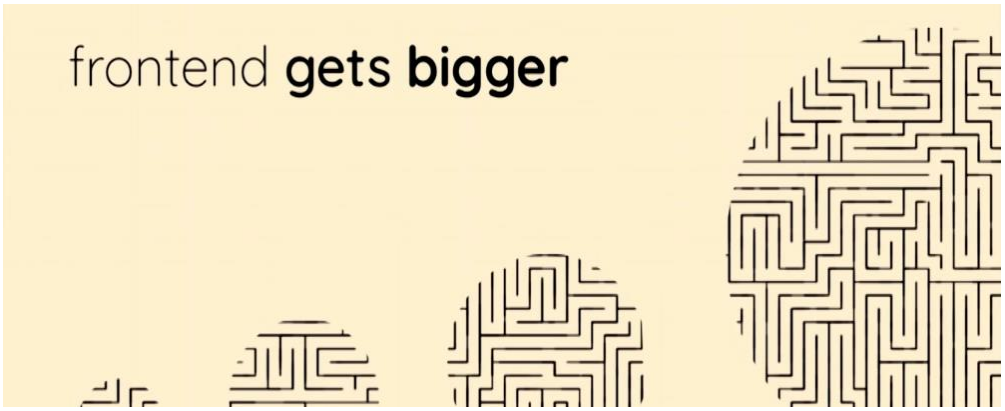
Microservices are the backend answer to handling complexity and work efficiently with multiple teams. Many frontend apps are also reaching the size where development gets harder and changing fundamental things is too risky. The software does not fit in one frontend developers head any more. Micro Frontends is a way of cutting your software into pieces that can live on their own and are each operated by an end-to-end team. It's not microservices for frontend people. Think of it as full stack microservices that come with database and user interface. This solves many issues. One of them is bridging the gap between frontend and backend developers.

Integrating the UI of multiple applications in the browser, so that the user still gets a seamless and coherent experience comes with a few challenges. This talk explains the Micro Frontends idea, when this concept is useful and what problems it tries to solve. It'll also highlight topics that you should think about when you build applications this way.



The frontend is also getting bigger

Michael Geers

Frontend Engineer

naltatis

on Twitter & GitHub

e-commerce

neuland:::

