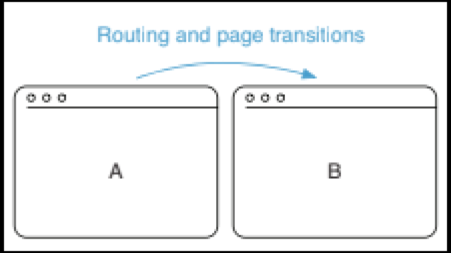
THE STATE OF MICRO FRONTENDS

Tommaso, Ruscica | Consultant

REPLY

01_PAGES_LINKS

- **CDN**
- **TEAM-DECIDE**
- **TEAM-INSPIRE**

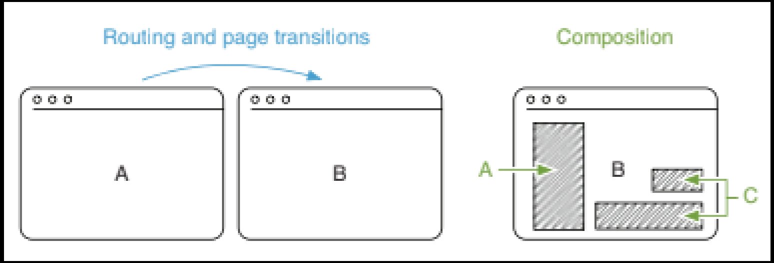


Team-decide owns the product page, team-inspire owns the recommendations.
Each team delivers their feature on a separate page.
They use (hard) links to connect the pages.

npm run 01_pages_links

02_IFRAME

- **CDN**
- **TEAM-DECIDE**
- **TEAM-INSPIRE**



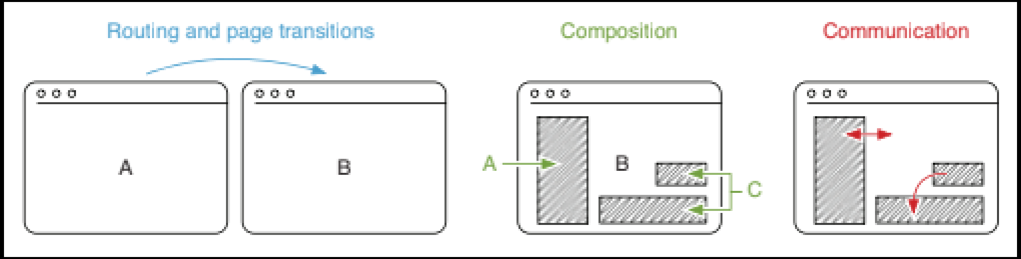
The plan is to integrate the recommendations into the product page itself using an iFrame.

```
<iframe src="http://localhost:3002/recommendations/porsche"></iframe>
```

npm run 02_iframe

03_AJAX

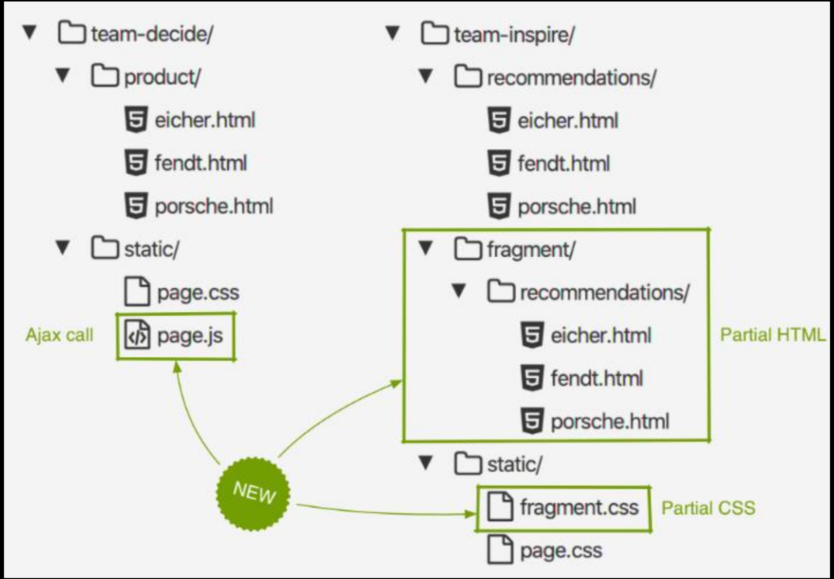
- CDN
- TEAM-DECIDE
- TEAM-INSPIRE



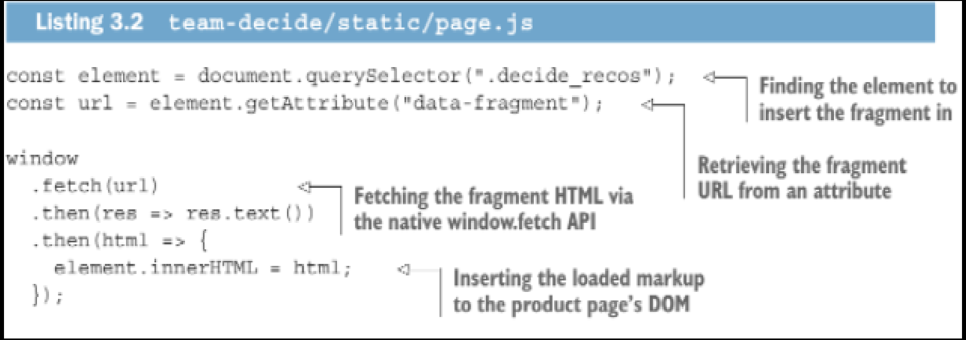
Team inspire will deliver the recommendations as a fragment - a snippet of HTML, loaded by other team (team decide) by ajax call (page.js) and integrated into DOM.
Templates inside "team-inspire\recommendations" are fallback in case AJAX call fails!

