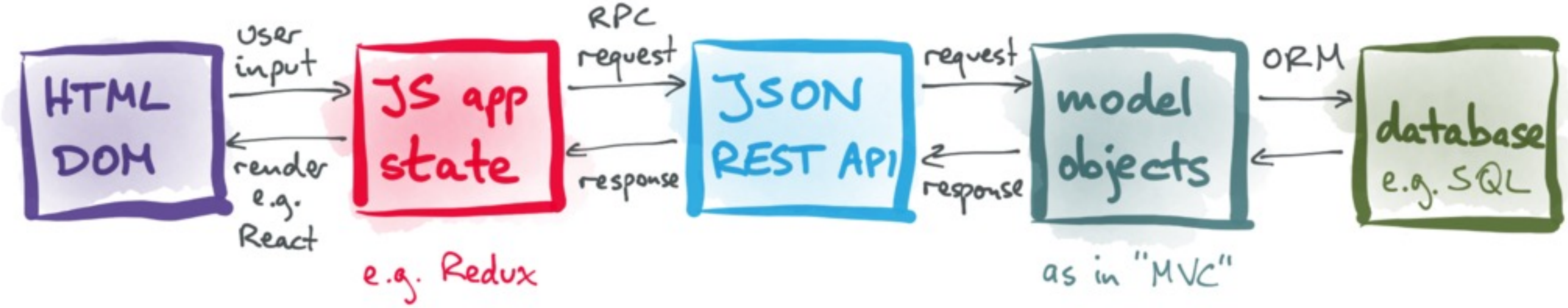
Figma

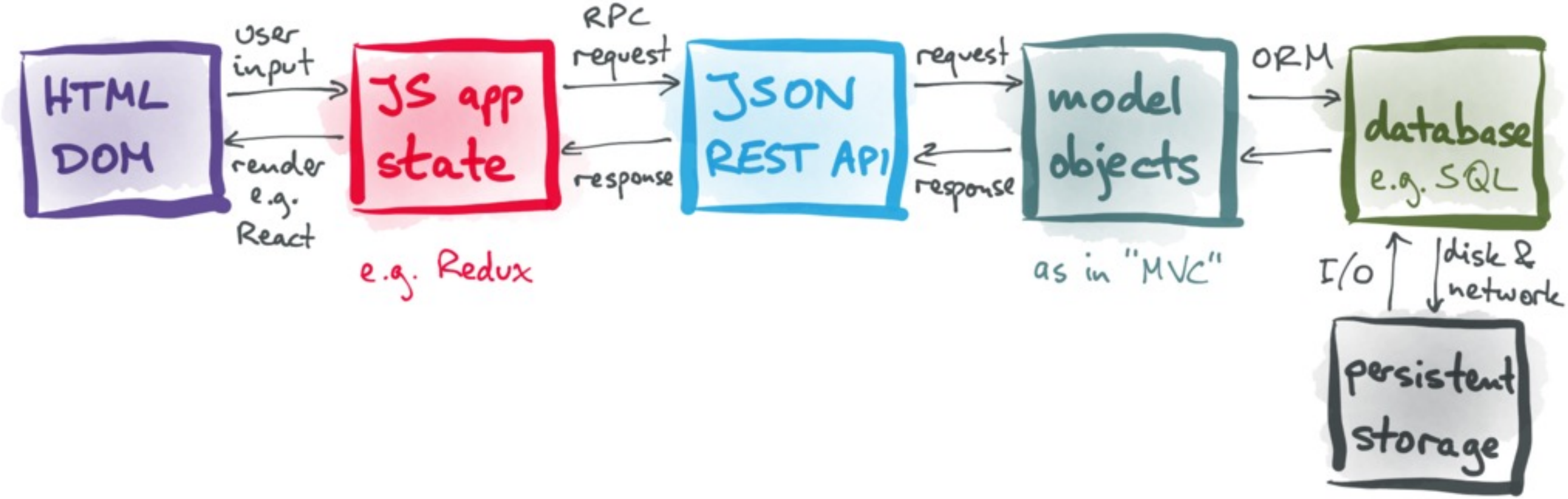
Trello


HTML DOM
user input
render e.g. React
JS app state
e.g. Redux

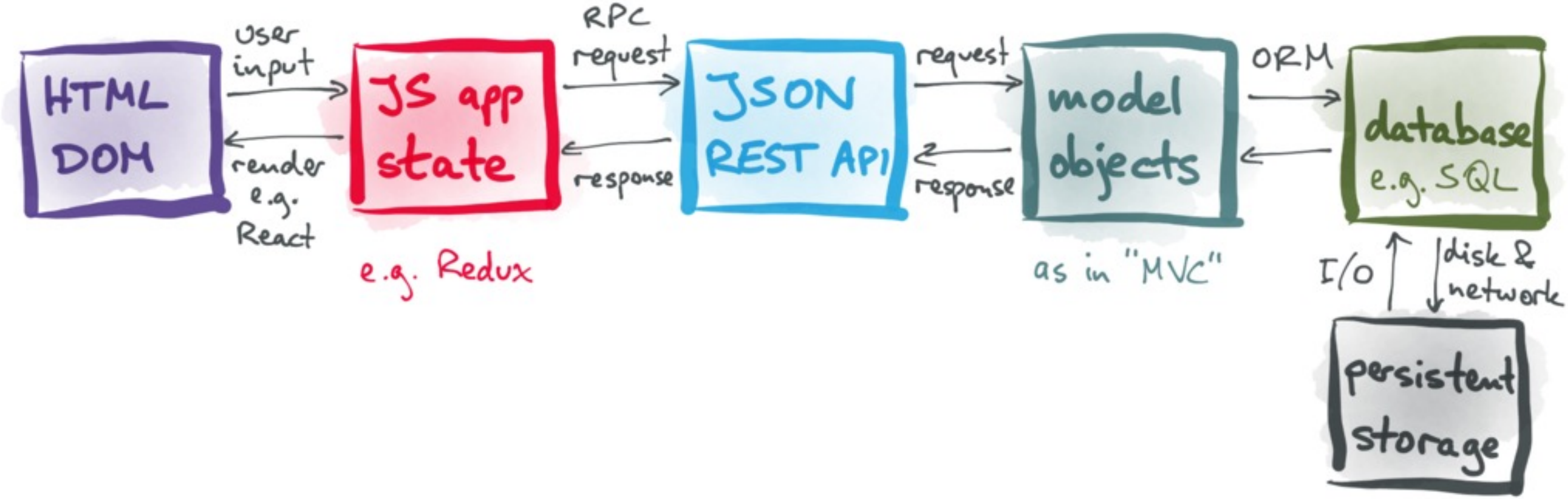

HTML DOM
user input
render e.g. React
JS app state
e.g. Redux
RPC request
response
JSON REST API


HTML DOM
user input
render e.g. React
JS app state
e.g. Redux
RPC request
response
JSON REST API
request
response
model objects
as in "MVC"

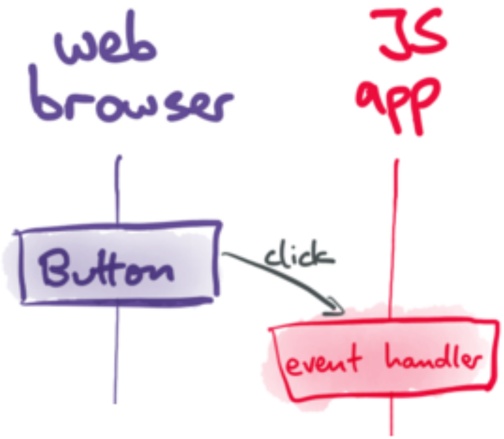

HTML DOM
user input
render e.g. React
JS app state
e.g. Redux
RPC request
response
JSON REST API
request
response
model objects
as in "MVC"
ORM
database
e.g. SQL

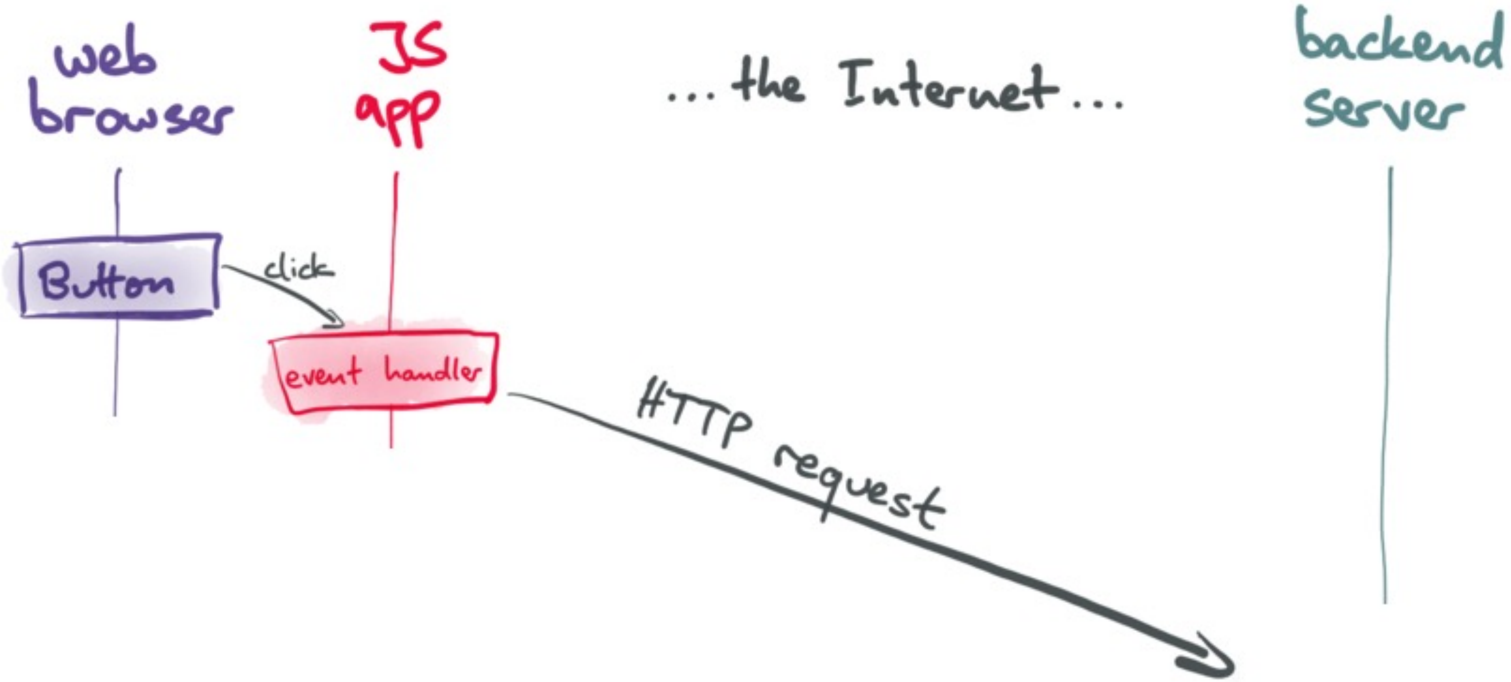

HTML DOM
user input
render e.g. React
JS app state
e.g. Redux
RPC request
response
JSON REST API
request
response
model objects
as in "MVC"
ORM
database
e.g. SQL
I/O
disk & network
persistent storage

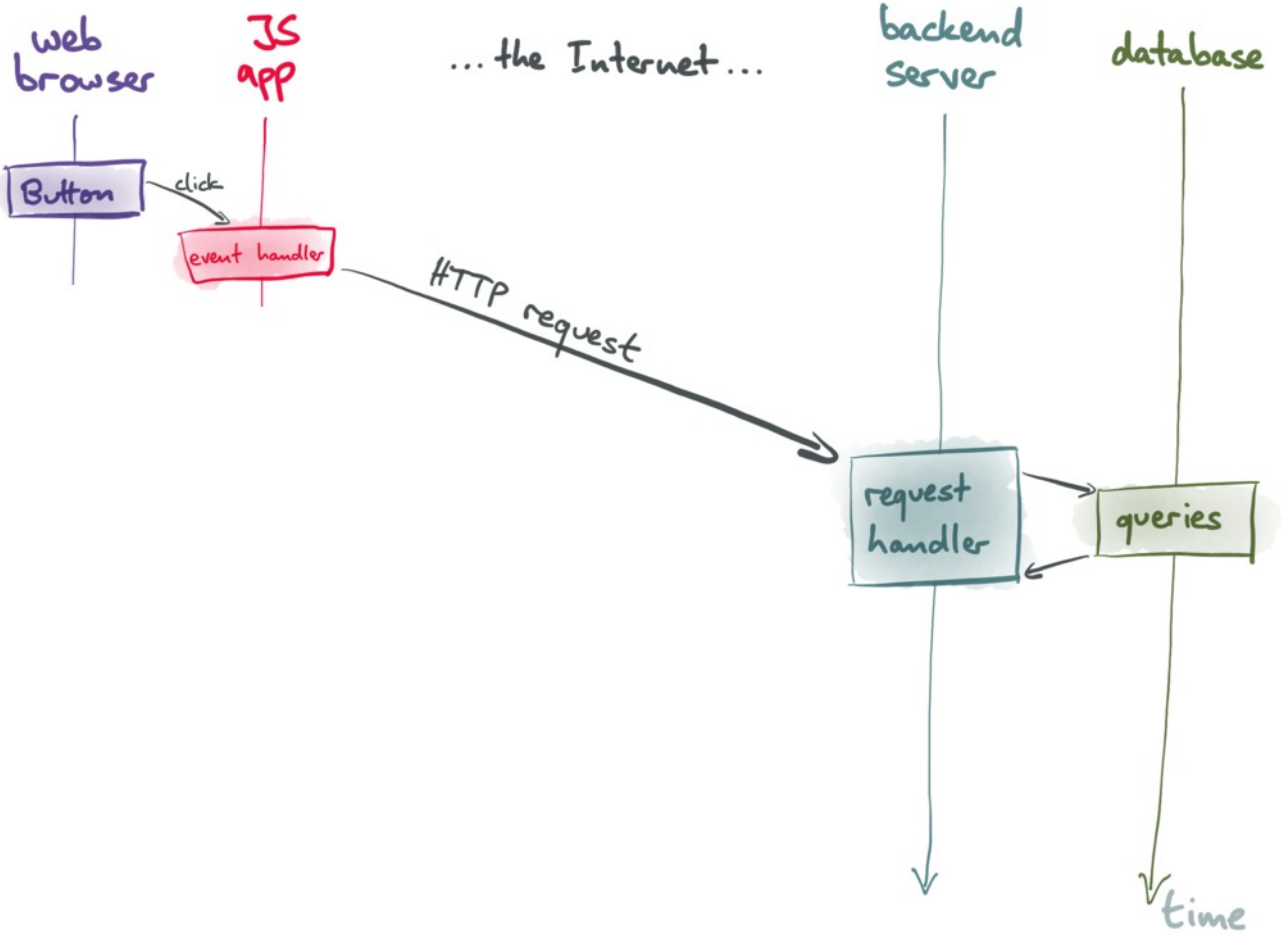
Six different representations of app state?!

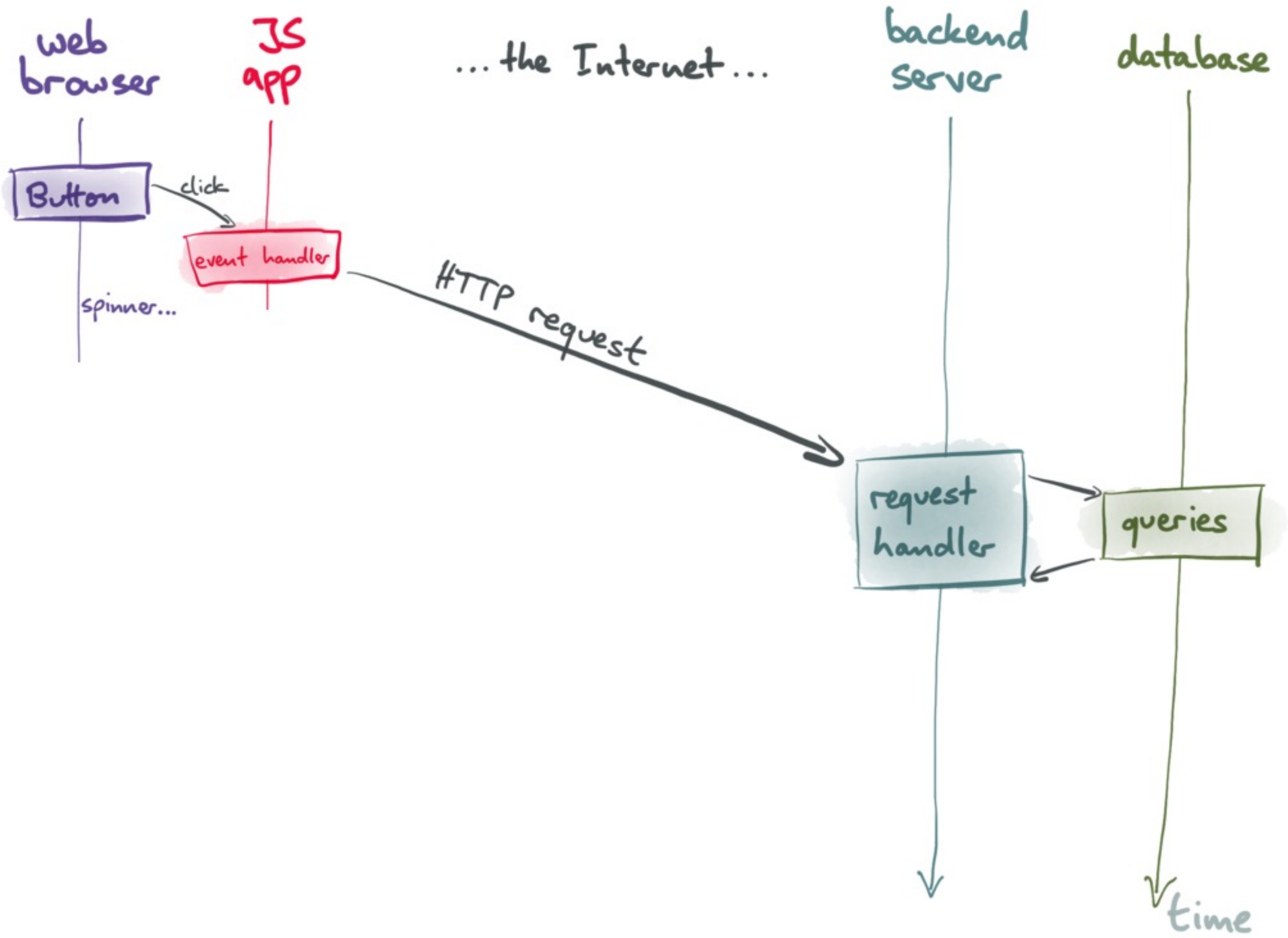


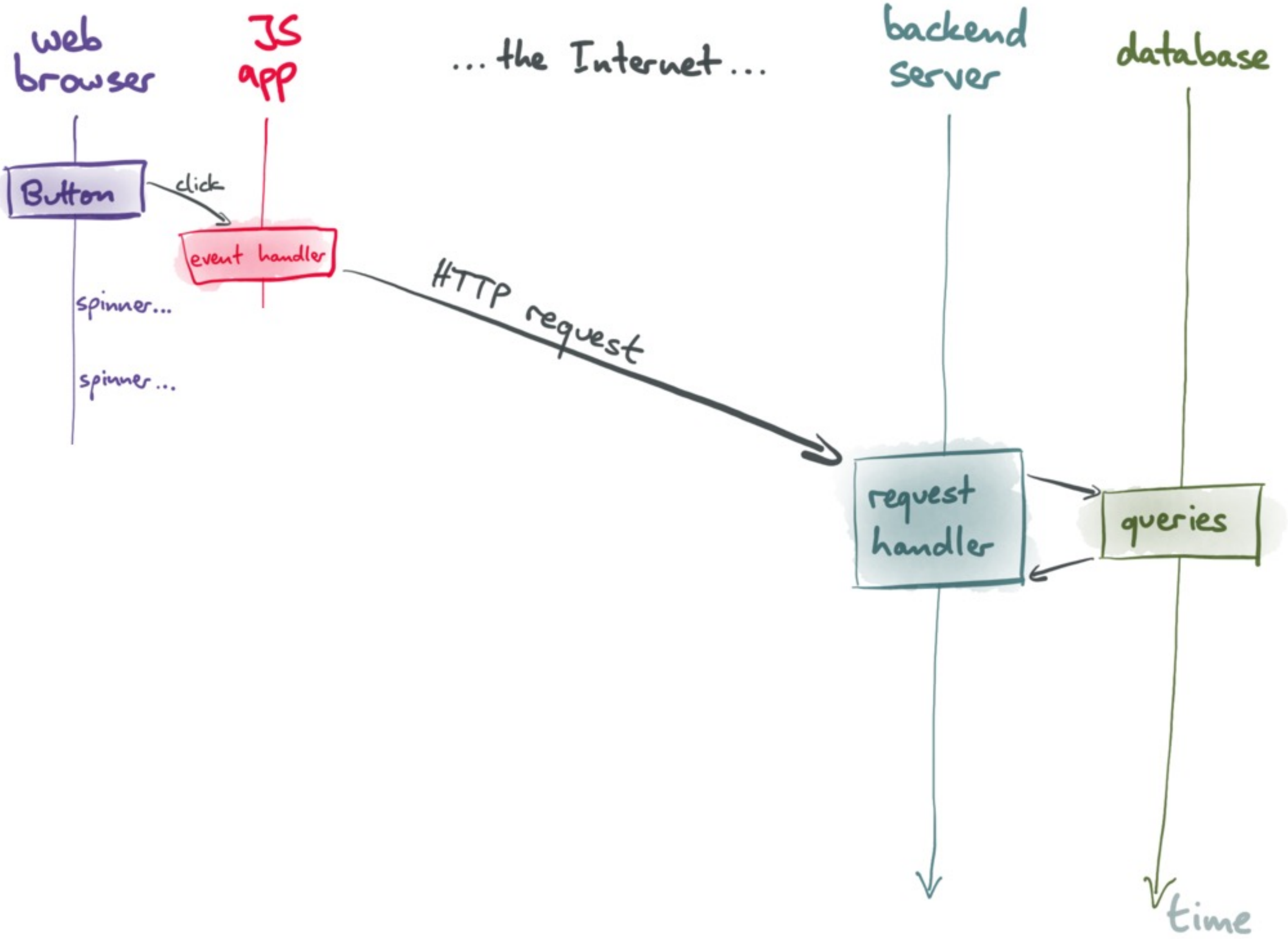
Lots of code is just converting data from one representation to another.