Büro für Informatik

Bremen, Germany

Micro Frontends

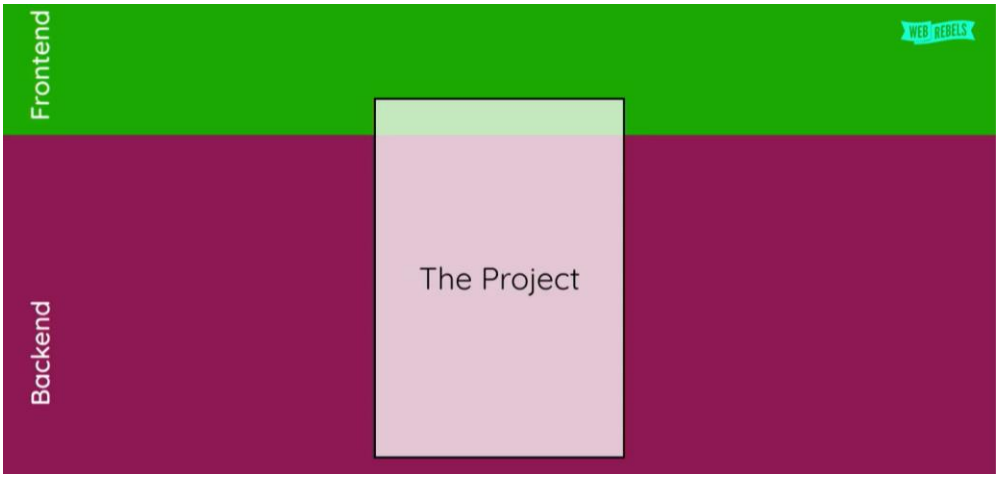
Verticalized Teams

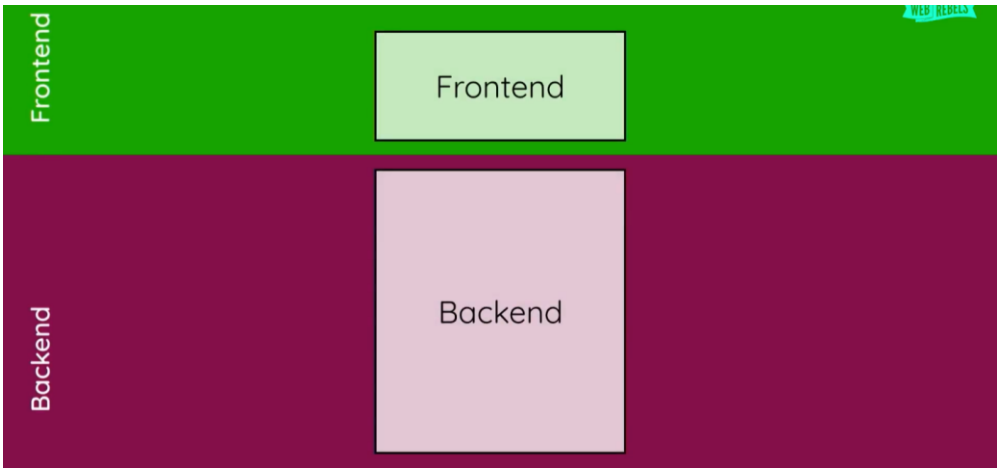
Self Contained Systems

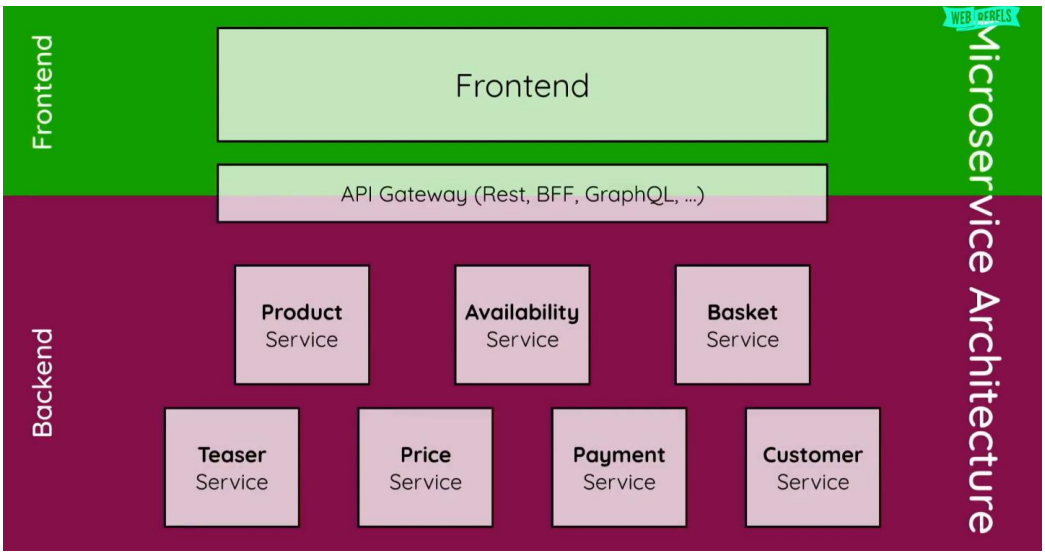
Vertical Decomposition

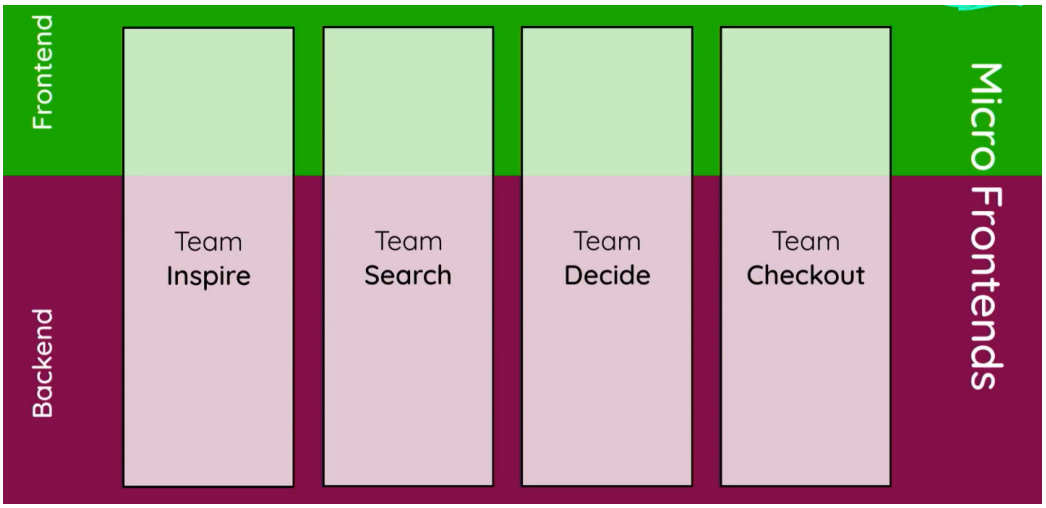
UI Composition

Architecture

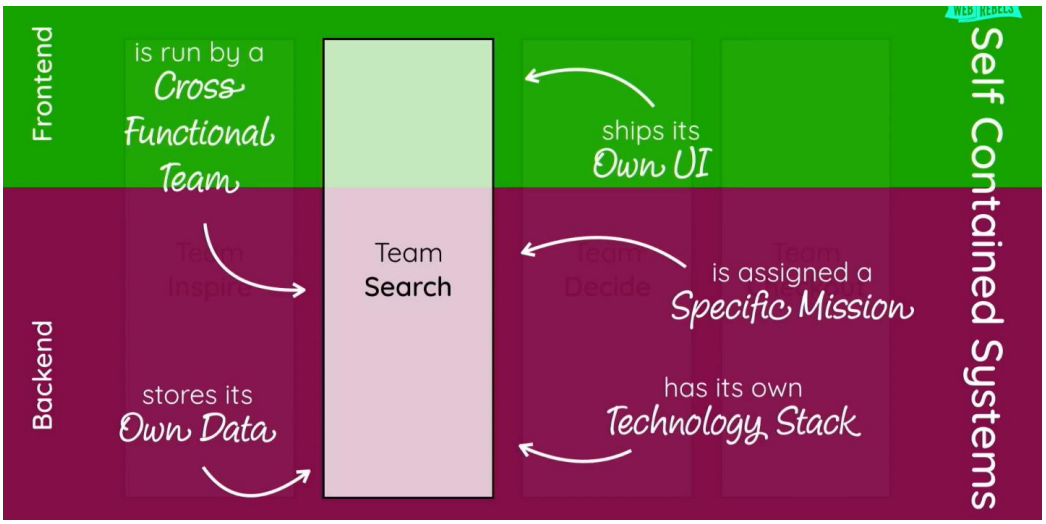

Frontend
Backend
The Project

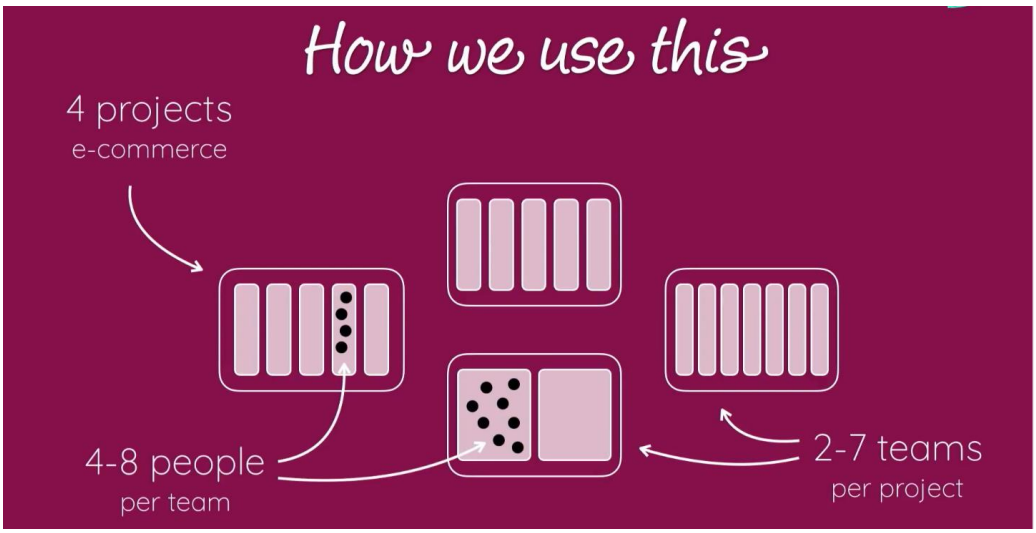

Frontend
Backend
Frontend
Backend


Frontend
Backend
Frontend
API Gateway (Rest, BFF, GraphQL, ...)
Product Service
Availability Service
Basket Service
Teaser Service
Price Service
Payment Service
Customer Service
Microservice Architecture



We can instead use a micro-frontend approach and have vertical slices like the above with each team building out





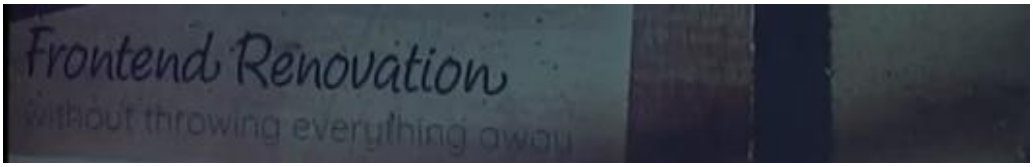

Others who work like this
Otto
Ikea
Facebook
Hello Fresh
Zalando
Amazon?
Office 365
AutoScout24
Spotify
FINN.no
gutefrage

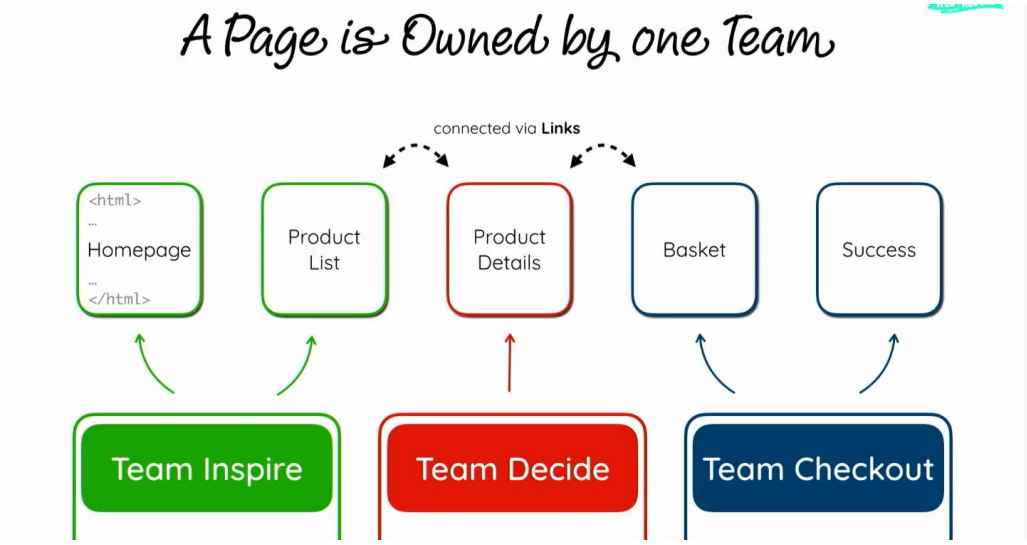

Why Micro Frontends?