npm run 03_ajax

03_AJAX



<link href="http://localhost:3002/static/fragment.css" rel="stylesheet" />



Listing 3.2 `team-decide/static/page.js`

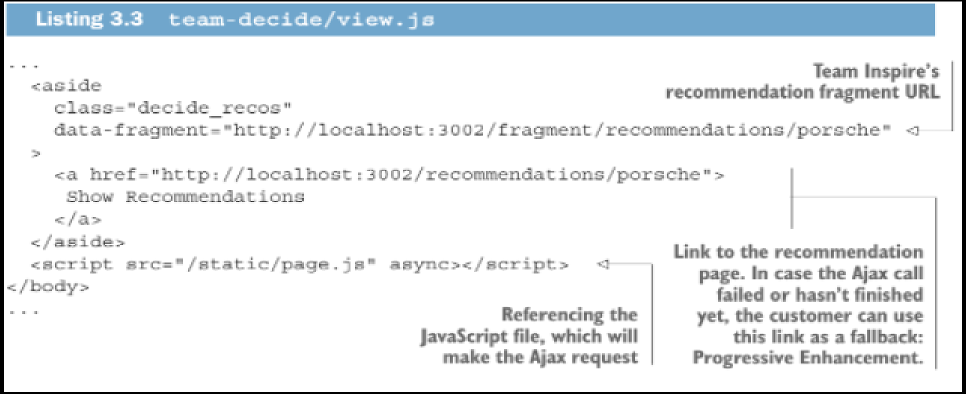
```
const element = document.querySelector(".decide_recos");
const url = element.getAttribute("data-fragment");

window
  .fetch(url)
  .then(res => res.text())
  .then(html => {
    element.innerHTML = html;
  });
```

Finding the element to insert the fragment in

Retrieving the fragment URL from an attribute

Fetching the fragment HTML via the native window.fetch API

Inserting the loaded markup to the product page's DOM



Listing 3.3 `team-decide/view.js`

```
...
  <aside
    class="decide_recos"
    data-fragment="http://localhost:3002/fragment/recommendations/porsche"
  >
    <a href="http://localhost:3002/recommendations/porsche">
     Show Recommendations
    </a>
  </aside>
  <script src="/static/page.js" async></script>
</body>
...
```

Team Inspire's recommendation fragment URL

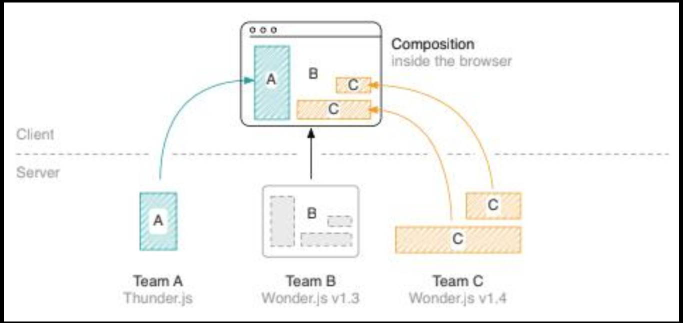
Link to the recommendation page. In case the Ajax call failed or hasn't finished yet, the customer can use this link as a fallback: Progressive Enhancement.

Referencing the JavaScript file, which will make the Ajax request

npm run 03_ajax

04_WEB_COMPONENTS

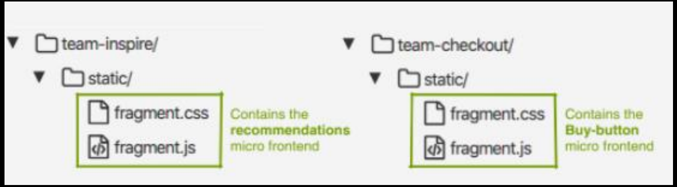
- **CDN**
- **TEAM-DECIDE**
- **TEAM-INSPIRE**
- **TEAM-CHECKOUT**

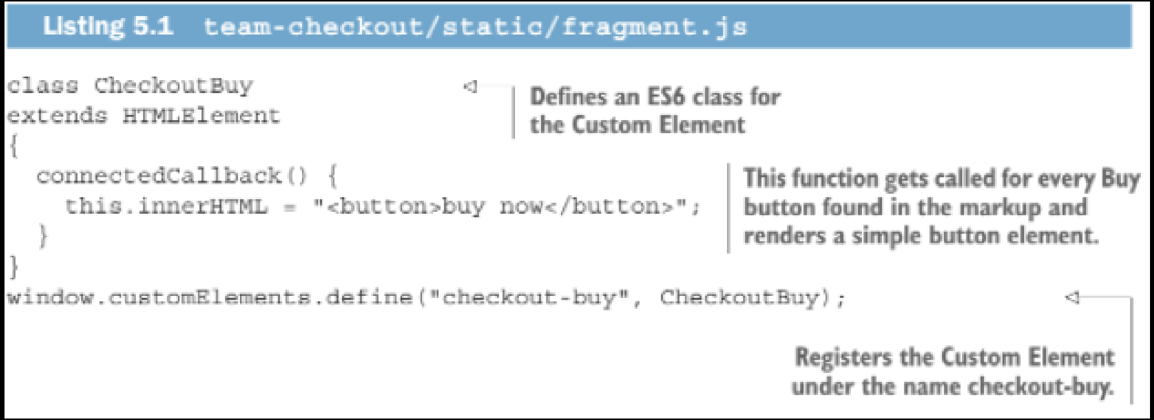


Team Checkout chose to go with client-side rendering for their user interfaces. They've implemented the checkout pages as a single page app (SPA). The Buy-button fragment is available as a standalone Web Component. Same for team Inspire.

