Building fast web applications

Thierry Sans

Users respond to speed

"Amazon found every 100ms of latency cost them 1% in sales"

"Google found an extra .5 seconds in search page generation time dropped traffic by 20%"

"A broker could lose $4 million in revenues per millisecond if their electronic trading platform is 5 milliseconds behind the competition"

http://blog.gigaspaces.com/amazon-found-every-100ms-of-latency-cost-them-1-in-sales/

Several Techniques

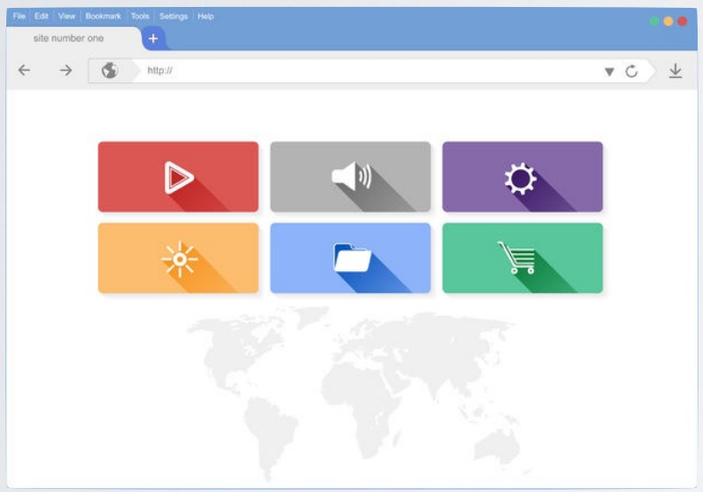
- Backend templates
- Frontend packing
- HTTP2
- Long polling

