



JavaScript Days 2019, Berlin

What's wrong with Single-Page Apps (and what to do about it)

Stefan Tilkov

stefan.tilkov@innoq.com

@stilkov

INNOQ

Annoying your app users in 10 easy steps

1.

*Forbid the use of the back
and forward buttons*

2.

*Send them to the home
page when they hit
“refresh” ...*

3.

*... or at least ensure the
browser pops up a
warning window*

4.

*Make sure they can't open
a second browser window*

5.

*Let them see UI decoration
and ads first, content last*

6.

*Make sure they can't
bookmark or send a link*

7.

*Don't let Google index
anything*

8.

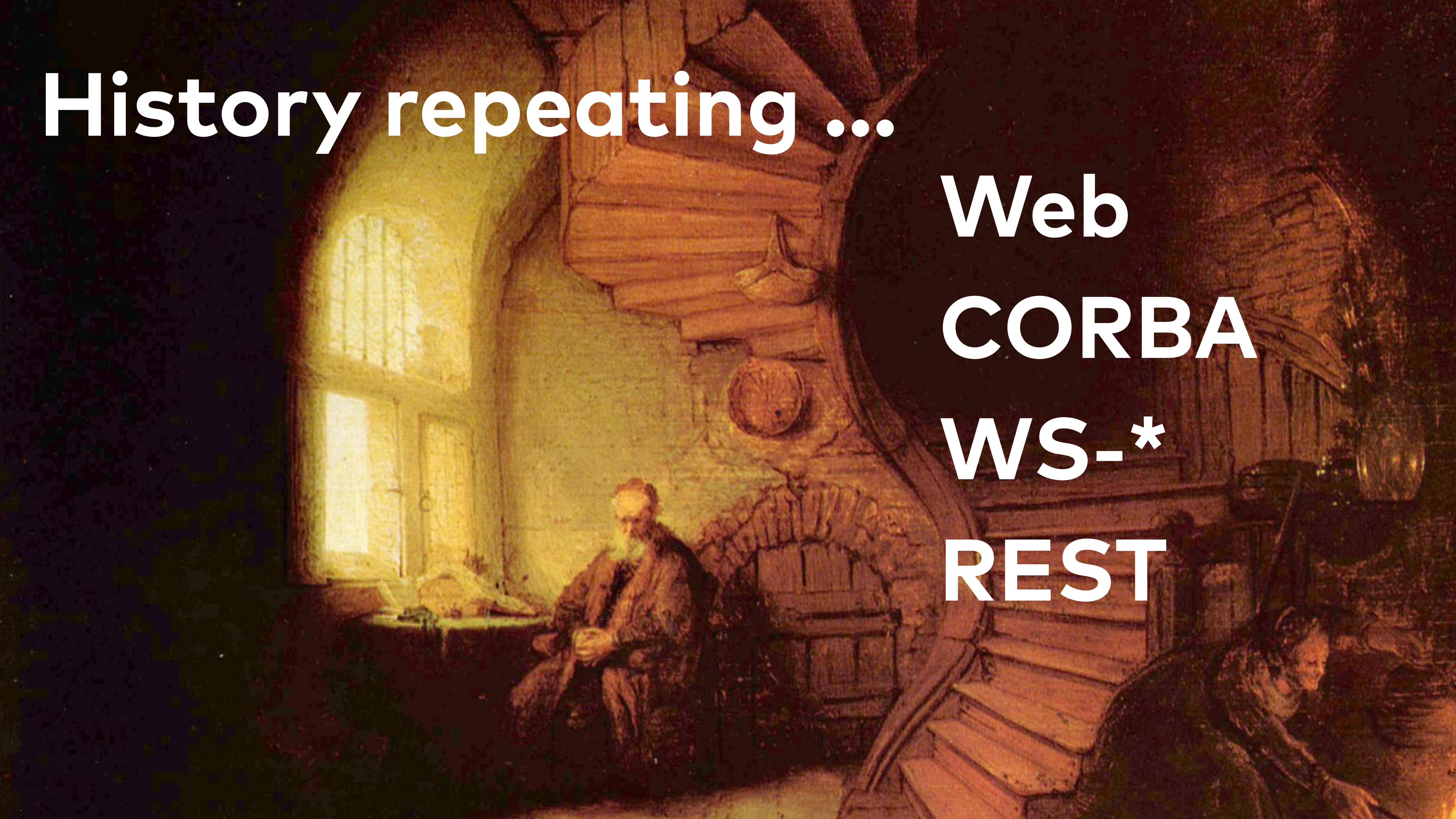
*Show users a picture of
your app – it's surely
better than nothing*

9.

*Disable assistive
technologies. Who needs a
screen reader, anyway?*

10.

*Ensure non-functioning
JavaScript gives them a
blank page*

A painting by Gustave Doré depicting a man in a dark, medieval-style setting. He is seated at a desk, illuminated by the light from a large, open book or manuscript he is holding. The scene is dimly lit, with light also coming from a window in the background and a small fire on the desk. The overall atmosphere is one of mystery and historical significance.

History repeating ...

Web
CORBA
WS-*
REST

What's the client side analogy?

“Web service”¹⁾

- > Uses HTTP as transport
- > Ignores HTTP verbs
- > Ignores URLs
- > Exposes single “endpoint”
- > Fails to embrace the Web

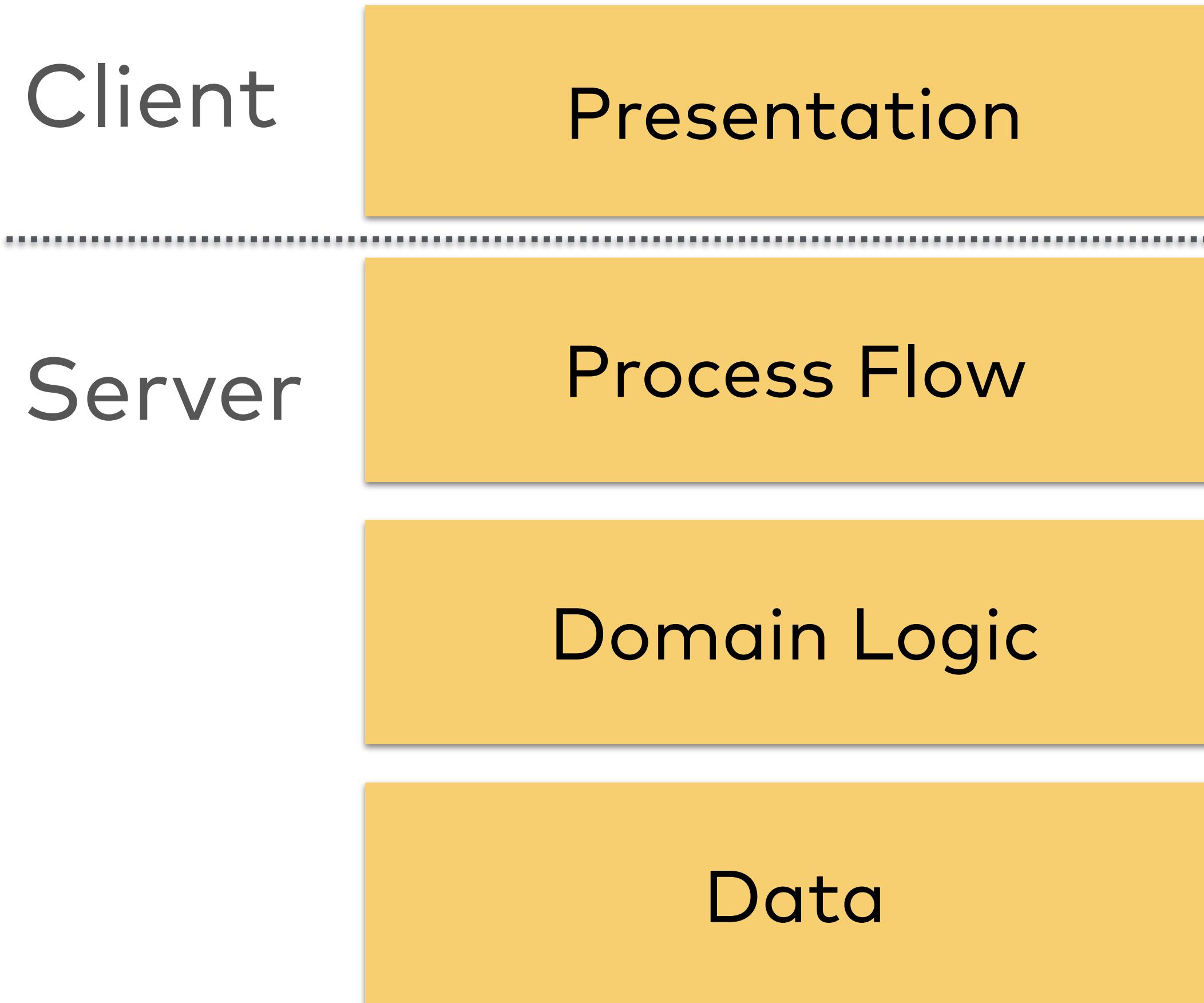
“Web app”²⁾

- > Uses browser as runtime
- > Ignores forward, back, refresh
- > Does not support linking
- > Exposes monolithic “app”
- > Fails to embrace the browser

¹⁾ in the SOAP/WSDL sense

²⁾ built as a careless SPA

The web-native way of distributing logic



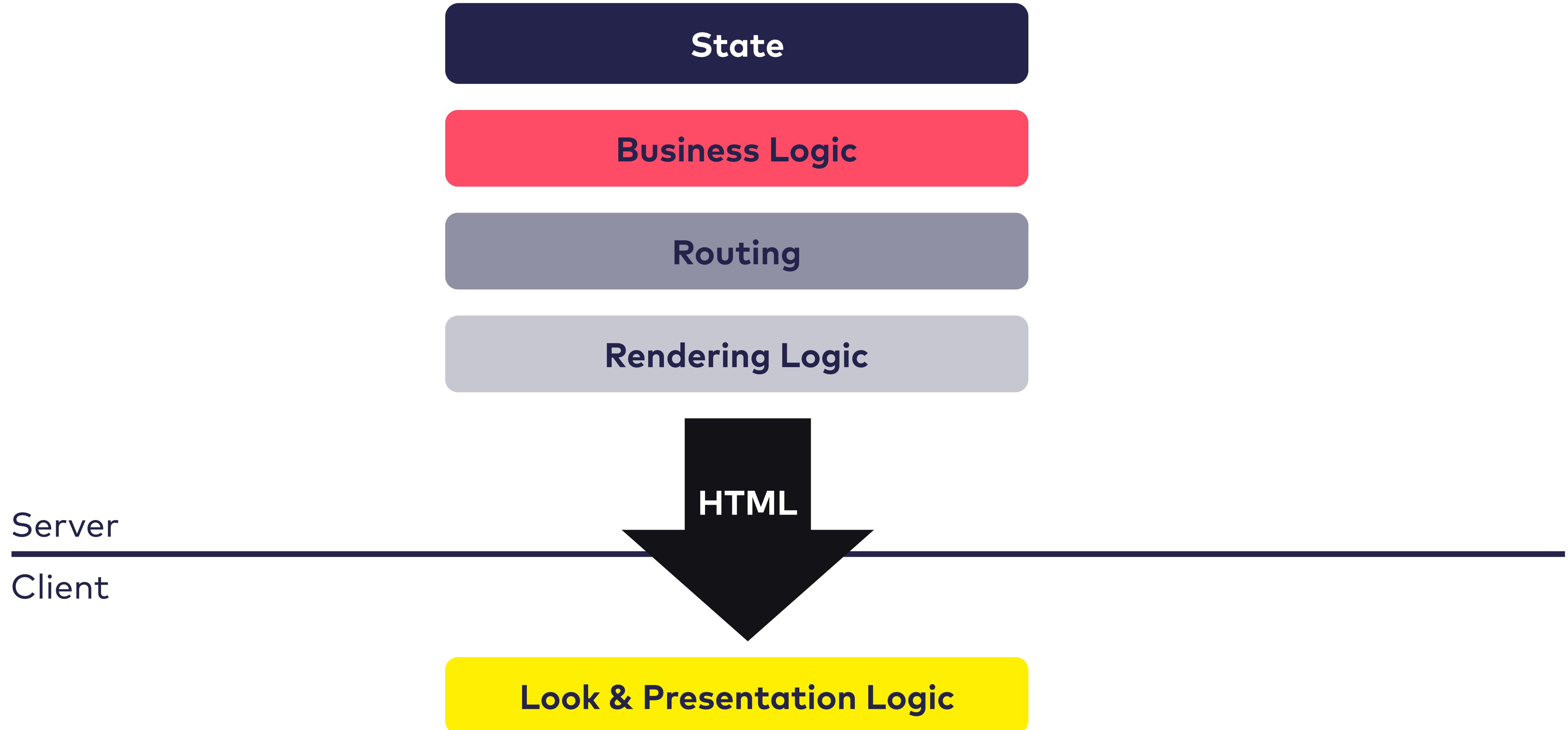
- > Rendering, layout, styling on an unknown client
- > Logic & state machine on server
- > Client user-agent extensible via code on demand