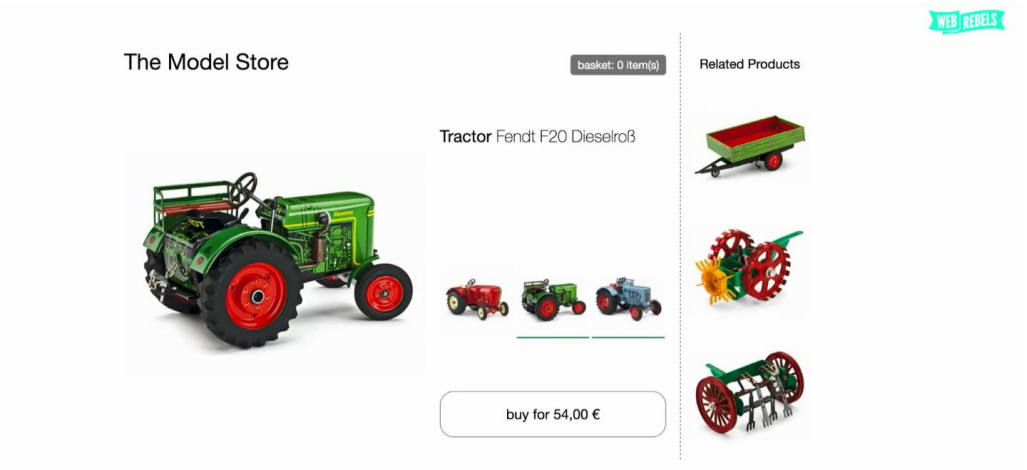

Small Autonomous Teams
with a clear mission

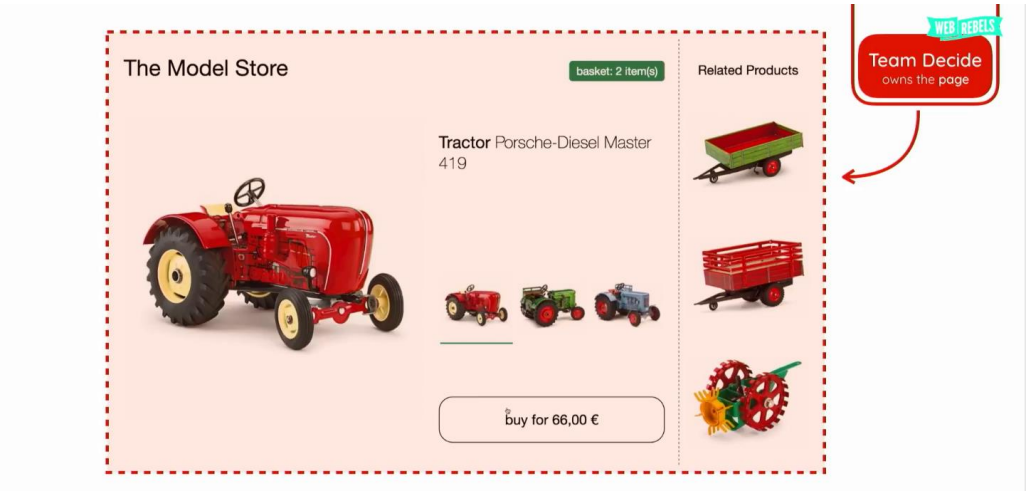

Customer Focus
every team ships directly
no pure api teams

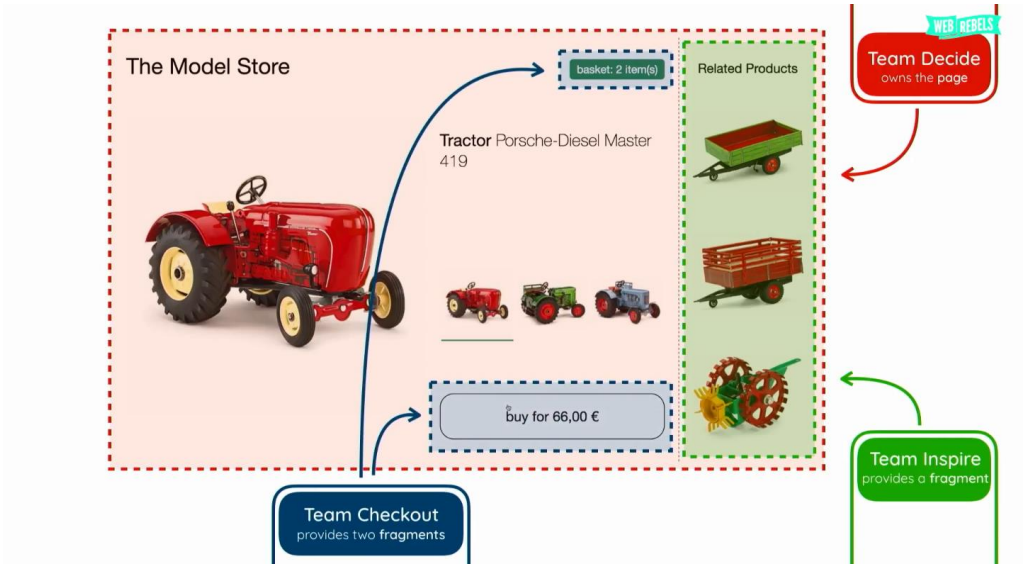

Reduced Scope
everything fits
into memory again


Frontend Renovation
without throwing everything away

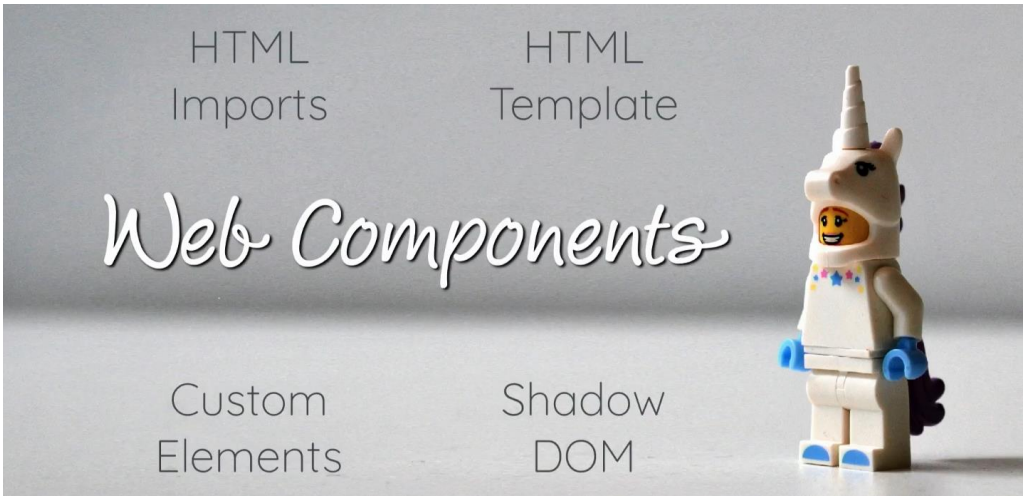

A Page is Owned by one Team
connected via Links
<html>
...
Homepage
...
</html>
Product List
Product Details
Basket
Success
Team Inspire
Team Decide
Team Checkout


WEB REBELS
The Model Store
basket: 0 item(s)
Related Products
Tractor Fendt F20 Dieselroß
buy for 54,00 €