npm run 04_web_components

04_WEB_COMPONENTS





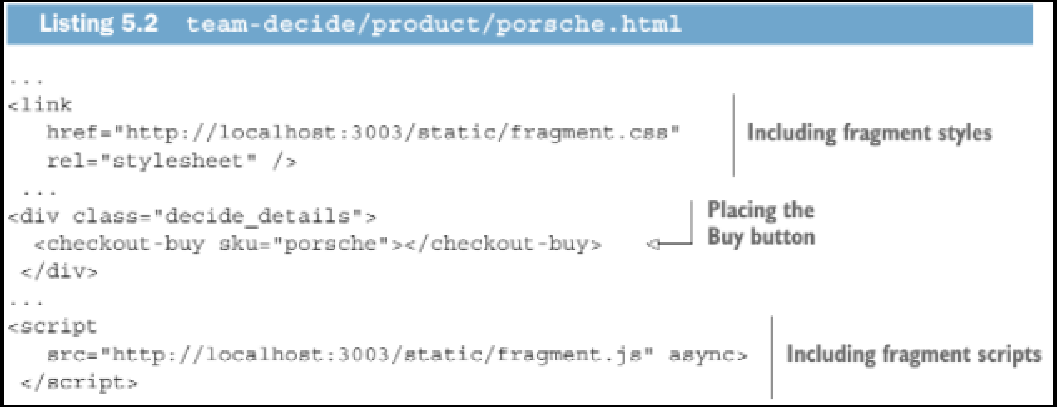
Listing 5.1 `team-checkout/static/fragment.js`

```
class CheckoutBuy
extends HTMLElement
{
  connectedCallback() {
    this.innerHTML = "<button>buy now</button>";
  }
}
window.customElements.define("checkout-buy", CheckoutBuy);
```

Defines an ES6 class for the Custom Element

This function gets called for every Buy button found in the markup and renders a simple button element.

Registers the Custom Element under the name checkout-buy.



Listing 5.2 `team-decide/product/porsche.html`

```
...
<link
   href="http://localhost:3003/static/fragment.css"
   rel="stylesheet" />
 ...
<div class="decide_details">
  <checkout-buy sku="porsche"></checkout-buy>
 </div>
...
<script
   src="http://localhost:3003/static/fragment.js" async>
 </script>
```