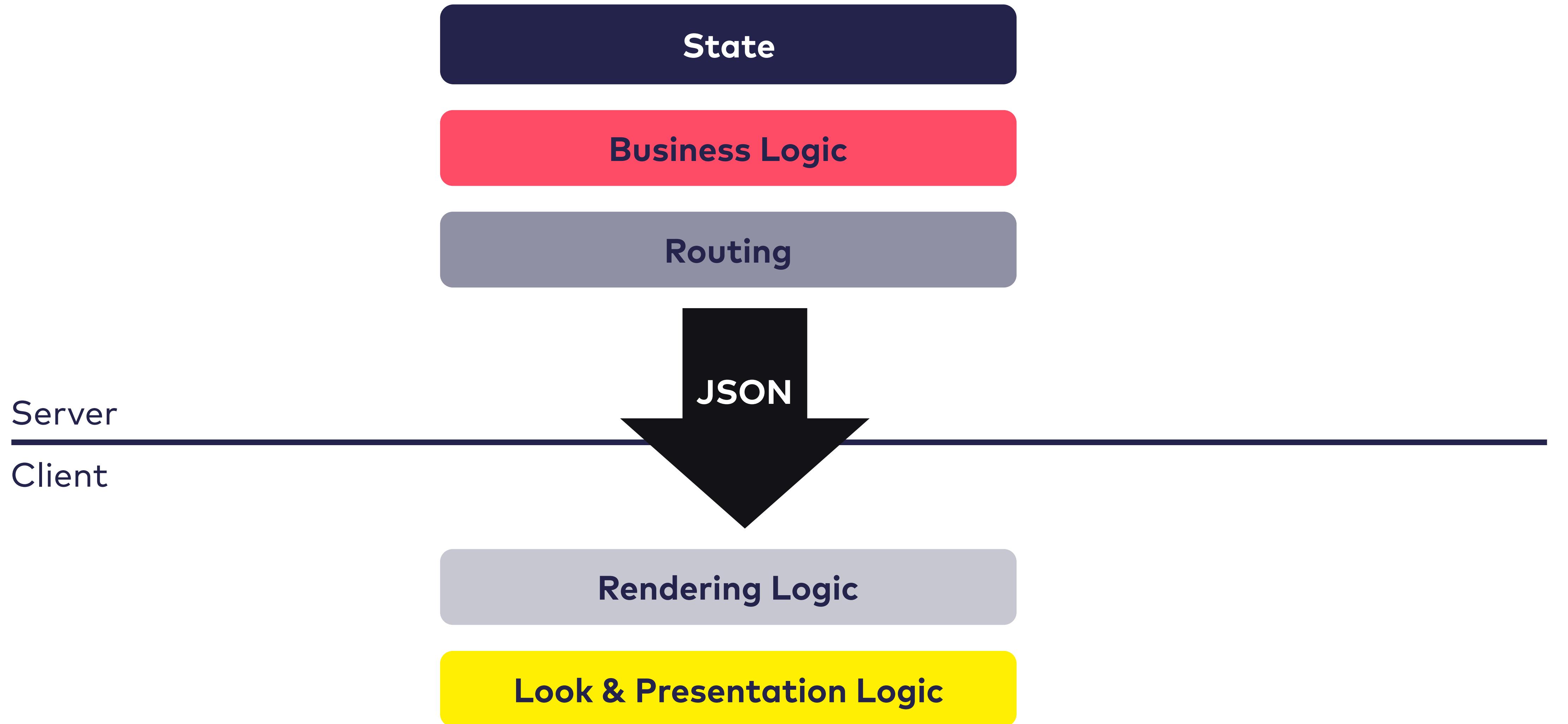
HTML & Hypermedia

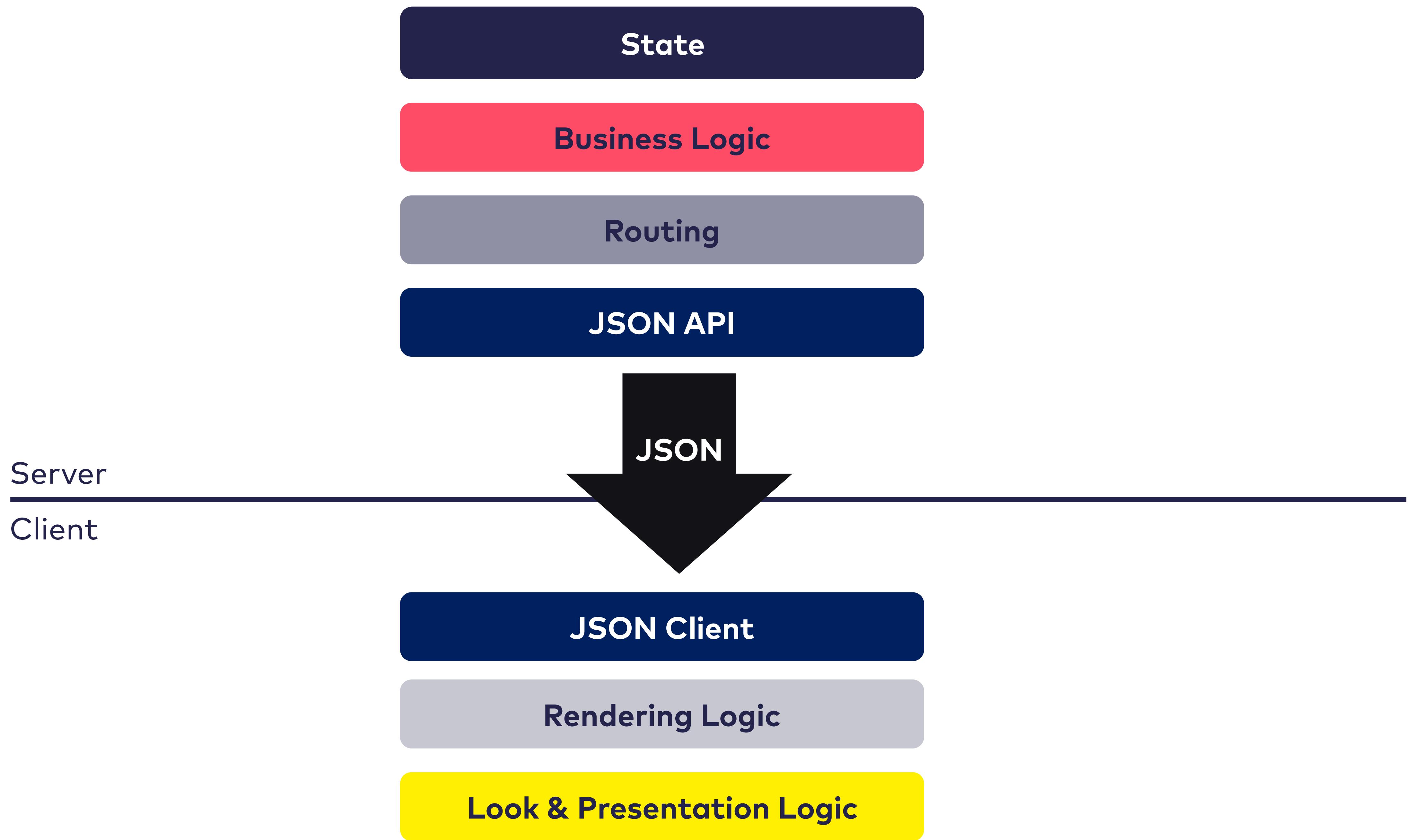
- In REST, servers expose a hypermedia format
 - Option 1: Just invent your own JSON-based, incomplete clone
 - Option 2: Just use HTML
- Clients need to be RESTful, too
 - Option 1: Invent your own, JS-based, buggy, incomplete implementation
 - Option 2: Use the browser

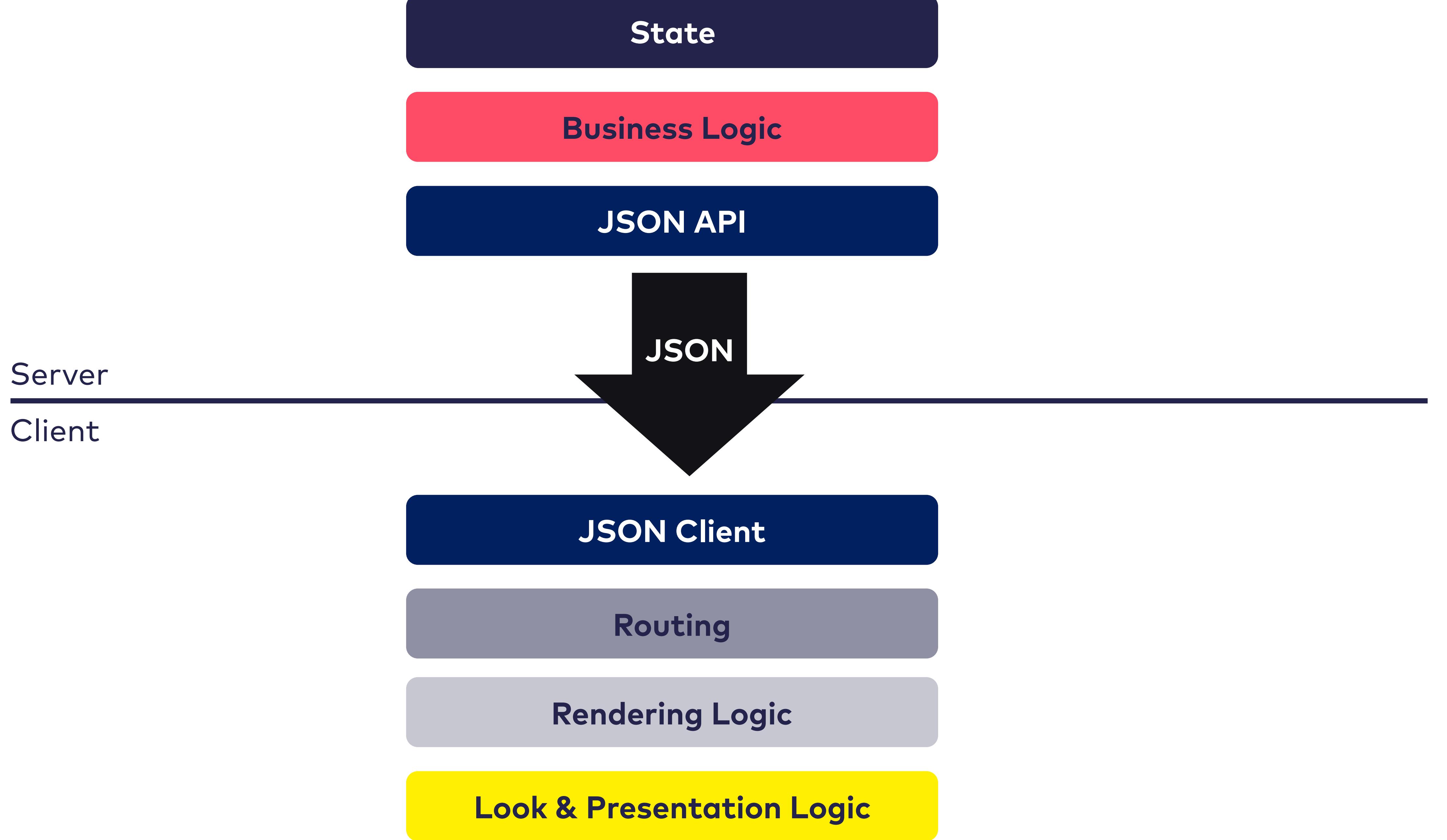
**A great REST hypermedia API is very similar to
a simple, server-sided rendered web application**

What does SPA even mean?

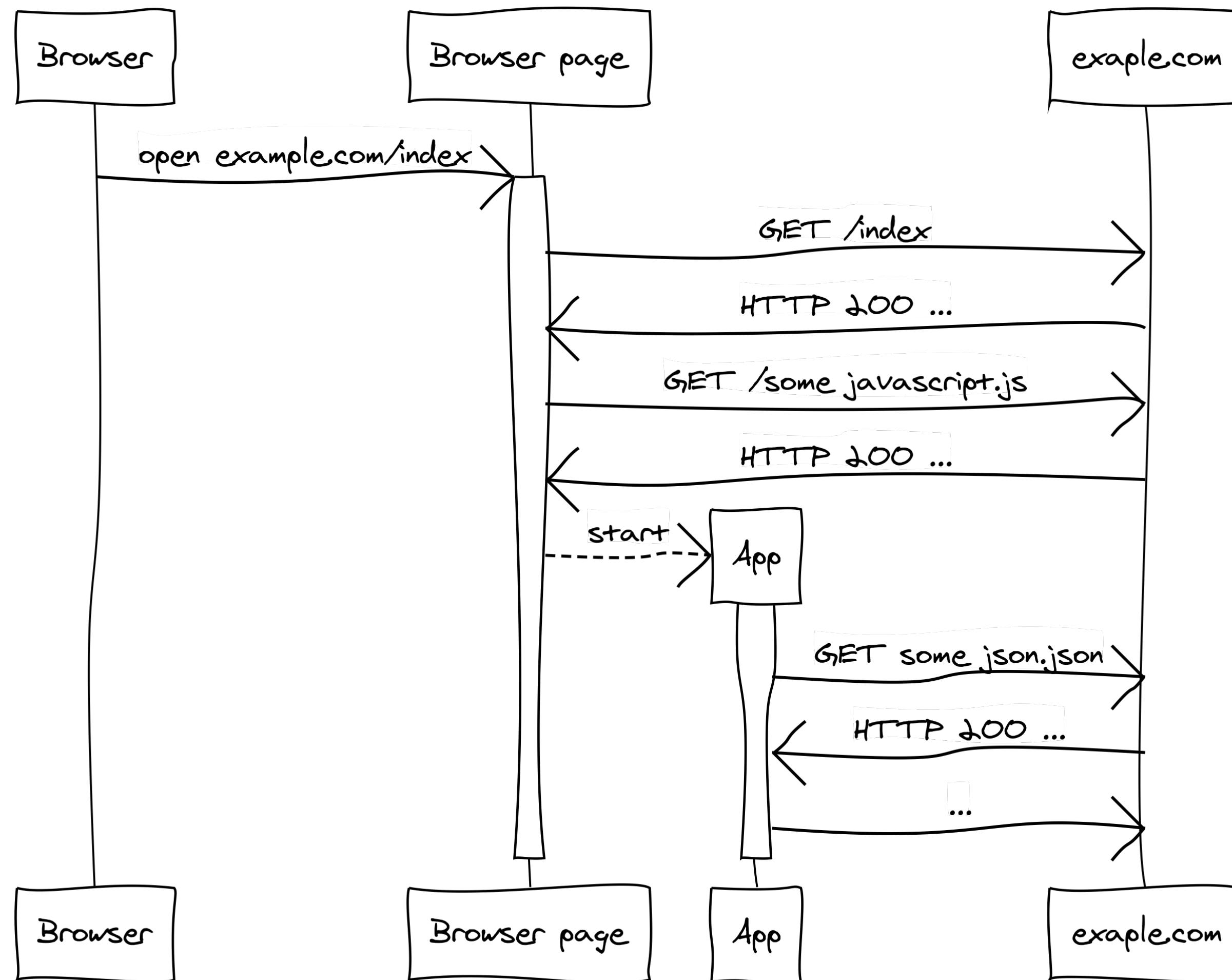








Single Page Apps – Why Routing?