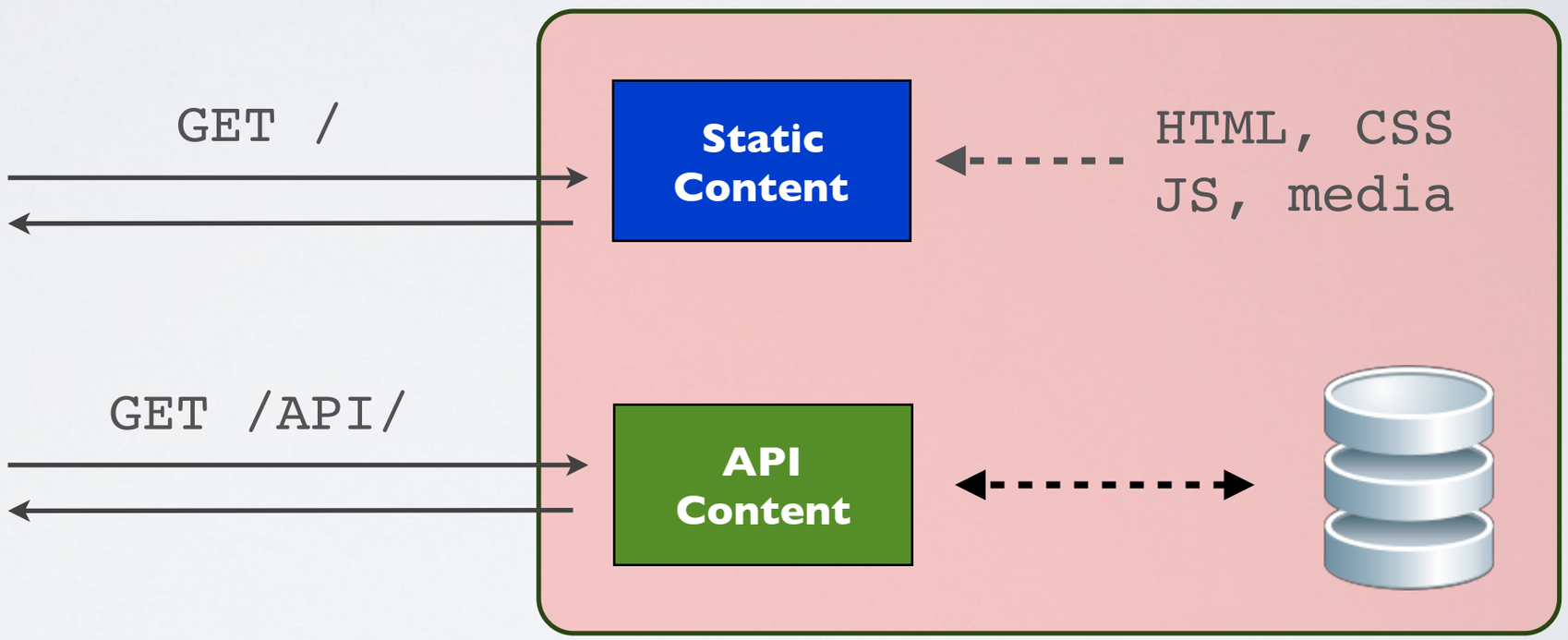
Backend Templates

Our application so far

Frontend

Backend

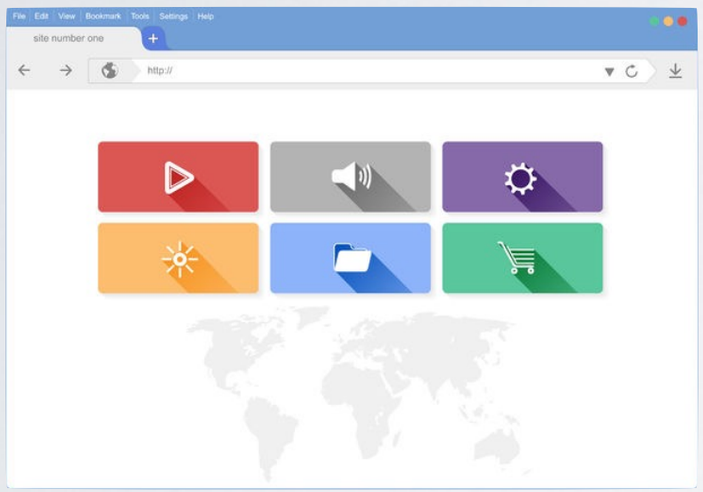




Dynamic content (using HTML templates)

Frontend

Backend



GET /

Static Content ◀--- CSS, JS, media

Dynamic content

HTML templates

API Content

Advantages of using templates in the backend

- Better code reuse and maintenance
- Faster loading time (avoid unnecessary ajax requests)

Better code reuse and maintenance

Some pages might share

- headers (title, JS and CSS)
- page organization (`div` tags structuring the page)
- footers (if any)

Faster loading time

- Dynamic pages are built on the server and can be retrieved with 1 HTTP requests (instead of 2 with the ajax API call)

- Dynamic pages can be cached on the server

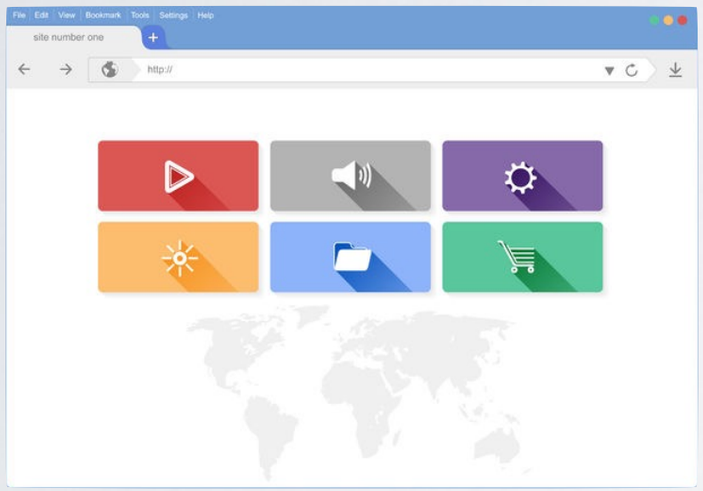
Template engines compatible with Express