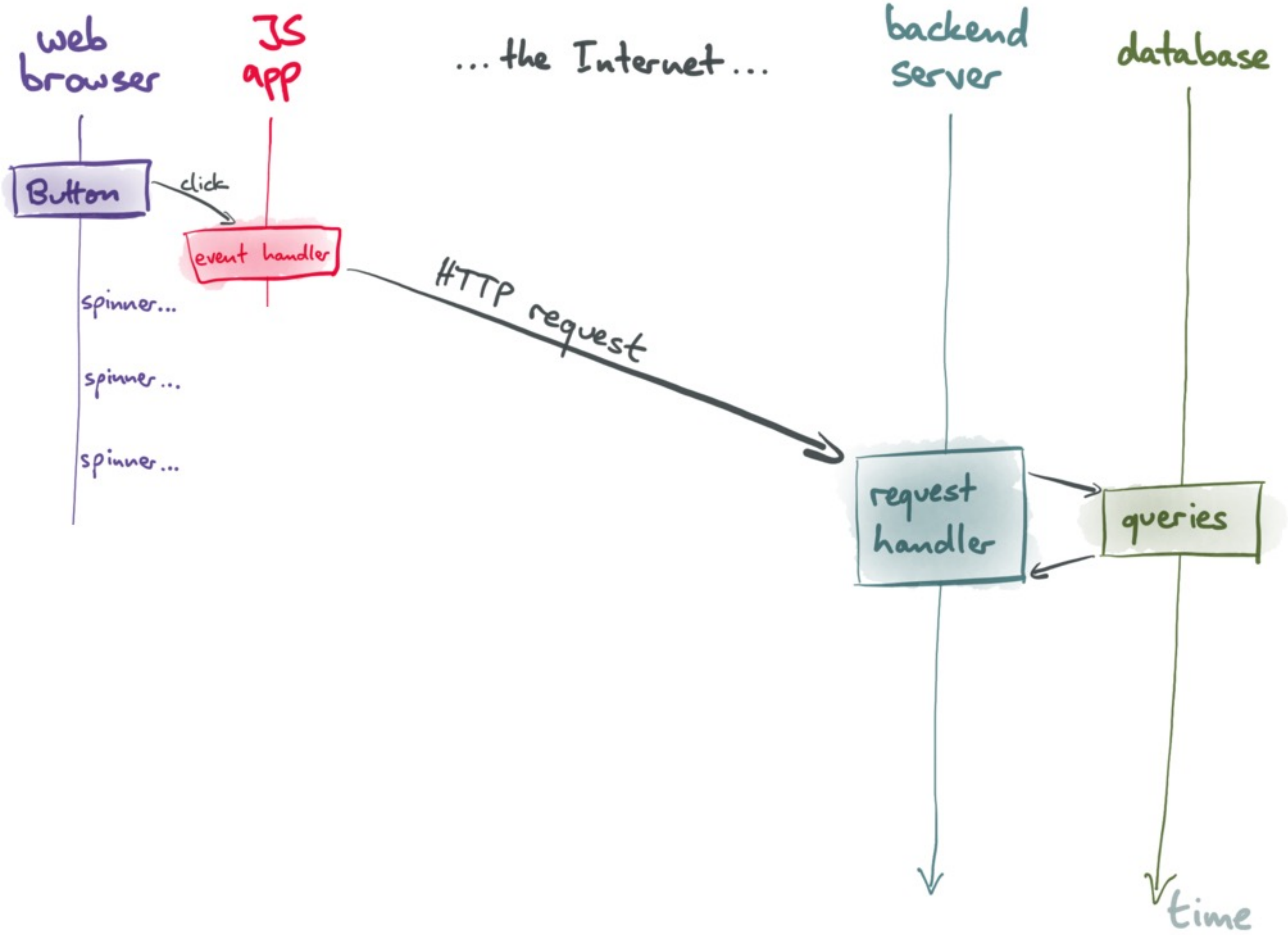

web browser
JS app
Button
click
event handler

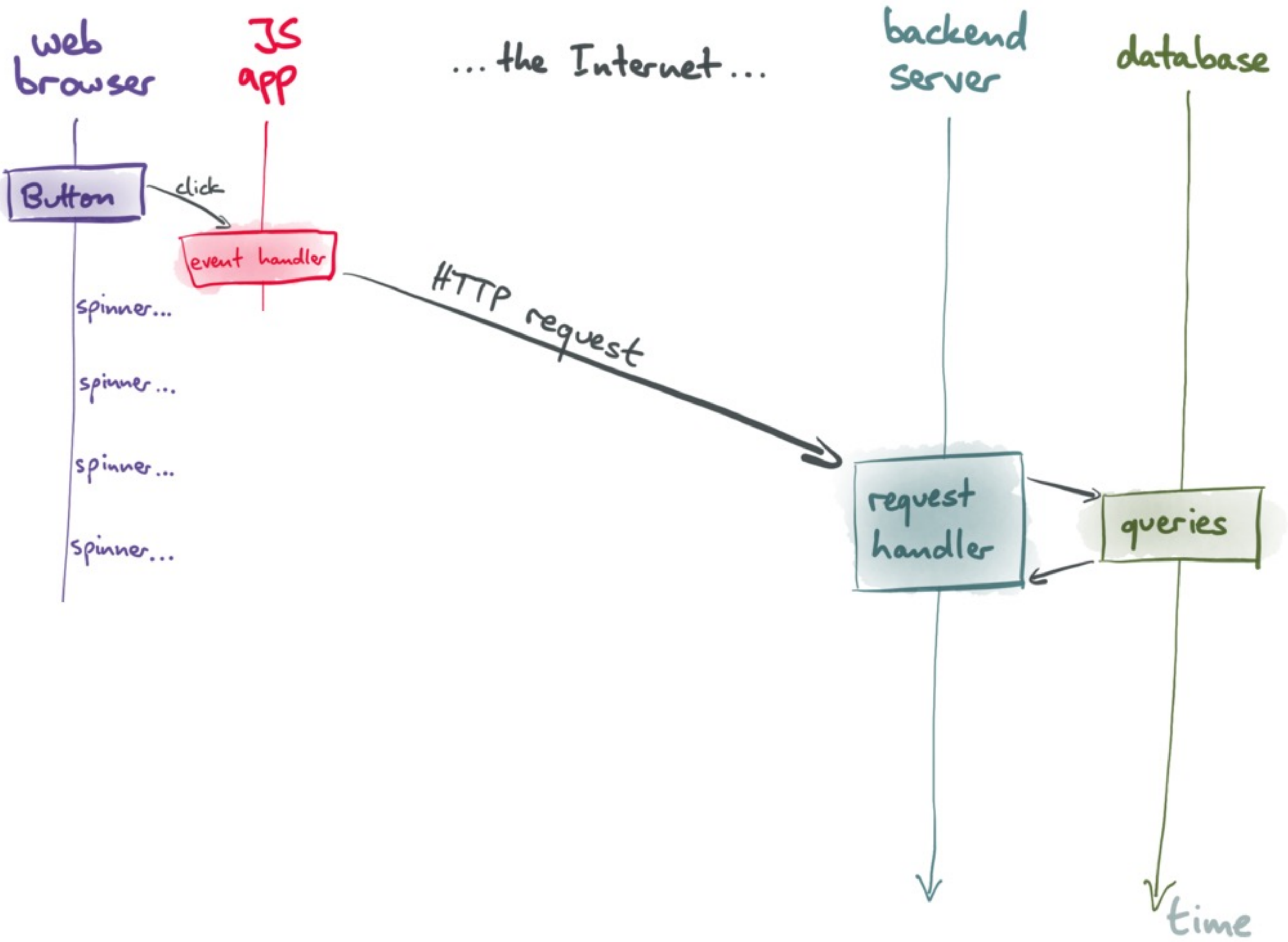

web browser
JS app
... the Internet ...
backend server
Button
click
event handler
HTTP request

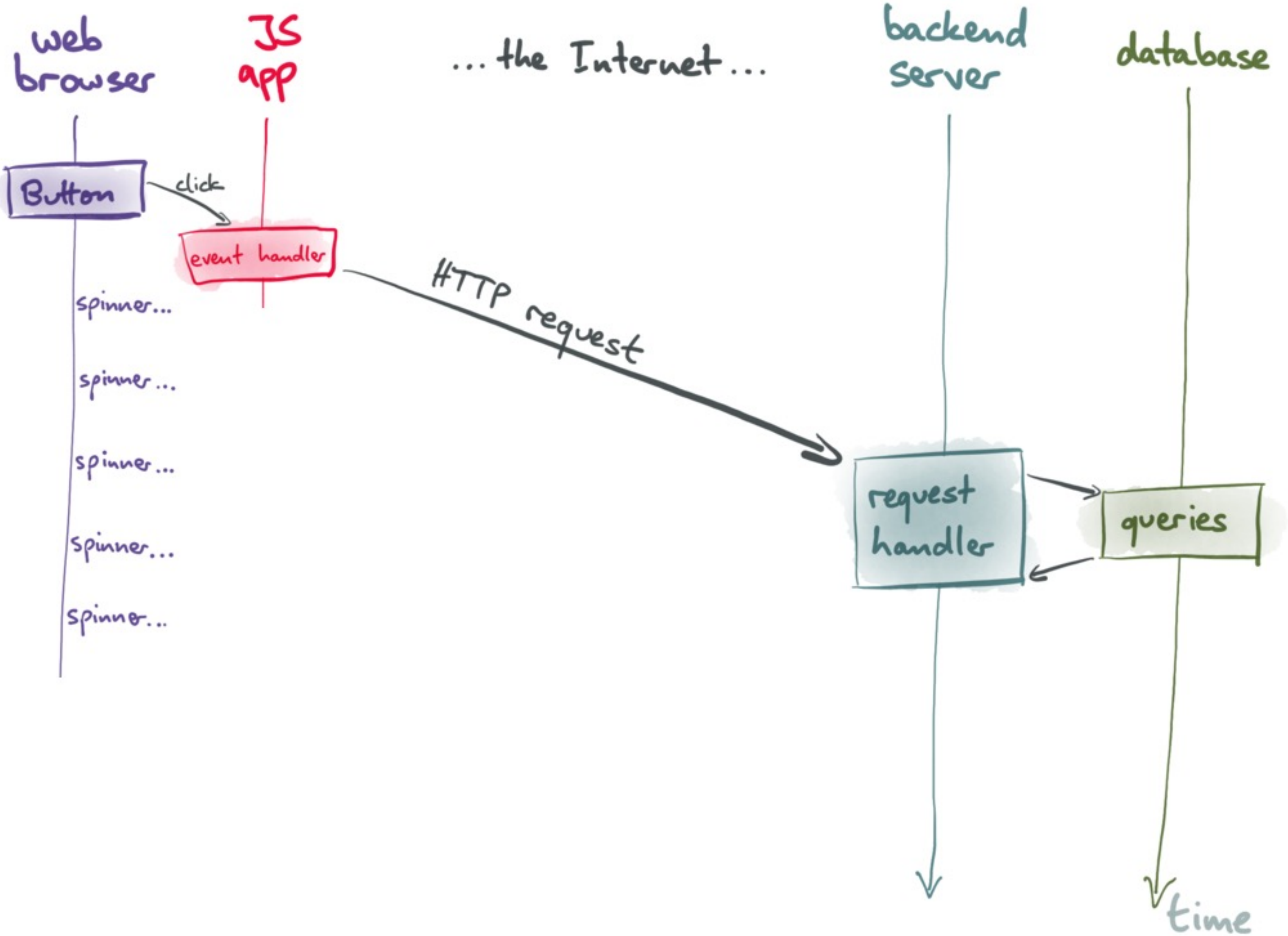

web browser
JS app
... the Internet ...
backend server
database
Button
click
event handler
HTTP request
request handler
queries
time

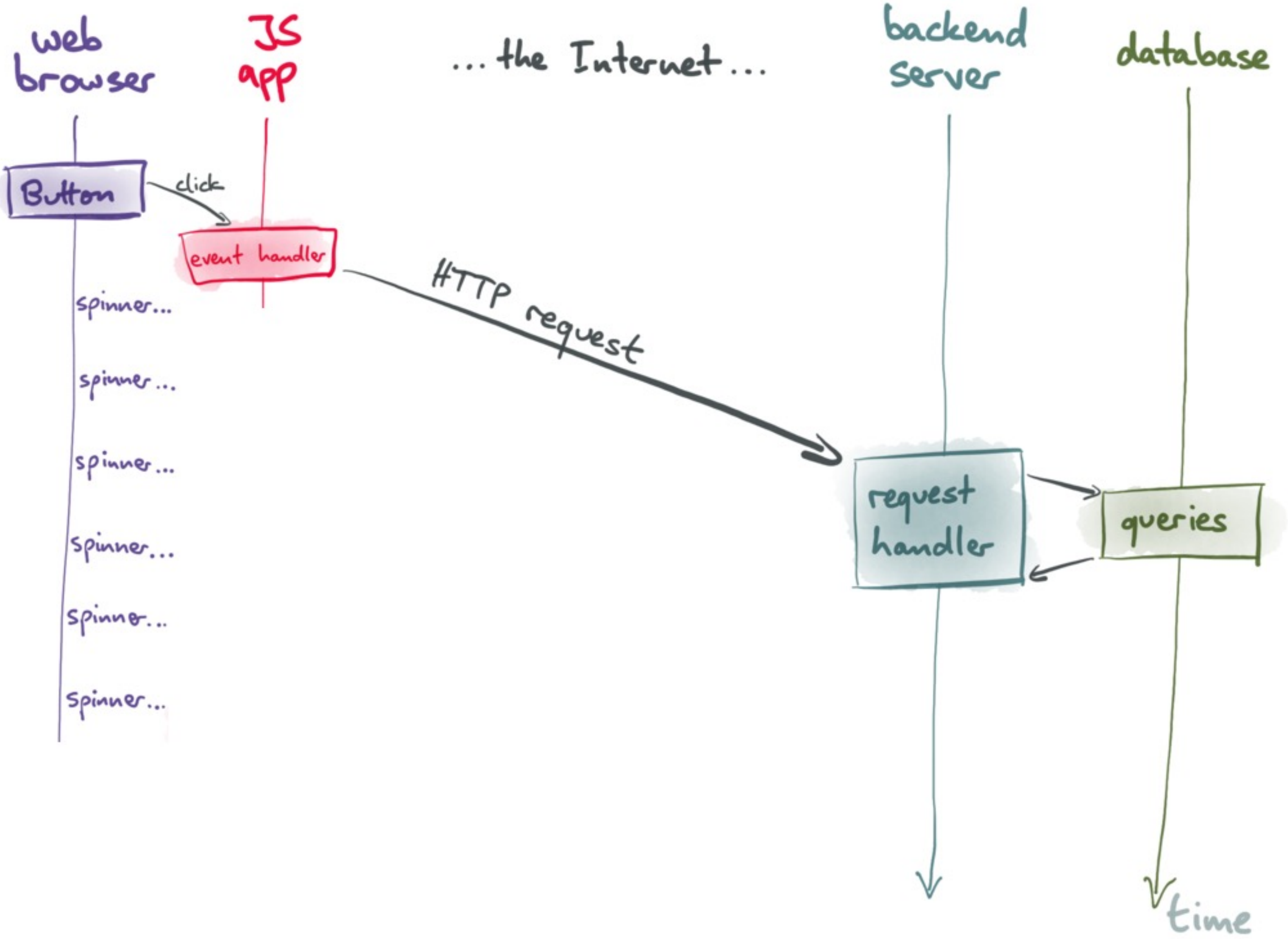

web browser
JS app
... the Internet ...
backend server
database
Button
click
event handler
spinner...
HTTP request
request handler
queries
time

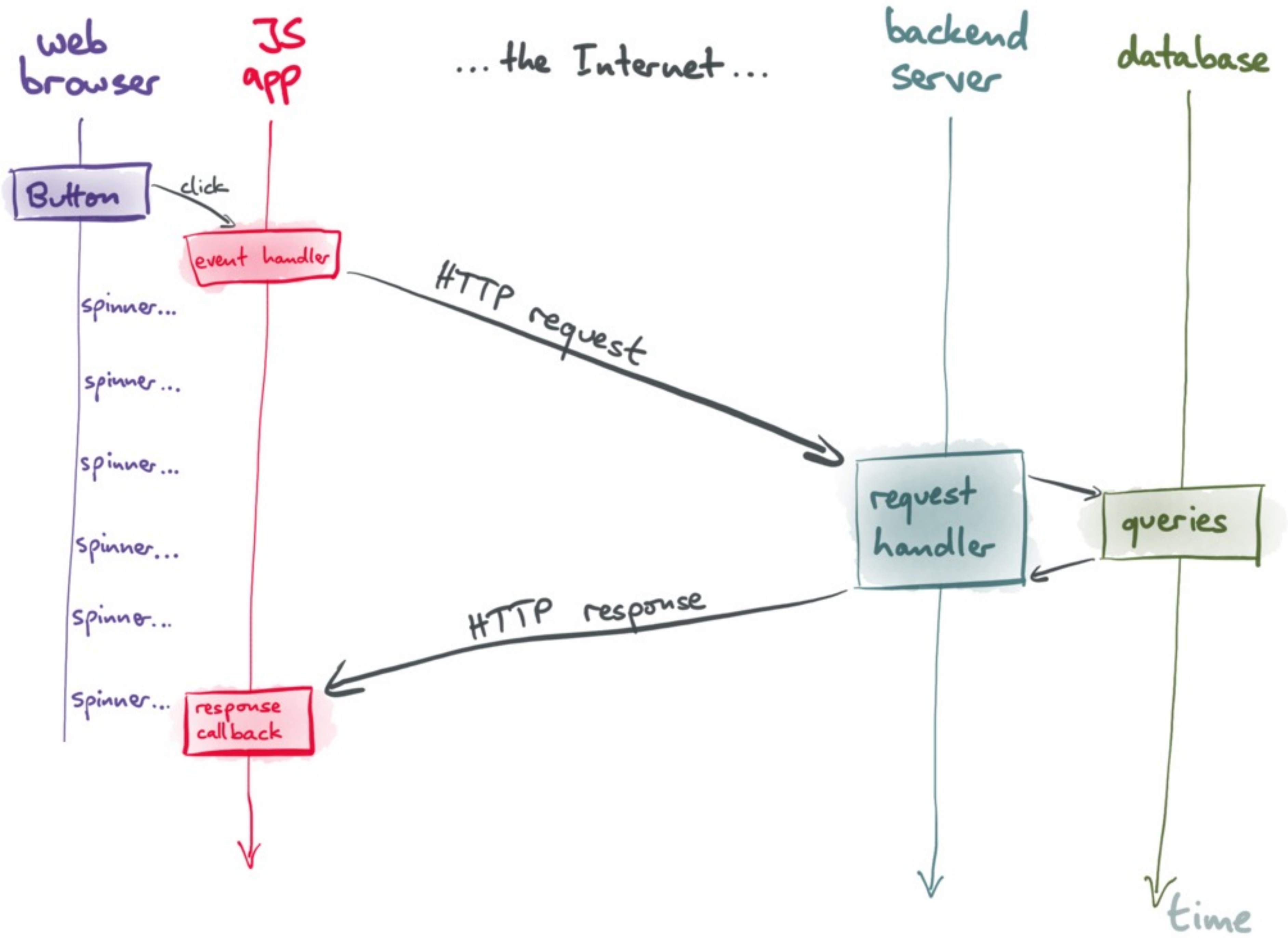

web browser
JS app
... the Internet ...
backend server
database
Button
click
event handler
spinner...
spinner...
HTTP request
request handler
queries
time

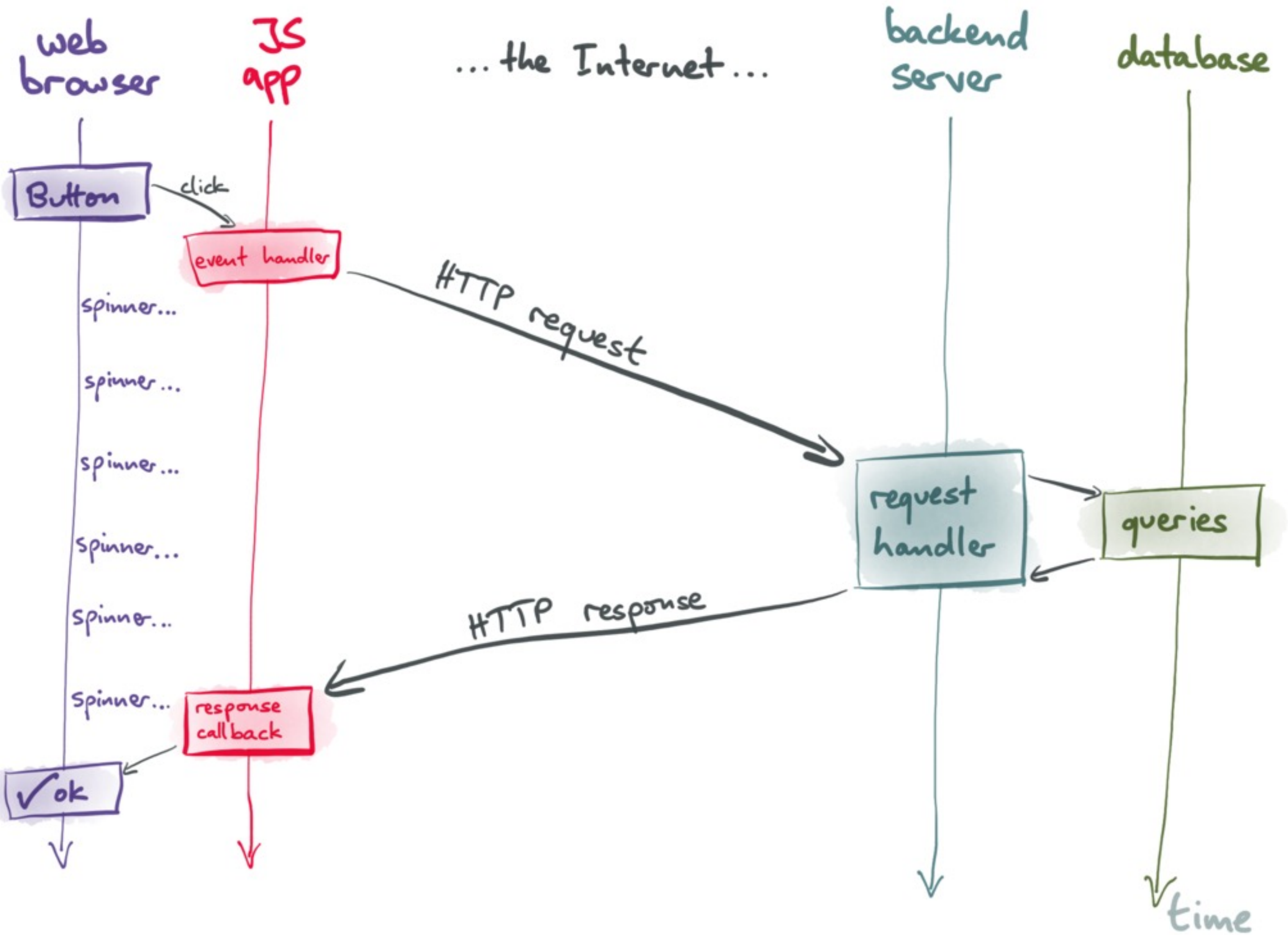

web browser
JS app
... the Internet ...
backend server
database
Button
click
event handler
HTTP request
spinner...
spinner...
spinner...
request handler
queries
time