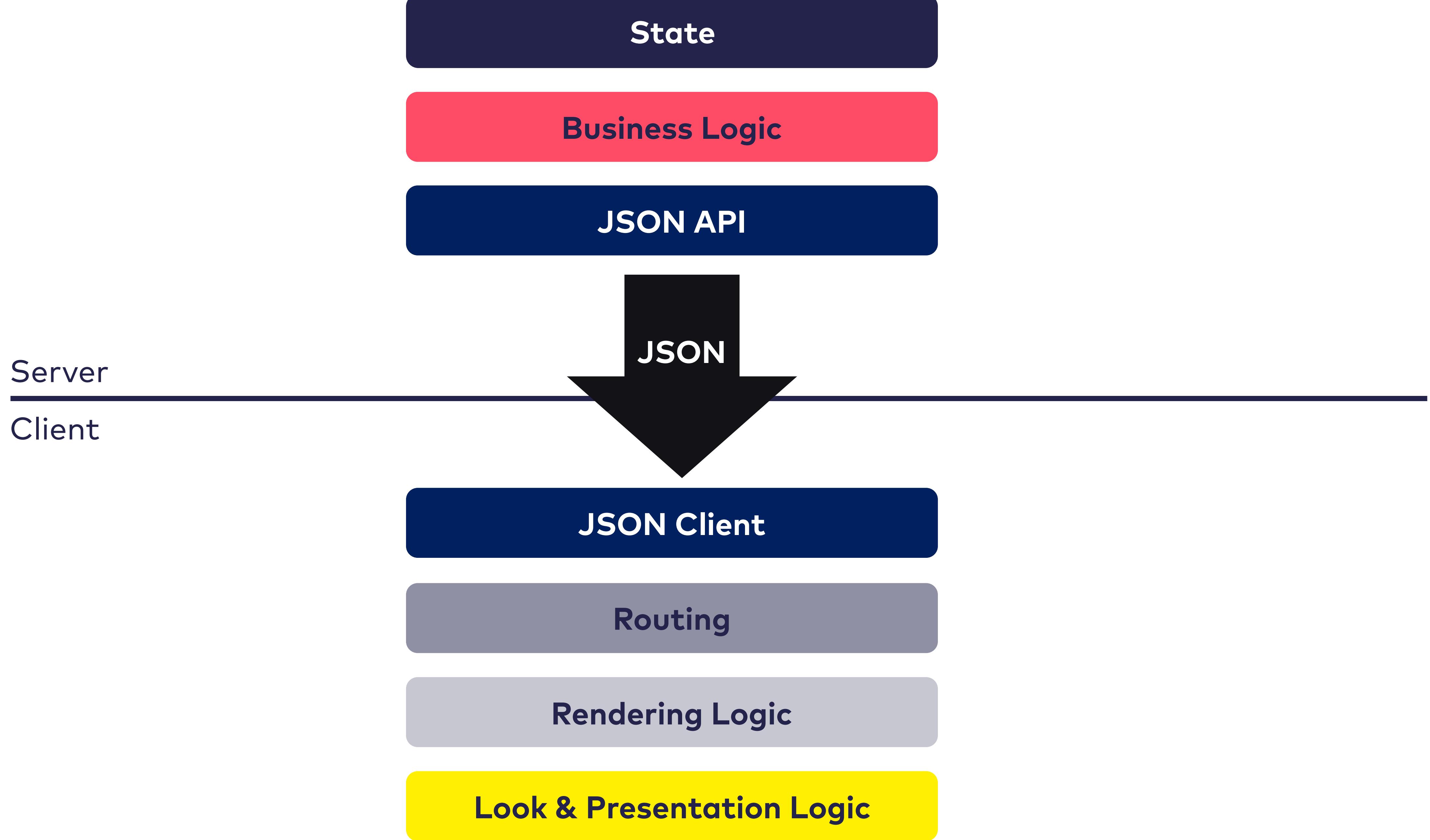


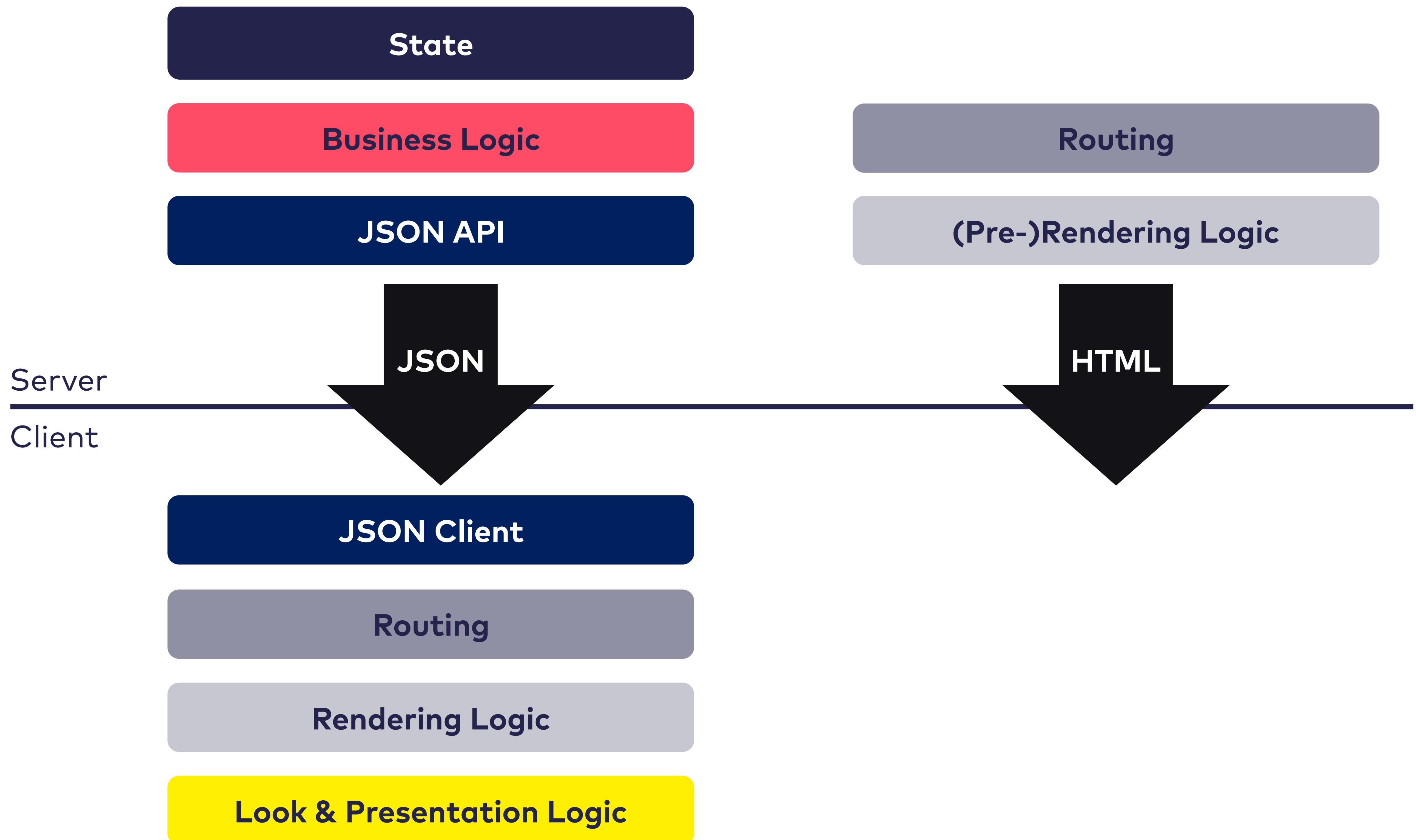
Bookmarks?

Deep links?

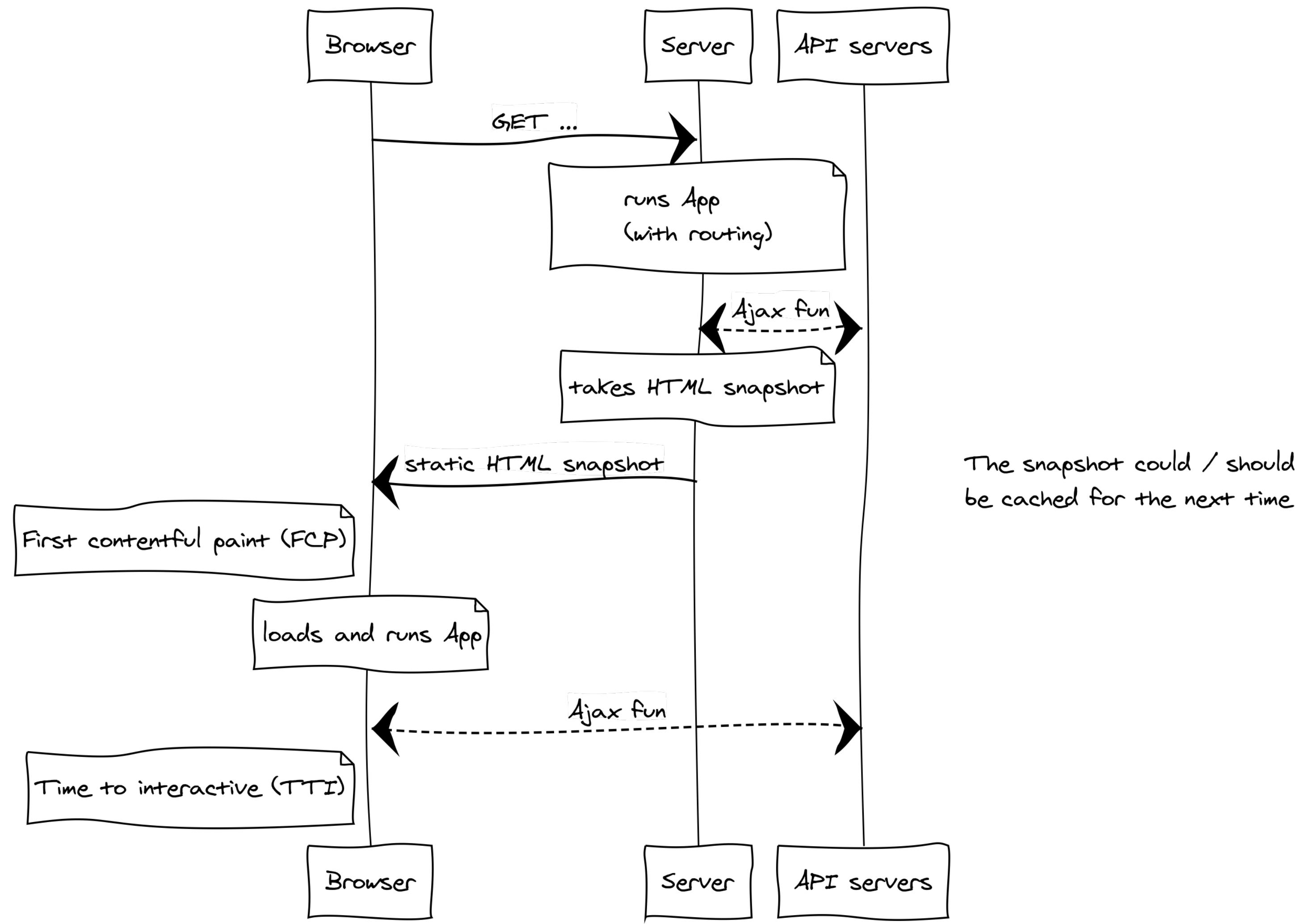
Reload?

Solution:
Store some app
state in the URI!





Prerendering

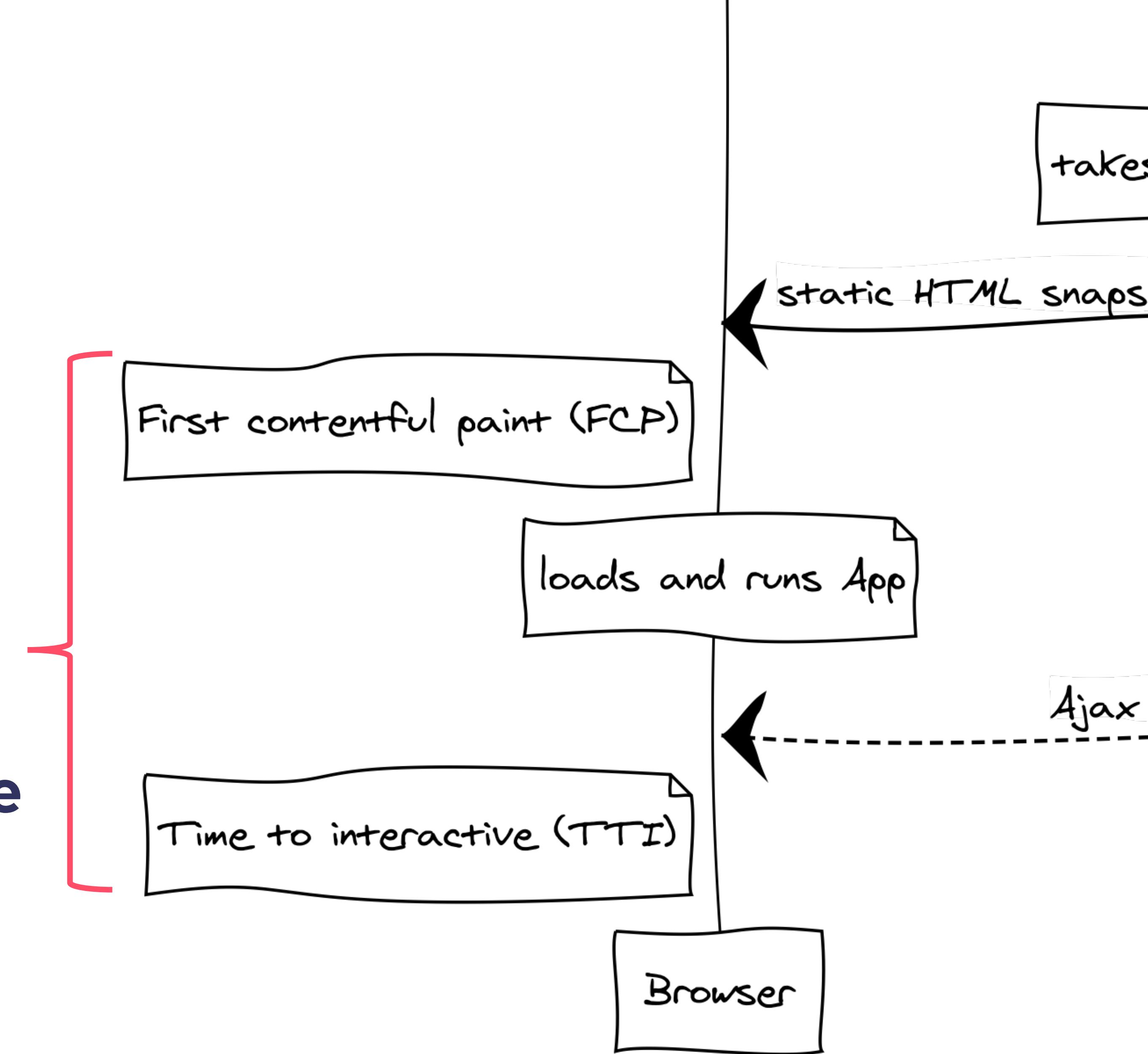


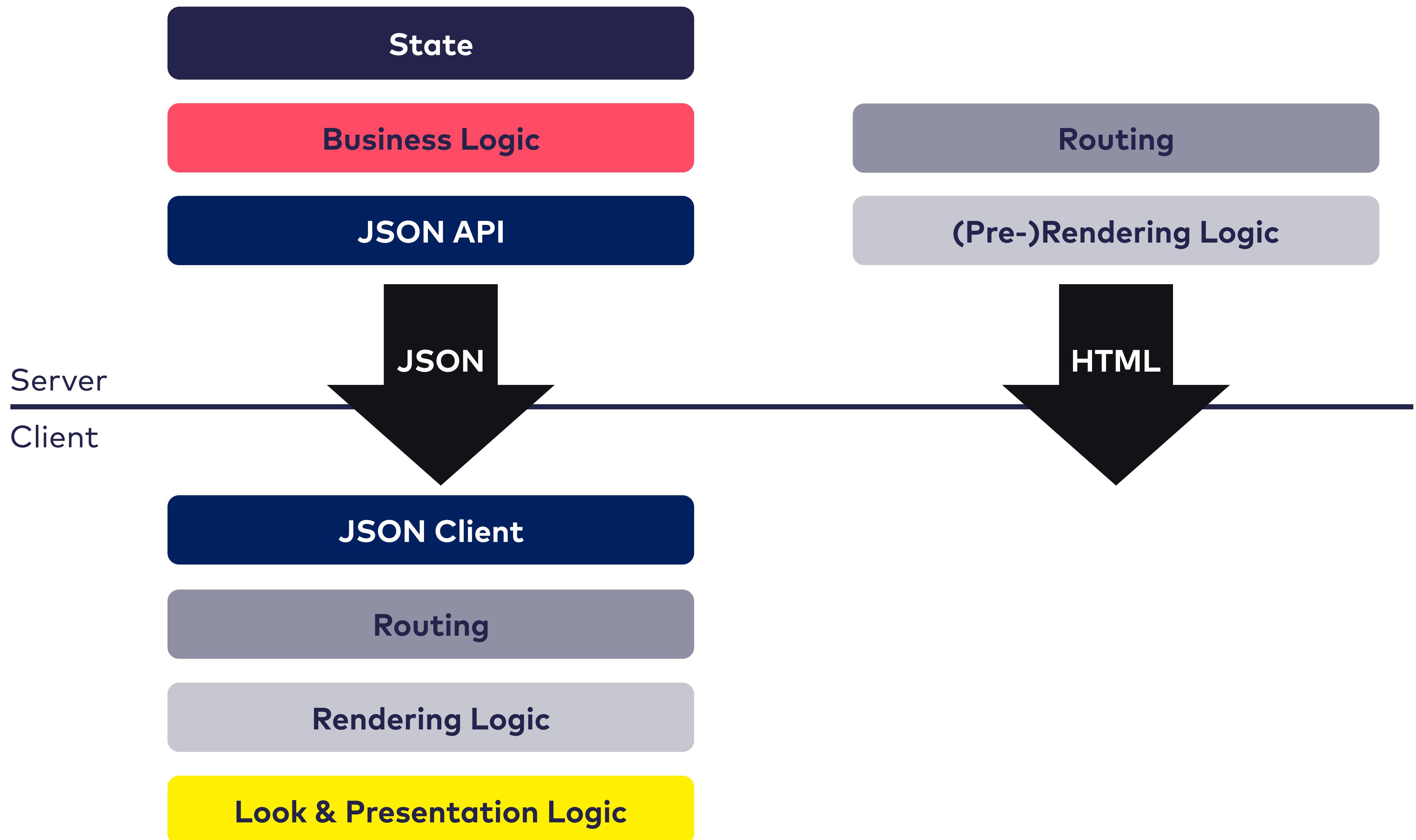
Hydration

How to simulate
readiness?

What about Events
(Clicks etc)?