WEB REBELS
Team Decide
owns the page
The Model Store
basket: 2 item(s)
Related Products
Tractor Porsche-Diesel Master 419
buy for 66,00 €



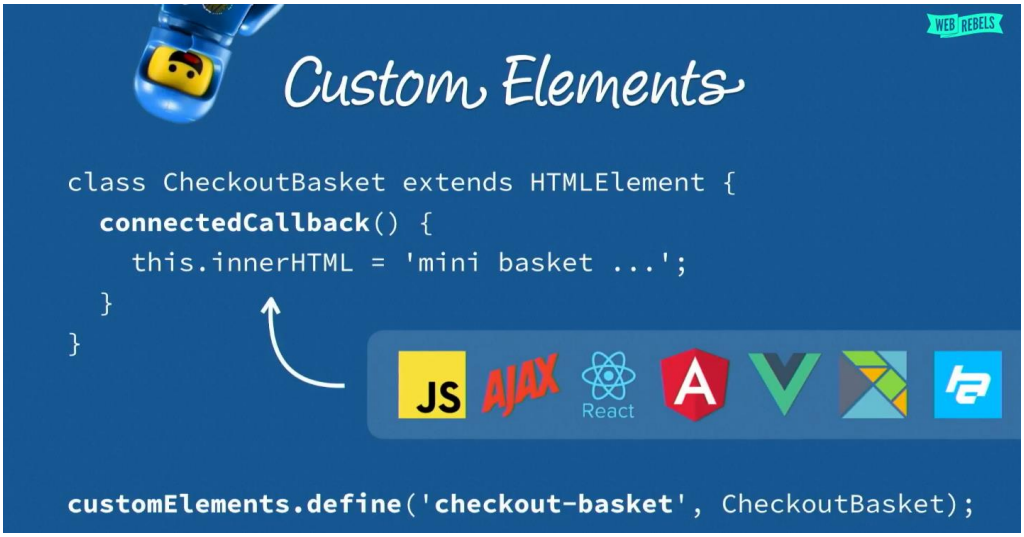
Integration in the Browser

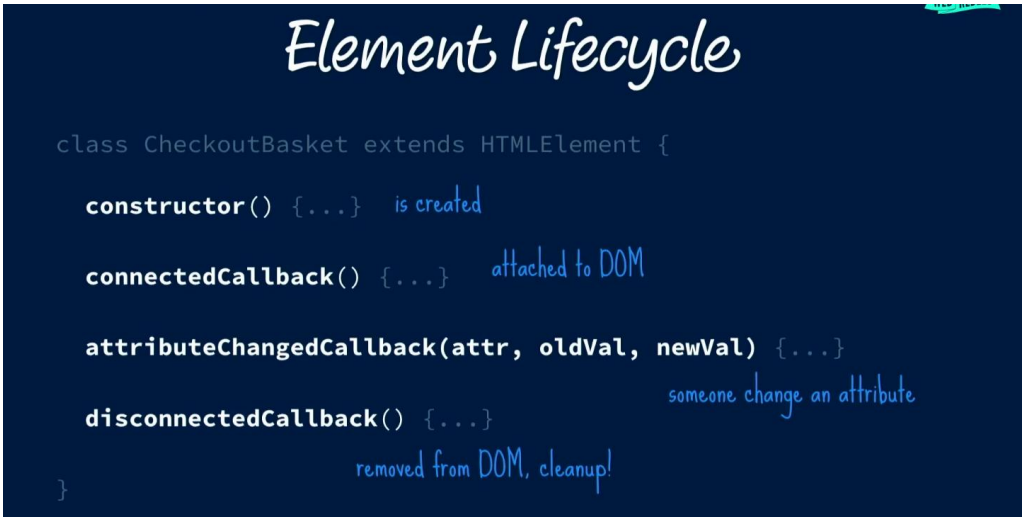


Custom Elements

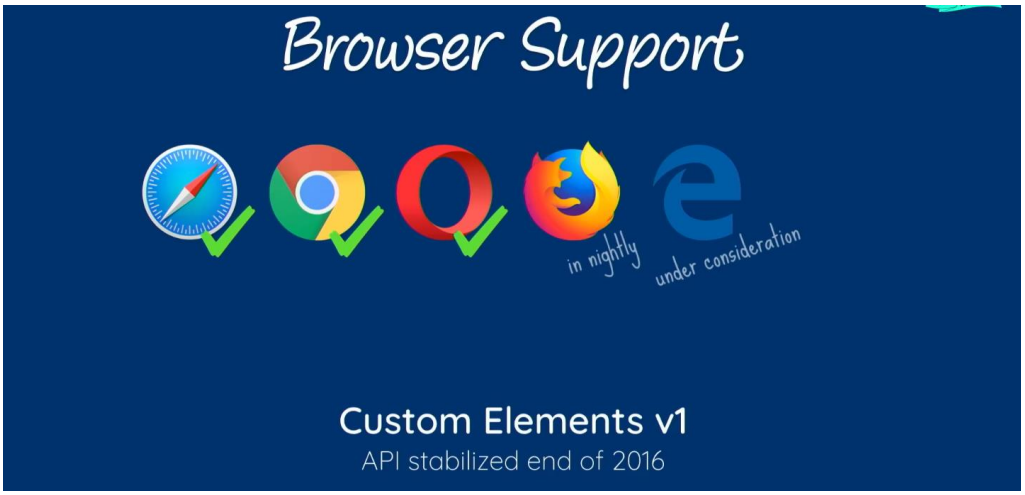
```
<checkout-basket></checkout-basket>
```

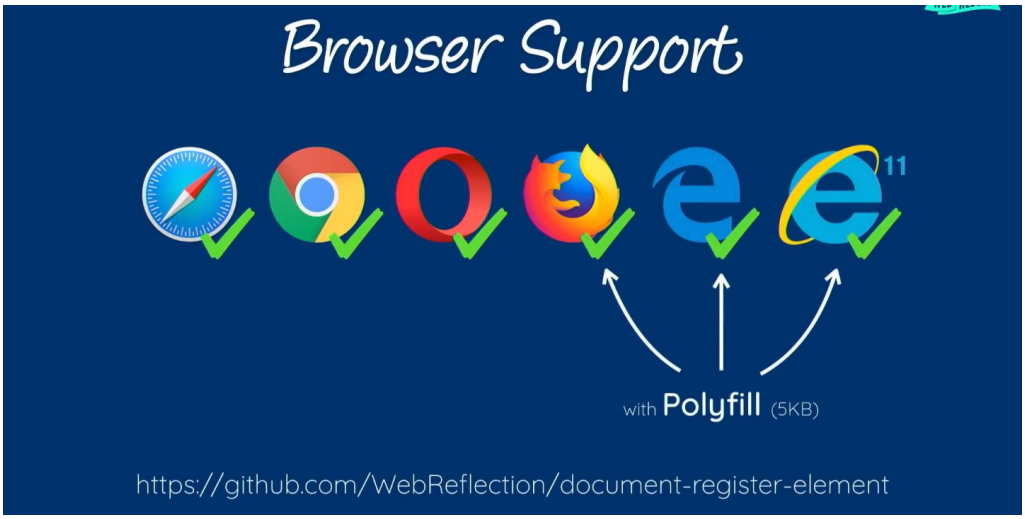
https://developers.google.com/web/fundamentals/getting-started/primers/customelements