Including fragment styles

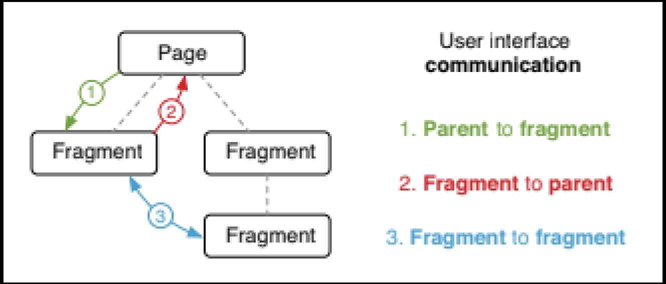
Placing the Buy button

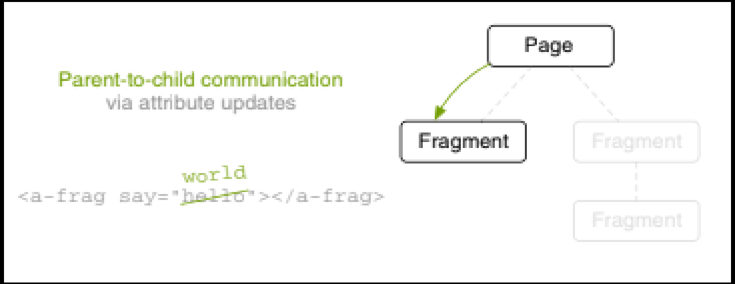
Including fragment scripts

npm run 04_web_components

05_PARENT_CHILD_FLOW

- CDN
- TEAM-DECIDE
- TEAM-INSPIRE
- TEAM-CHECKOUT

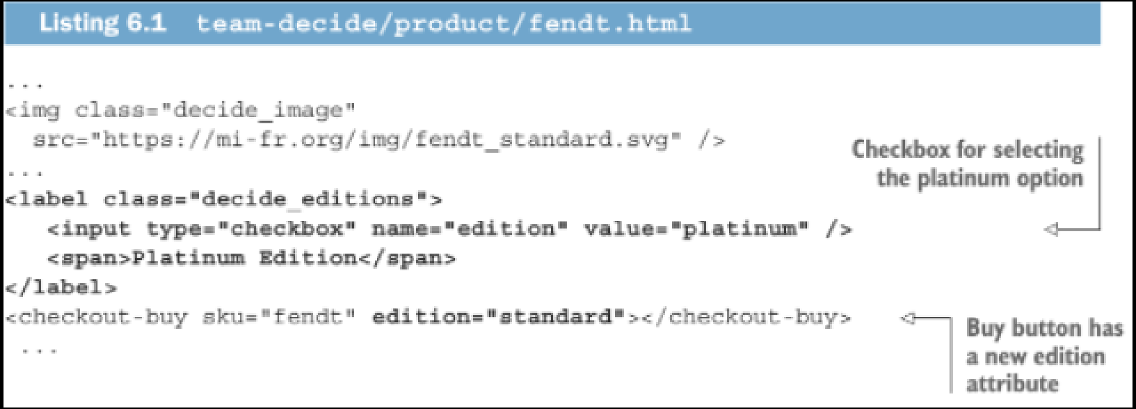



Parent-to-child communication via attribute updates
Page
Fragment
Fragment
Fragment
<a-frag say="~~hello~~ world"></a-frag>

Team Decide introduced a simple checkbox input element for choosing the upgrade item. The Buy-button component also received an edition attribute.

```
npm run 05_parent_child_flow
```

05_PARENT_CHILD_FLOW

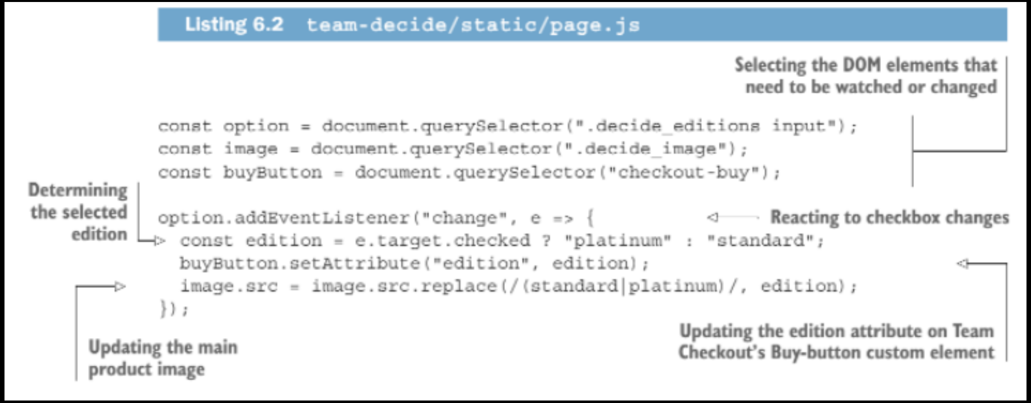


Listing 6.1 team-decide/product/fendt.html

```
...
<img class="decide_image"
  src="https://mi-fr.org/img/fendt_standard.svg" />
...
<label class="decide_editions">
   <input type="checkbox" name="edition" value="platinum" />
   <span>Platinum Edition</span>
</label>
<checkout-buy sku="fendt" edition="standard"></checkout-buy>
 ...
```

Checkbox for selecting the platinum option

Buy button has a new edition attribute



Listing 6.2 team-decide/static/page.js

Selecting the DOM elements that need to be watched or changed

```
const option = document.querySelector(".decide_editions input");
const image = document.querySelector(".decide_image");
const buyButton = document.querySelector("checkout-buy");

option.addEventListener("change", e => {
  const edition = e.target.checked ? "platinum" : "standard";
  buyButton.setAttribute("edition", edition);
  image.src = image.src.replace(/(standard|platinum)/, edition);
});
```