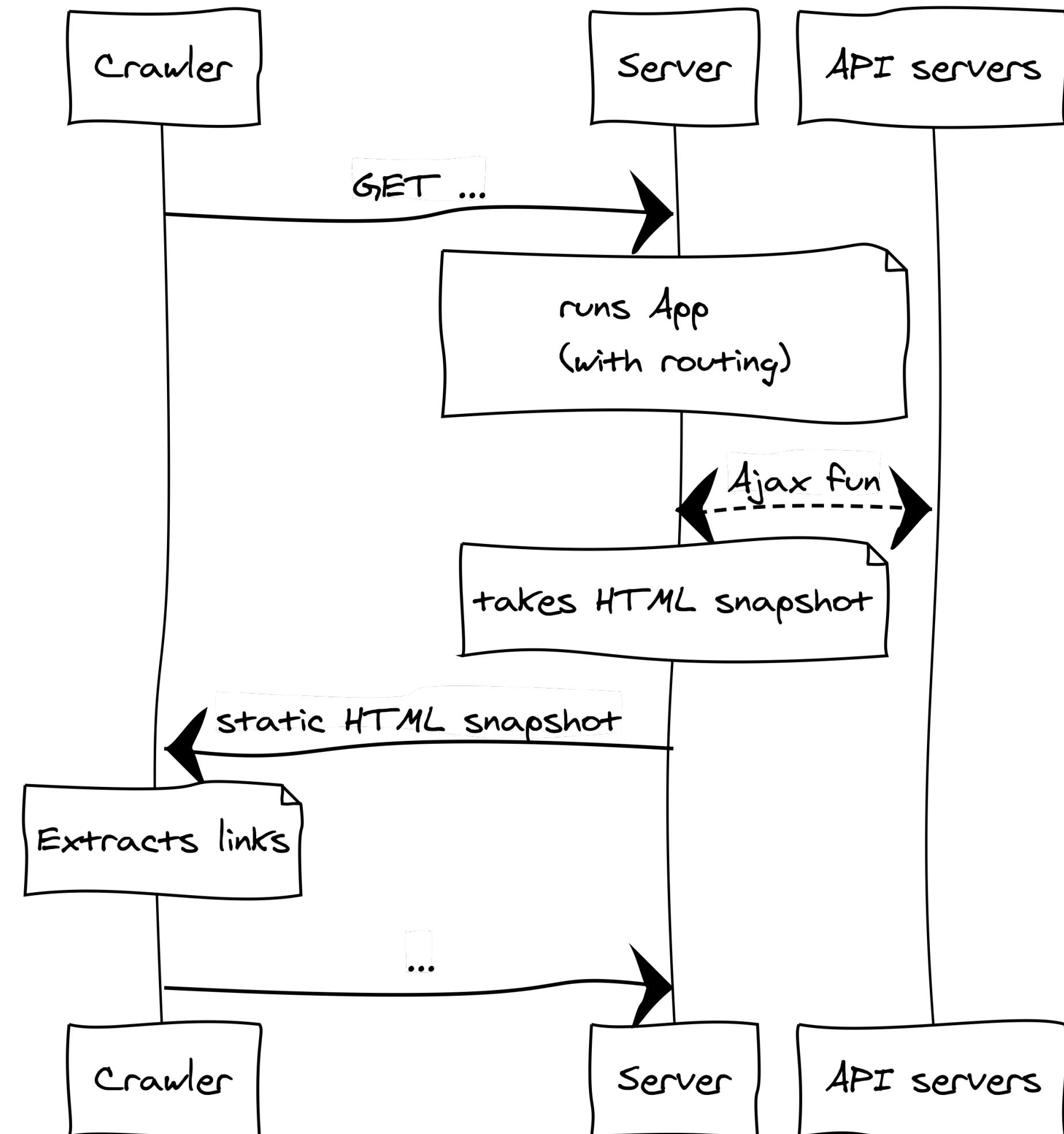
How to match server-side
HTML to client-side
DOM?

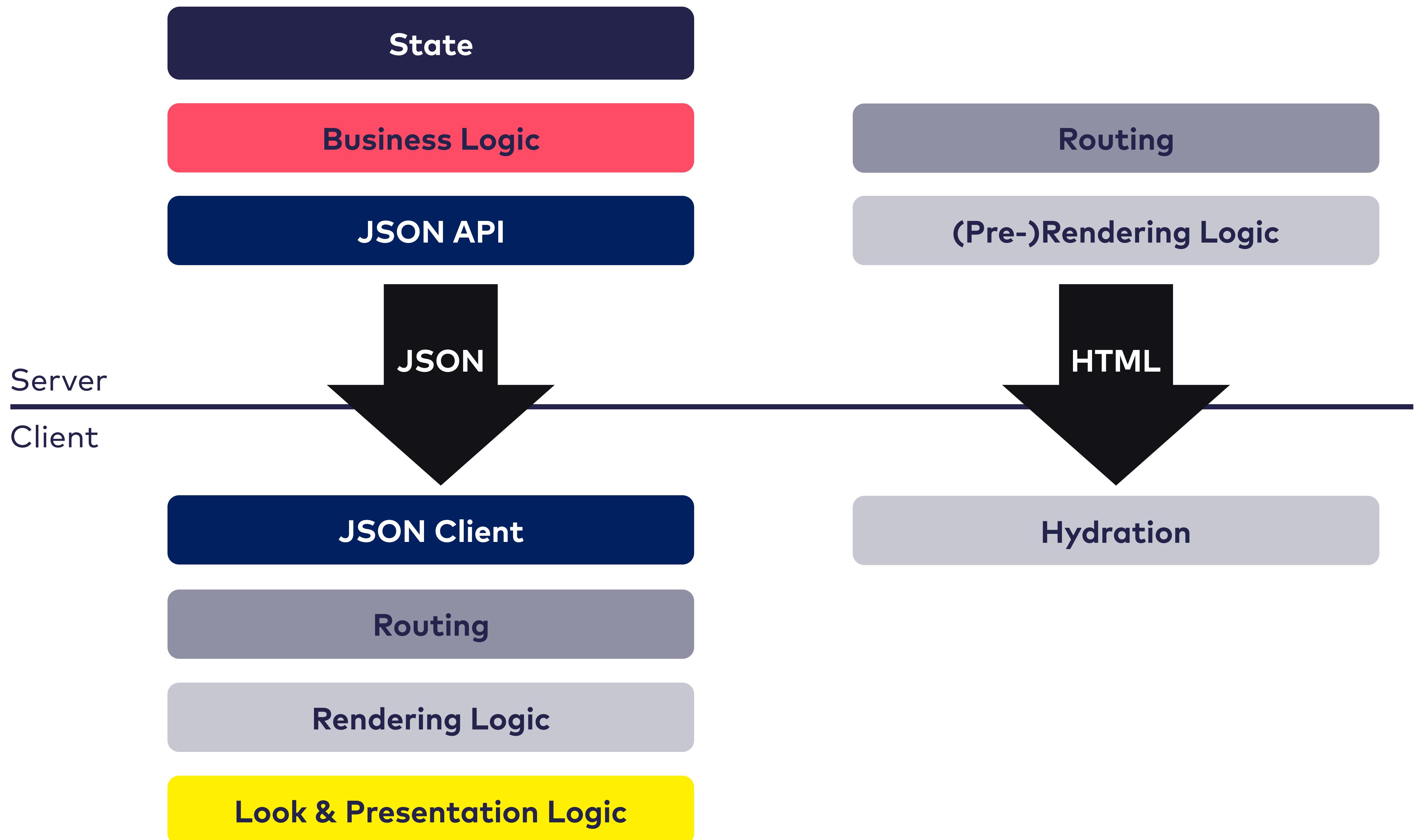




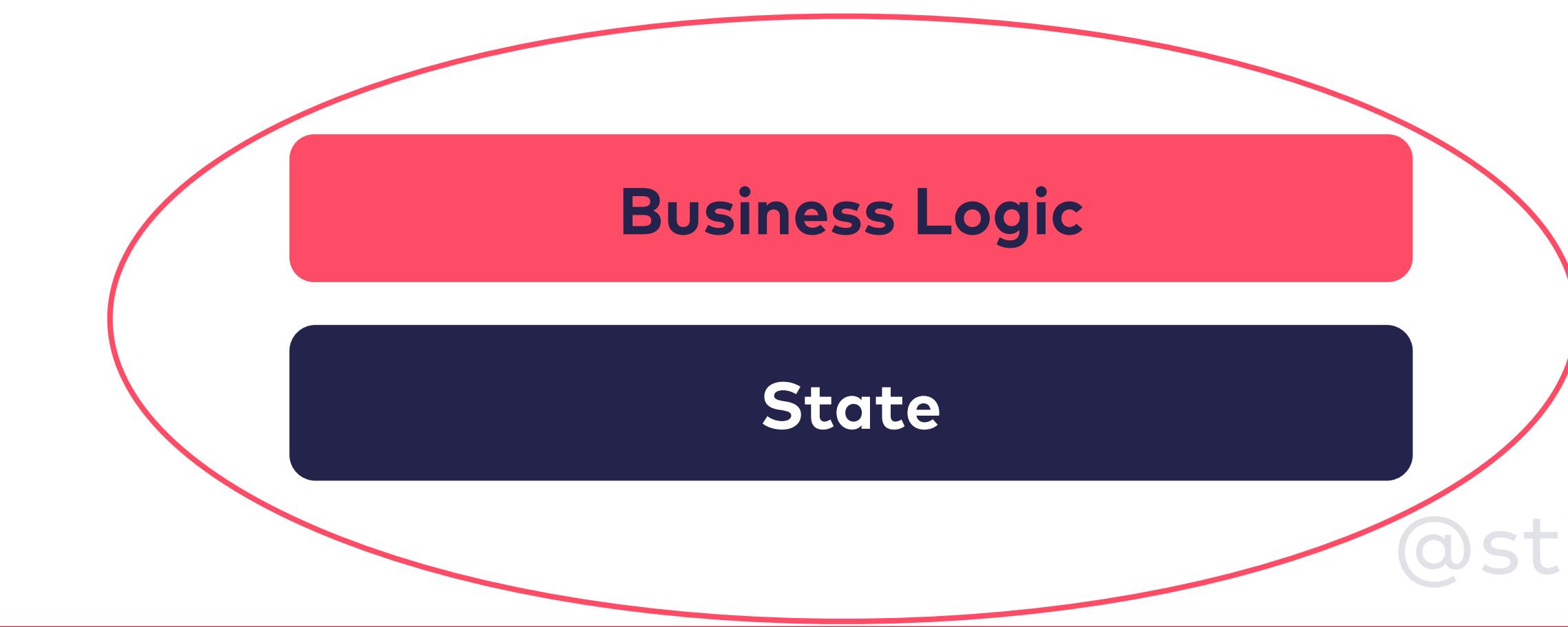
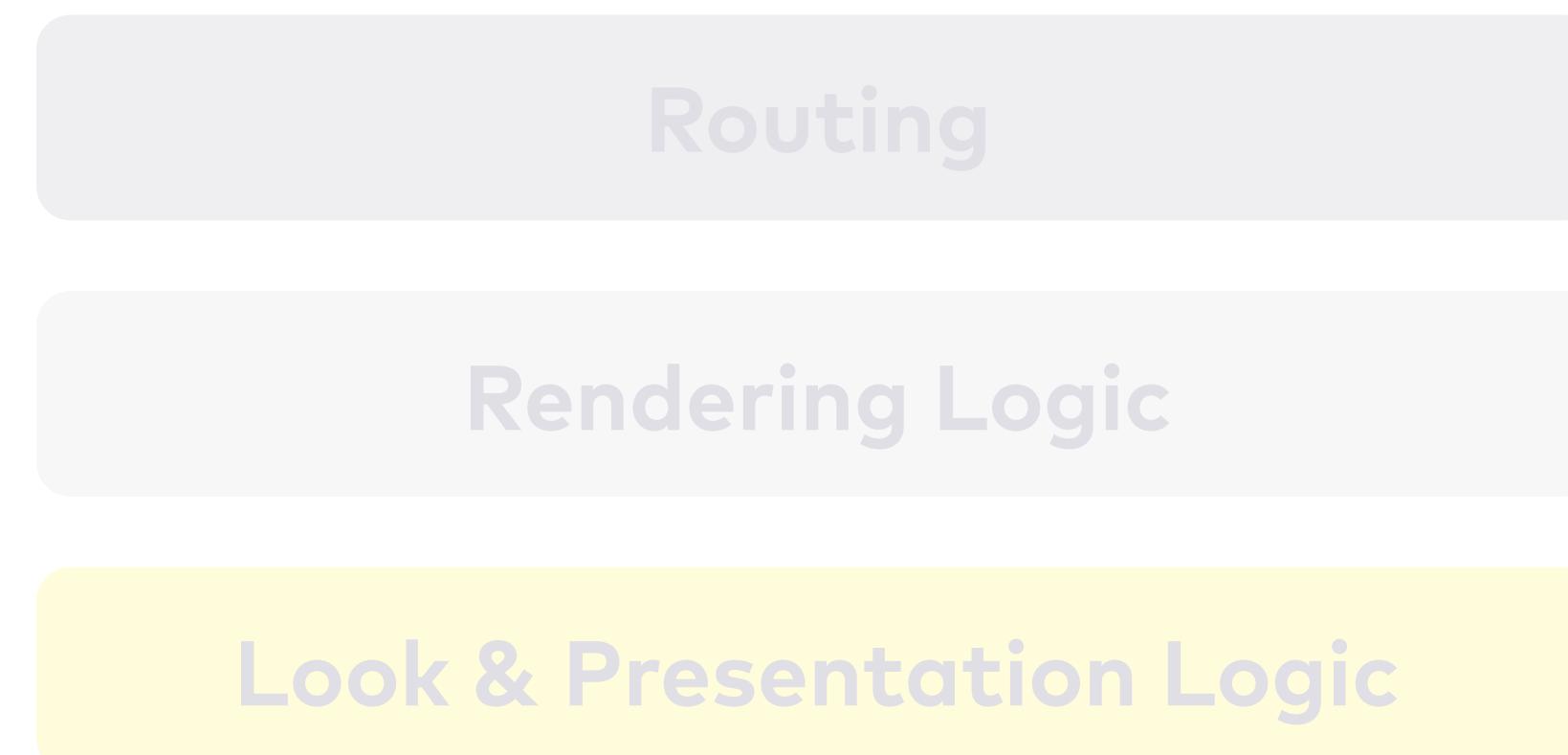
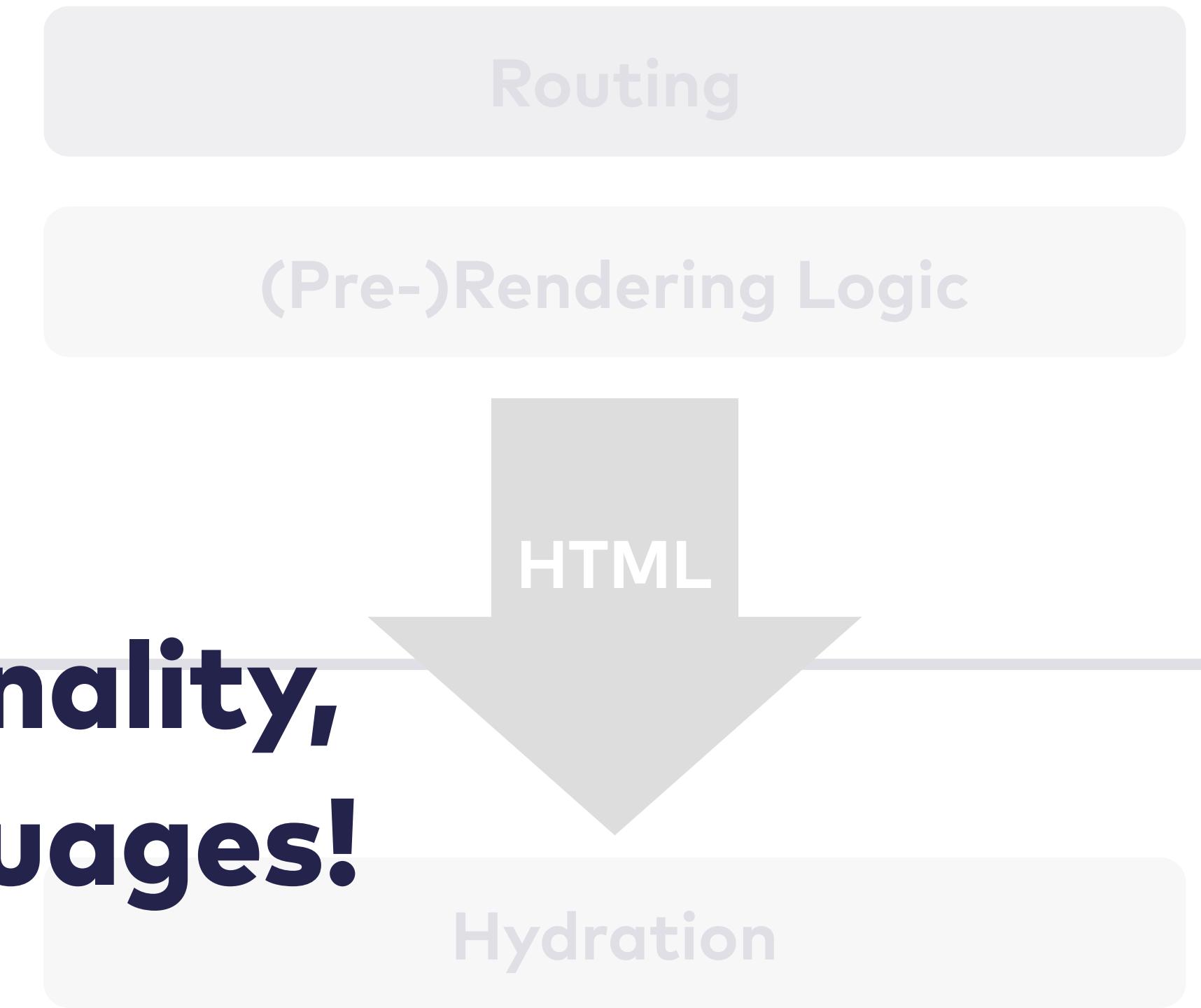
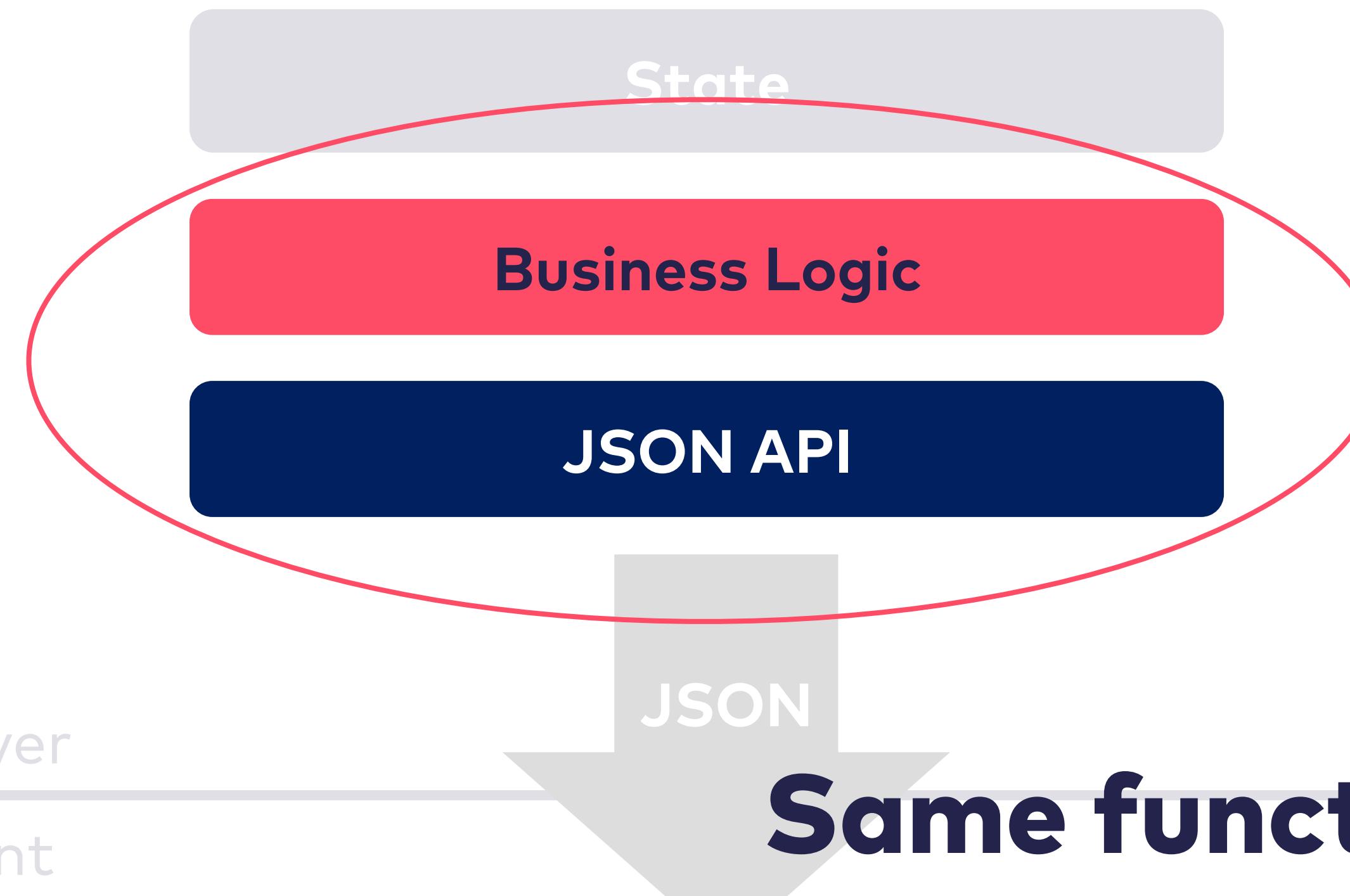
Special case: Searchability