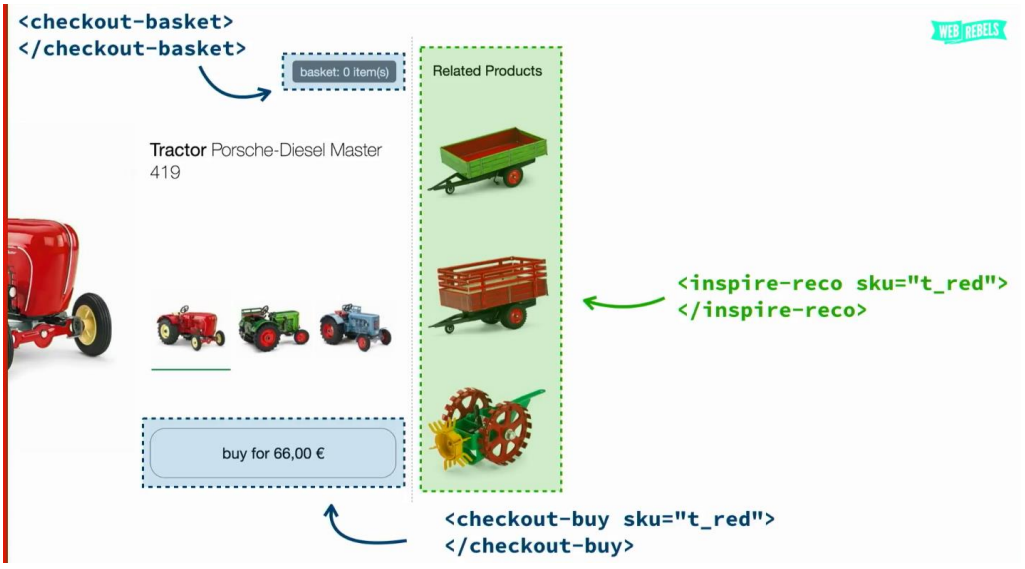

WEB REBELS
Custom Elements
class CheckoutBasket extends HTMLElement {
connectedCallback() {
this.innerHTML = 'mini basket ...';
}
}
JS
AJAX
React
customElements.define('checkout-basket', CheckoutBasket);

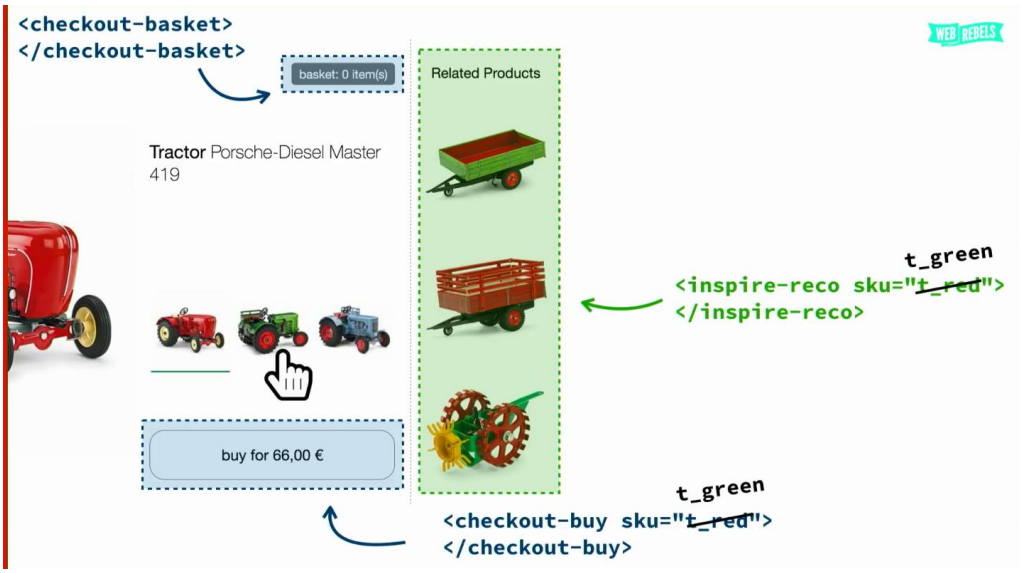

Element Lifecycle
class CheckoutBasket extends HTMLElement {
constructor() {...} is created
connectedCallback() {...} attached to DOM
attributeChangedCallback(attr, oldVal, newVal) {...}
someone change an attribute
disconnectedCallback() {...}
removed from DOM, cleanup!
}


Custom Elements by Default
stencil
svelte
SkateJS
Angular Elements
hyperHTML Element

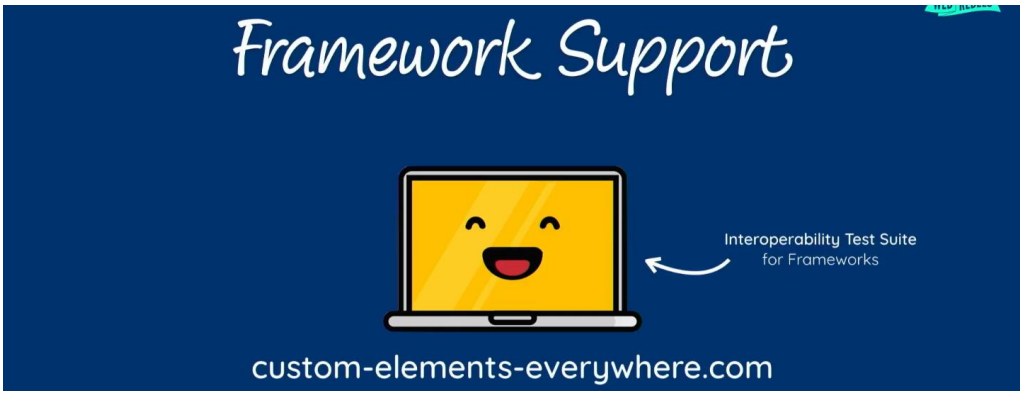

Browser Support
in nightly
under consideration
Custom Elements v1
API stabilized end of 2016

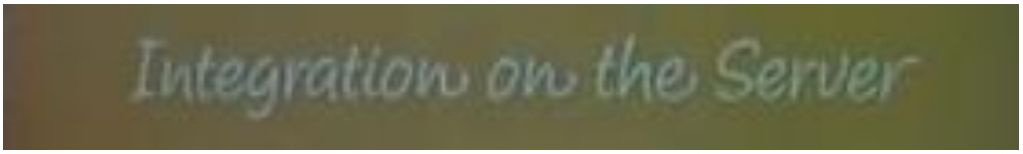

Browser Support
11
with Polyfill (5KB)
https://github.com/WebReflection/document-register-element


<checkout-basket>
</checkout-basket>
basket: 0 item(s)
Related Products
Tractor Porsche-Diesel Master 419
buy for 66,00 €
<inspire-reco sku="t_red">
</inspire-reco>
<checkout-buy sku="t_red">
</checkout-buy>



The red team just has to change the sku and emit an event to get the DOM to re-render again for all the affected fragments





Related Products

Server Initial Render

```
/inspire-reco?sku=t_red

res.send("<h3>Related Products</h3><ul>…")
```

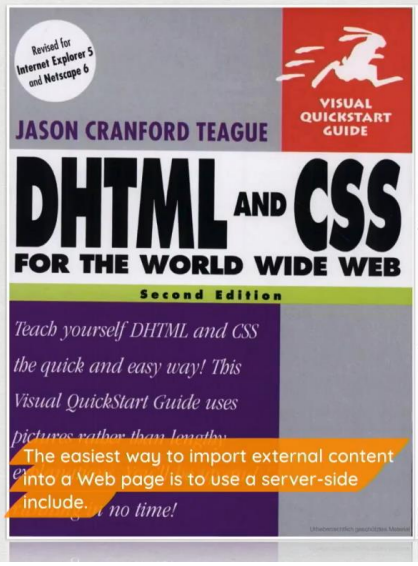
Browser Rehydration & User Interaction

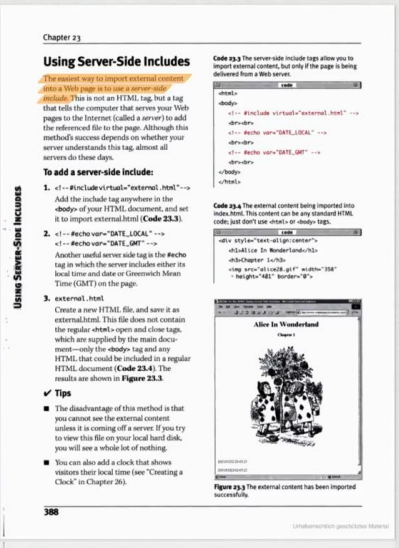
```
<inspire-reco sku="t_red"></inspire-reco>

el.innerHTML = "<h3>Related Products</h3><ul>…")
```