Reacting to checkbox changes

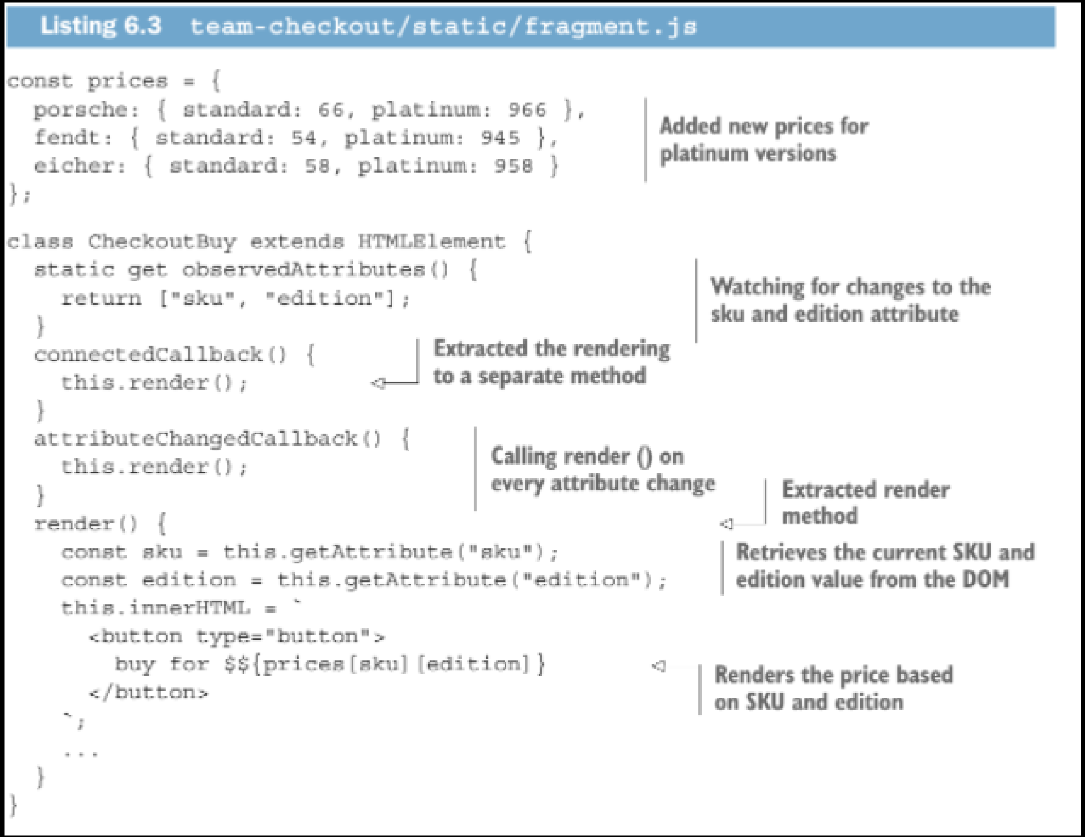
Determining the selected edition

Updating the edition attribute on Team Checkout's Buy-button custom element

Updating the main product image

npm run 05_parent_child_flow

05_PARENT_CHILD_FLOW



Listing 6.3 `team-checkout/static/fragment.js`

```
const prices = {
  porsche: { standard: 66, platinum: 966 },
  fendt: { standard: 54, platinum: 945 },
  eicher: { standard: 58, platinum: 958 }
};

class CheckoutBuy extends HTMLElement {
  static get observedAttributes() {
    return ["sku", "edition"];
  }
  connectedCallback() {
    this.render();
  }
  attributeChangedCallback() {
    this.render();
  }
  render() {
    const sku = this.getAttribute("sku");
    const edition = this.getAttribute("edition");
    this.innerHTML = `
      <button type="button">
        buy for $${prices[sku][edition]}
      </button>
    `;
    ...
  }
}
```

Added new prices for platinum versions

Watching for changes to the sku and edition attribute

Extracted the rendering to a separate method

Calling render () on every attribute change

Extracted render method

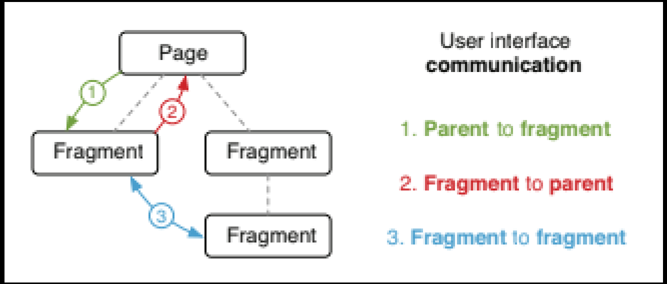
Retrieves the current SKU and edition value from the DOM

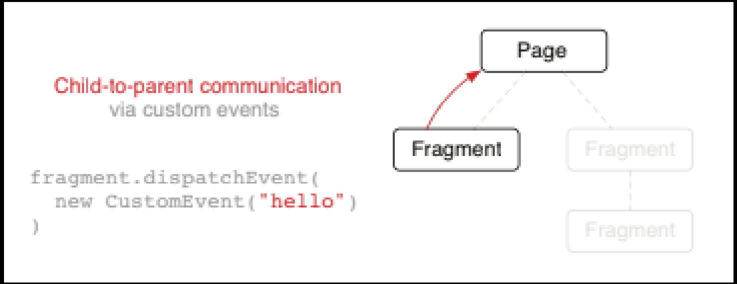
Renders the price based on SKU and edition