No
Hydration
needed

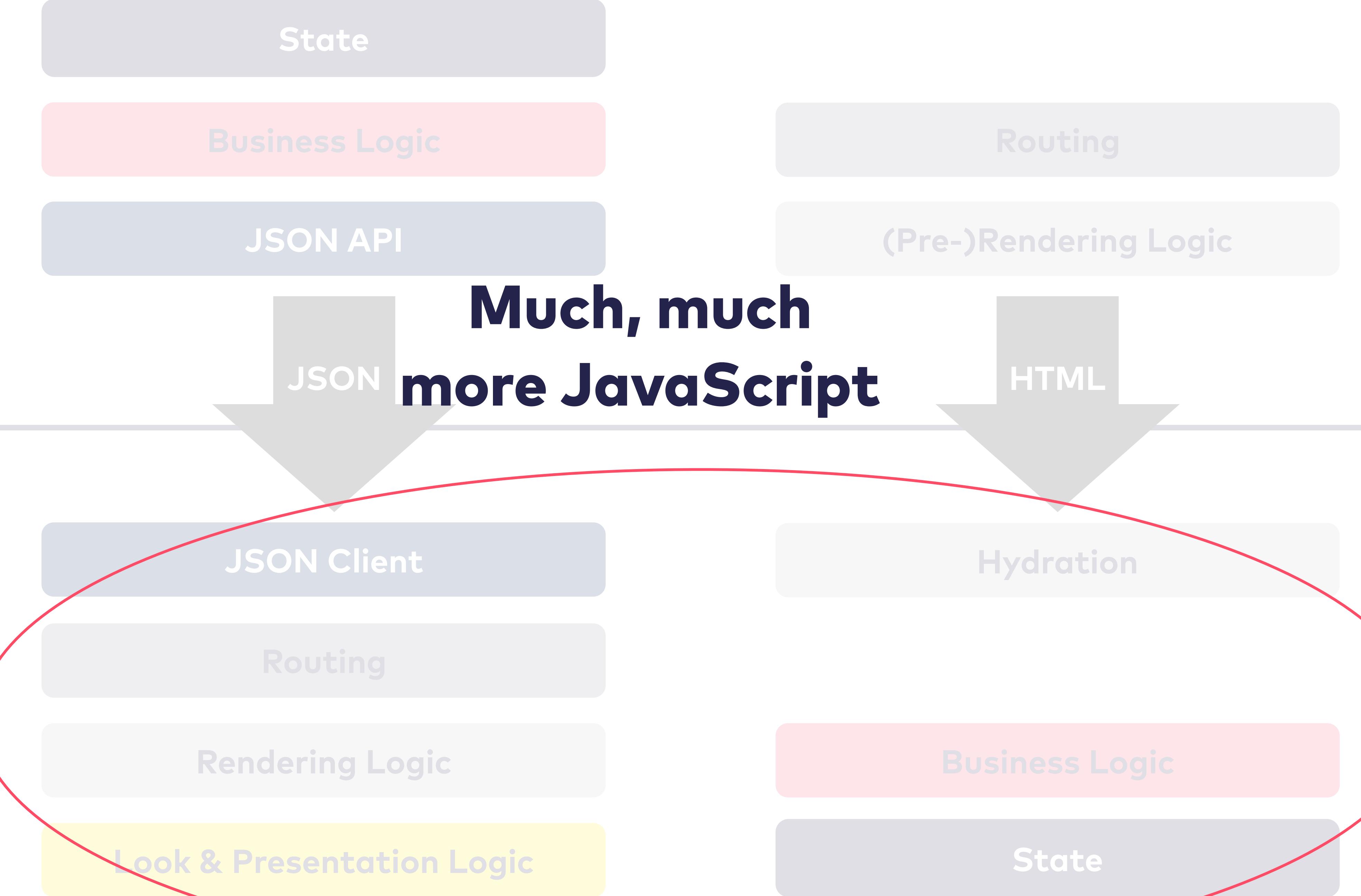




Server
Client



Server
Client





@stilkov

All your users are non-JS while they're downloading your JS

Does the corporate firewall block JavaScript?

Did the HTTP request for the JavaScript complete?

Does their browser support the JavaScript you've written?

Do your users get the JavaScript?

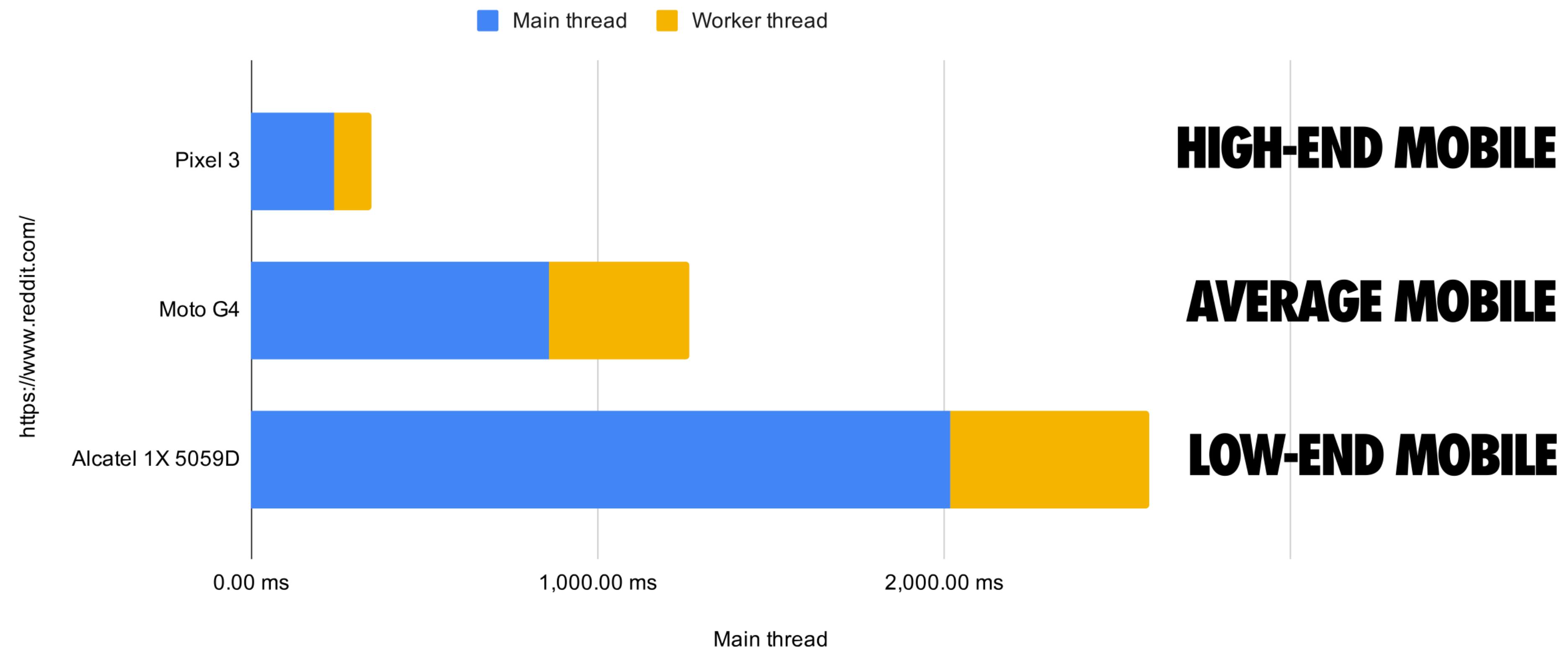
If they're on a train and their net connection goes away before your JavaScript loads, then there's no JavaScript.

Does their ISP or mobile operator interfere with downloaded JavaScript?

COST OF JAVASCRIPT

REDDIT.COM

JavaScript Processing times for Reddit.com



What's the alternative?

ROCA: Resource-oriented Client Architecture

<http://roca-style.org>

ROCA

RESTful Server-side HTML (SSR)

Client-side (self contained) JavaScript components

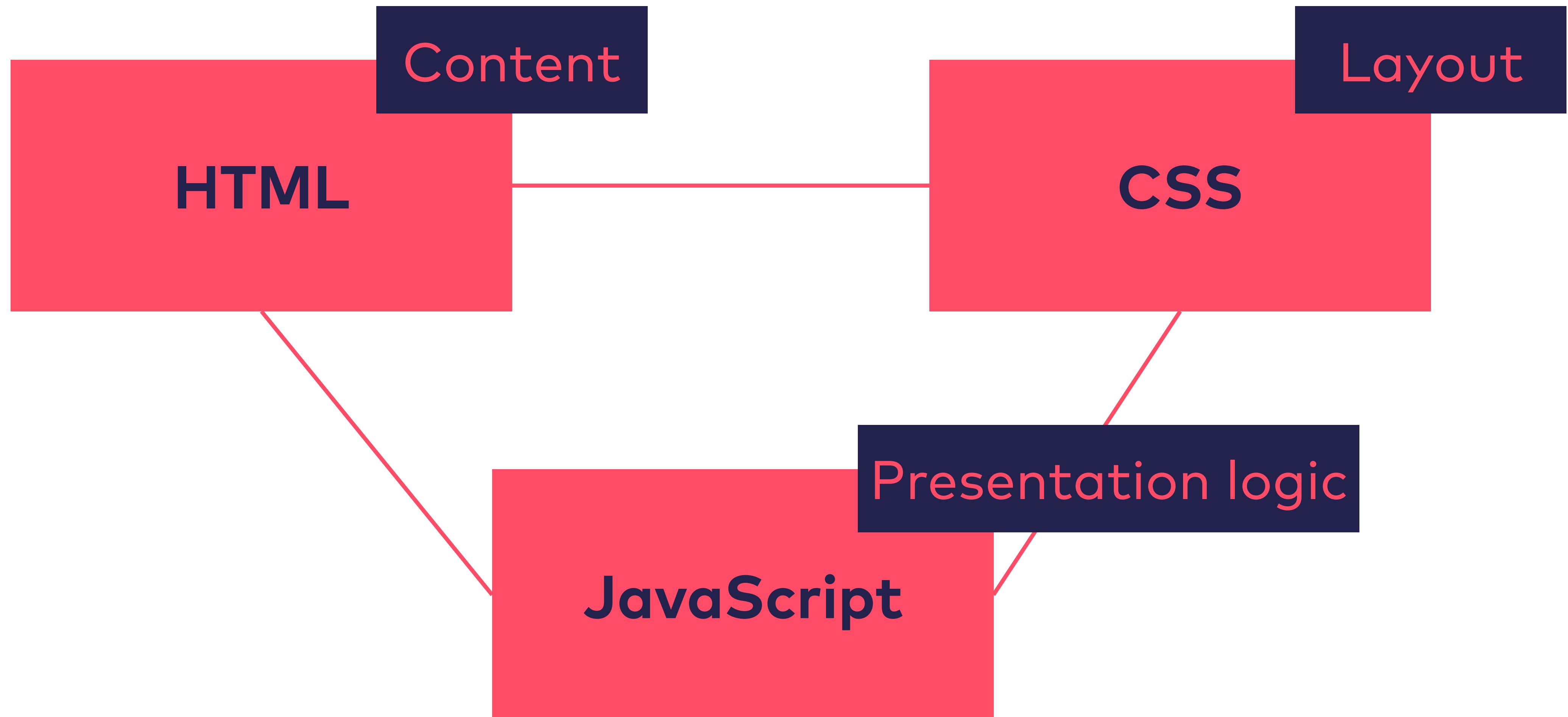
Application logic only on server

+

No duplicated logic on client

=

No application logic on client!