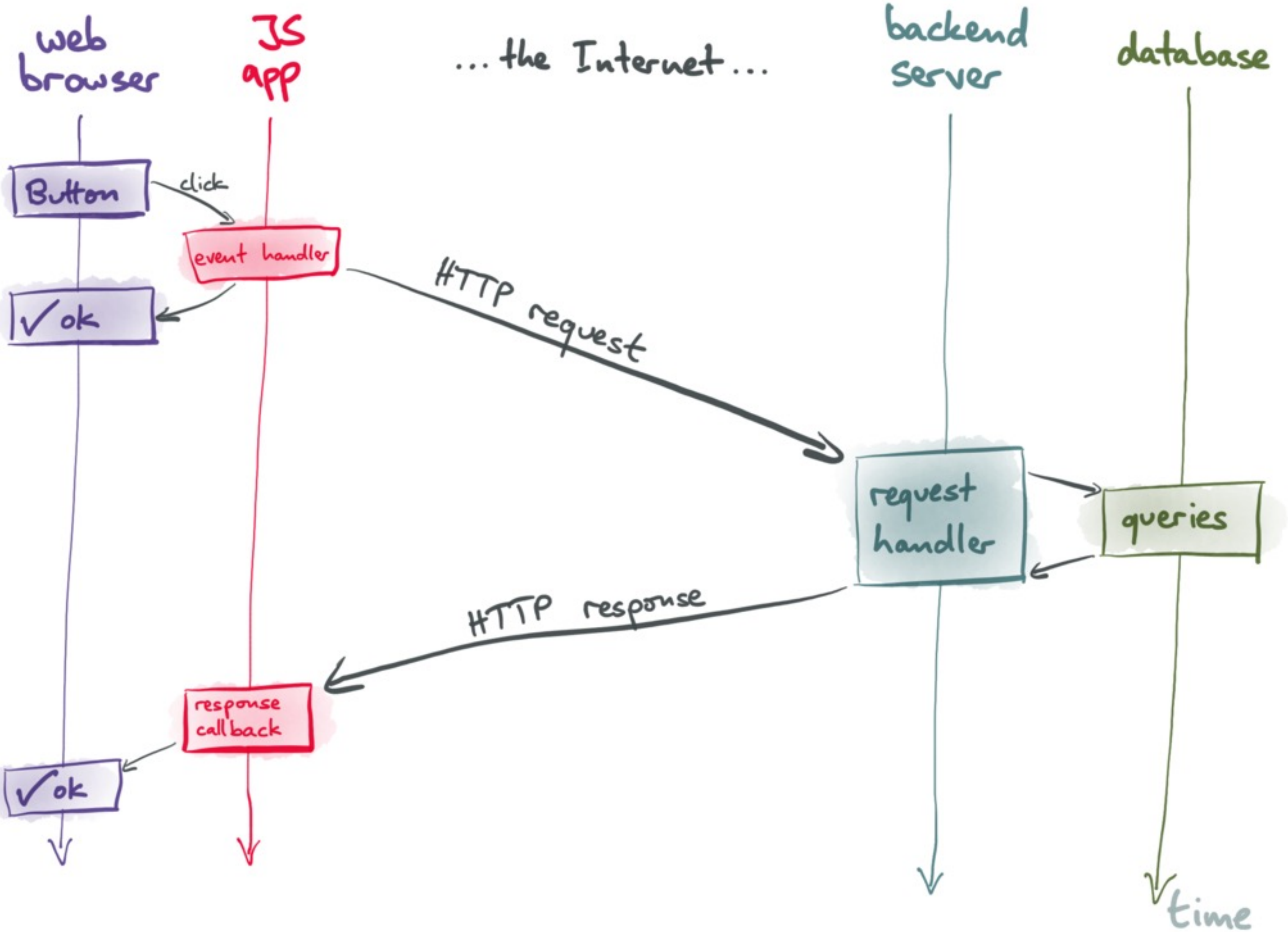

web browser
JS app
... the Internet ...
backend server
database
Button
click
event handler
HTTP request
spinner...
spinner...
spinner...
spinner...
request handler
queries
time

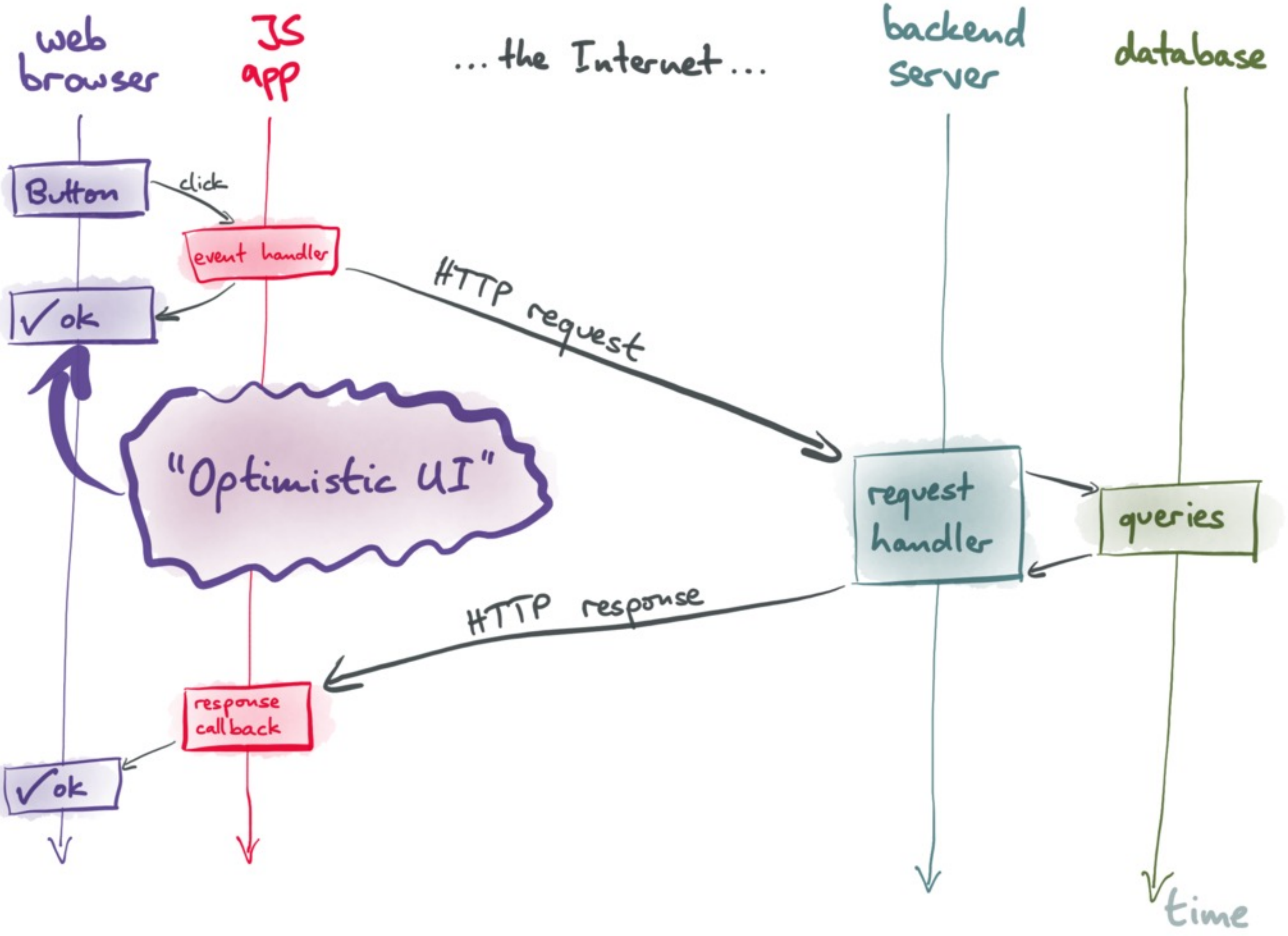

web browser
JS app
... the Internet ...
backend server
database
Button
click
event handler
HTTP request
spinner...
spinner...
spinner...
spinner...
spinner...
request handler
queries
time

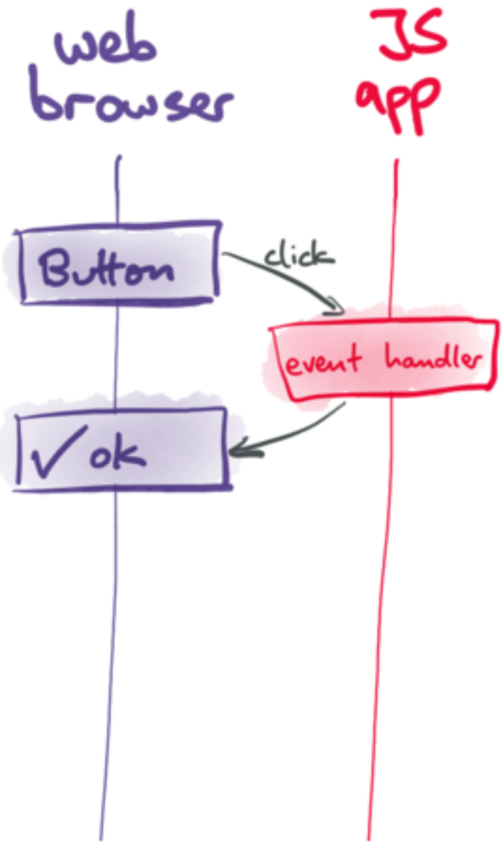

web browser
JS app
... the Internet ...
backend server
database
Button
click
event handler
HTTP request
spinner...
spinner...
spinner...
spinner...
spinner...
spinner...
request handler
queries
time

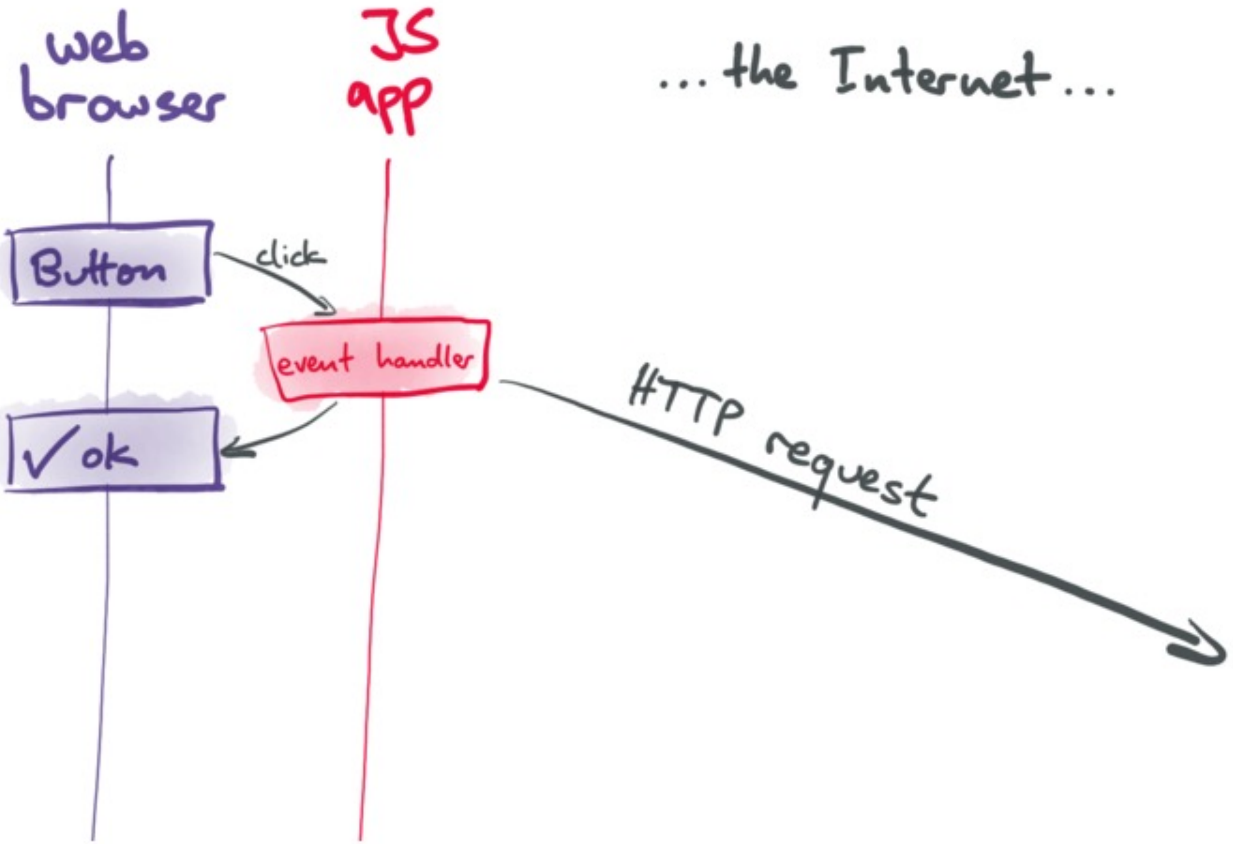

web browser
JS app
... the Internet ...
backend server
database
Button
click
event handler
spinner...
spinner...
spinner...
spinner...
spinner...
spinner...
HTTP request
request handler
queries
HTTP response
response callback
time

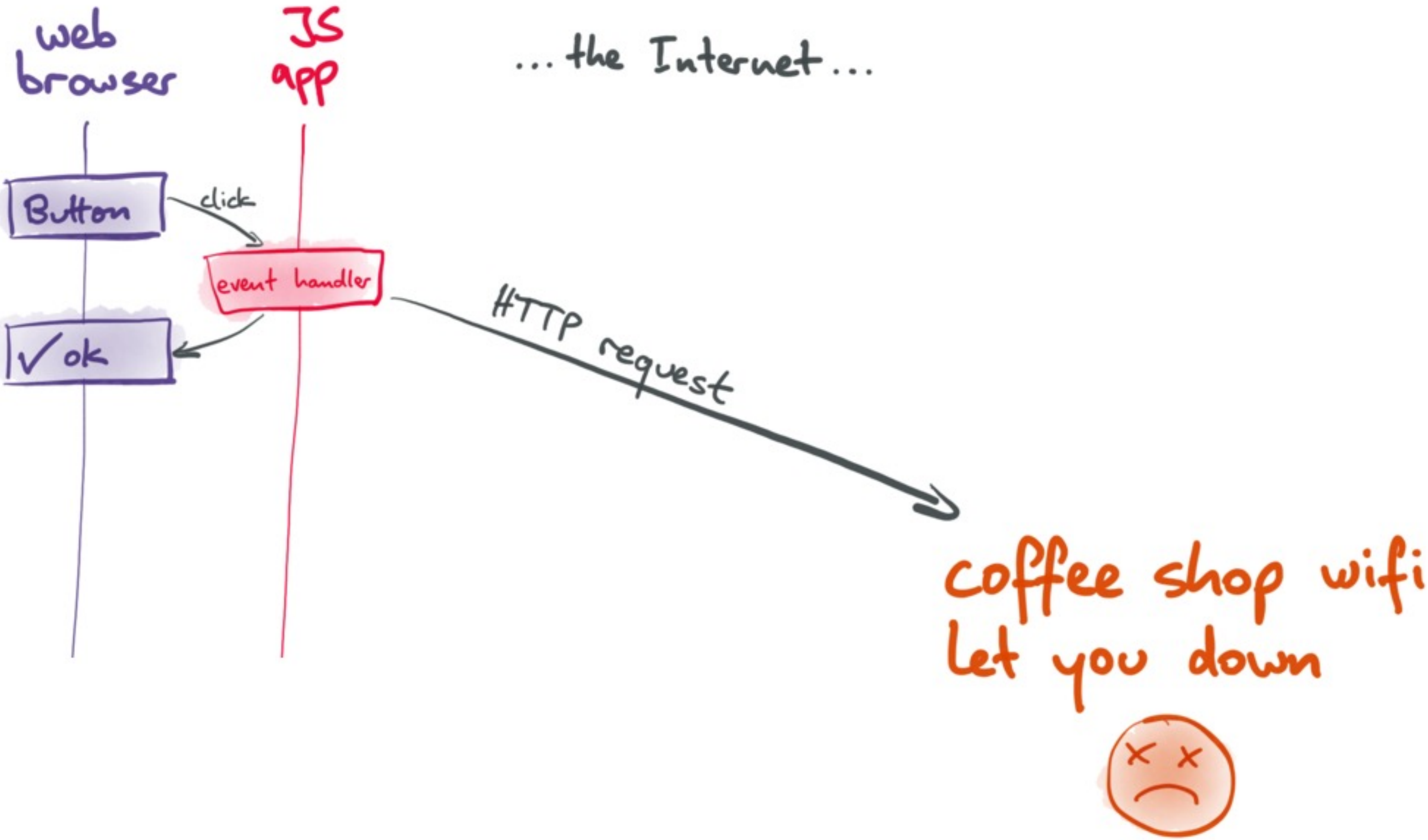

web browser
JS app
... the Internet ...
backend server
database
Button
click
event handler
spinner...
spinner...
spinner...
spinner...
spinner...
spinner...
HTTP request
request handler
queries
HTTP response
response callback
✓ok
time

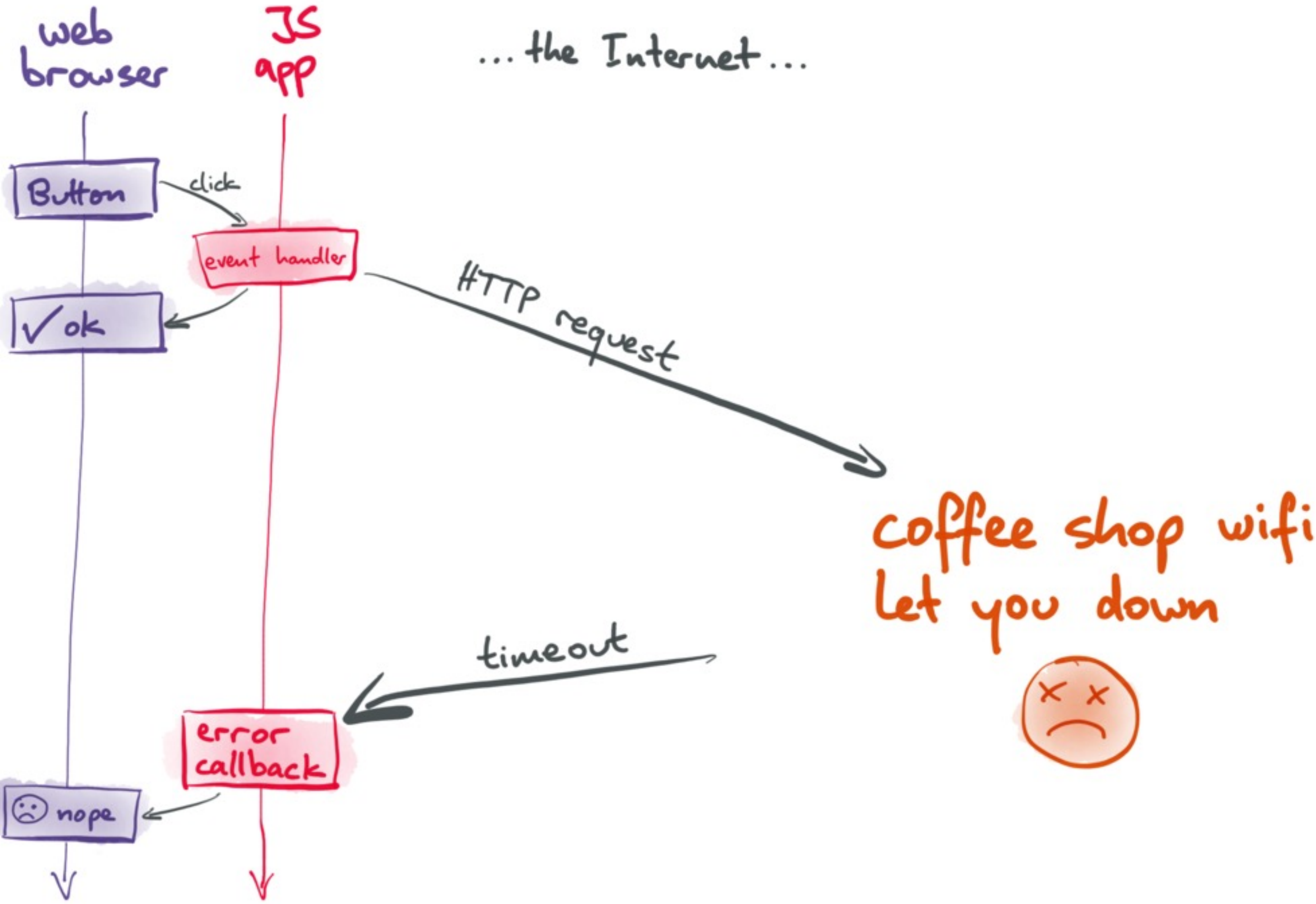

web browser
JS app
... the Internet ...
backend server
database
Button
click
event handler
✓ok
HTTP request
request handler
queries
HTTP response
response callback
✓ok
time


web browser
JS app
... the Internet ...
backend server
database
Button
click
event handler
✓ok
HTTP request
"Optimistic UI"
request handler
queries
HTTP response
response callback
✓ok
time


web browser
JS app
Button
click
event handler
✓ok


web browser
JS app
... the Internet ...
Button
click
event handler
✓ok
HTTP request


web browser
JS app
... the Internet ...
Button
click
event handler
✓ok
HTTP request
coffee shop wifi let you down


web browser
JS app
... the Internet ...
Button
click
event handler
✓ ok
HTTP request
coffee shop wifi let you down
timeout
error callback
☹ nope

You own your data, in spite of the cloud

Local-first

SOFTWARE

https://inkandswitch.com/local-first/

TRADITIONAL WEB APP MODEL:

"If it's not stored in the server database, it didn't really happen"

Can't reach the server? Can't do anything!