npm run 05_parent_child_flow

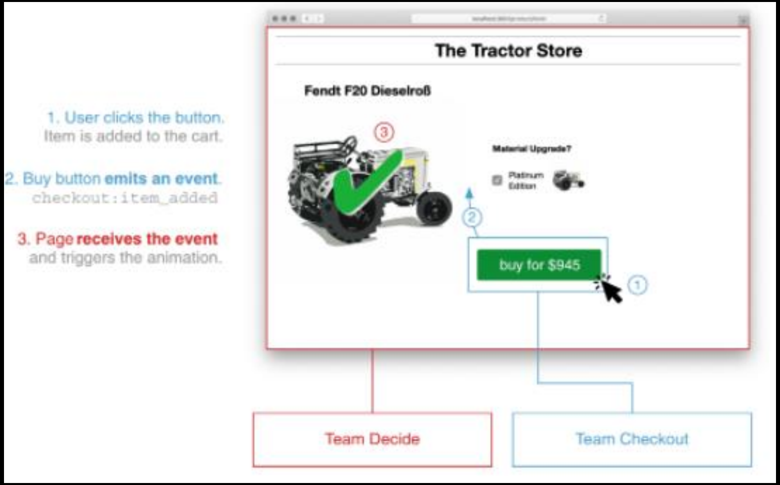
06_CHILD_PARENT_FLOW

- CDN
- TEAM-DECIDE
- TEAM-INSPIRE
- TEAM-CHECKOUT



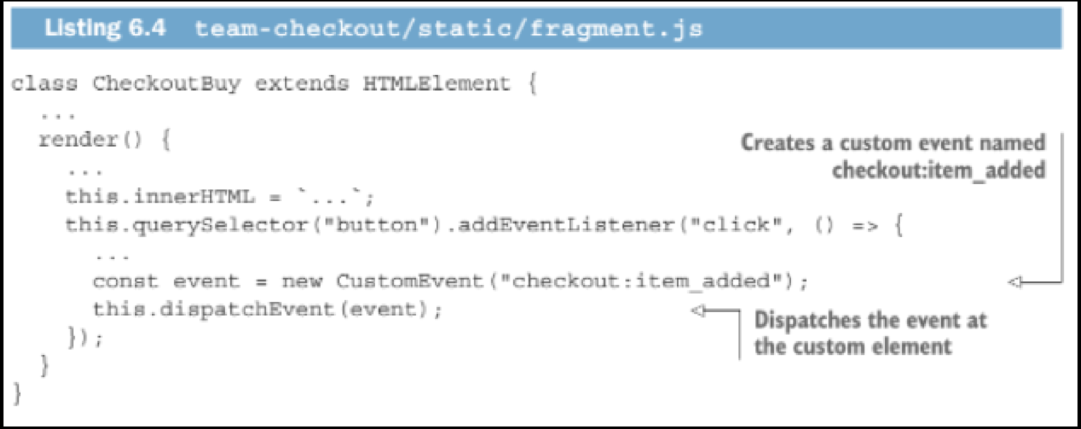


```
fragment.dispatchEvent(
  new CustomEvent("hello")
)
```


1. User clicks the button. Item is added to the cart.
2. Buy button emits an event.
checkout:item_added
3. Page receives the event and triggers the animation.
The Tractor Store
Fendt F20 Dieselroß
Material Upgrade?
Platinum Edition
buy for $945
Team Decide
Team Checkout

npm run 06_child_parent_flow

06_CHILD_PARENT_FLOW

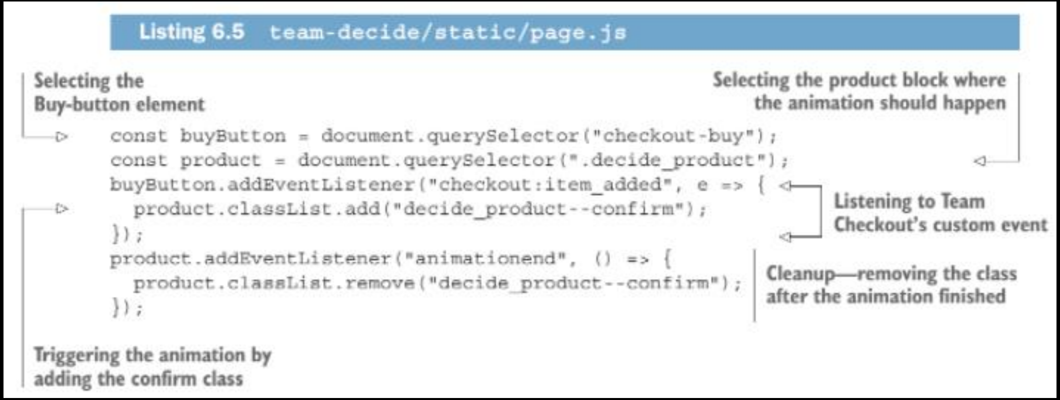


Listing 6.4 `team-checkout/static/fragment.js`

```
class CheckoutBuy extends HTMLElement {
  ...
  render() {
    ...
    this.innerHTML = `...`;
    this.querySelector("button").addEventListener("click", () => {
      ...
      const event = new CustomEvent("checkout:item_added");
      this.dispatchEvent(event);
    });
  }
}
```

Creates a custom event named checkout:item_added

Dispatches the event at the custom element



Listing 6.5 `team-decide/static/page.js`

Selecting the Buy-button element

Selecting the product block where the animation should happen

```
const buyButton = document.querySelector("checkout-buy");
const product = document.querySelector(".decide_product");
buyButton.addEventListener("checkout:item_added", e => {
  product.classList.add("decide_product--confirm");
});
product.addEventListener("animationend", () => {
  product.classList.remove("decide_product--confirm");
});
```

Listening to Team Checkout's custom event

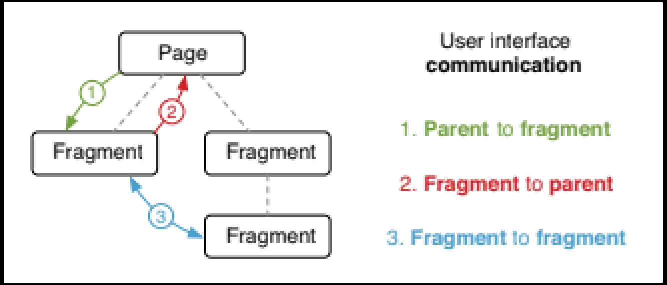
Cleanup—removing the class after the animation finished