Client Side Logic is generic Presentation logic only. It enhances HTML

Progressive Enhancement

A web page needs to work without graphical elements, CSS and JS

This ensures our page will even work under unfavorable circumstances

We also lay the foundation for accessibility

Does **not** mean that everything needs to work without CSS and JavaScript

It means that the fundamental functionality of our web page relies on HTML only

Every thing that goes beyond that is seen as an enhancement

```

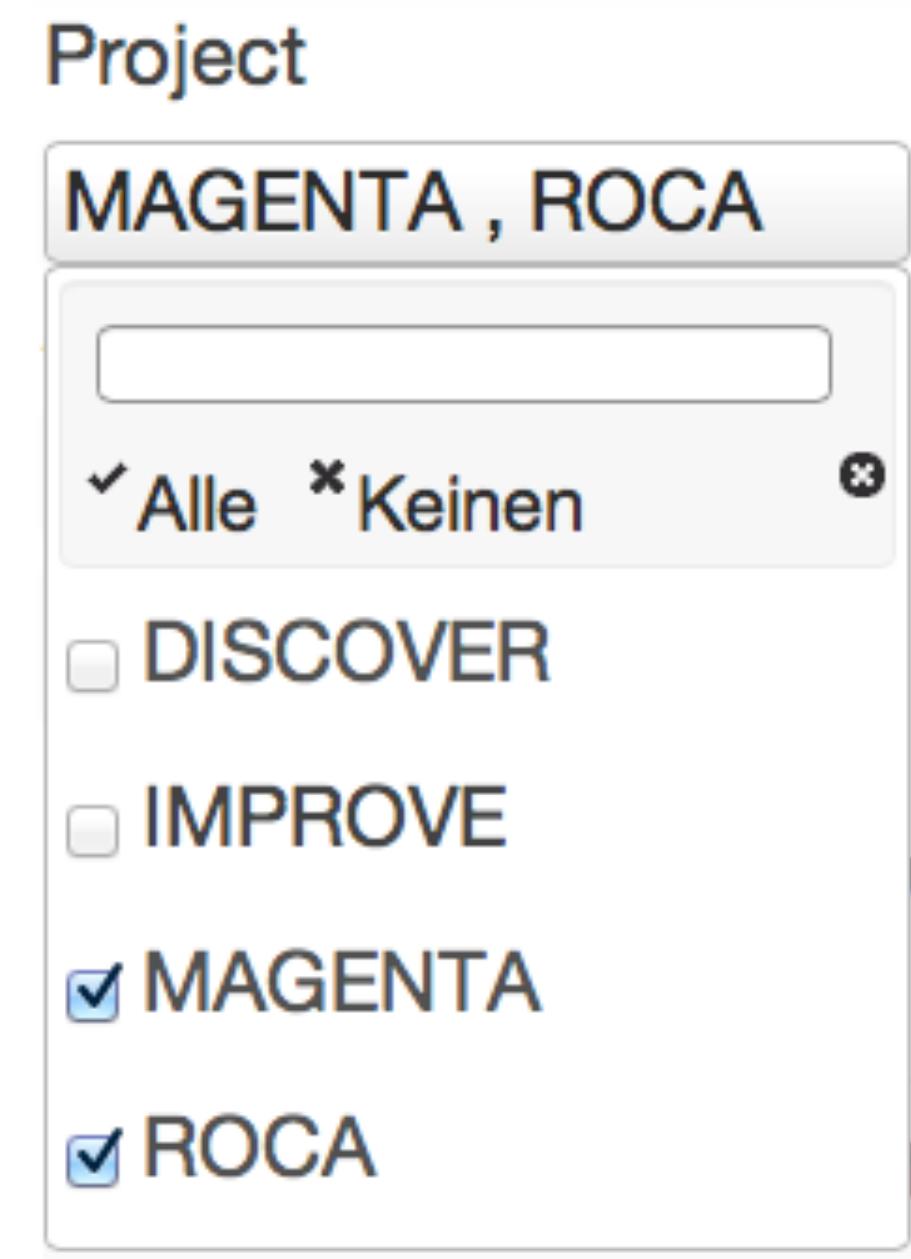
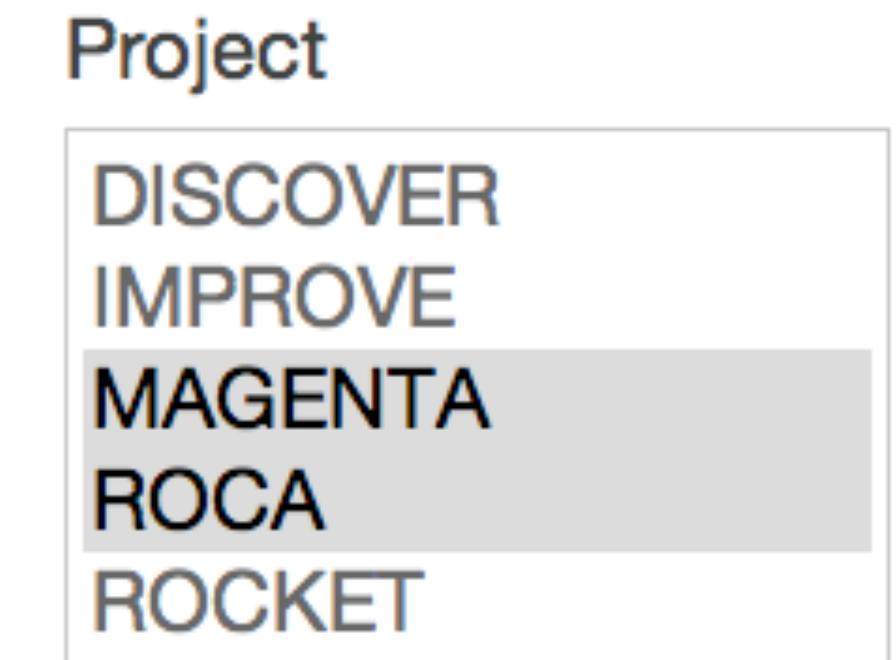
<div class="filter-column">
  <label for="project">Project</label>
  <select class="multiselect" id="project"
    name="project" size="5" multiple>
    <option>DISCOVER</option>
    <option>IMPROVE</option>
    <option>MAGENTA</option>
    <option>ROCA</option>
    <option>ROCKET</option>
  </select>
</div>

```

```

$('.multiselect', context).each(function() {
  $(this).multiselect({
    selectedList: 2,
    checkAllText: "Alle",
    uncheckAllText: "Keinen"
  }).multiselectfilter({label:"", width:"200px"});
});

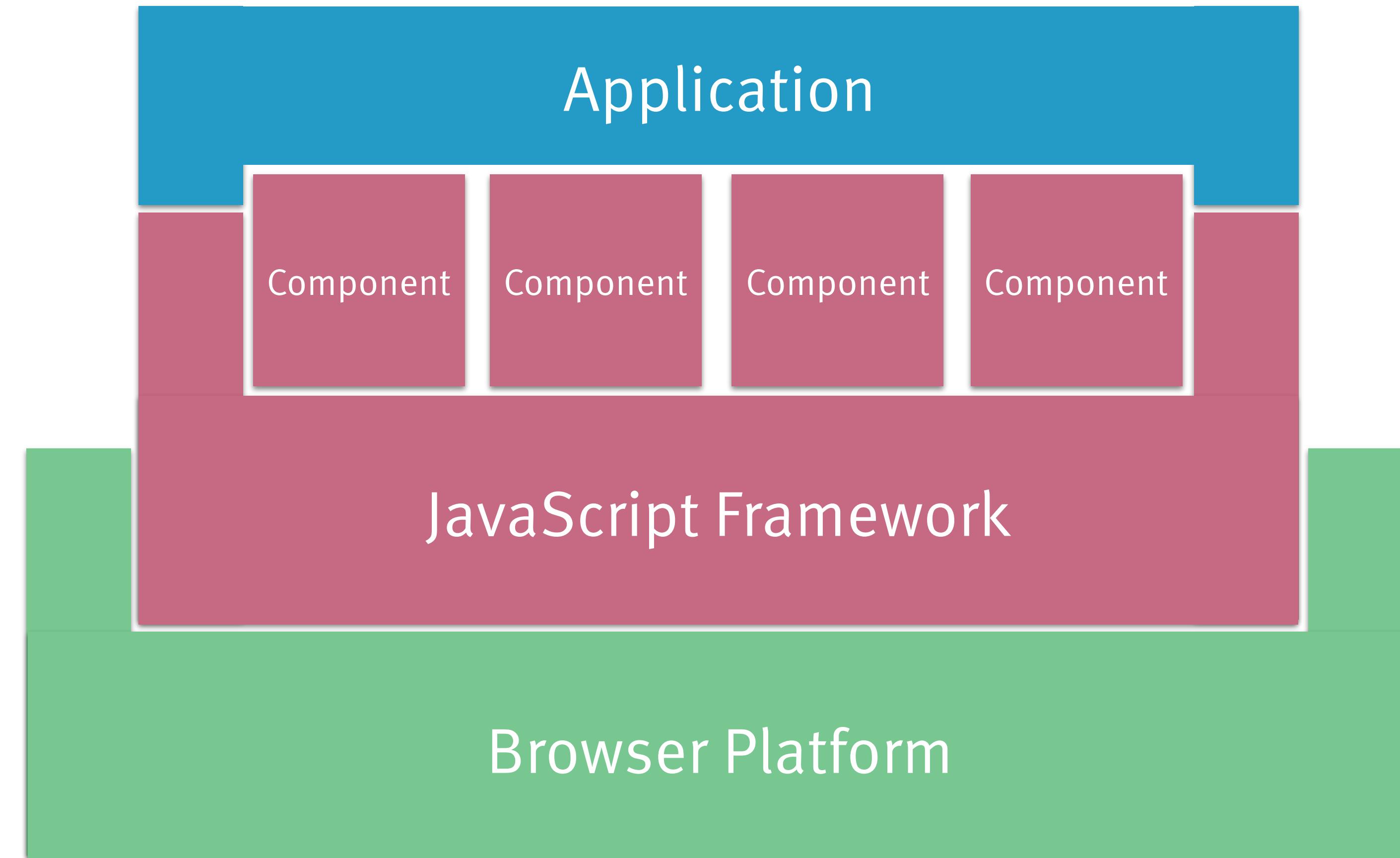
```