TRADITIONAL WEB APP MODEL:

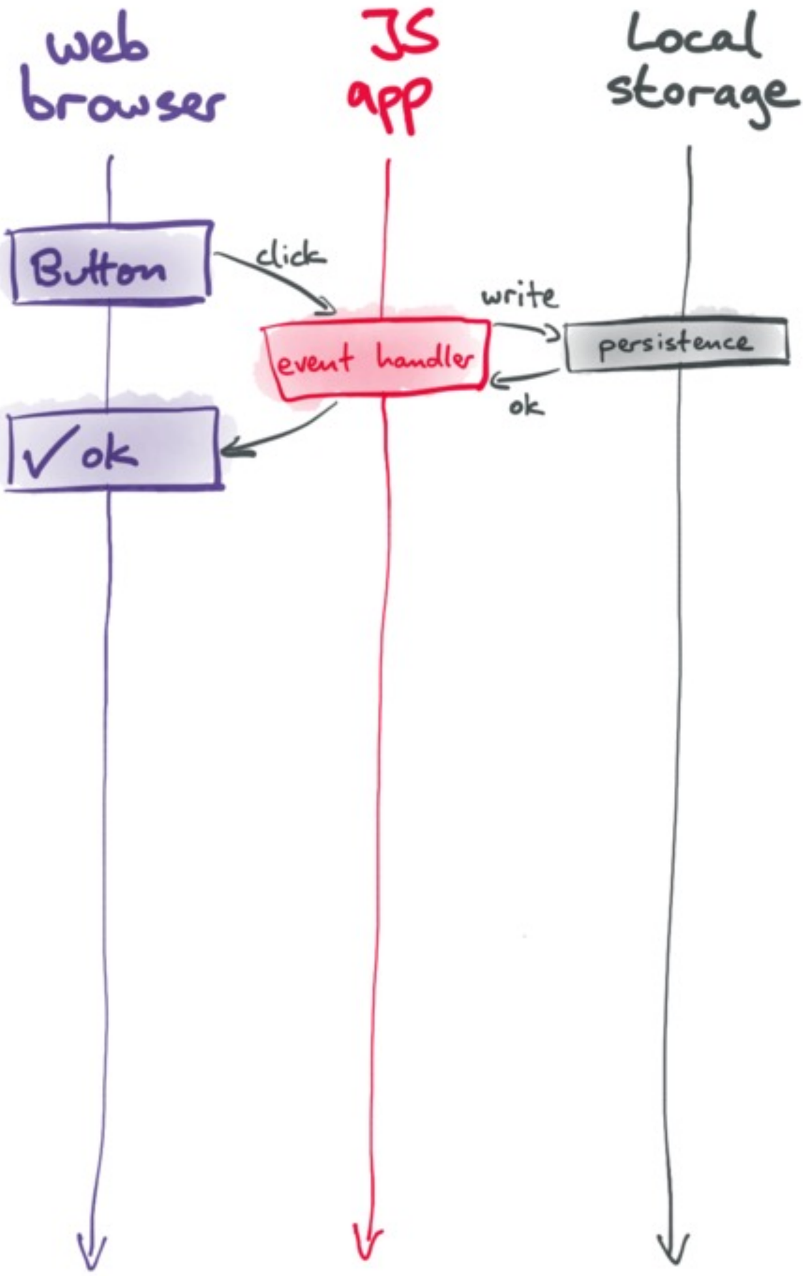
"If it's not stored in the server database, it didn't really happen"

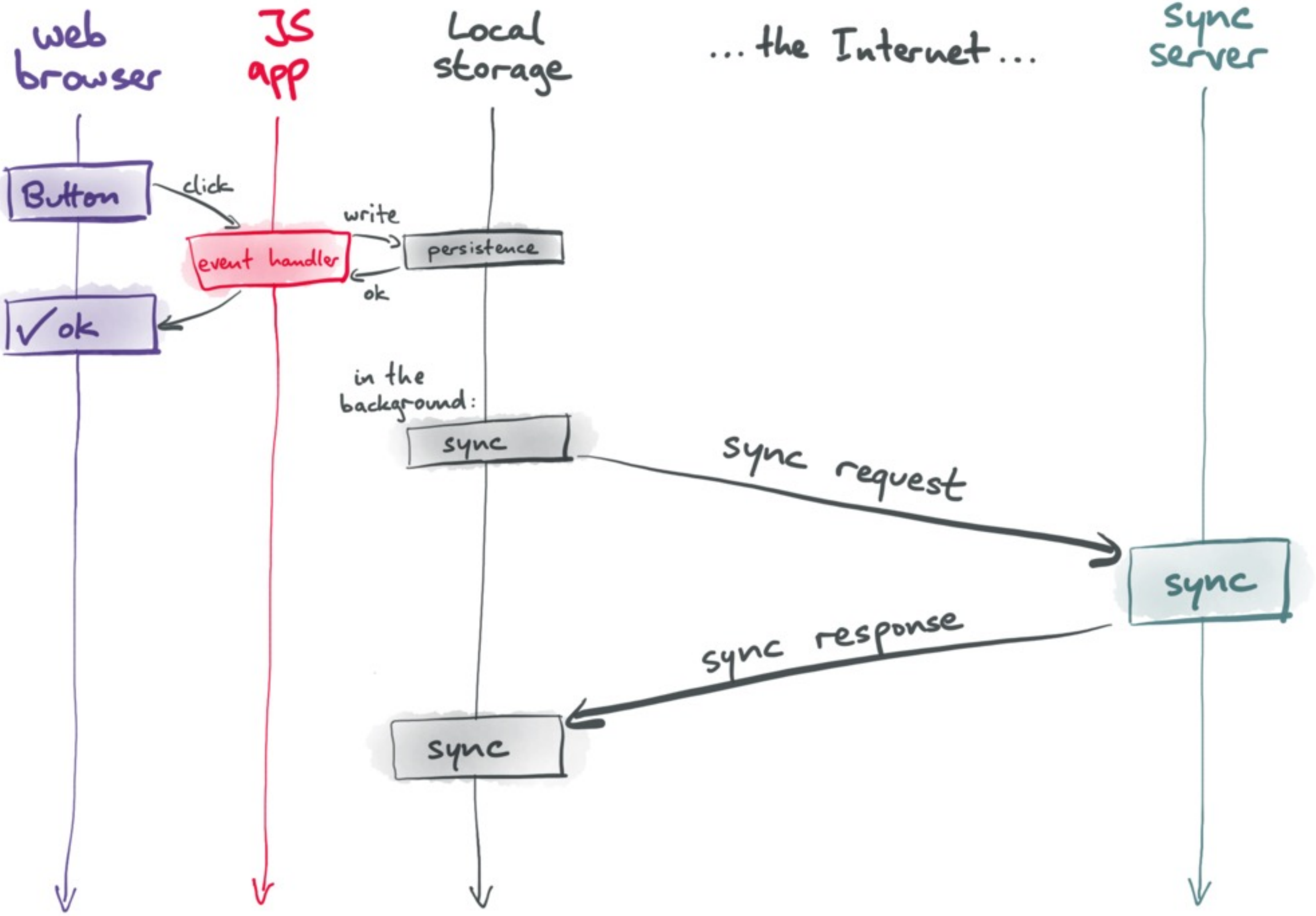
Can't reach the server? Can't do anything!

LOCAL-FIRST MODEL:

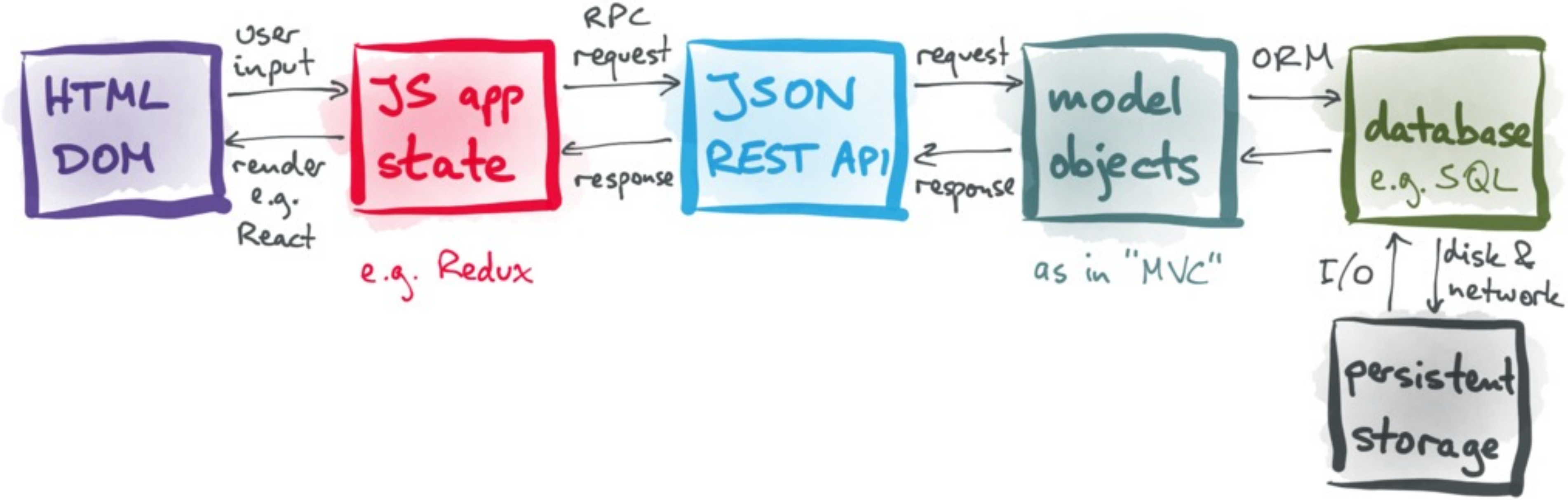
"The client's local storage is what matters — the server is just for multi-user sync and backup"

Don't care if we're online or offline right now!

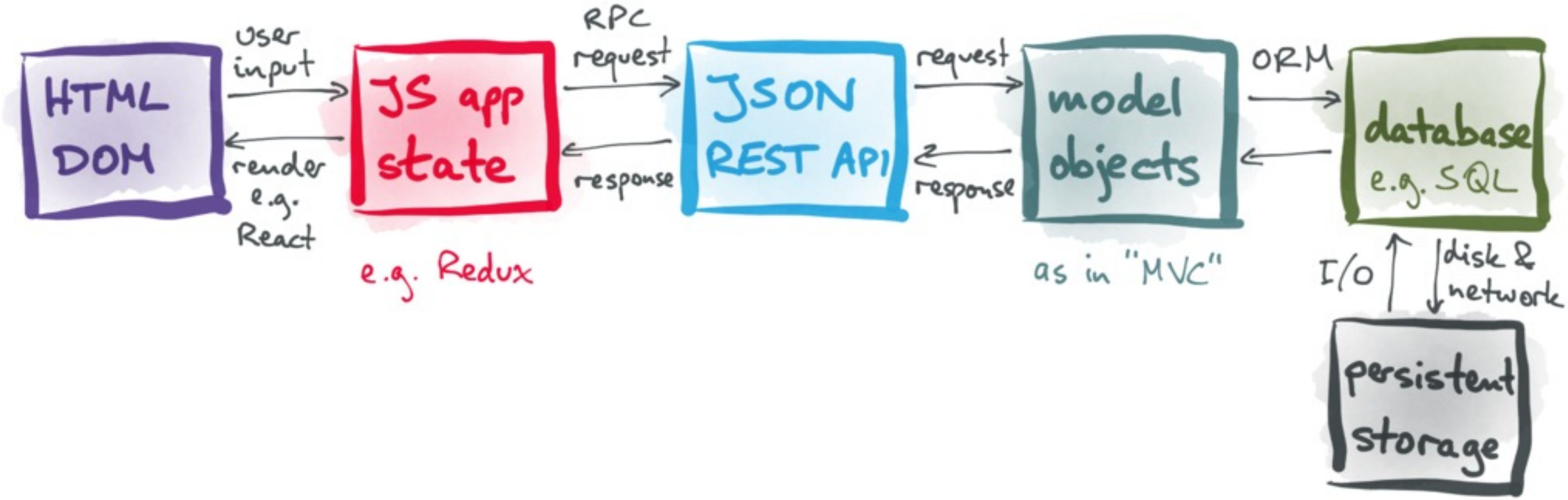

web browser
JS app
Local storage
Button
click
event handler
write
persistence
ok
✓ok


web browser
JS app
Local storage
... the Internet ...
Sync server
Button
click
event handler
write
persistence
ok
✓ ok
in the background:
sync
sync request
sync
sync response
sync

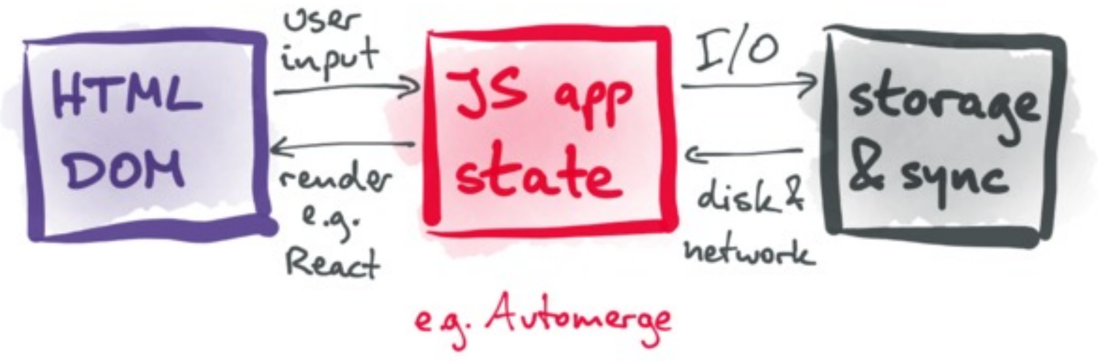
TRADITIONAL WEB APP MODEL:



TRADITIONAL WEB APP MODEL:



LOCAL-FIRST MODEL:



Self-contained!

CLOUD SOFTWARE	LOCAL-FIRST SOFTWARE
☹ Real-time collaboration is hard to implement	☺ Built for real-time sync

CLOUD SOFTWARE	LOCAL-FIRST SOFTWARE
☹ Real-time collaboration is hard to implement	☺ Built for real-time sync
☹ Does not work offline	☺ Works offline

CLOUD SOFTWARE	LOCAL-FIRST SOFTWARE
☹ Real-time collaboration is hard to implement	☺ Built for real-time sync
☹ Does not work offline	☺ Works offline
☹ Service shuts down? Lose everything!	☺ Users can continue using local copy of software + data

CLOUD SOFTWARE	LOCAL-FIRST SOFTWARE
☹ Real-time collaboration is hard to implement	☺ Built for real-time sync
☹ Does not work offline	☺ Works offline
☹ Service shuts down? Lose everything!	☺ Users can continue using local copy of software + data
☹ Custom service for each app (infra, ops, on-call rotation, ...)	☺ Sync service is generic ⇒ outsource to cloud vendor

Local-first is a good fit for:	Local-first is a bad fit for:
"File editing" software (text editor, word processor, spreadsheet, presentation slides, graphics editor, video editing, music production, CAD software for engineering, Jupyter notebooks, ...)	

Local-first is a good fit for:

"File editing" software

(text editor, word processor, spreadsheet,
presentation slides, graphics editor,
video editing, music production,
CAD software for engineering,
Jupyter notebooks, ...)

Productivity software

(notes, to-do lists, issue trackers,
calendar, time tracking, group messaging,
bookkeeping...)

Basically, apps where the user can edit the data however they like.

Local-first is a bad fit for:

Local-first is a good fit for:

"File editing" software

(text editor, word processor, spreadsheet, presentation slides, graphics editor, video editing, music production, CAD software for engineering, Jupyter notebooks, ...)

Productivity software

(notes, to-do lists, issue trackers, calendar, time tracking, group messaging, bookkeeping...)

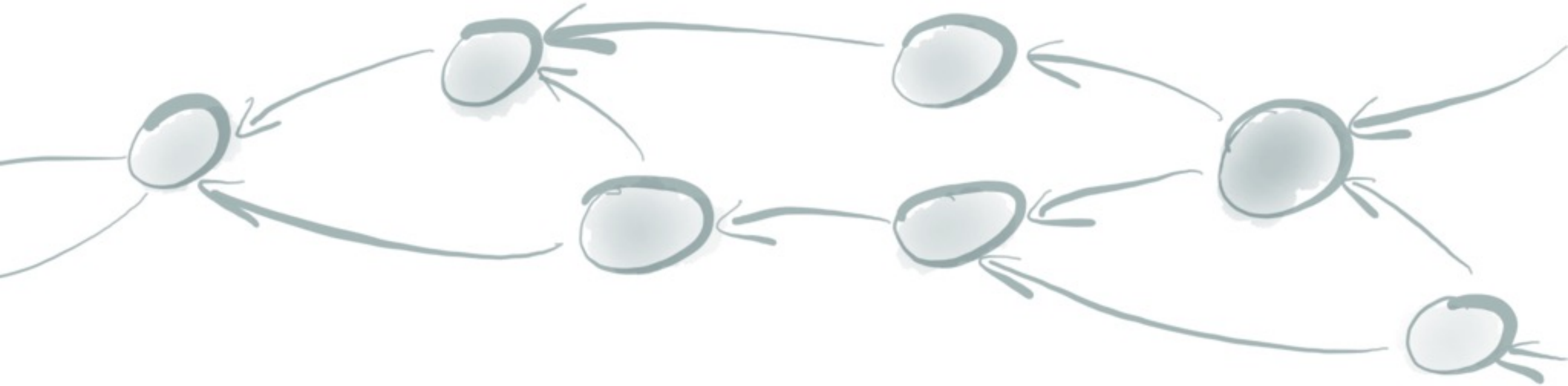
Basically, apps where the user can edit the data however they like.

Local-first is a bad fit for:

Managing a real-world resource, e.g.

- money (bank account, payments, ad impressions)
- physical products (e-commerce, warehouse inventory)
- vehicles (car-sharing/rental, freight/logistics)

For these apps, a centralised cloud/server model works best.



Automerge

https://automerge.org

AUTOMERGE: "Git for your app's data"

```
{"todos": [
    {"title": "buy milk", "done": false},
    {"title": "water plants", "done": false}
]}
```

AUTOMERGE: "Git for your app's data"

```
{"todos": [
   {"title": "buy milk", "done": false},
   {"title": "water plants", "done": false}
]}
```

```
after = Automerge.change(before, "mark item as done", doc => {
      doc.todos[1].done = true;
});
```

AUTOMERGE: "Git for your app's data"

{"todos": [
  {"title": "buy milk", "done": false},
  {"title": "water plants", "done": ~~false~~}
                                                      true
]}

```
after = Automerge.change(before, "mark item as done", doc => {
    doc.todos[1].done = true;
});
```

AUTOMERGE: "Git for your app's data"

```
{"todos": [
    {"title": "buy milk", "done": false},
    {"title": "water plants", "done": ~~false~~}
                                      true
]}
```

reflects updated state

immutable

```
after = Automerge.change(before, "mark item as done", doc => {
    doc.todos[1].done = true;
});
```

AUTOMERGE: "Git for your app's data"

```
{"todos": [
    {"title": "buy milk",  "done": false},
    {"title": "water plants",  "done": ~~false~~}
                                        true
]}
```

reflects updated state

immutable

"commit message" (optional)

```
after = Automerge.change(before, "mark item as done", doc => {
      doc.todos[1].done = true;
});
```

AUTOMERGE: "Git for your app's data"

```
{"todos": [
    {"title": "buy milk", "done": false},
    {"title": "water plants", "done": ~~false~~}
                                      true
]}
```

reflects updated state

immutable

"commit message" (optional)

```
after = Automerge.change(before, "mark item as done", doc => {
    doc.todos[1].done = true;
});
```

record exactly what changed

AUTOMERGE: "Git for your app's data"

```
{"todos": [
    {"title": "buy milk", "done": false},
    {"title": "water plants", "done": false}
]}
```

AUTOMERGE: "Git for your app's data"

```
{"todos": [
    {"title": "buy milk", "done": false},
    {"title": "water plants", "done": false}
]}
```

```
after = Automerge.change(before, "add new item", doc => {
        doc.todos.push({title: "do laundry", done: false});
});
```

AUTOMERGE: "Git for your app's data"

```
{"todos": [
    {"title": "buy milk", "done": false},
    {"title": "water plants", "done": false},
    {"title": "do laundry", "done": false}
]}
```

← added

```
after = Automerge.change(before, "add new item", doc => {
      doc.todos.push({title: "do laundry", done: false});
});
```

AUTOMERGE: "Git for your app's data"

```
{"todos": [
    {"title": "buy milk", "done": false},
    {"title": "water plants", "done": false},
    {"title": "do laundry", "done": false}
]}
```

added

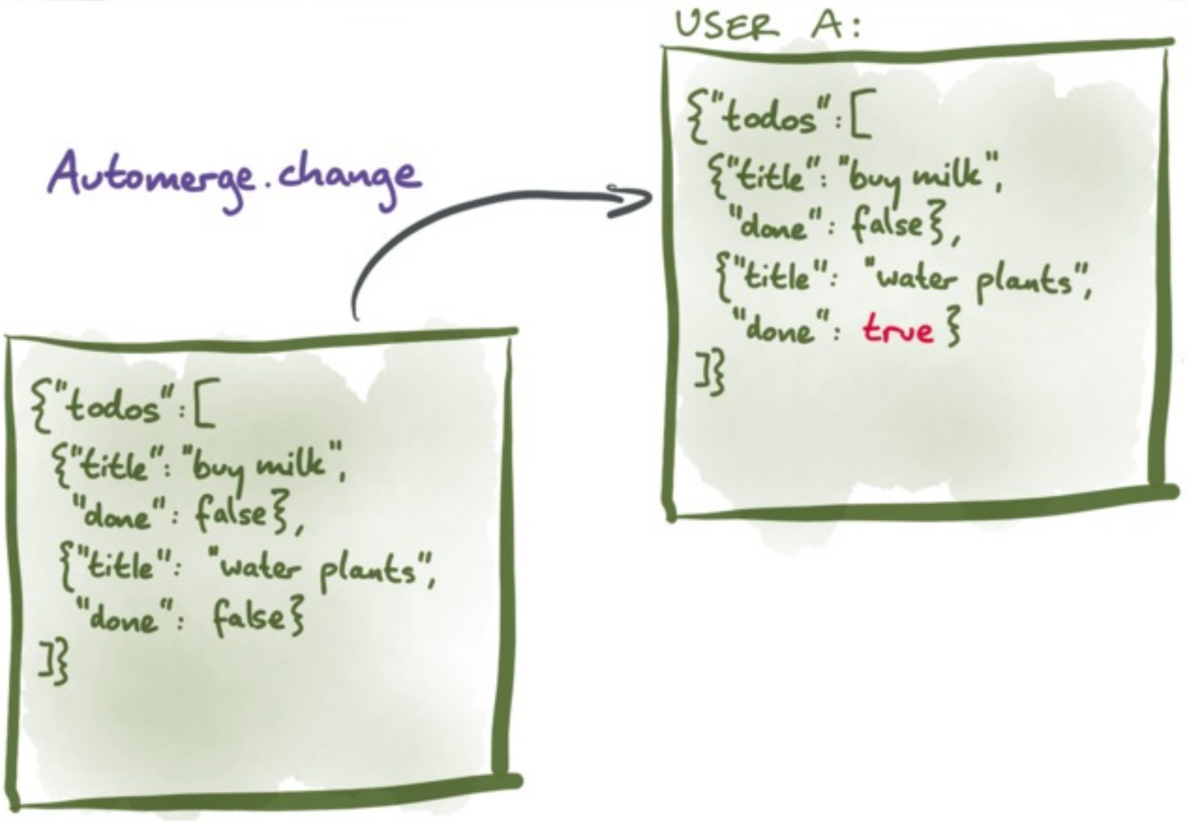
```
after = Automerge.change(before, "add new item", doc => {
    doc.todos.push({title: "do laundry", done: false});
});
```

append item to list

AUTOMERGE: Branching and merging

```
{"todos":[
 {"title":"buy milk",
  "done": false},
 {"title": "water plants",
  "done": false}
]}
```