SSI Server Side Includes

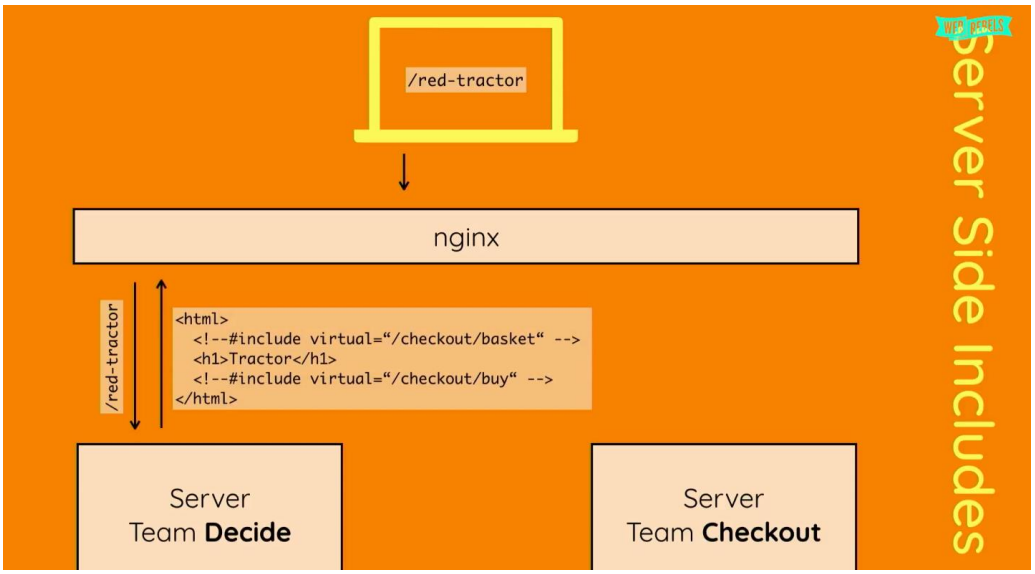
```
<!--#include virtual="/some-url" -->
```



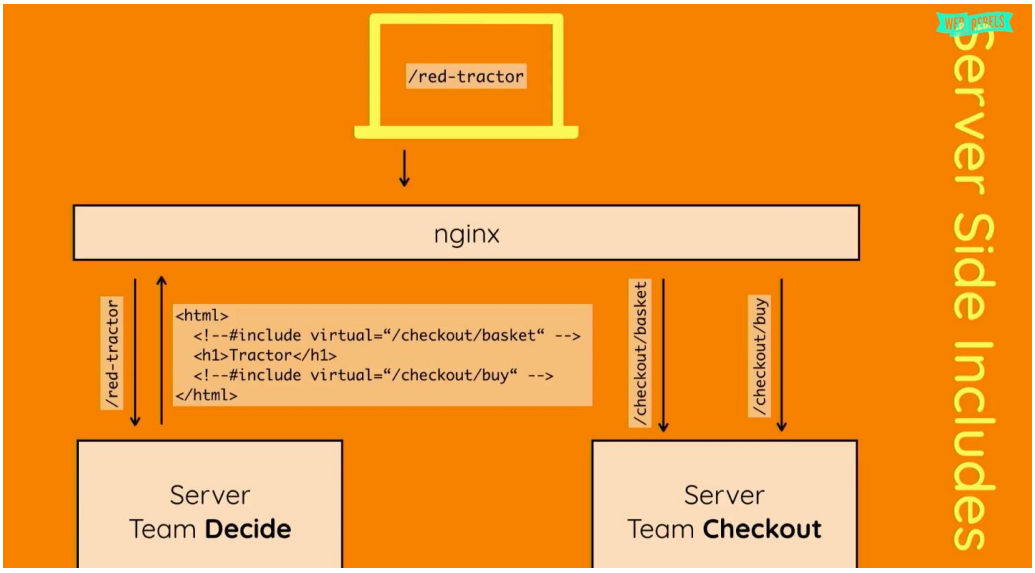


2001

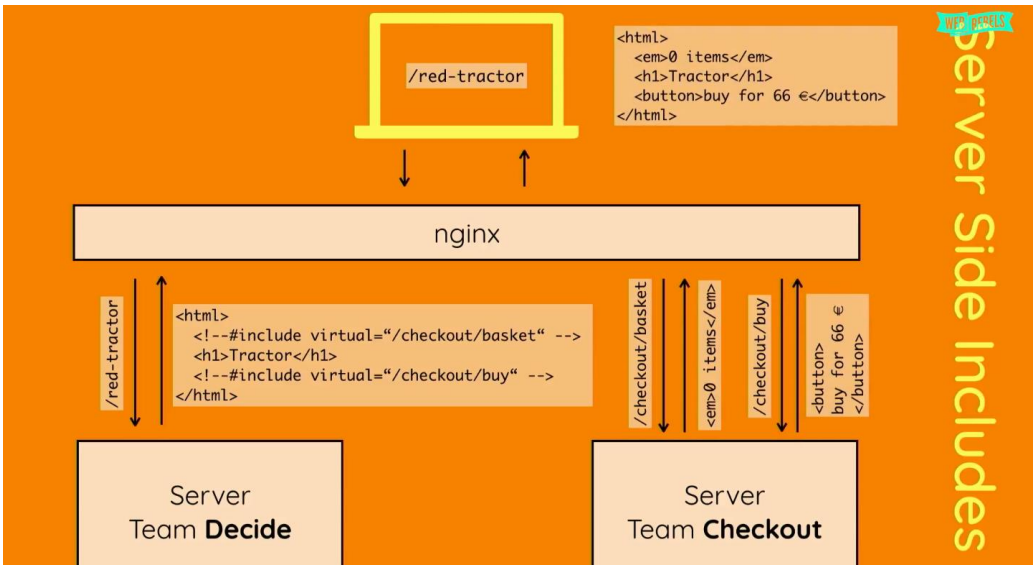
Google Books



When a request is made, the nginx web server decides which team is responsible for the request's URL and forwards the request to the correct endpoint. The complete page HTML is returned by the team server with any fragments needed as SSI tags.