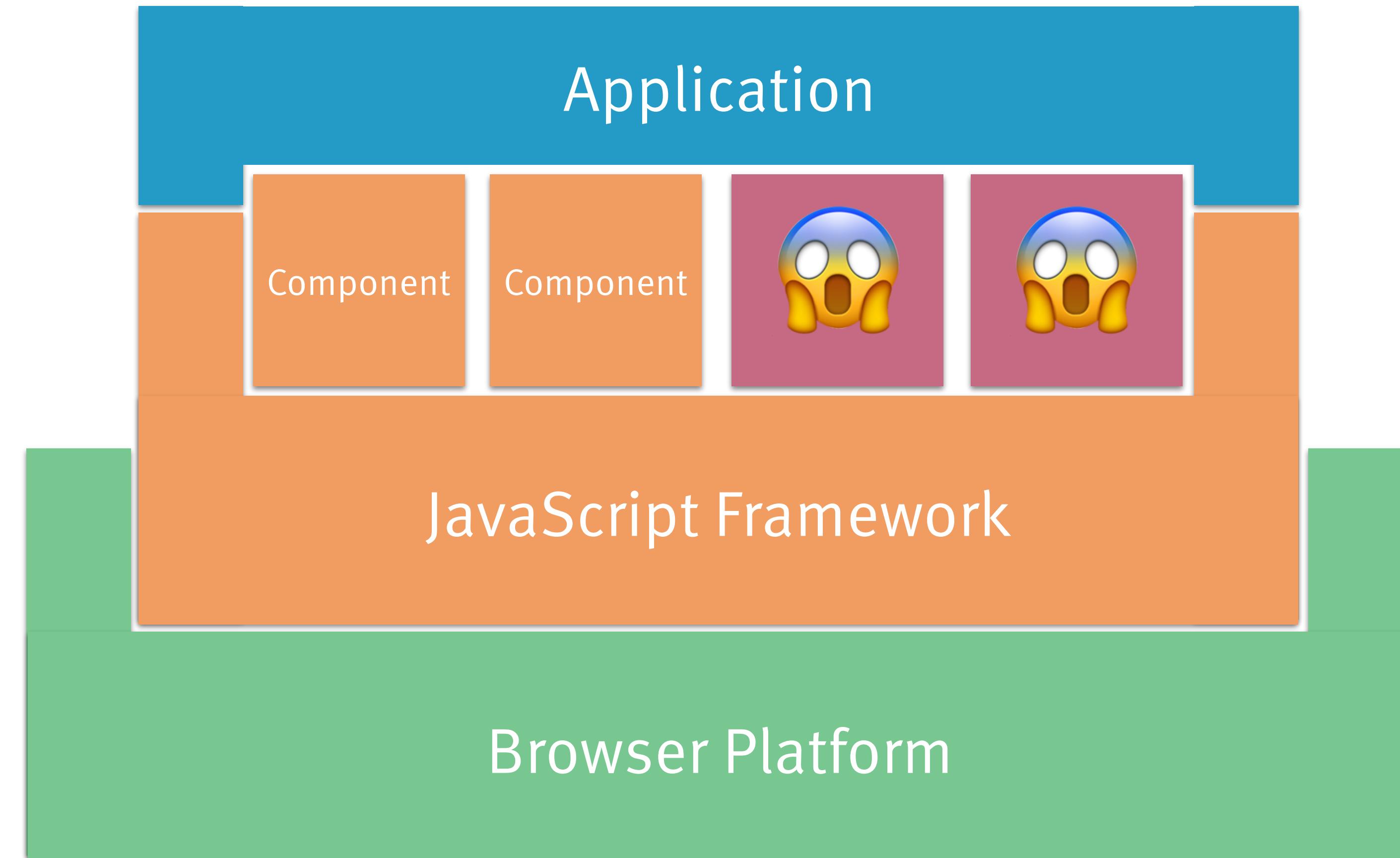


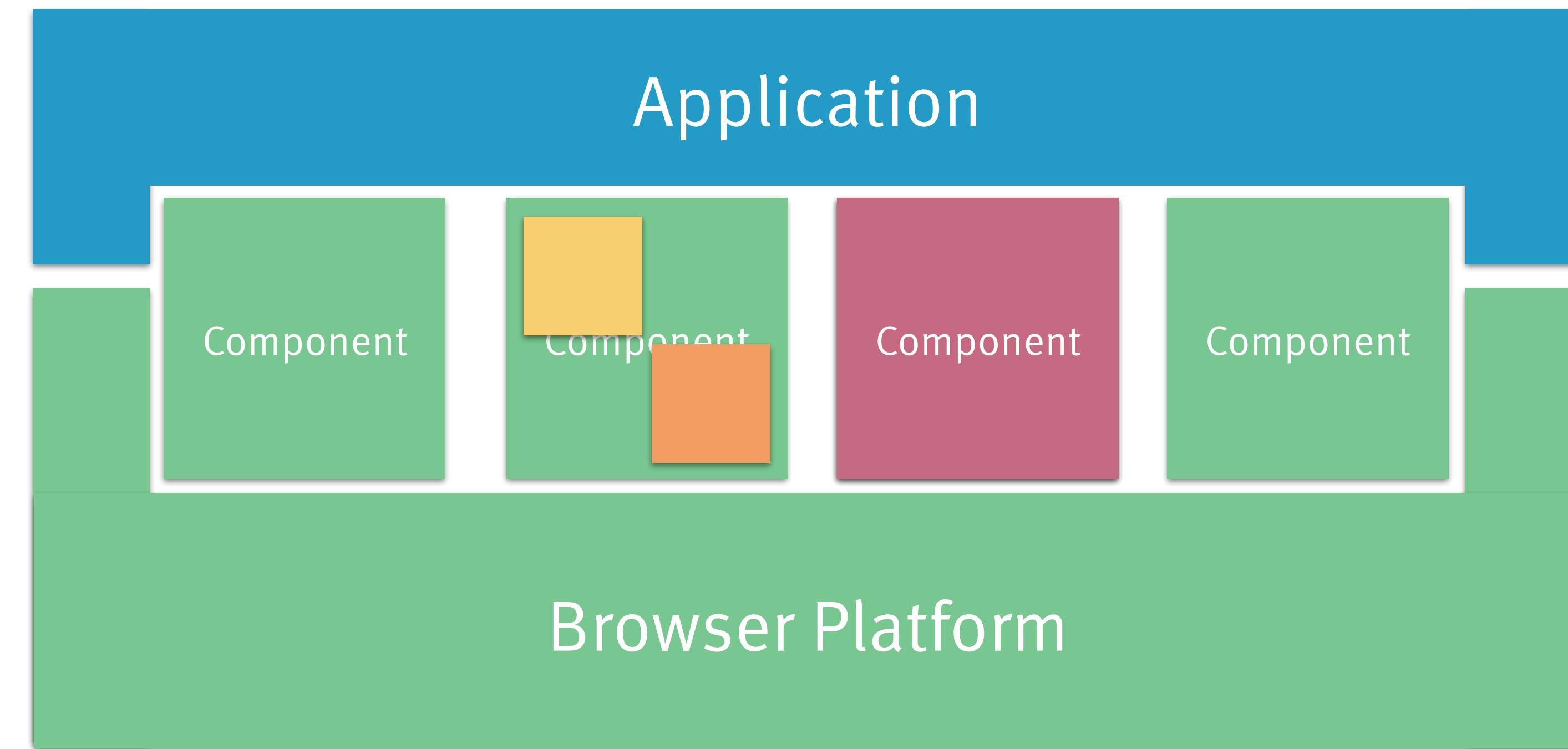
http://roca-airways.herokuapp.com

http://rocair.herokuapp.com

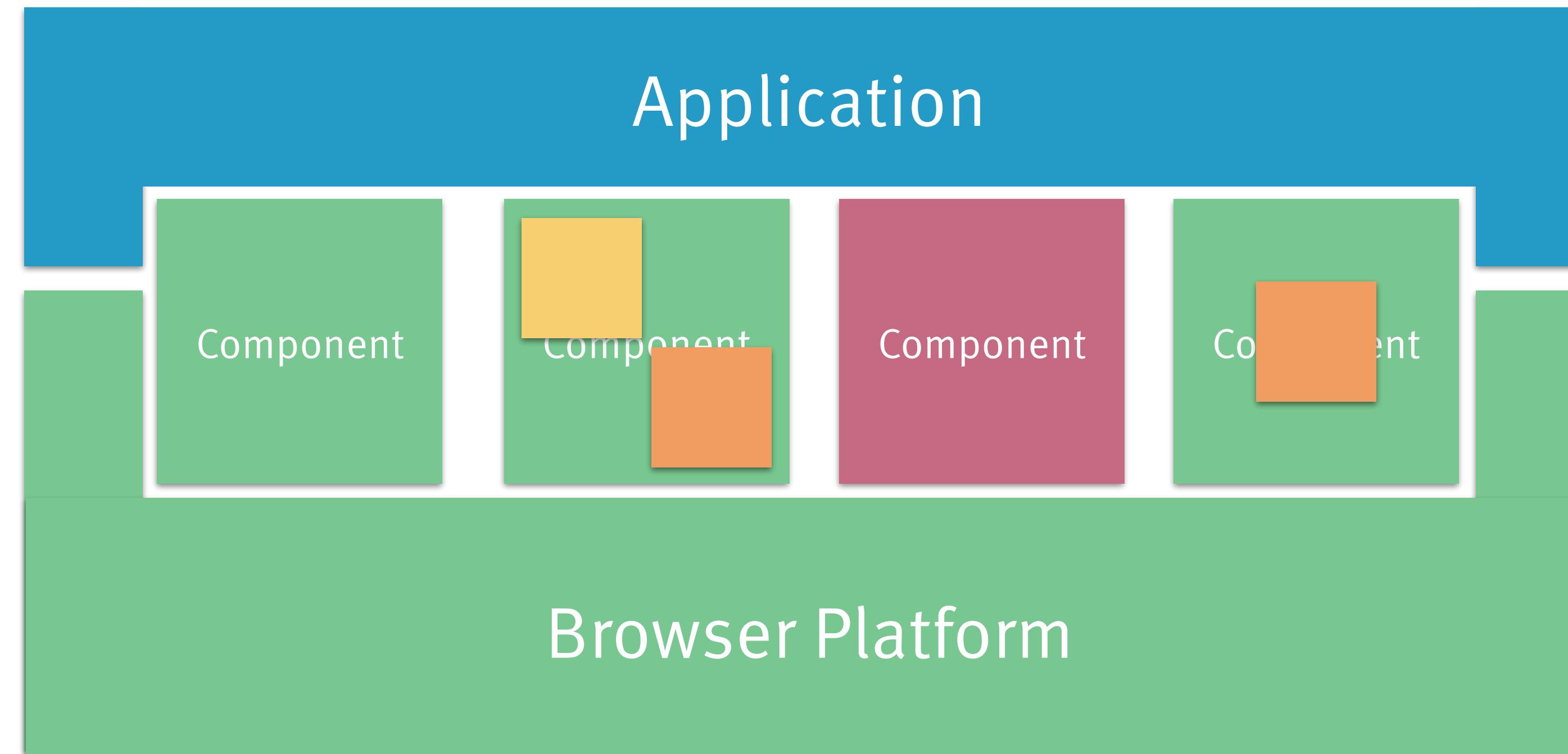
Components







```
<audio src="meow.mp4" controls>
```



Application

Glue Code

Component

Component

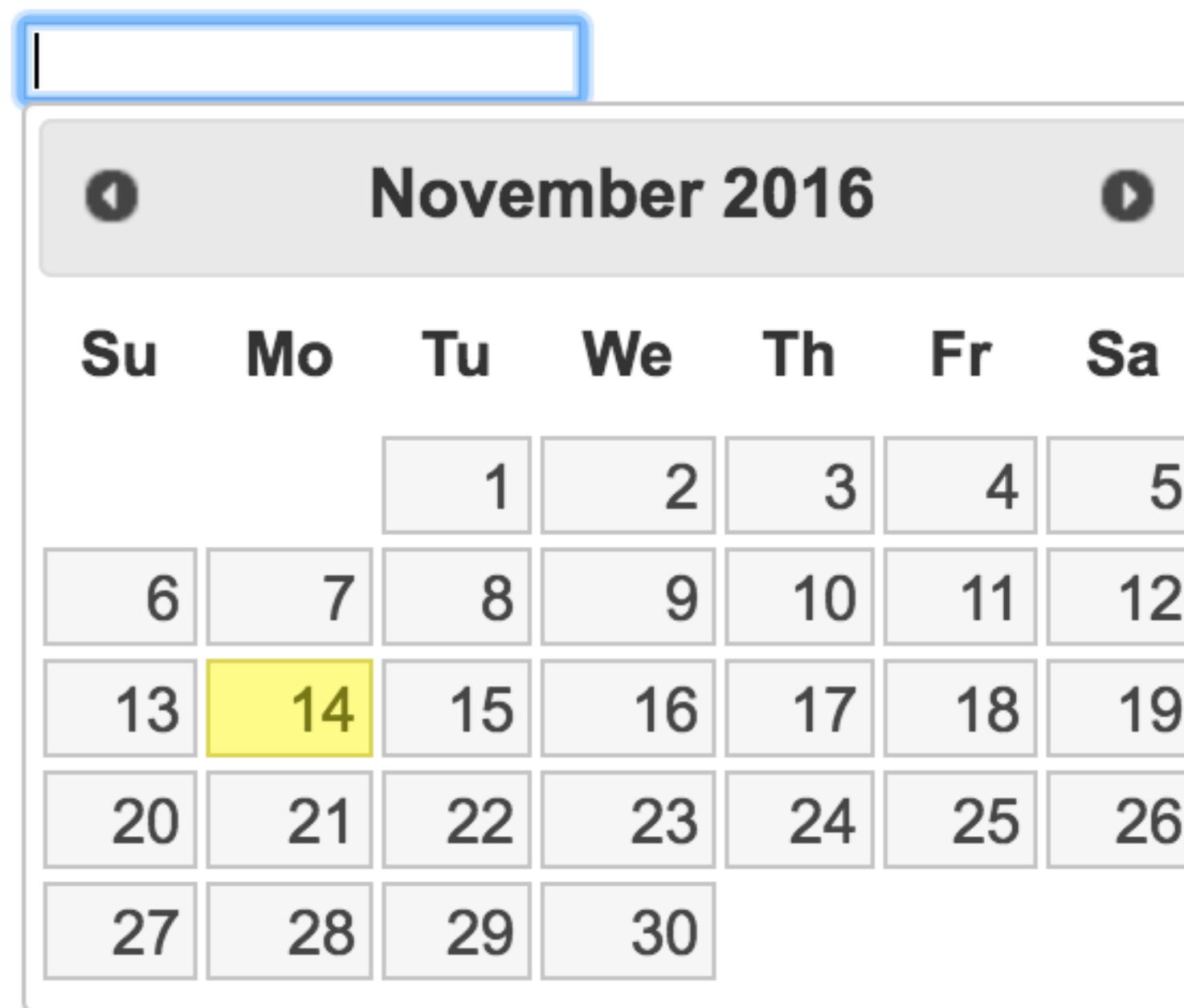
Component

Component

Browser Platform

```
<input type="text" class="date">
```

```
$( "input.date" ).datepicker();
```

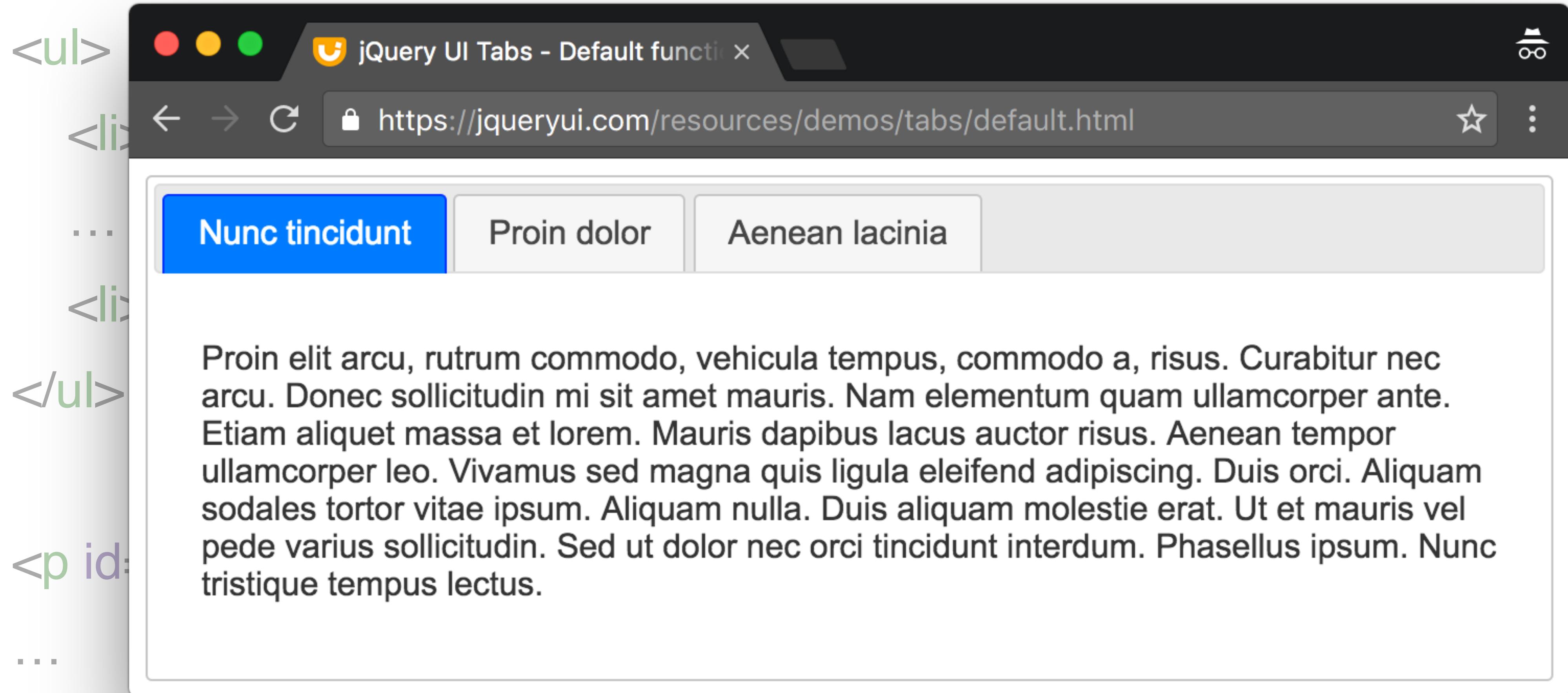


```
<div class="tabs">  
  <ul>  
    <li> <a href="#about">About</a>  
    ...  
    <li> <a href="#comments">Discussion</a>  
  </ul>  
  
<p id="about">lorem ipsum dolor sit amet</p>  
...  
<ol id="comments"> ... </ol>  
  
</div>
```

```
$(".tabs").tabs();
```

```
<div class="tabs">
```

```
$(".tabs").tabs();
```



```
<ol id="comments"> ... </ol>
```

```
</div>
```

```
<div class="tabs">
```

```
$(".tabs").tabs();
```

The screenshot shows a web browser window with the title "jQuery UI Tabs - Default function". The address bar shows the URL <https://jqueryui.com/resources/demos/tabs/default.html>. The page displays three tabs: "Nunc tincidunt", "Proin dolor", and "Aenean lacinia". The "Proin dolor" tab is currently selected, revealing its content area which contains placeholder text from the jQuery UI documentation.

```
<ul>
<li>• Nunc tincidunt
...
<li>• Proin dolor
<li>• Aenean lacinia
</ul>
<p id="content">
...
<ol id="list">
...
</div>
```

Proin elit arcu, rutrum commodo, vehicula tempus, commodo a, risus. Curabitur nec arcu. Donec sollicitudin mi sit amet mauris. Nam elementum quam ullamcorper ante. Etiam aliquet massa et lorem. Mauris dapibus lacus auctor risus. Aenean tempor ullamcorper leo. Vivamus sed magna quis ligula eleifend adipiscing. Duis orci. Aliquam sodales tortor vitae ipsum. Aliquam nulla. Duis aliquam molestie erat. Ut et mauris vel pede varius sollicitudin. Sed ut dolor nec orci tincidunt interdum. Phasellus ipsum. Nunc tristique tempus lectus.

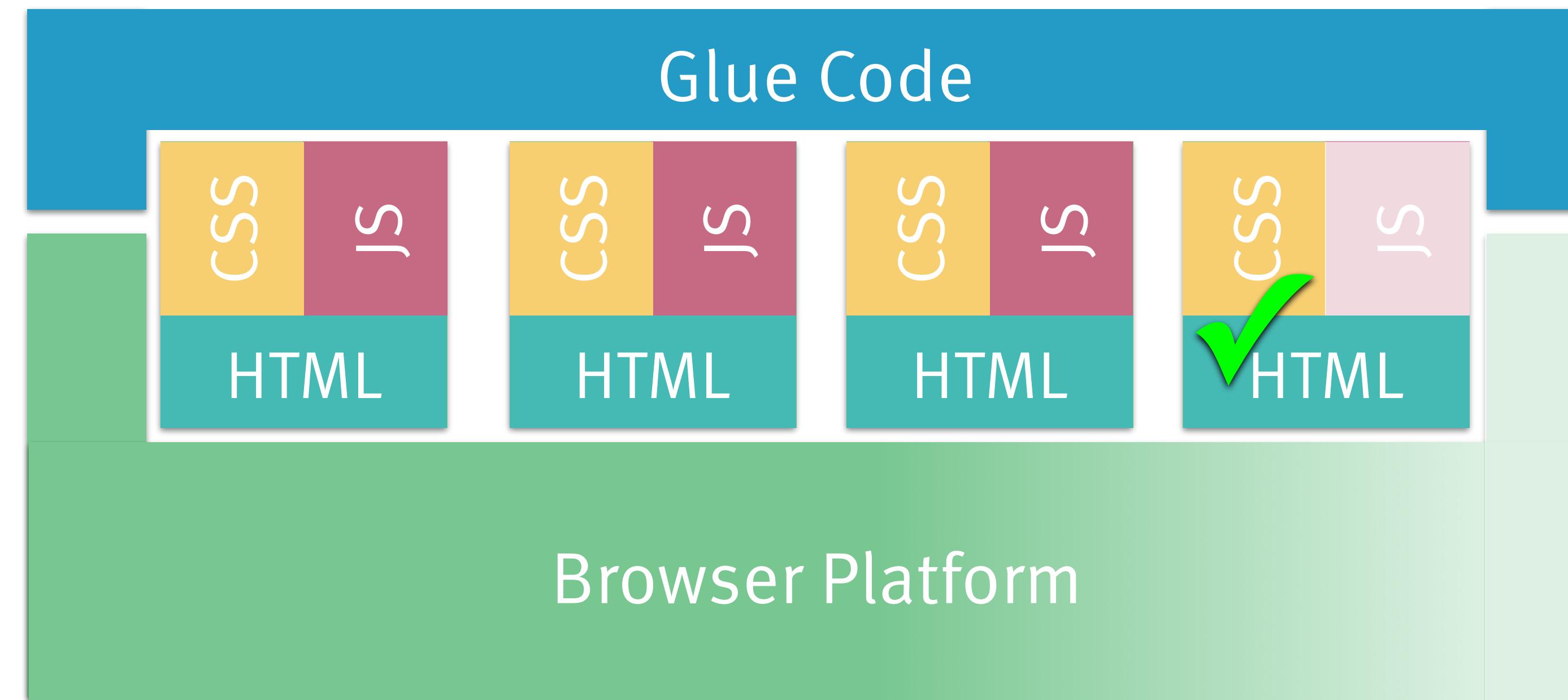
Unobtrusive JavaScript

Morbi tincidunt, dui sit amet facilisis feugiat, odio metus gravida ante, ut pharetra massa metus id nunc. Duis scelerisque molestie turpis. Sed fringilla, massa eget luctus malesuada, metus eros molestie lectus, ut tempus eros massa ut dolor. Aenean aliquet fringilla sem. Suspendisse sed ligula in ligula suscipit aliquam. Praesent in eros vestibulum mi adipiscing adipiscing. Morbi facilisis. Curabitur ornare consequat nunc. Aenean vel metus. Ut posuere viverra nulla. Aliquam erat volutpat. Pellentesque convallis. Maecenas feugiat, tellus pellentesque pretium posuere, felis lorem euismod felis, eu ornare leo nisi vel felis. Mauris consectetur tortor et purus.