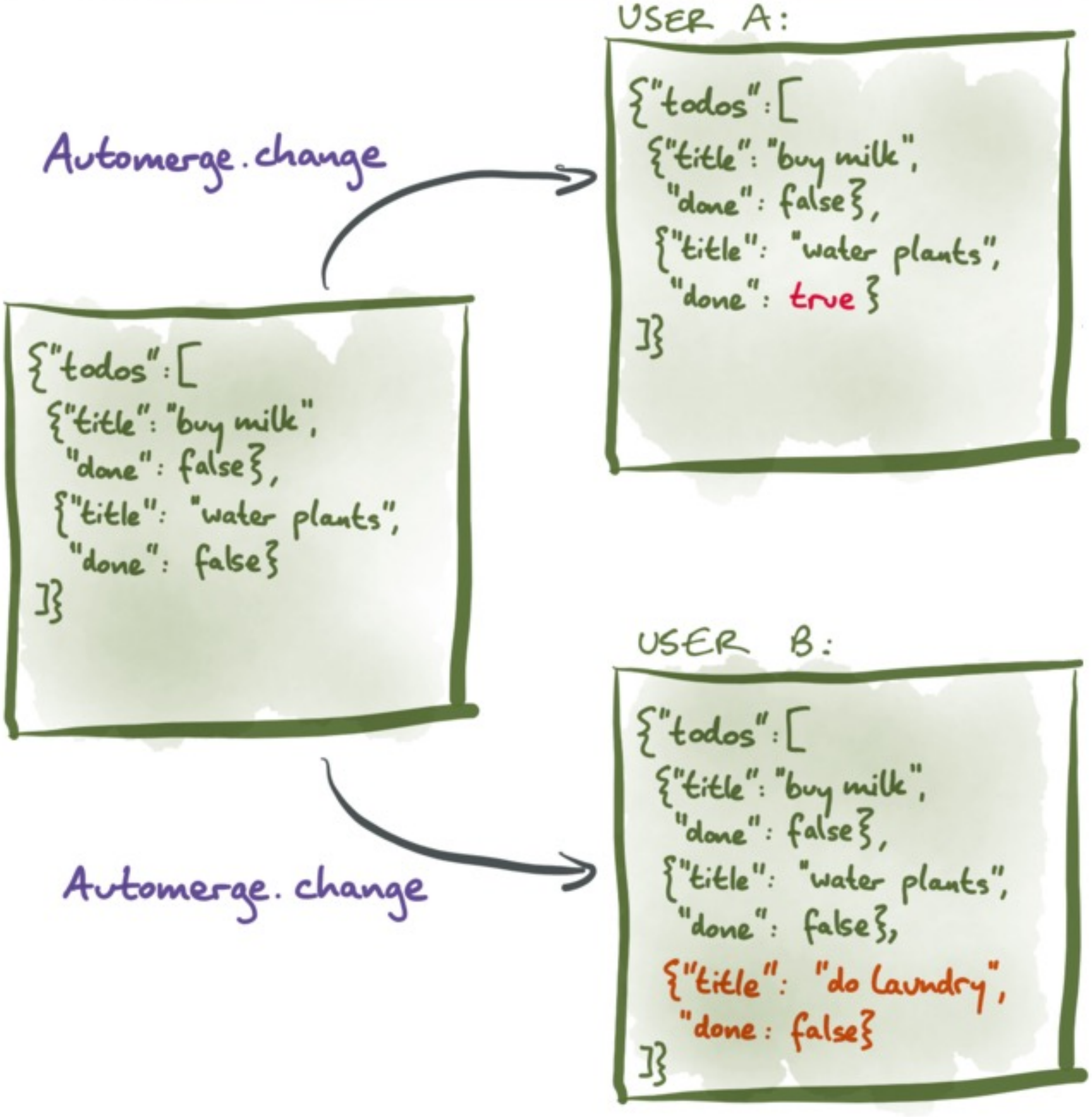
AUTOMERGE: Branching and merging



```
{"todos":[
 {"title": "buy milk",
  "done": false},
 {"title": "water plants",
  "done": false}
]}
```

```
{"todos":[
 {"title": "buy milk",
  "done": false},
 {"title": "water plants",
  "done": true}
]}
```

AUTOMERGE: Branching and merging



```
{"todos":[
 {"title": "buy milk",
  "done": false},
 {"title": "water plants",
  "done": false}
]}
```

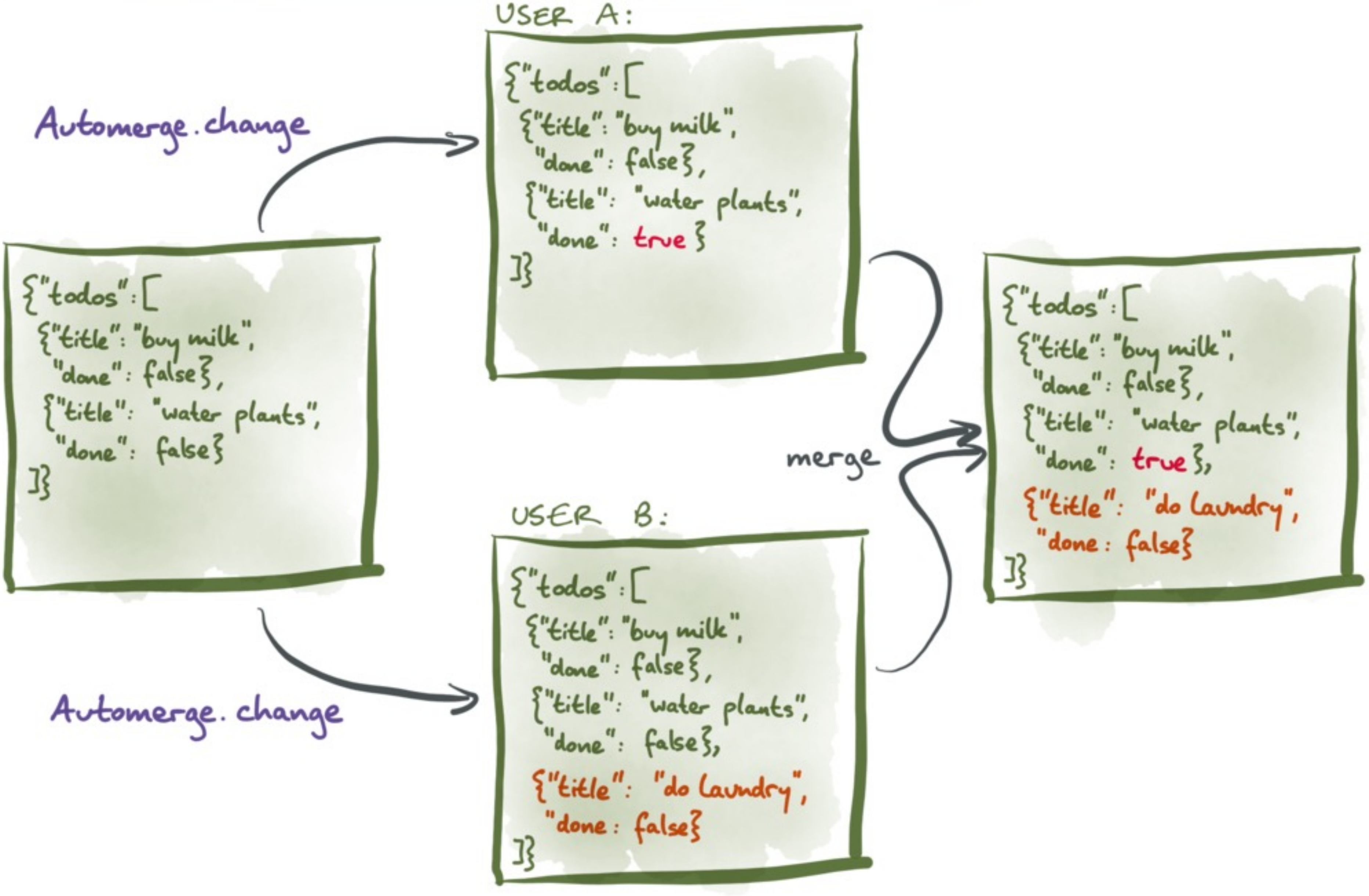
USER A:

```
{"todos":[
 {"title": "buy milk",
  "done": false},
 {"title": "water plants",
  "done": true}
]}
```

USER B:

```
{"todos":[
 {"title": "buy milk",
  "done": false},
 {"title": "water plants",
  "done": false},
 {"title": "do Laundry",
  "done : false}
]}
```

AUTOMERGE: Branching and merging



```
{"todos":[
 {"title":"buy milk",
  "done": false},
 {"title": "water plants",
  "done": false}
]}
```

USER A:

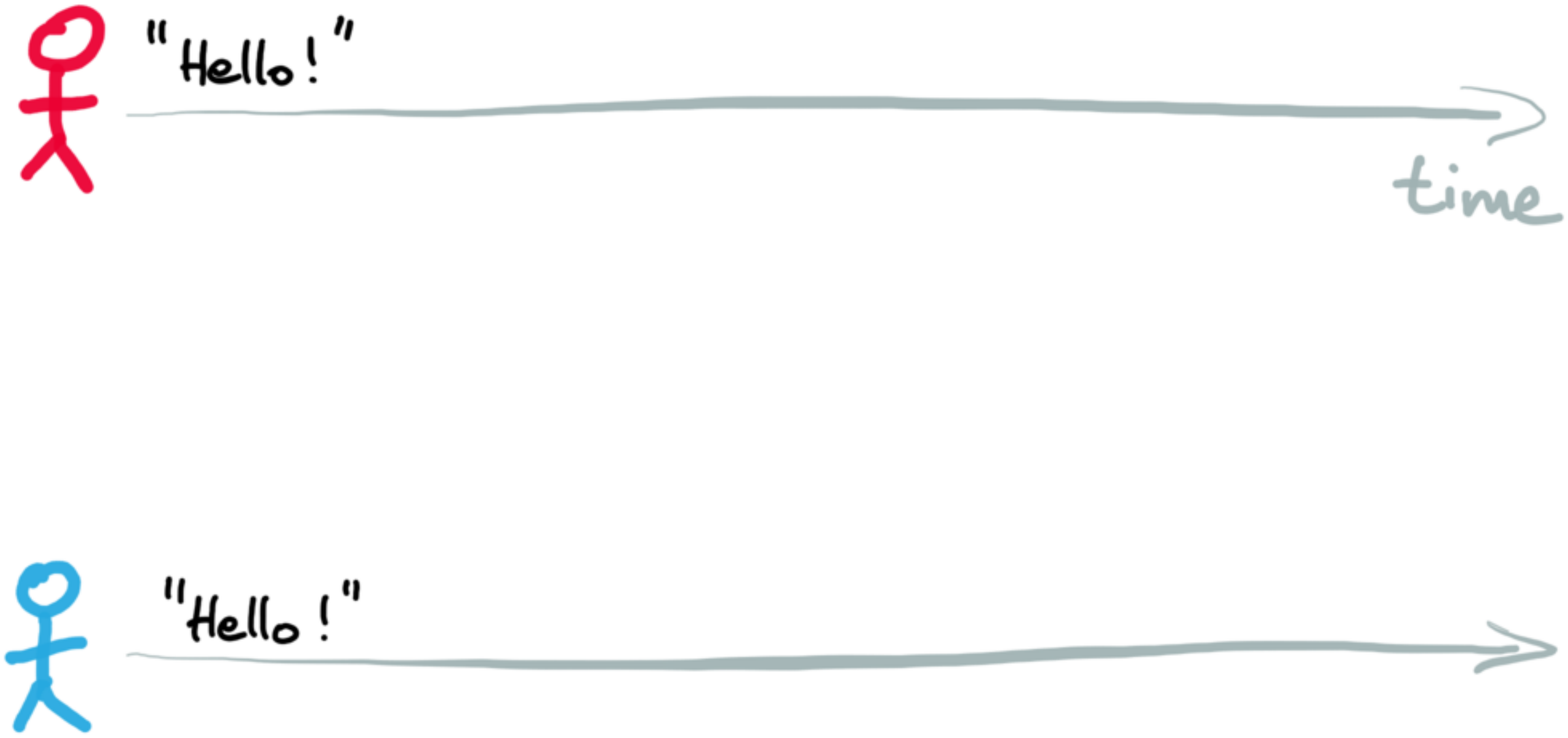
```
{"todos":[
 {"title":"buy milk",
  "done": false},
 {"title": "water plants",
  "done": true }
]}
```

USER B:

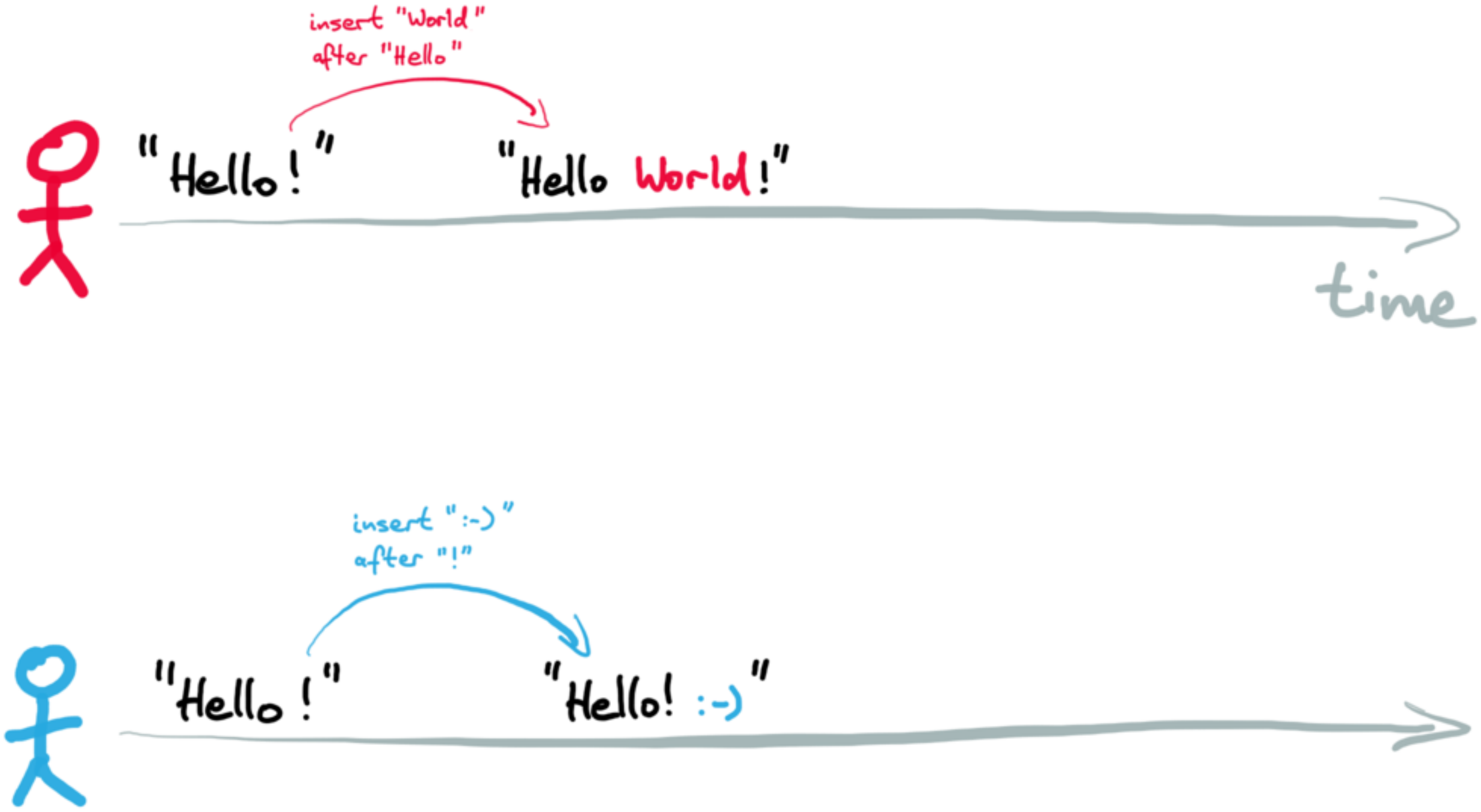
```
{"todos":[
 {"title":"buy milk",
  "done": false},
 {"title": "water plants",
  "done": false},
 {"title": "do laundry",
  "done : false}
]}
```

```
{"todos":[
 {"title":"buy milk",
  "done": false},
 {"title": "water plants",
  "done": true},
 {"title": "do laundry",
  "done : false}
]}
```

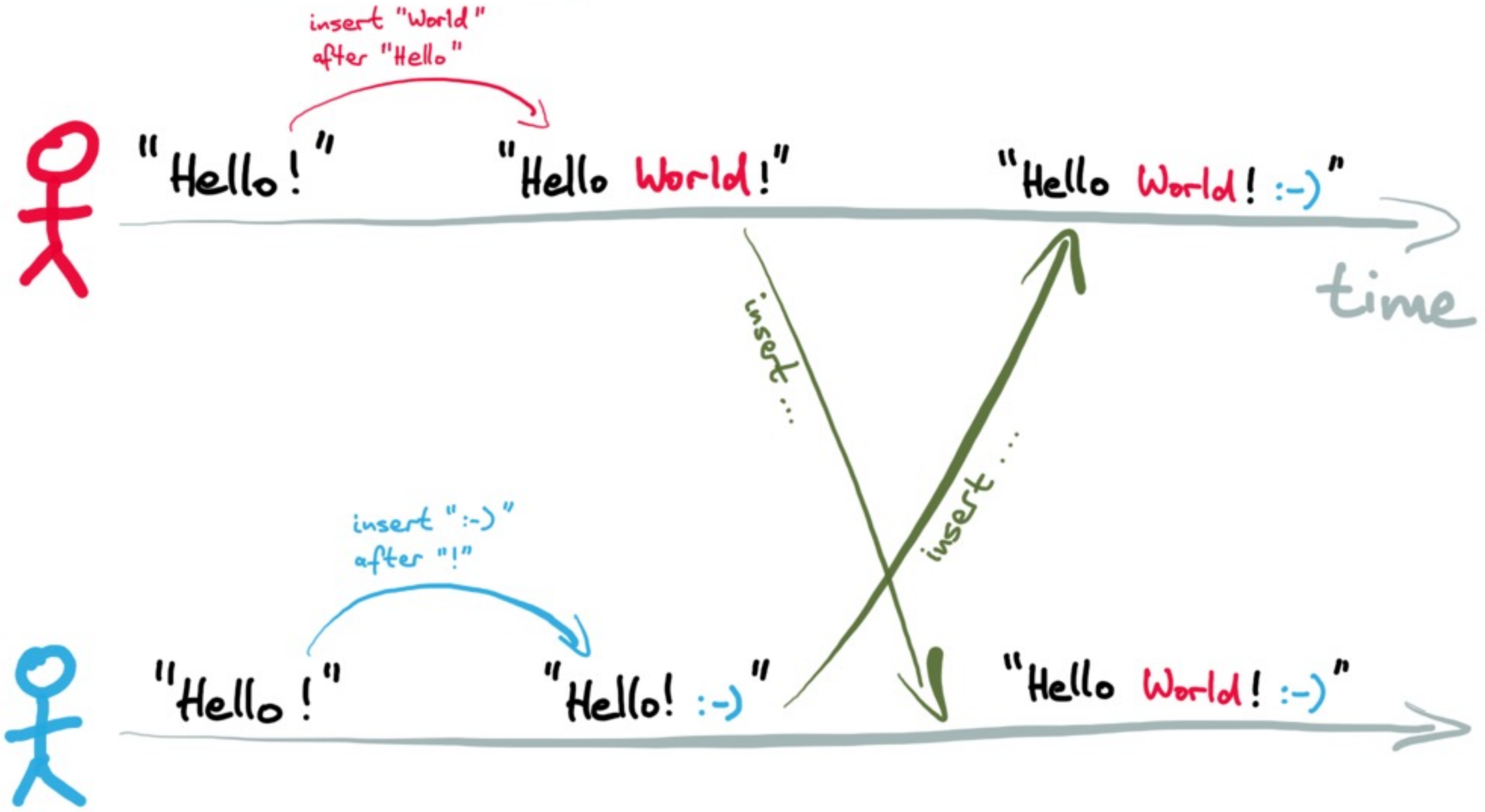
Example: Text editing



Example: Text editing



Example: Text editing

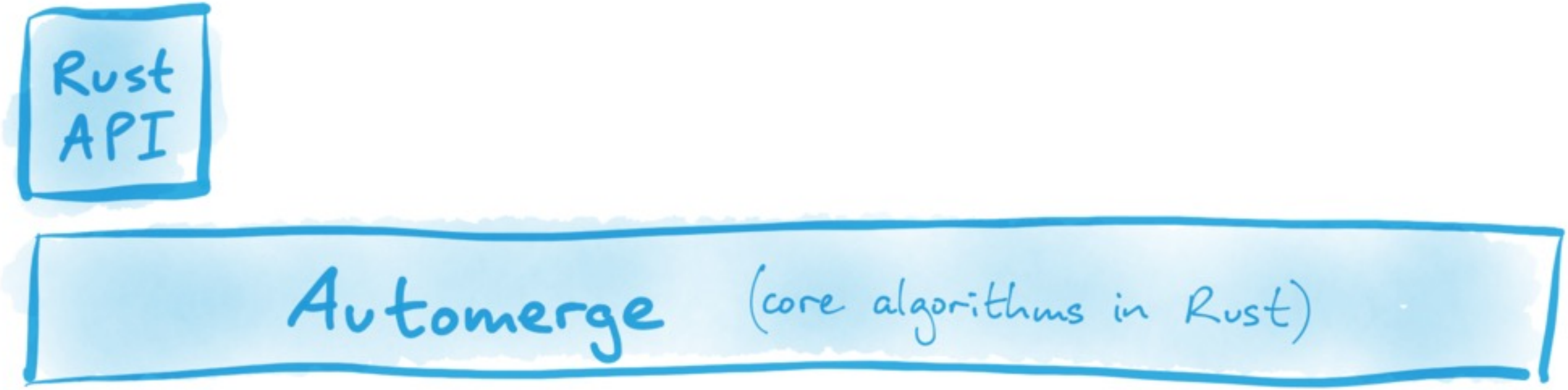


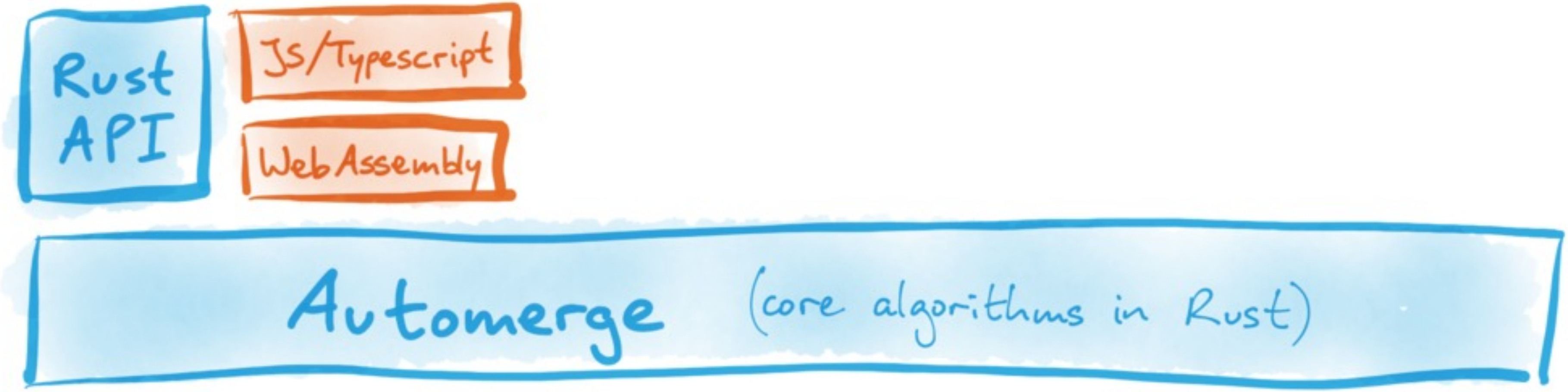
Automerge guarantees:

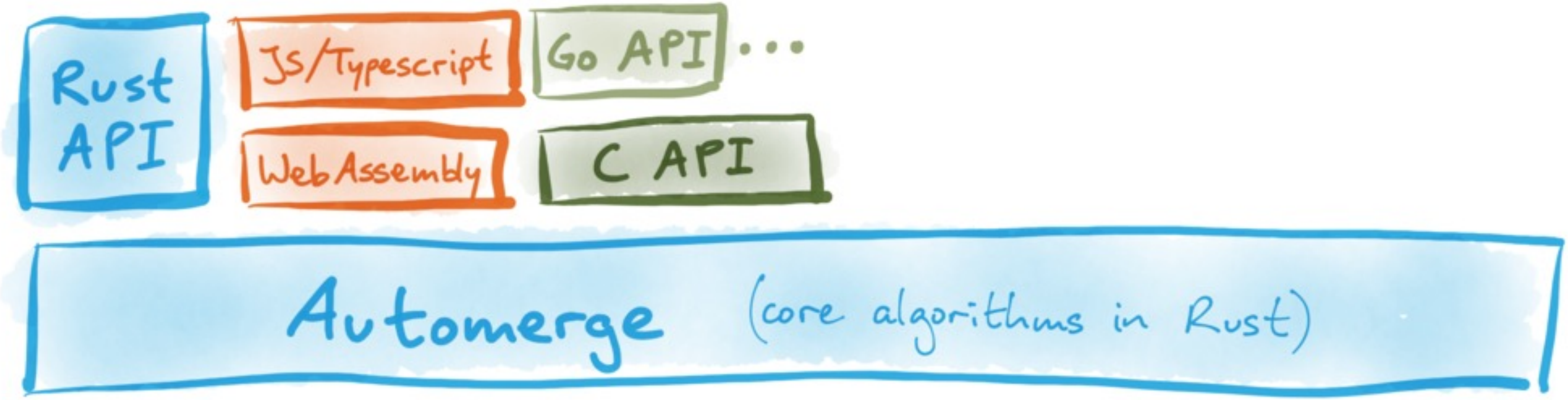
- All changes are preserved
- If two users have seen the same set of changes (in any order), then they are in the same state
- Branches (=concurrent updates) can be merged automatically
- Can branch & merge arbitrarily, can inspect/compare versions

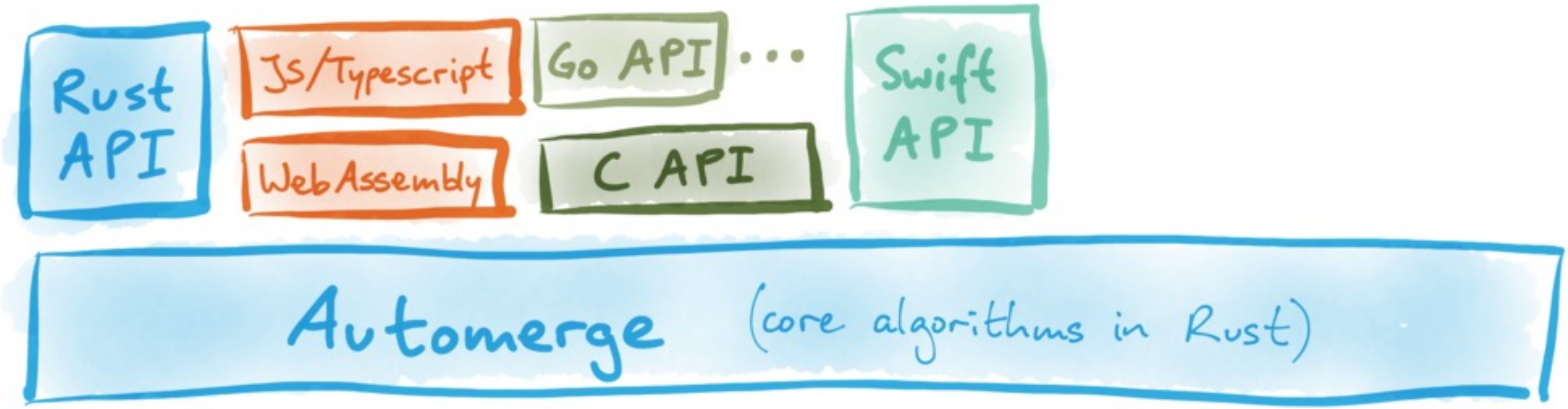
Automerge is a CRDT

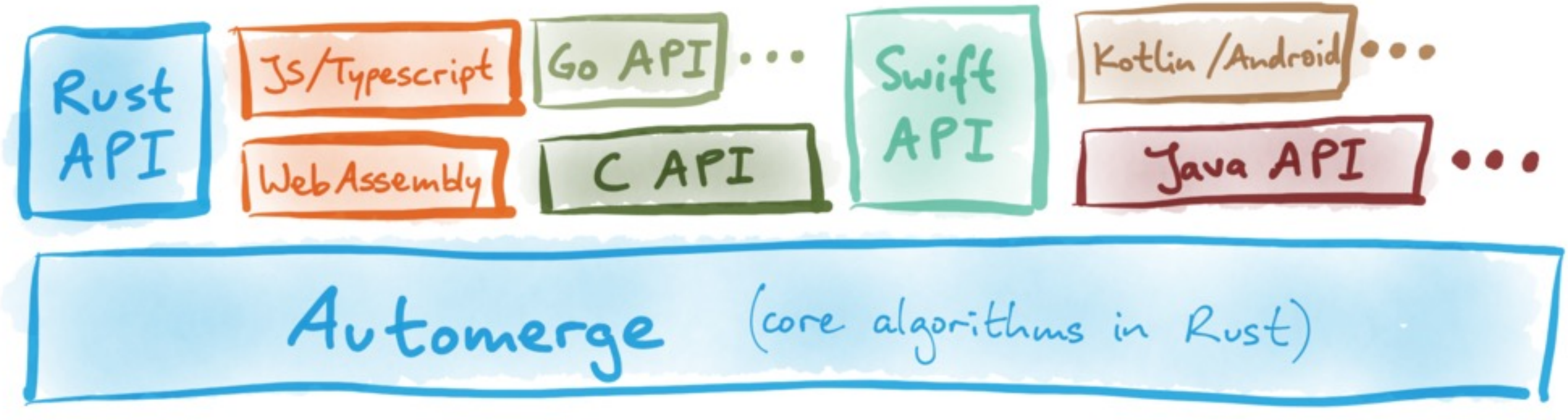
(conflict-free replicated data type)

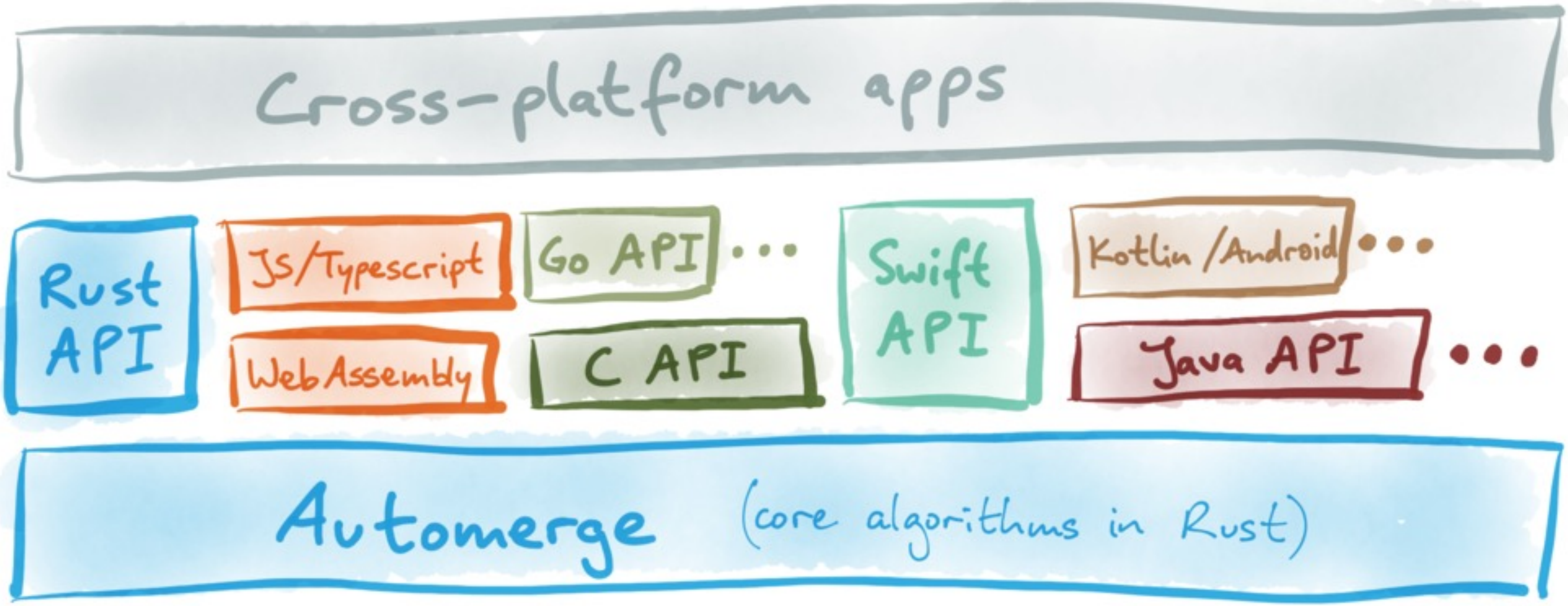

Rust API
Automerge (core algorithms in Rust)

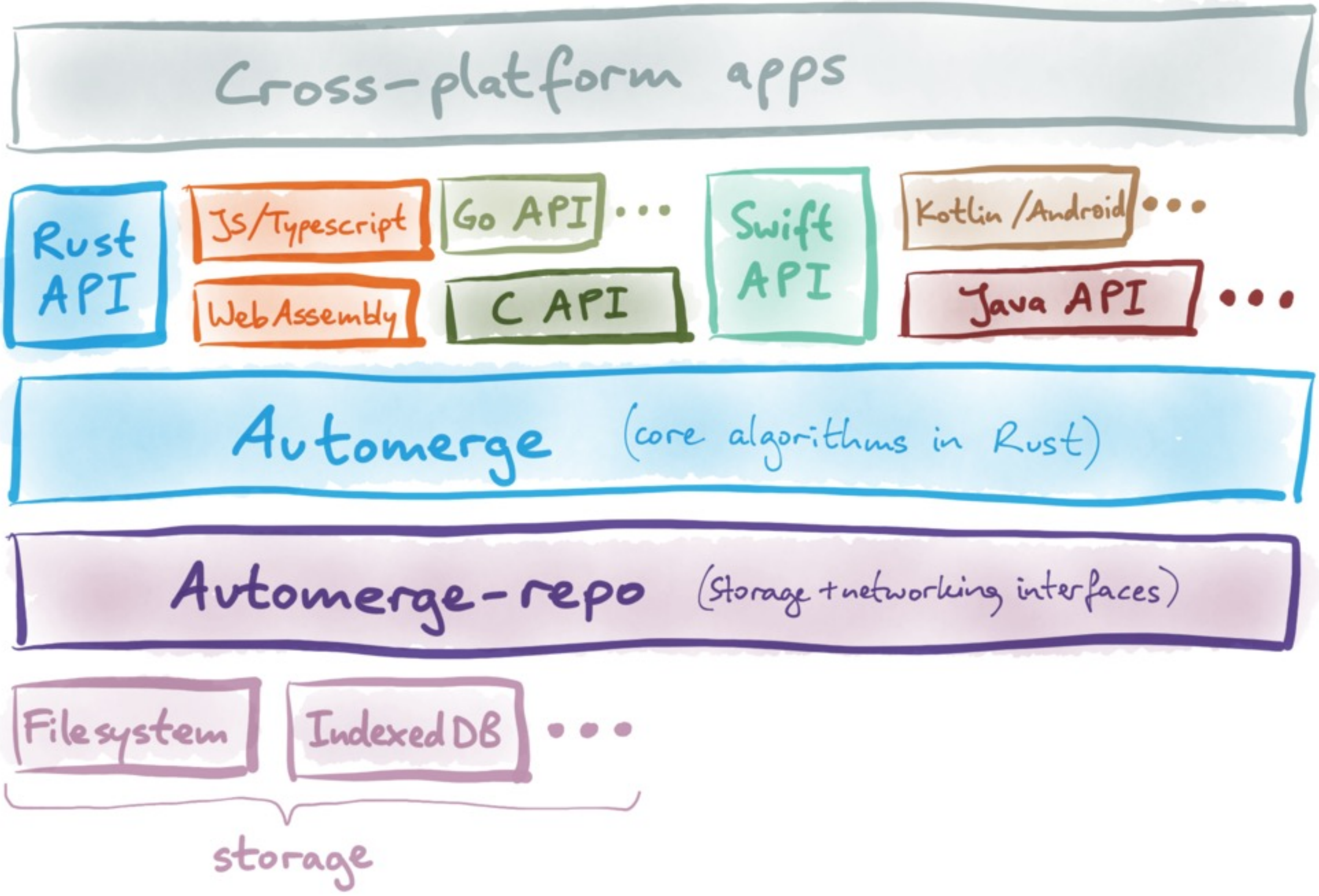

Rust API
JS/Typescript
WebAssembly
Automerge (core algorithms in Rust)

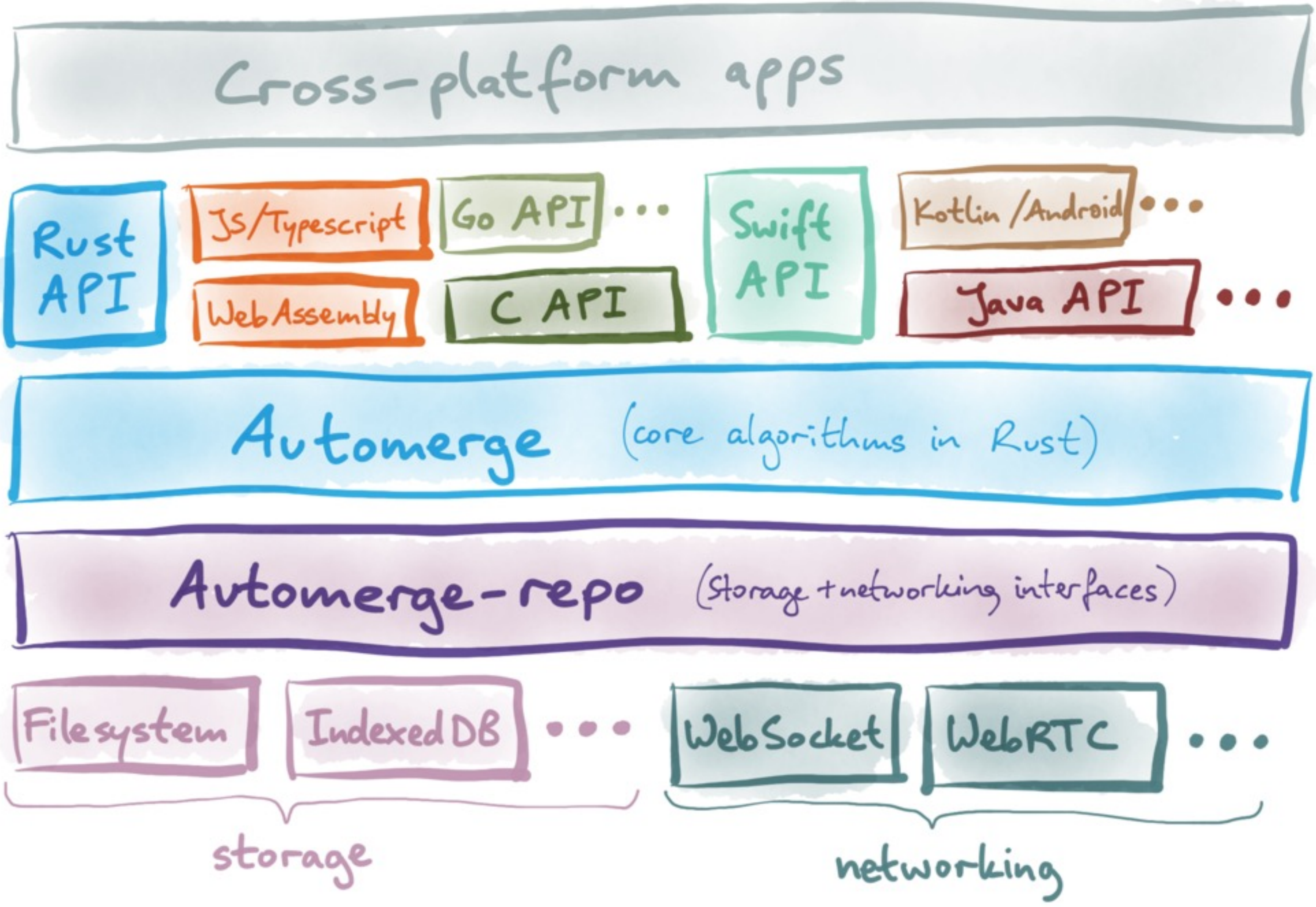

Rust API
JS/Typescript
Go API
...
WebAssembly
C API
Automerge (core algorithms in Rust)


Rust API
JS/Typescript
WebAssembly
Go API
...
C API
Swift API
Automerge (core algorithms in Rust)


Rust API
JS/Typescript
WebAssembly
Go API
C API
Swift API
Kotlin /Android
Java API
Automerge (core algorithms in Rust)


Cross-platform apps
Rust API
JS/Typescript
WebAssembly
Go API
...
C API
Swift API
Kotlin/Android
...
Java API
...
Automerge (core algorithms in Rust)


Cross-platform apps
Rust API
JS/Typescript
WebAssembly
Go API
C API
...
Swift API
Kotlin /Android
Java API
...
...
Automerge (core algorithms in Rust)
Automerge-repo (Storage + networking interfaces)
Filesystem
IndexedDB
...
storage


Cross-platform apps
Rust API
JS/Typescript
WebAssembly
Go API
C API
...
Swift API
Kotlin /Android
Java API
...
...
Automerge (core algorithms in Rust)
Automerge-repo (Storage + networking interfaces)
Filesystem
IndexedDB
...
WebSocket
WebRTC
...
storage
networking

A brief history of Automerge

Version 0.1–0.14 (2017–2020)

Research prototype in JavaScript

A brief history of Automerge

Version 0.1 – 0.14 (2017 – 2020)

Research prototype in JavaScript

Version 1.0 (2021)

Compressed data format, sync protocol

A brief history of Automerge

Version 0.1 – 0.14 (2017 – 2020)

Research prototype in JavaScript

Version 1.0 (2021)

Compressed data format, sync protocol

Version 2.0 (2022)

Moved to Rust, performance, production ready, commercial support

A brief history of Automerge

Version 0.1–0.14 (2017–2020)

Research prototype in JavaScript

Version 1.0 (2021)

Compressed data format, sync protocol

Version 2.0 (2022)

Moved to Rust, performance, production ready, commercial support

Version 2.1 (2023)

Rich text, automerge-repo, performance

Peritext

A CRDT for Rich-Text Collaboration

Ink & Switch

Geoffrey Litt

Slim Lim

Martin Kleppmann

Peter van Hardenberg

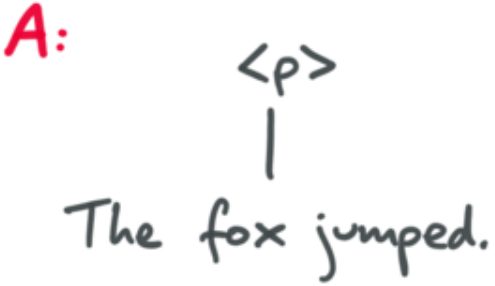
November 2021

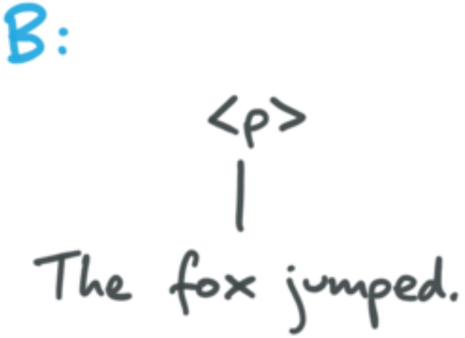
Collaborative editors like Google Docs allow people to work on a rich-text document in real-time, which is convenient when users want to immediately see each others' changes. However, sometimes people prefer a more *asynchronous* collaboration style, where they can work on a private copy of a document for a while and share their updates later. The algorithms underpinning services like Google Docs are not designed to support this use case.

In this article we present Peritext, an algorithm for rich-text collaboration that provides greater flexibility: it allows users to edit independent copies of a document, and it provides a mechanism for automatically merging those versions back together in a way that preserves the users' intent as much as possible. Once the versions are merged, the algorithm guarantees that all users converge towards the same merged result.

We provide a detailed analysis of various edge cases that need to be handled in collaborative rich-text editors, and explain why existing algorithms for plain text collaboration are not able to handle them correctly. We then explain how Peritext handles these issues, and demonstrate our prototype implementation of the algorithm.

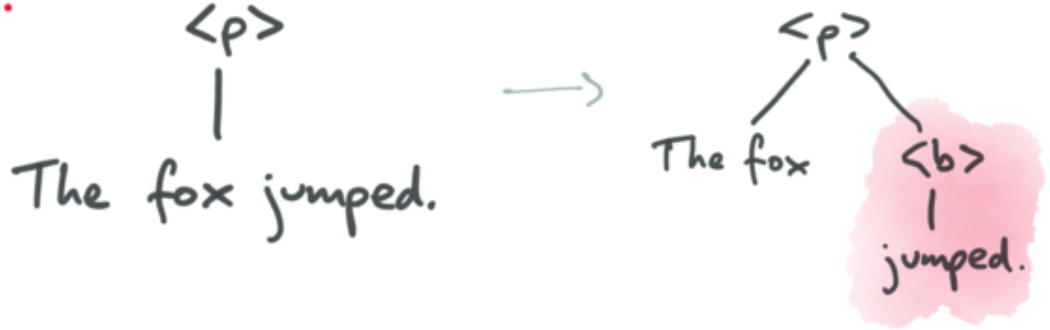
Rich text as a tree?



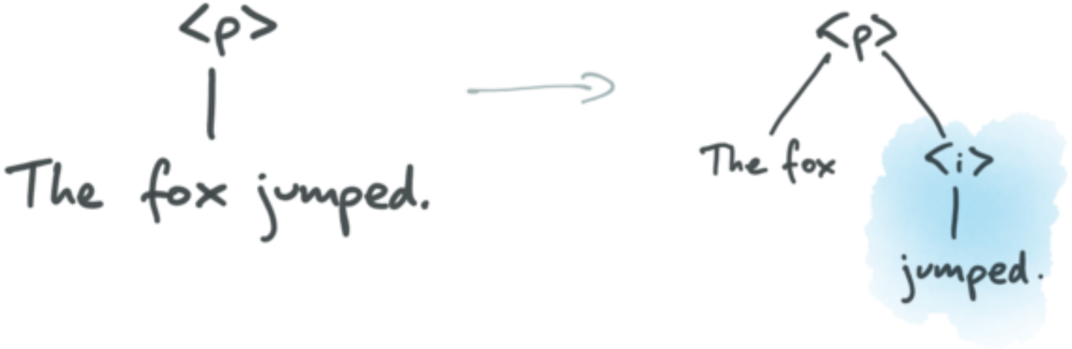


Rich text as a tree?