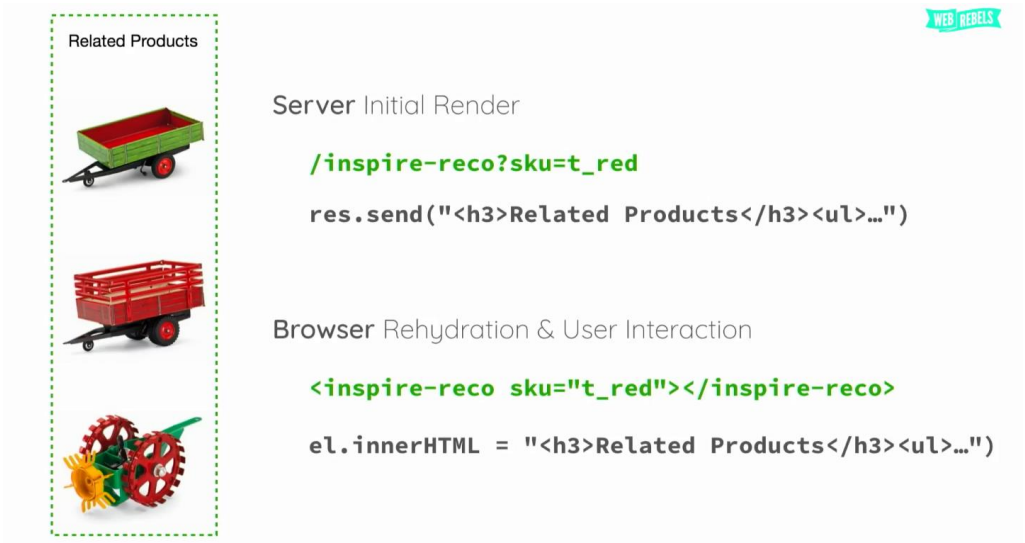


The nginx web server parses the returned HTML and makes the needed parallel calls to get the fragments from other team server locations.

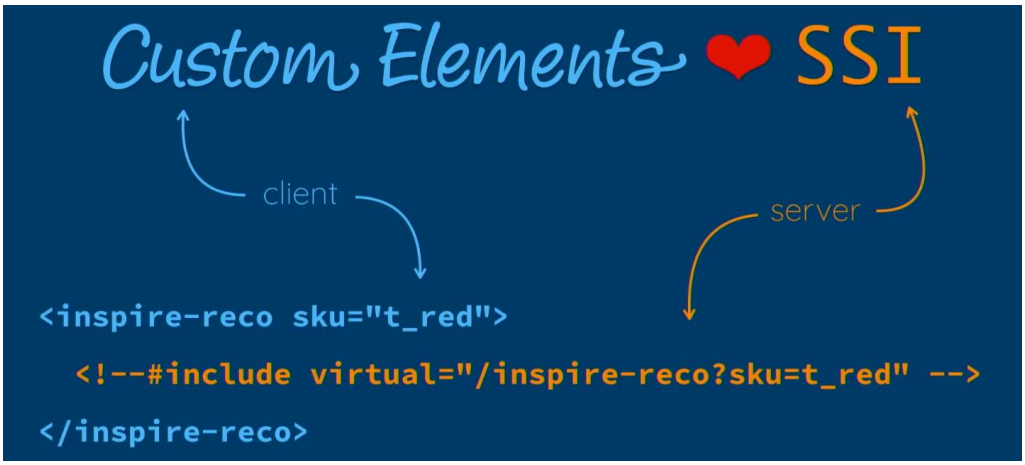


The fragments are inserted into the page by the nginx web server and the complete HTML is returned to the client

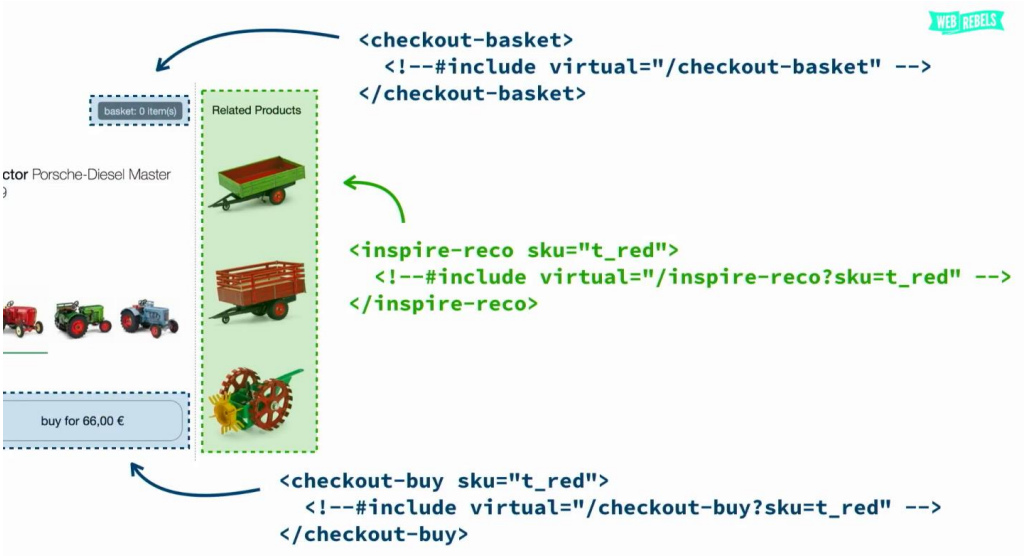
Web components are just a concept that exist in the browser only, how can we benefit using the server?

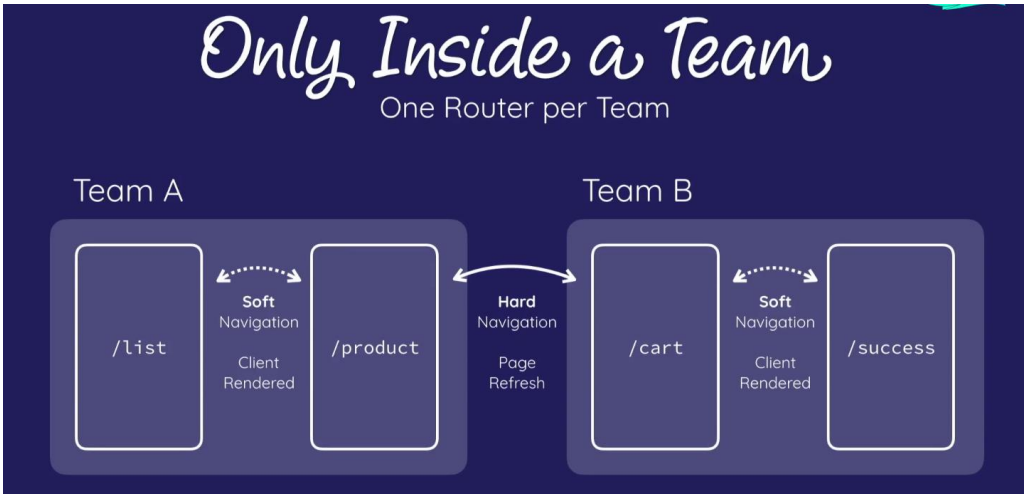


It is possible to render the content of a complete fragment on a server and do the same on the client

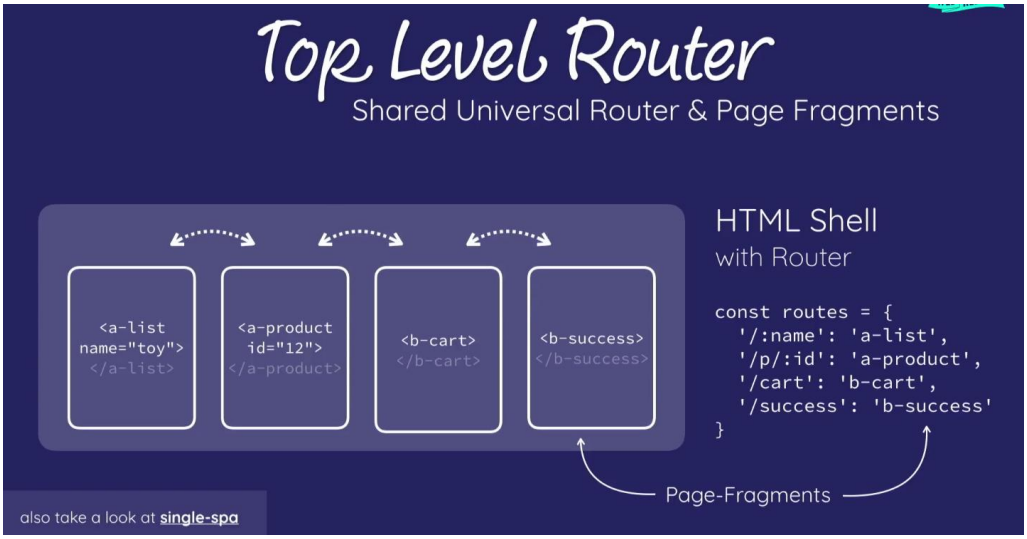


Once the server has initially inserted the SSI fragment, the client can then take over the re-render of that fragment whenever the content changes due to an event or change within the browser