- Pug
- Moustache
- EJS
- Jade
- ... and others https://expressjs.com/en/guide/using-template-engines.html

Frontend packing

The problem

Frontend



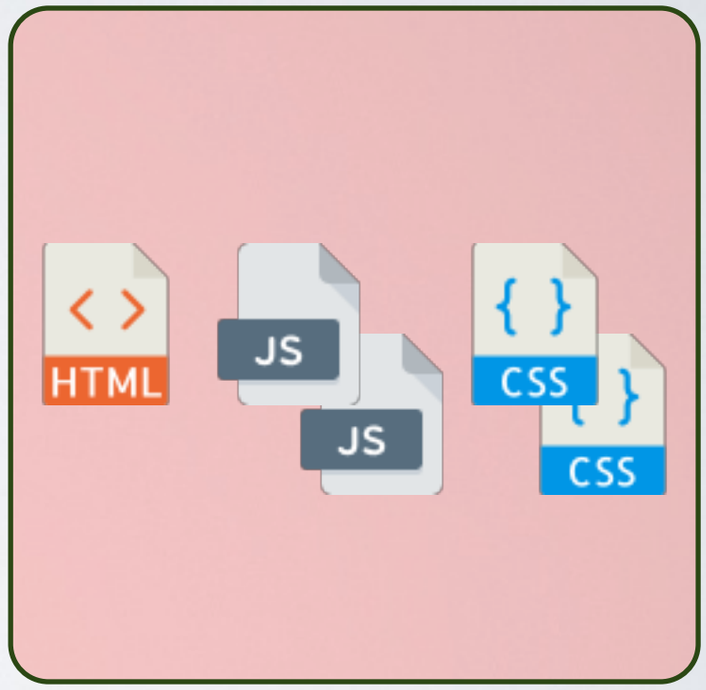
GET /

GET /js/lib.js

GET /js/index.js

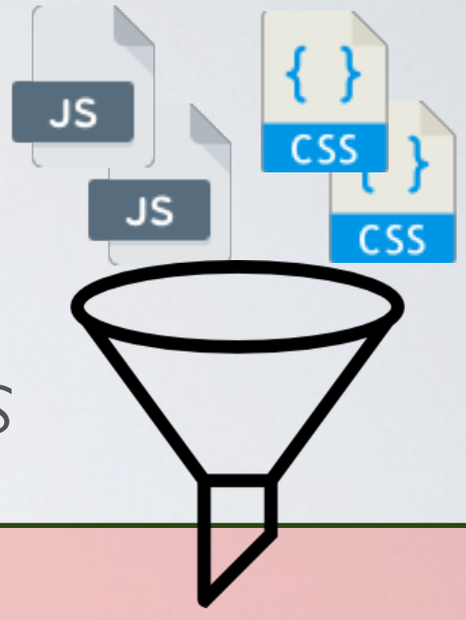
GET /style/index.css

GET /style/generic.css



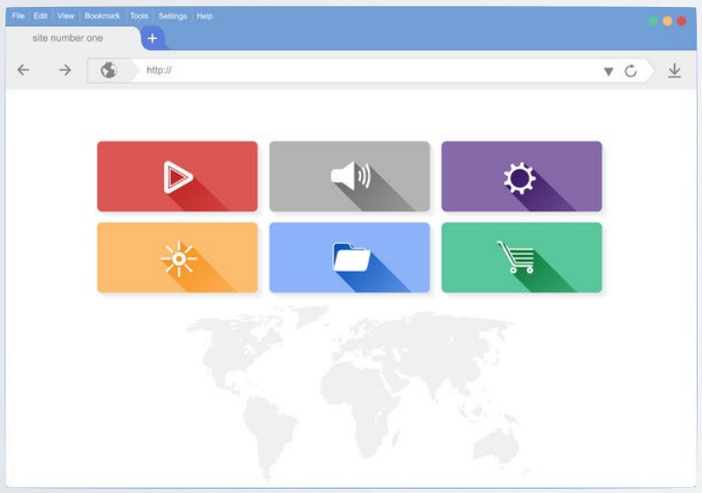
Backend

The solution - using a frontend packer



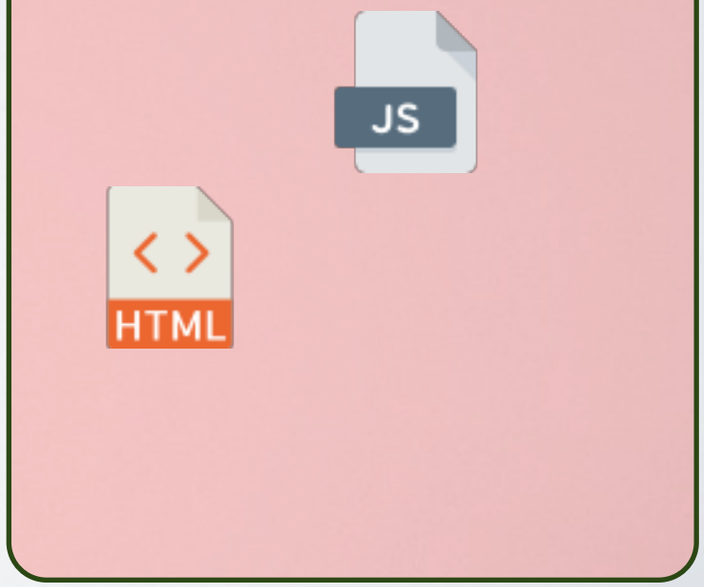