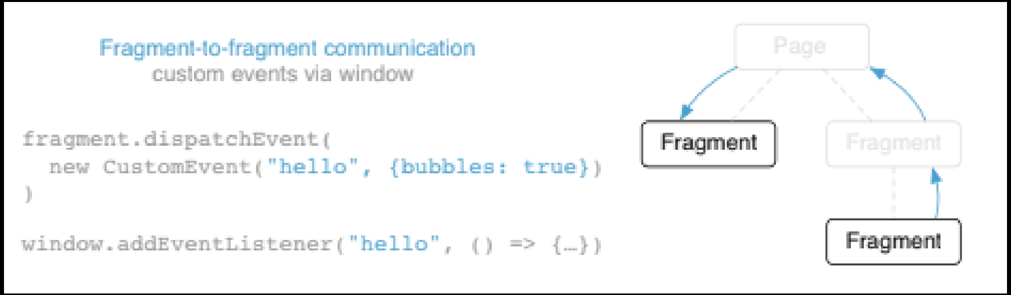
Triggering the animation by adding the confirm class

npm run 06_child_parent_flow

07_FRAGMENT_FRAGMENT_FLOW

- CDN
- TEAM-DECIDE
- TEAM-INSPIRE
- TEAM-CHECKOUT



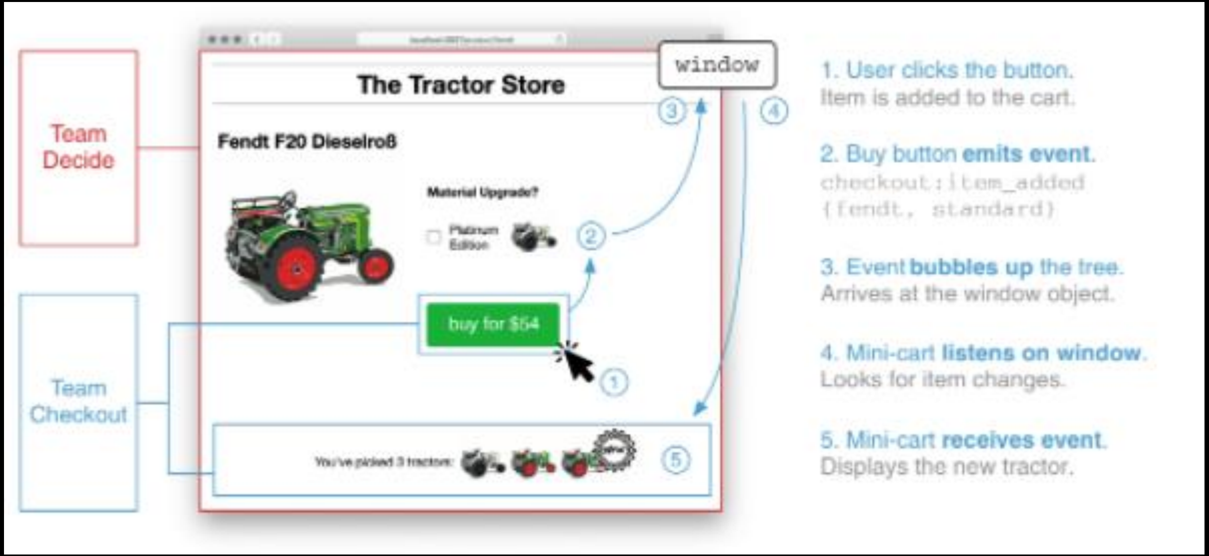


```
fragment.dispatchEvent(
  new CustomEvent("hello", {bubbles: true})
)

window.addEventListener("hello", () => {…})
```

Team Checkout wants to add a mini-cart to the product page to reduce the number of product returns.

npm run 07_fragment_fragment_flow

07_FRAGMENT_FRAGMENT_FLOW



1. User clicks the button. Item is added to the cart.

2. Buy button **emits event**. `checkout:item_added {fendt, standard}`

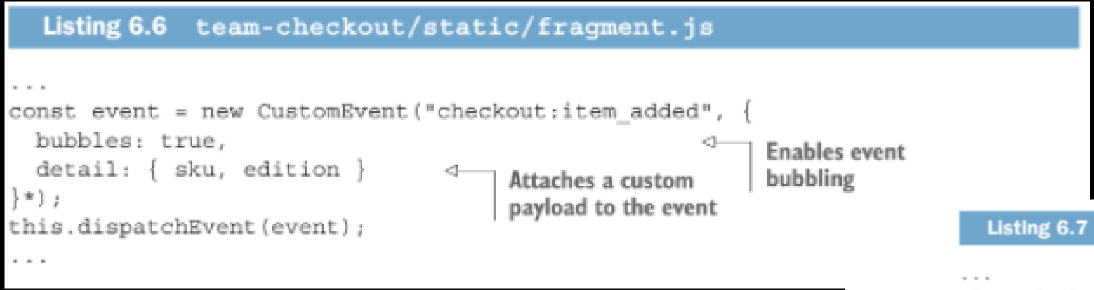
3. Event **bubbles up** the tree. Arrives at the window object.