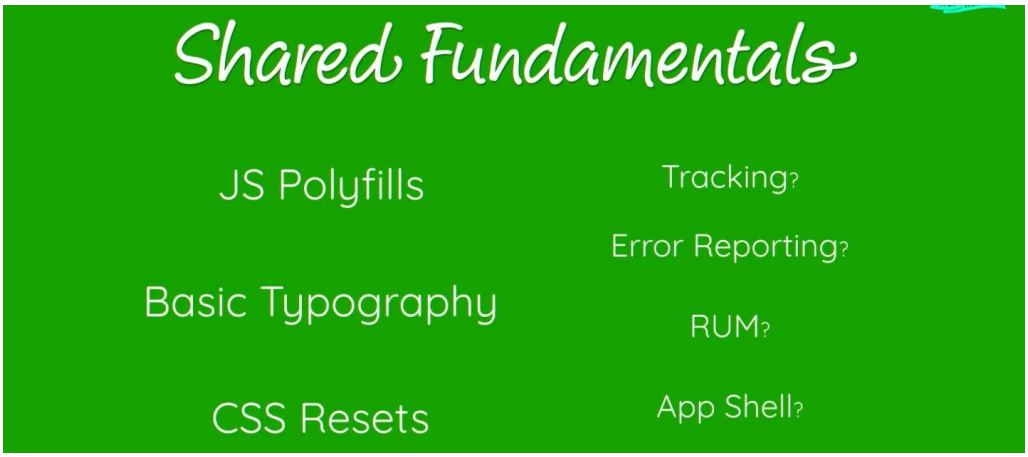


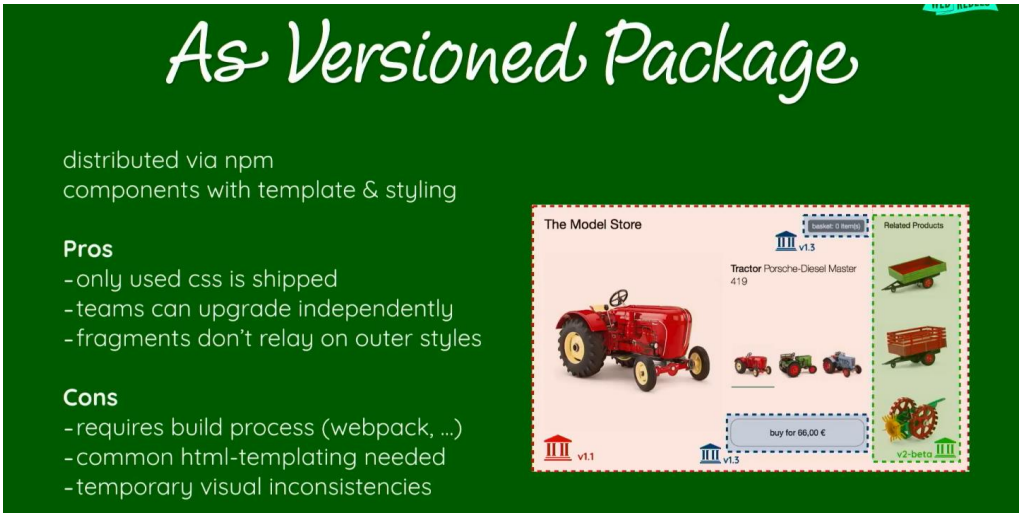


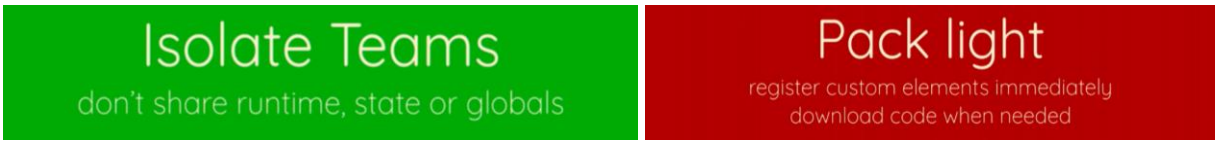
The downside with this approach is that a large number of teams will lead to more page refreshes and hard navigations that degrade user experience

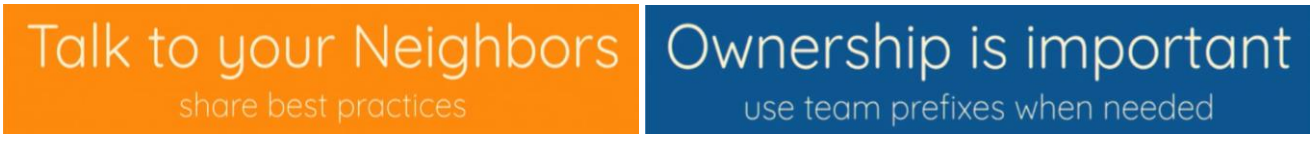


In this approach, there is a HTML Shell that handles the outer page and navigation. Other teams will have to deliver fragments and routes without the header and body HTML tags to be inserted into the shell.









HTTP/2 allows to load multiple JS files/bundles in parallel if needed. You don't need to include all of Angular for small app, only load library when the user needs it. Since all team code share the same runtime, it is important to use team prefixes to name event and CSS to identify when things go wrong. Try to make your design style into small libraries that teams can pull in when needed.


#perfmatters


Be reasonable!
don't use all frameworks together!
do you really need a big framework?


Measure Performance
set a performance budget & know your numbers


Don't Share State
no shared globals


Value Independence
its tempting to unify everything


Learn from Colleagues
teach them how CSS works


Talk to your Neighbors
share best practices


Ownership is important


Credits
Colorful Wood Structure Alexas_Fotos
https://pixabay.com/en/background-colorful-wood-structure-3412048/
Clocks Pixabay
https://www.pexels.com/photo/london-new-york-tokyo-and-moscow-clocks-48770/
Octan Tower The Lego Movie
https://www.youtube.com/watch?v=GZT5z4K6uw4
recaptioned by @peeofive
Action Bicycling Roman Pohorecki
https://www.pexels.com/photo/action-bicycling-bike-biking-287398/
Four Playing Pieces Pixabay
https://www.pexels.com/photo/set-of-4-wooden-elongated-accessory-209651/
Wooden Counter David Siglin
https://www.pexels.com/photo/blur-blurry-bokeh-close-up-347139/
The Tool Guy Tirachard Kumtanom
https://unsplash.com/photos/UuW4psOb388
Woodshop Igor Ovsyannykov
https://unsplash.com/photos/iXV0i4Wo4yc
Unicorn d97jro
https://pixabay.com/de/lego-einhorn-spielzeug-671593/
Browser Logos Cătălin Mariş
https://github.com/alrra/browser-logos
Broken Escalator Skitterphoto
https://pixabay.com/photo-1746279
https://freesound.org/people/dobroide/sounds/15227/
https://freesound.org/people/sittstav/sounds/171665/
Sad Lego Benny Cheezburger
http://gph.is/1BCBCKh
Wired Elements
https://wiredjs.com/
Tractors Manufactum
https://www.manufactum.com/tin-toys-c193667/
built with Micro Frontends