e.g. *webpack.js*

Frontend




```
GET /
```

```
GET /js/bundle.js
```



Backend

HTTP 2

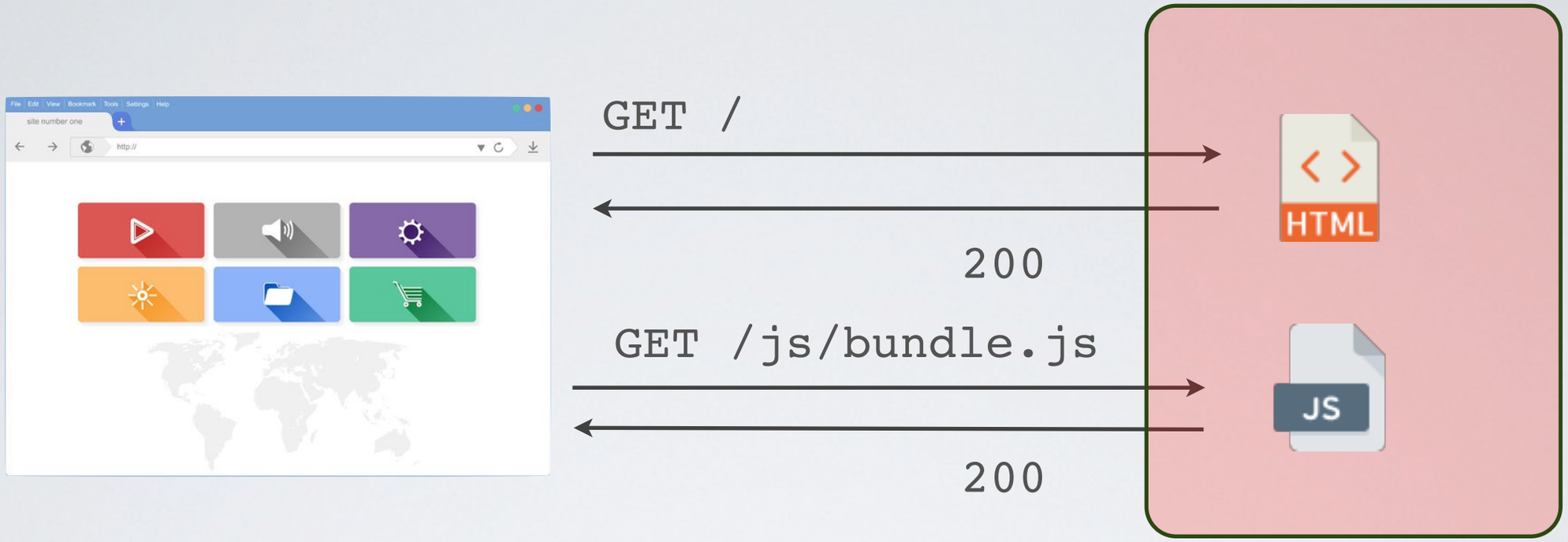
HTTP/2

HTTP/2 enables multiplexing

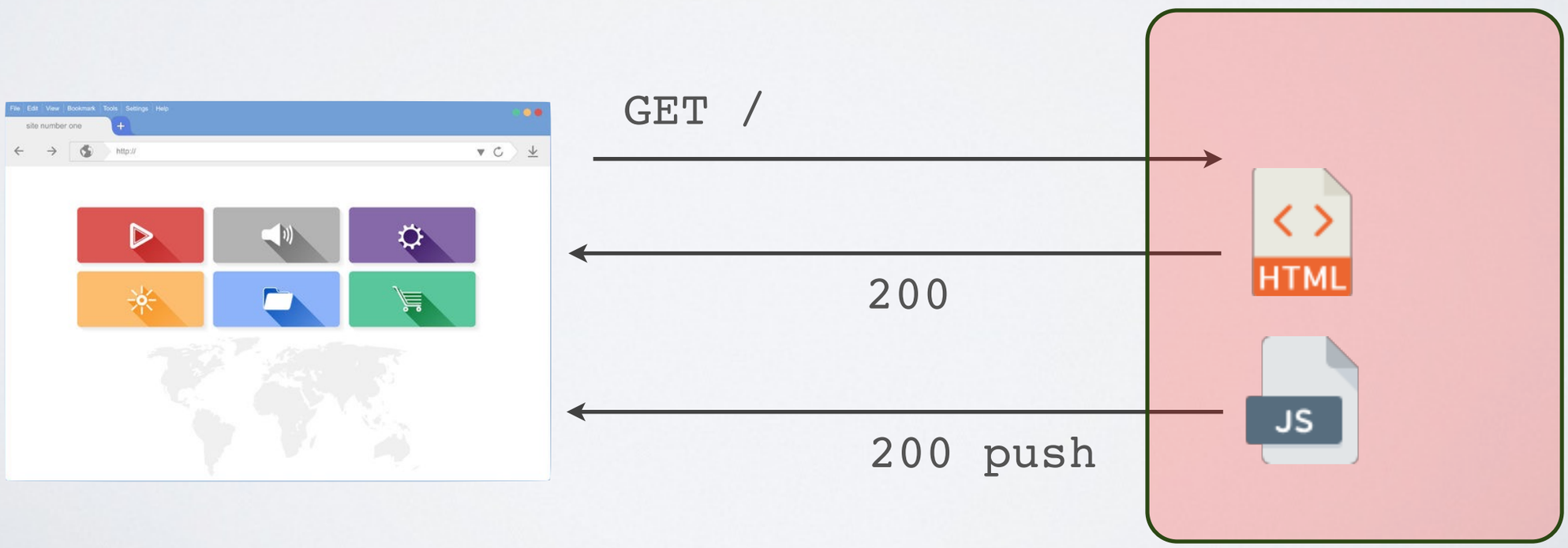
➡ send multiple HTTP responses for a given request (a.ka `push`)

- Proposed by Google (called SPDY)
- Adopted as an standard in 2015 (RFC 7540)
- HTTP/2 is compatible with HTTP/1 (same protocol)

HTTP 1.1



HTTP 2.0



Long Polling

Short Polling vs Long Polling

Short Polling

- The frontend request an update from the backend every few seconds
- The backend replies right away regardless if there is an update or not

◉ Many request/responses are wasted

Long Polling

- The frontend request an update from the backend and wait for the response
- The backend replies to the update request only when there is an update

✓ No request/response wasted

✓ Updates are processed as soon as they arrived

Long Polling