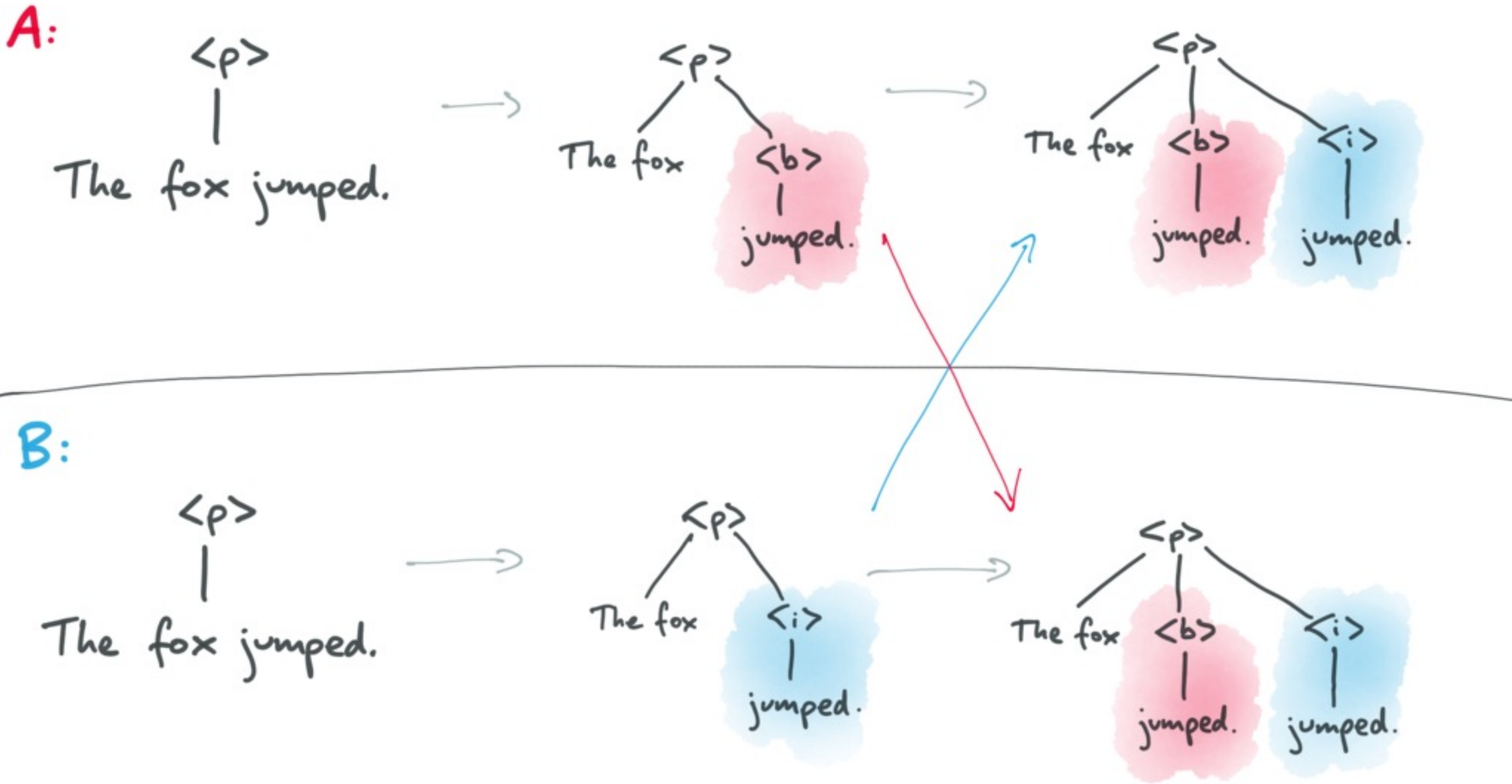
A:



B:



Rich text as a tree?



Rich text as markup?

A:

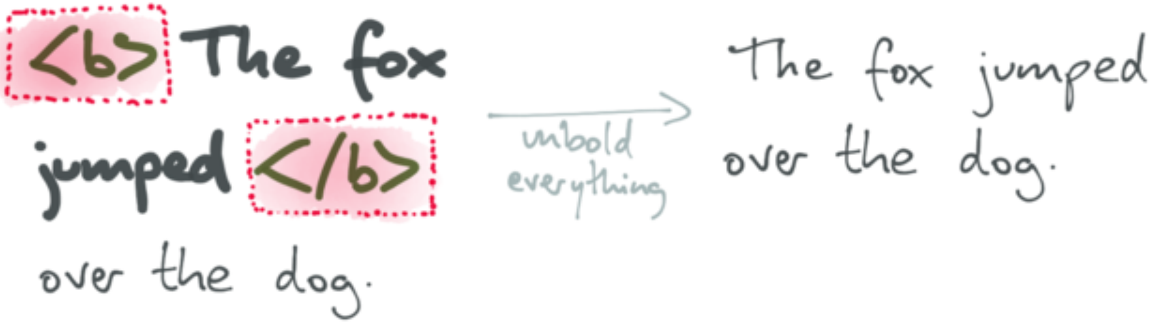
```
<b> The fox
jumped </b>
over the dog.
```

B:

```
<b> The fox
jumped </b>
over the dog.
```

Rich text as markup?

A:

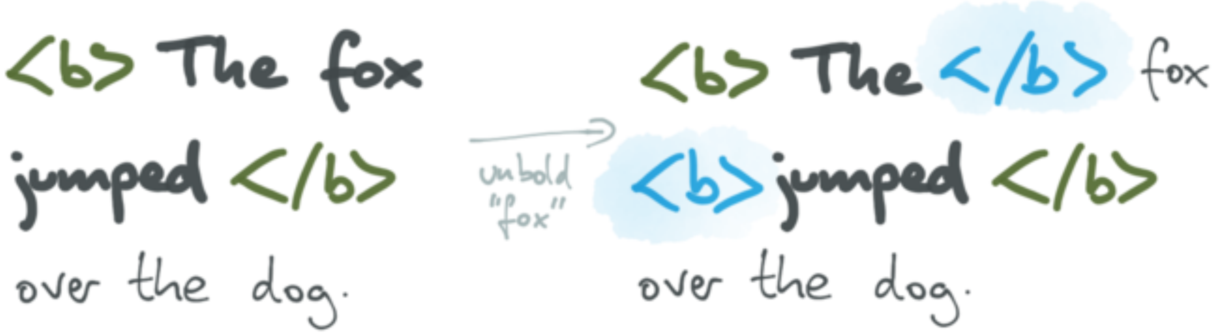


<b> The fox
jumped </b>
over the dog.

unbold everything

The fox jumped
over the dog.

B:

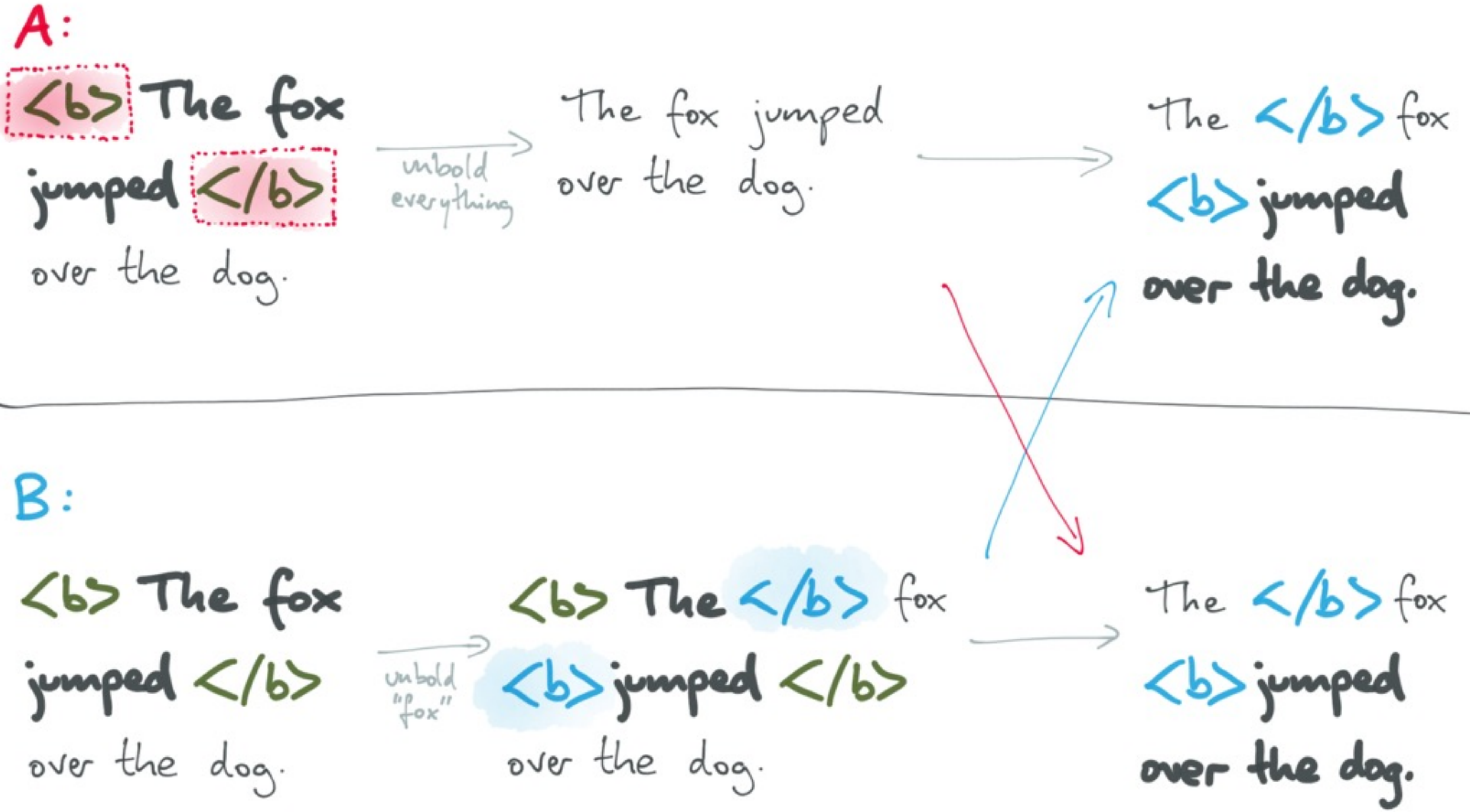


<b> The fox
jumped </b>
over the dog.

unbold "fox"

<b> The </b> fox
<b> jumped </b>
over the dog.

Rich text as markup?



A:

<b> The fox jumped </b> over the dog.

unbold everything

The fox jumped over the dog.

The </b> fox <b> jumped over the dog.

B:

<b> The fox jumped </b> over the dog.

unbold "fox"

<b> The </b> fox <b> jumped </b> over the dog.

The </b> fox <b> jumped over the dog.

Rich text as per-character properties?

A:

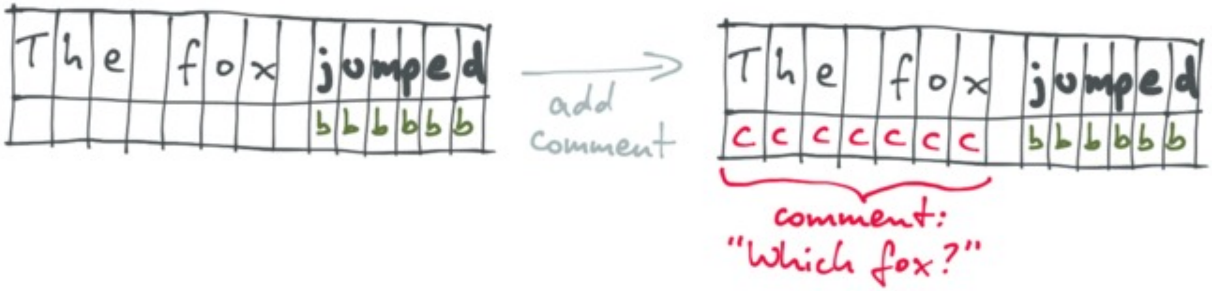
T	h	e		f	o	x		**j**	**u**	**m**	**p**	**e**	**d**
								b	b	b	b	b	b

B:

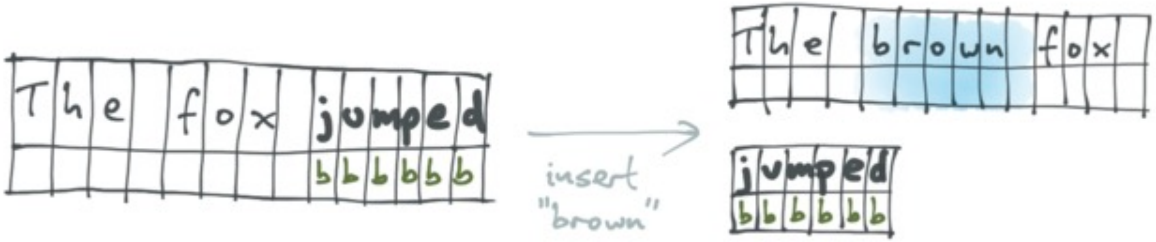
T	h	e		f	o	x		**j**	**u**	**m**	**p**	**e**	**d**
								b	b	b	b	b	b

Rich text as per-character properties?

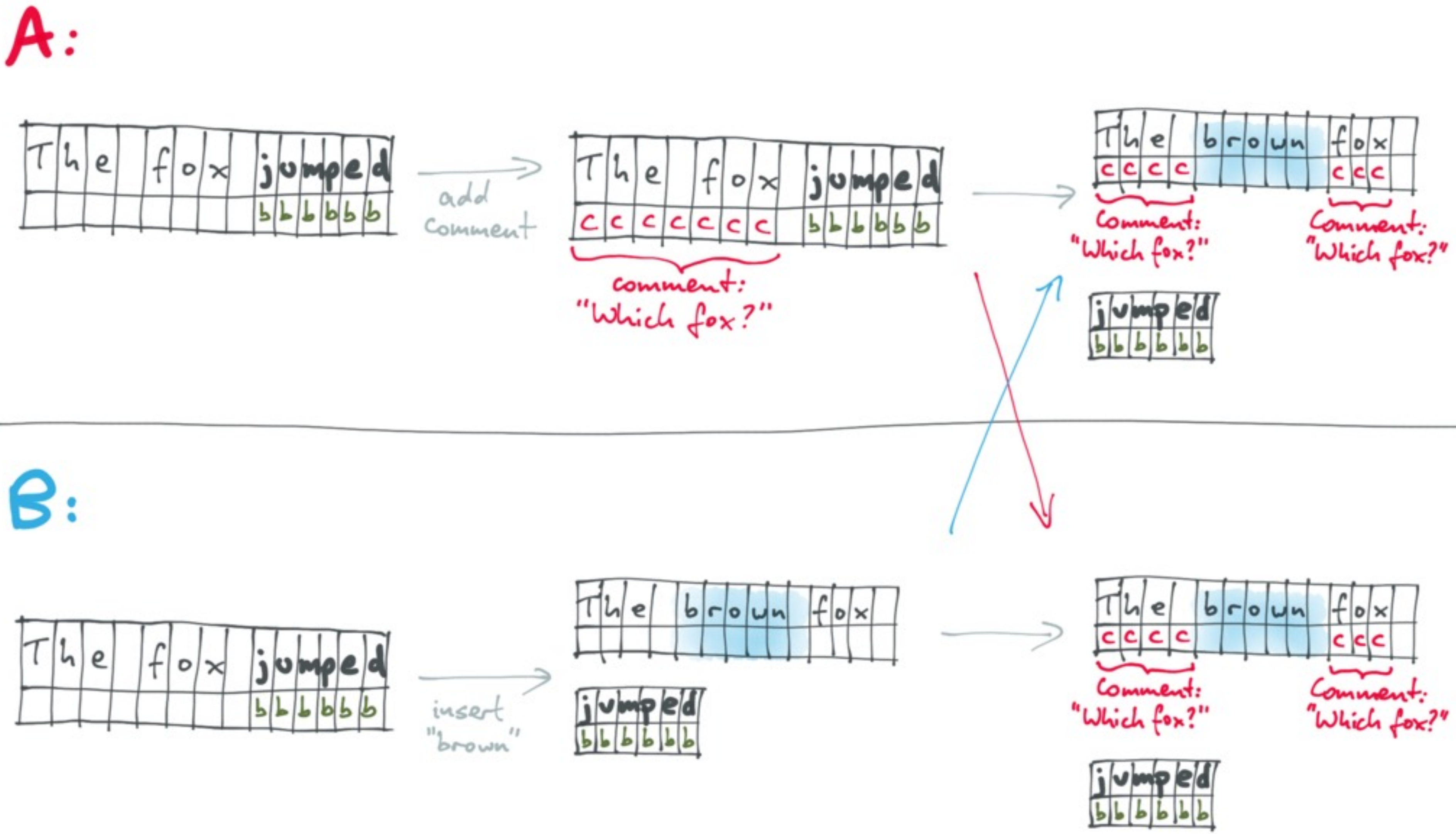
A:



B:



Rich text as per-character properties?



Rich text as text with annotation spans

A:

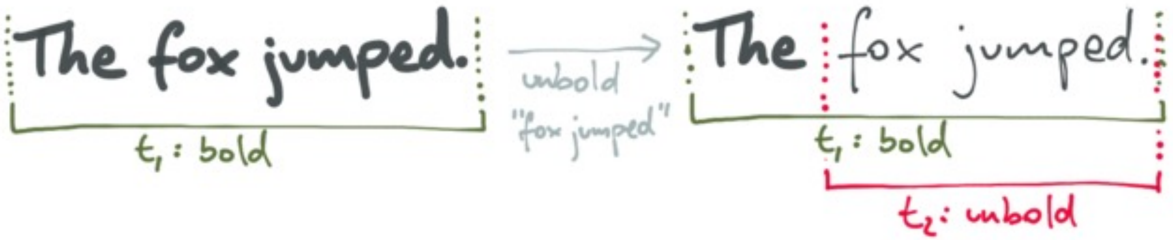
$$\underbrace{\text{The fox jumped.}}_{t_1 : \text{bold}}$$

B:

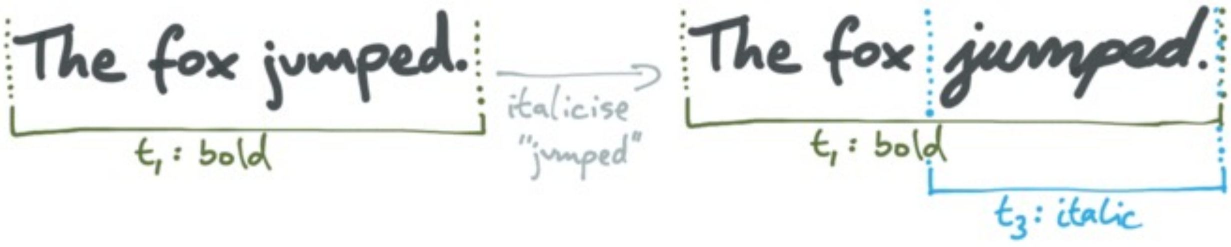
$$\underbrace{\text{The fox jumped.}}_{t_1 : \text{bold}}$$

Rich text as text with annotation spans

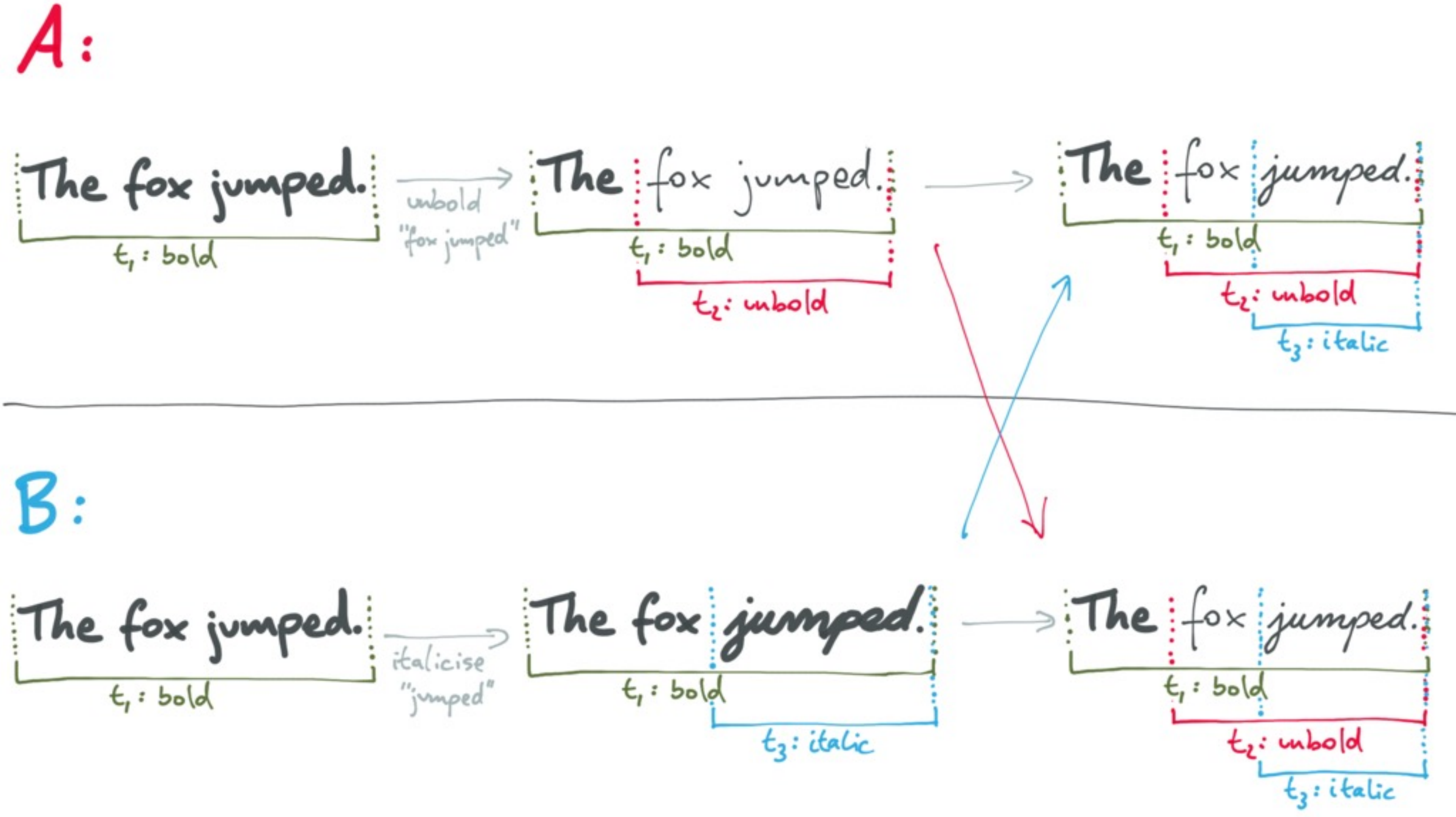
A:

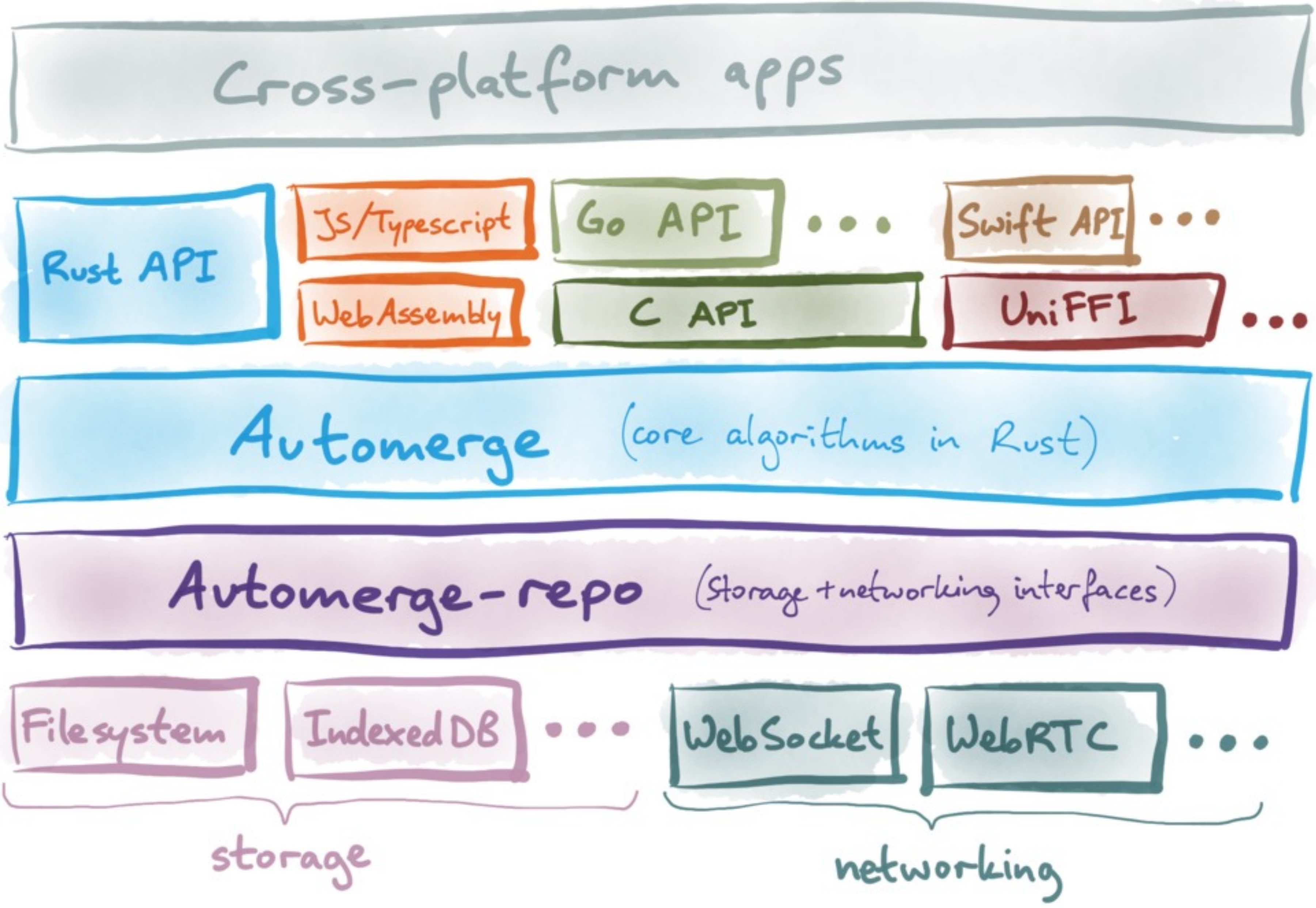


B:



Rich text as text with annotation spans




Cross-platform apps
Rust API
JS/Typescript
WebAssembly
Go API
...
C API
Swift API
...
UniFFI
...
Automerge (core algorithms in Rust)
Automerge-repo (Storage + networking interfaces)
Filesystem
IndexedDB
...
WebSocket
WebRTC
...
storage
networking

Resources

Automerge	https://automerge.org/
My work	https://martin.kleppmann.com/
Email	martin@kleppmann.com
Twitter	@martinkl
Bluesky	@martinkl.com
Mastodon	@martin@nondeterministic.computer

Huge thanks to the Automerge community and contributors, especially Alex Good, Peter van Hardenberg, Orion Henry, Andrew Jeffery, Herb Caudill, Alex Currie-Clark, Jason Kankiewicz, Conrad Irwin, and many others!

Thank you to my Patreon supporters and institutional funders:

VolkswagenStiftung

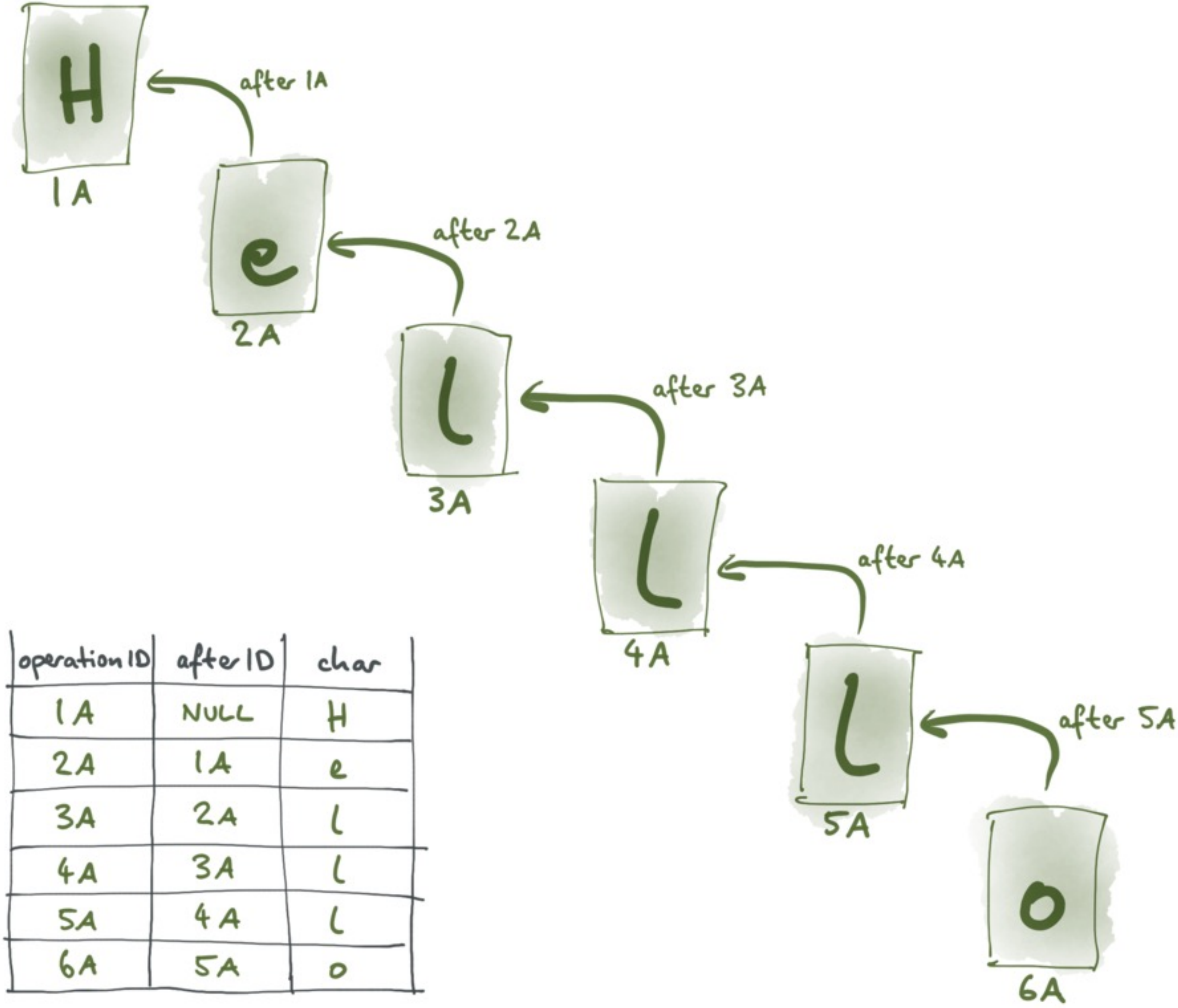
LEVERHULME TRUST

Isaac Newton Trust

Ink & Switch

NOKIA Bell Labs

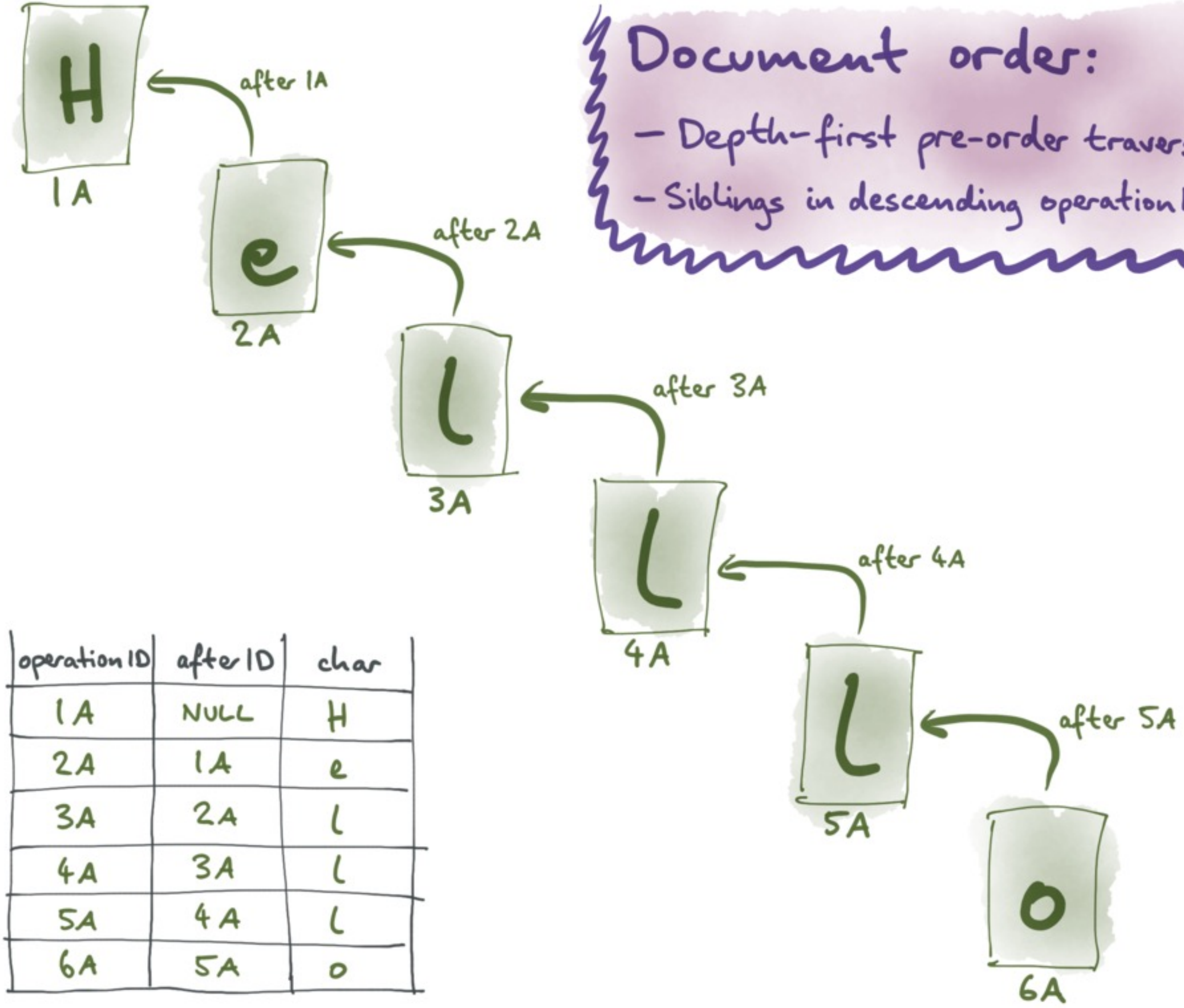
HOW AUTOMERGE WORKS



operation ID	after ID	char
1A	NULL	H
2A	1A	e
3A	2A	l
4A	3A	l
5A	4A	l
6A	5A	o

Document order:

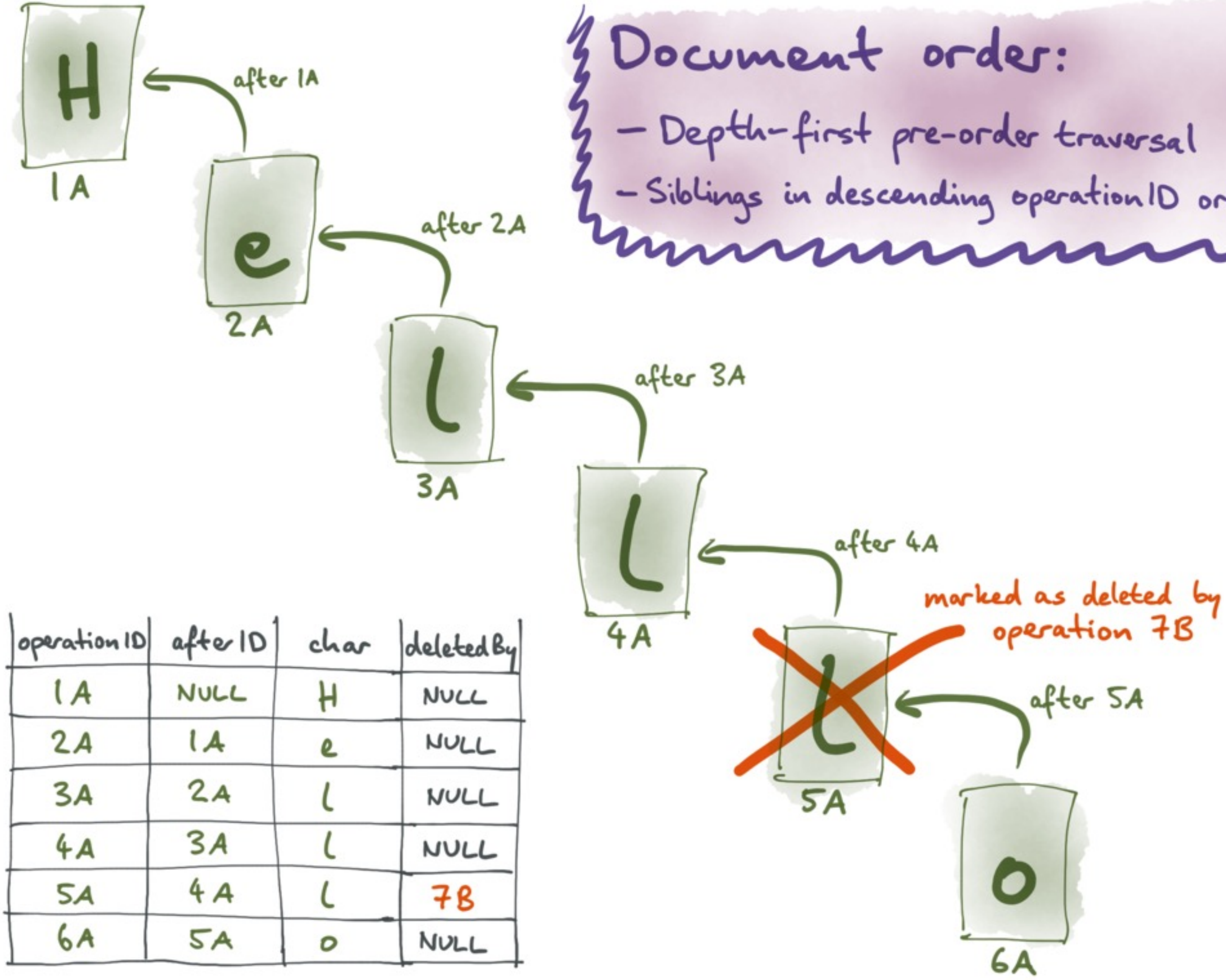
- Depth-first pre-order traversal
- Siblings in descending operationID order



operationID	afterID	char
1A	NULL	H
2A	1A	e
3A	2A	l
4A	3A	l
5A	4A	l
6A	5A	o

Document order:

- Depth-first pre-order traversal
- Siblings in descending operationID order



operationID	afterID	char	deletedBy
1A	NULL	H	NULL
2A	1A	e	NULL
3A	2A	l	NULL
4A	3A	l	NULL
5A	4A	l	7B
6A	5A	o	NULL

COLUMNAR ENCODING (simplified)

operation ID		reference element ID		inserted character		deleted by op ID	
counter	actor	counter	actor	length	UTF-8	counter	actor
1	A	—	—	1	"H"	—	—
2	A	1	A	1	"e"	—	—
3	A	2	A	1	"l"	—	—
4	A	3	A	1	"l"	—	—
5	A	4	A	1	"l"	7	B
6	A	5	A	1	"o"	—	—

COLUMNAR ENCODING (simplified)

operation ID		reference element ID		inserted character		deleted by op ID	
counter	actor	counter	actor	length	UTF-8	counter	actor
1	A	—	—	1	"H"	—	—
2	A	1	A	1	"e"	—	—
3	A	2	A	1	"l"	—	—
4	A	3	A	1	"l"	—	—
5	A	4	A	1	"l"	7	B
6	A	5	A	1	"o"	—	—

→ 1, 2, 3, 4, 5, 6

delta-encode to 1, 1, 1, 1, 1, 1

run-length encode to (6, 1)

LEB128 encodes this in 2 bytes

COLUMNAR ENCODING (simplified)

operation ID		reference element ID		inserted character		deleted by op ID	
counter	actor	counter	actor	length	UTF-8	counter	actor
1	A	—	—	1	"H"	—	—
2	A	1	A	1	"e"	—	—
3	A	2	A	1	"l"	—	—
4	A	3	A	1	"l"	—	—
5	A	4	A	1	"l"	7	B
6	A	5	A	1	"o"	—	—

→ make a lookup table: {"A": 0, "B": 1}

→ 0, 0, 0, 0, 0, 0

→ run-length encode to (6, 0)

→ LEB128 encodes in 2 bytes

COLUMNAR ENCODING (simplified)

operation ID		reference element ID		inserted character		deleted by op ID	
counter	actor	counter	actor	length	UTF-8	counter	actor
1	A	—	—	1	"H"	—	—
2	A	1	A	1	"e"	—	—
3	A	2	A	1	"l"	—	—
4	A	3	A	1	"l"	—	—
5	A	4	A	1	"l"	7	B
6	A	5	A	1	"o"	—	—

just concatenate the UTF-8 byte sequences → "Hellllo" (6 bytes)
(use length column to separate again)

Automerge compression benchmark

Test document: keystroke-by-keystroke editing trace of a text file (LaTeX source of a research paper)

- 182,315 single-character insertions
- 77,463 single-character deletions

Final text (uncompressed, no edit history): 105 kB

Automerge compression benchmark

Test document: keystroke-by-keystroke editing trace of a text file (LaTeX source of a research paper)

- 182,315 single-character insertions
- 77,463 single-character deletions

Final text (uncompressed, no edit history): 105 kB

Full edit history (columnar compression): 184 kB

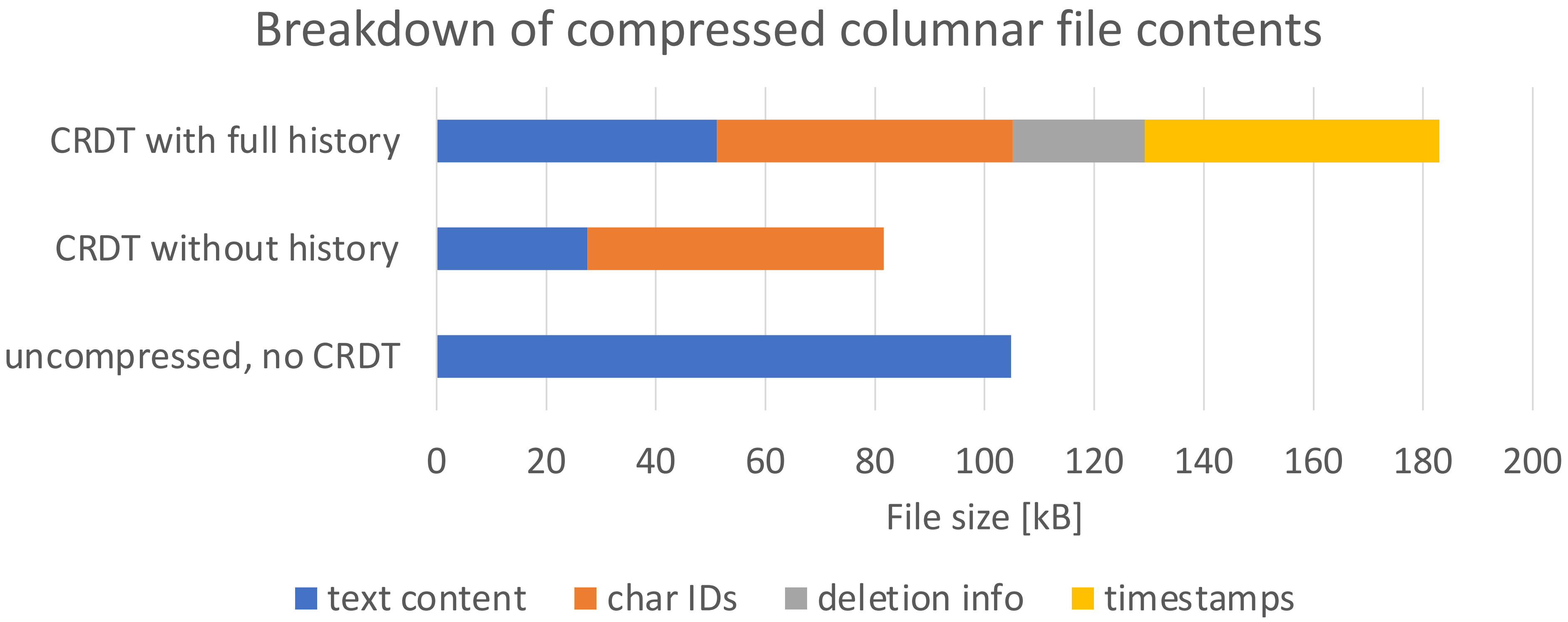
(0.7 bytes/operation!)

Can look at any past document version, diffing, branching, merging ...

Automerge compression benchmark

Benchmark data: keystroke-by-keystroke editing trace of a text file (LaTeX source of a research paper) containing 182,315 single-character insertions and 77,463 single-character deletions, timestamped with 1-second granularity.

As individual changes: 33.7 MB (130 bytes/operation)
As compressed document with full edit history: 184 kB (0.7 bytes/operation)


Breakdown of compressed columnar file contents
CRDT with full history
CRDT without history
uncompressed, no CRDT
File size [kB]
text content
char IDs
deletion info
timestamps