Mauris eleifend est et turpis. Duis id erat. Suspendisse potenti. Aliquam vulputate, pede vel



Progressive Enhancement





Progressive Enhancement



Progressive enhancement is not about dealing with old browsers, it's about dealing with new browsers.

— Jeremy Keith (@adactio)

Browser Platform

```
<div class="tabs">
  <ul>
    <li> <a href="#about">About</a>
    ...
    <li> <a href="#comments">Discussion</a>
  </ul>

  <p id="about">lorem ipsum dolor sit amet</p>
  ...
  <ol id="comments"> ... </ol>

</div>
```

```
$(".tabs").tabs();
```

```
<tab-nav>                               $(".tabs").tabs();  
  <ul>  
    <li> <a href="#about">About</a>  
    ...  
    <li> <a href="#comments">Discussion</a>  
  </ul>  
</tab-nav>  
  
<p id="about">lorem ipsum dolor sit amet</p>  
...  
<ol id="comments"> ... </ol>
```

```
<tab-nav>                               $(".tabs").tabs();  
  <ul>  
    <li> <a href="#about">About</a>  
    ...  
    <li> <a href="#comments">Discussion</a>  
  </ul>  
</tab-nav>  
  
<tab-contents>  
  <p id="about">lorem ipsum dolor sit amet</p>  
  ...  
  <ol id="comments"> ... </ol>  
</tab-contents>
```

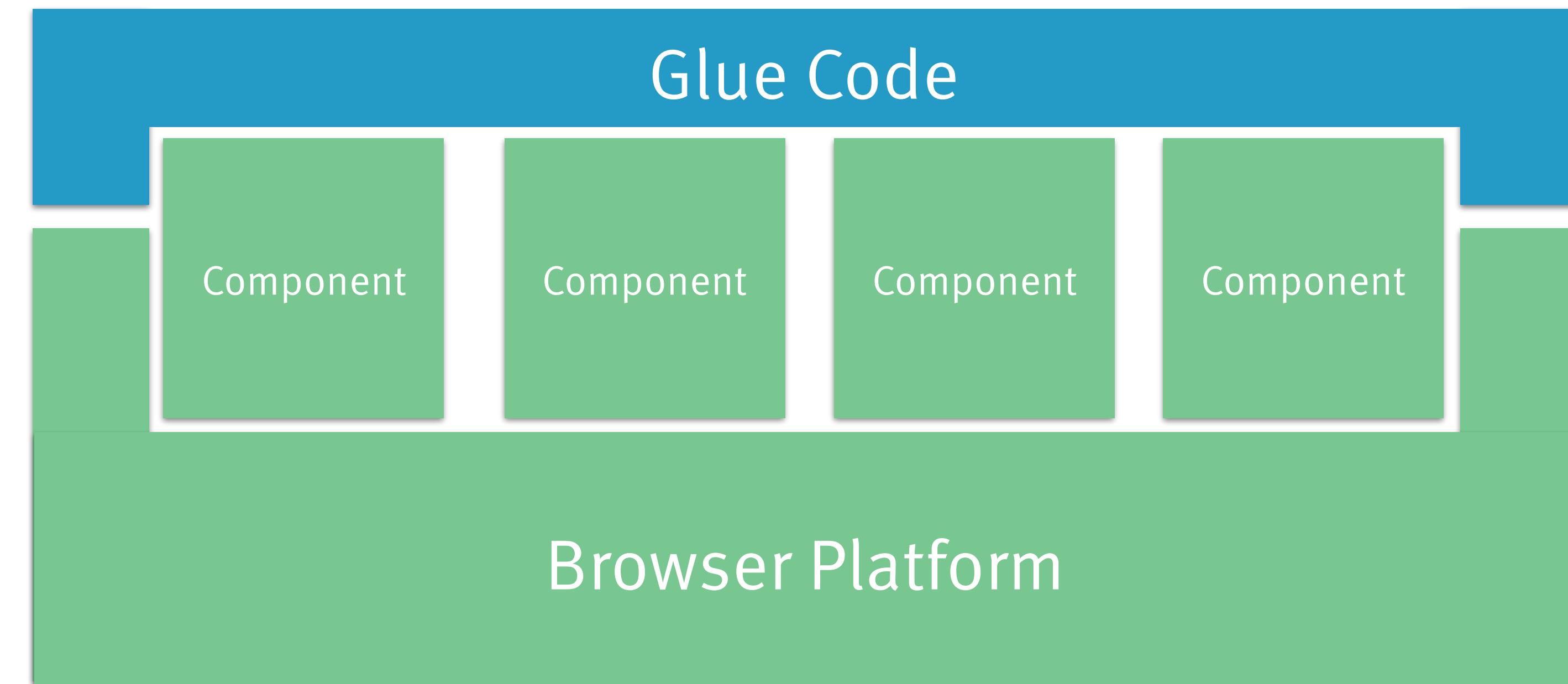
```
<tab-nav>
<ul>
  <li> <a href="#about">About</a>
  ...
  <li> <a href="#comments">Discussion</a>
</ul>
</tab-nav>

<tab-contents>
  <p id="about">lorem ipsum dolor sit amet</p>
  ...
  <ol id="comments"> ... </ol>
</tab-contents>
```

\$(".tabs").tabs();



Custom Elements



OLD MAN YELLS AT CLOUD



Old man yells at cloud
with mouth wide open
yellow cloud looks back
and says 'Kah'
old man goes 'Huh?'
yellow cloud says 'I'm
not a cloud I'm a cloud'
old man goes 'Huh?'

Why the hate?

General benefits

**Classic, server-side rendered web applications
are much simpler, use less bandwidth, scale
better, are more resilient, and a good match
for most user needs**

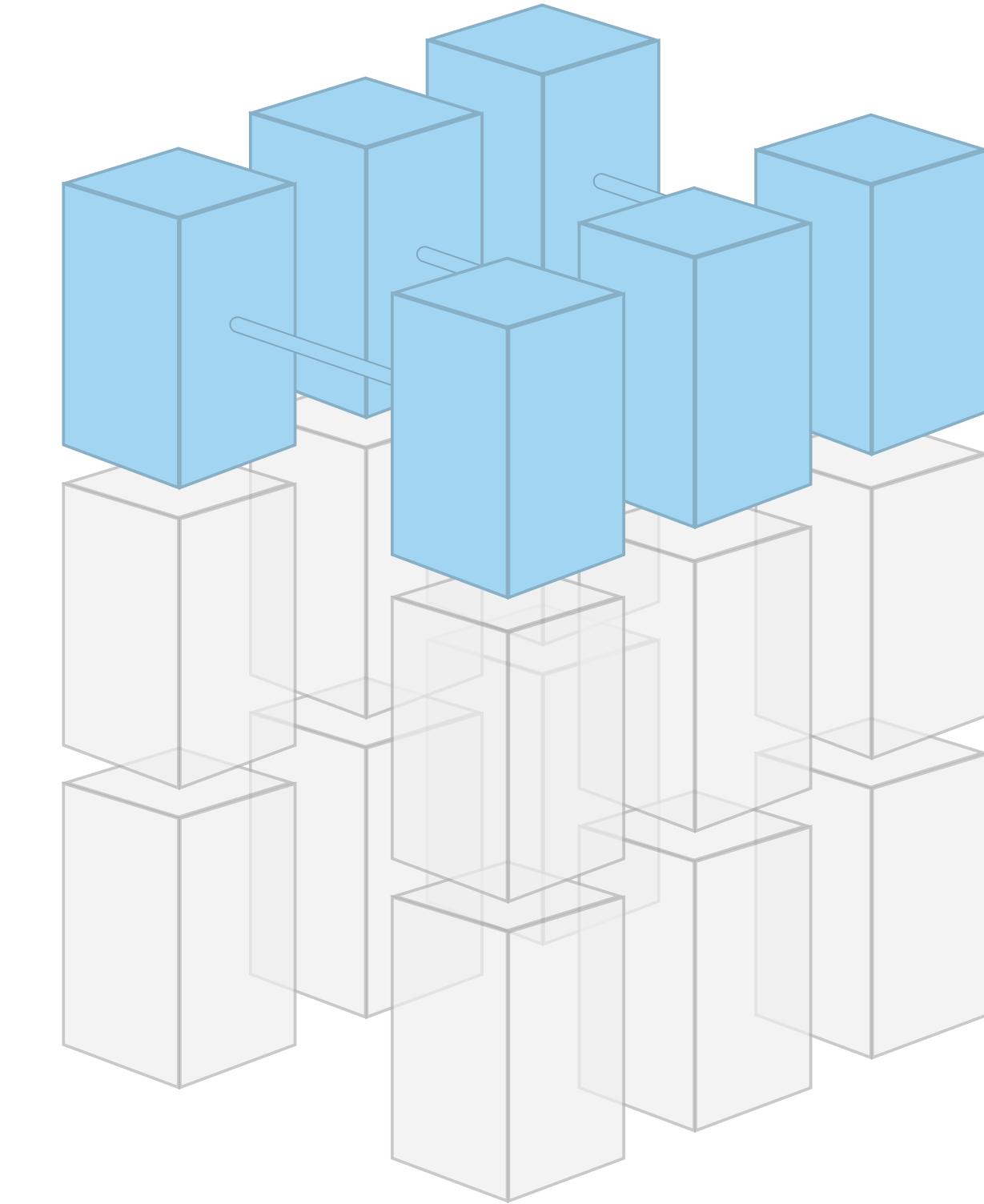
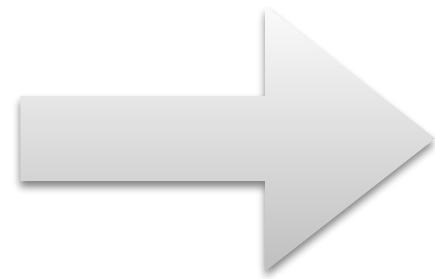
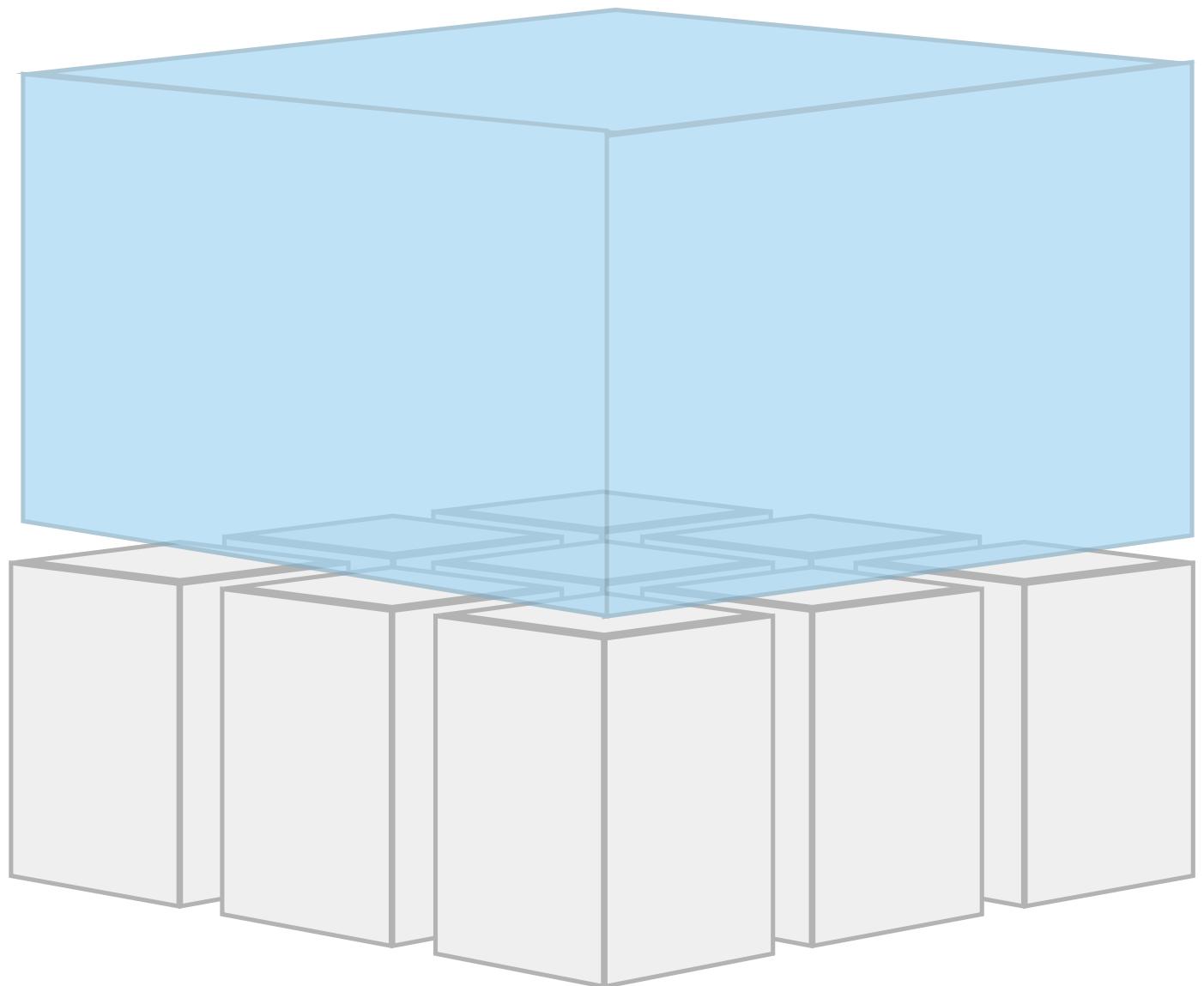
**Many single-page apps are built carelessly
and improve developer experience (if at all) at
the cost of decreased usability and
architectural complexity**

Large-scale benefits

Why choose a monolith if you don't have to?

Any sufficiently complicated JavaScript client application contains an ad hoc, informally-specified, bug-ridden, slow implementation of half a browser.

(Me, with apologies to Phillip Greenspun)



In-page
JavaScript method calls
Shared abstractions & frameworks
Common language runtime
HTML 5 JS platform

Cross-page
Links & redirects
Micro-architecture
HTTP
Standard Browser

**Sometimes, JS-centric applications/SPAs are
the right choice, but very rarely for every part
of your system**

If you're a fan of single page apps, at least build more than one

- Don't reinvent browser integration features
- Accept some inefficiency
- Trade-off for framework independence
- Avoid modularity à la Java EE, OSGi etc.

Use your favorite SPA framework – when it's appropriate. Stick to basic, simpler stuff, when it's sufficient – likely, most of the time.

**That's all I have.
Thanks for listening!**

Stefan Tilkov
@stilkov
stefan.tilkov@innoq.com
Phone: +49 170 471 2625

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim am Rhein
Germany
Phone: +49 2173 3366-0

Ohlauer Straße 43
10999 Berlin
Germany
Phone: +49 2173 3366-0

Ludwigstr. 180E
63067 Offenbach
Germany
Phone: +49 2173 3366-0

Kreuzstraße 16
80331 München
Germany
Phone: +49 2173 3366-0

innoQ Schweiz GmbH

Gewerbestr. 11
CH-6330 Cham
Switzerland
Phone: +41 41 743 0116



www.innoq.com

SERVICES

- Strategy & technology consulting
- Digital business models
- Software architecture & development
- Digital platforms & infrastructures
- Knowledge transfer, coaching & trainings

FACTS

- ~150 employees
- Privately owned
- Vendor-independent

OFFICES

- Monheim
- Berlin
- Offenbach
- Munich
- Hamburg
- Zurich

CLIENTS

- Finance
- Telecommunications
- Logistics
- E-commerce
- Fortune 500
- SMBs
- Startups