4. Mini-cart **listens on window**. Looks for item changes.

5. Mini-cart **receives event**. Displays the new tractor.

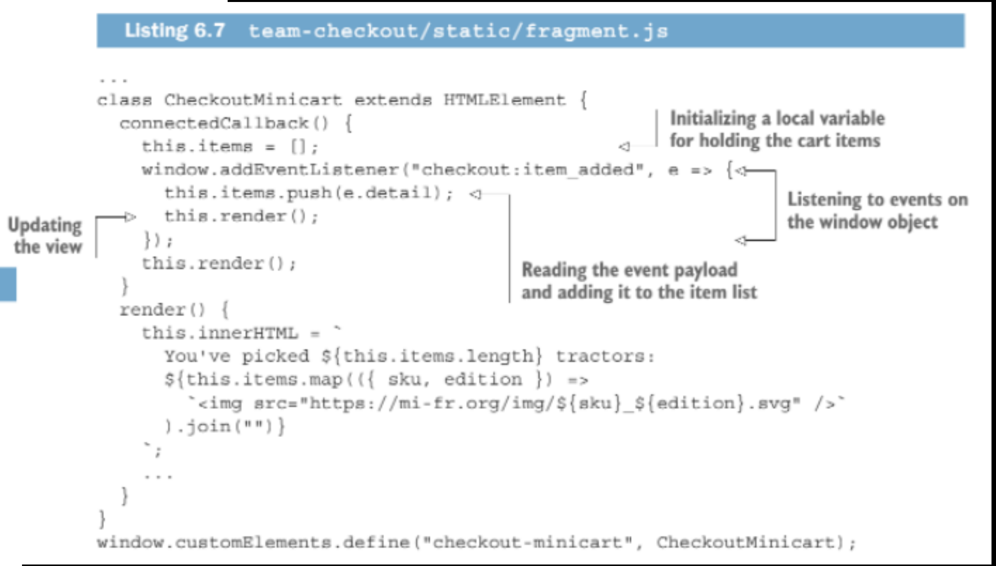
npm run 07_fragment_fragment_flow

07_FRAGMENT_FRAGMENT_FLOW



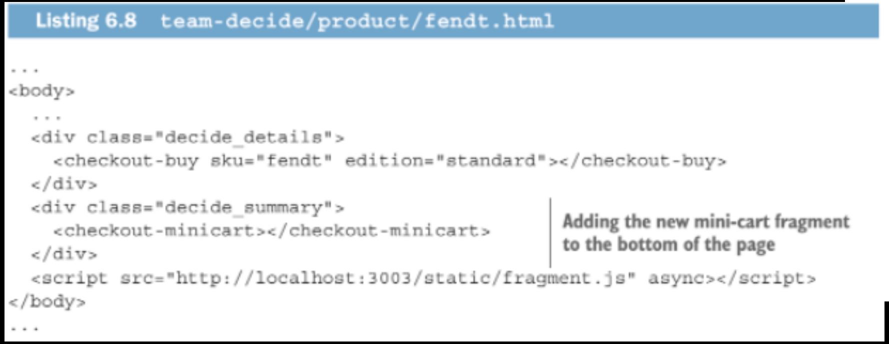
Listing 6.6 team-checkout/static/fragment.js

```
...
const event = new CustomEvent("checkout:item_added", {
  bubbles: true,                                    ◁ Enables event bubbling
  detail: { sku, edition }        ◁ Attaches a custom payload to the event
});
this.dispatchEvent(event);
...
```



Listing 6.7 team-checkout/static/fragment.js

```
...
class CheckoutMinicart extends HTMLElement {
  connectedCallback() {
    this.items = [];                                   ◁ Initializing a local variable for holding the cart items
    window.addEventListener("checkout:item_added", e => {◁ Listening to events on the window object
      this.items.push(e.detail);  ◁ Reading the event payload and adding it to the item list
      this.render();              ◁ Updating the view
    });
    this.render();
  }
  render() {
    this.innerHTML = `
      You've picked ${this.items.length} tractors:
      ${this.items.map(({ sku, edition }) =>
        `<img src="https://mi-fr.org/img/${sku}_${edition}.svg" />`
      ).join("")}
    `;
    ...
  }
}
window.customElements.define("checkout-minicart", CheckoutMinicart);
```



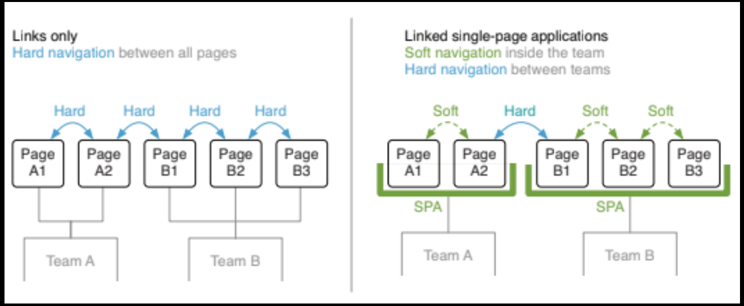
Listing 6.8 team-decide/product/fendt.html

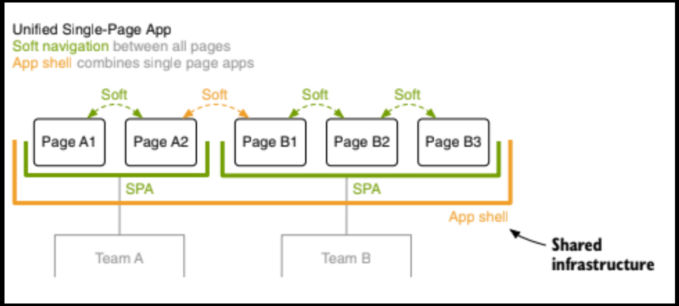
```
...
<body>
  ...
  <div class="decide_details">
    <checkout-buy sku="fendt" edition="standard"></checkout-buy>
  </div>
  <div class="decide_summary">
    <checkout-minicart></checkout-minicart>    Adding the new mini-cart fragment to the bottom of the page
  </div>
  <script src="http://localhost:3003/static/fragment.js" async></script>
</body>
...
```

npm run 07_fragment_fragment_flow

08_FLAT_ROUTING

- CDN
- TEAM-DECIDE
- TEAM-INSPIRE
- TEAM-CHECKOUT
- APP-SHELL

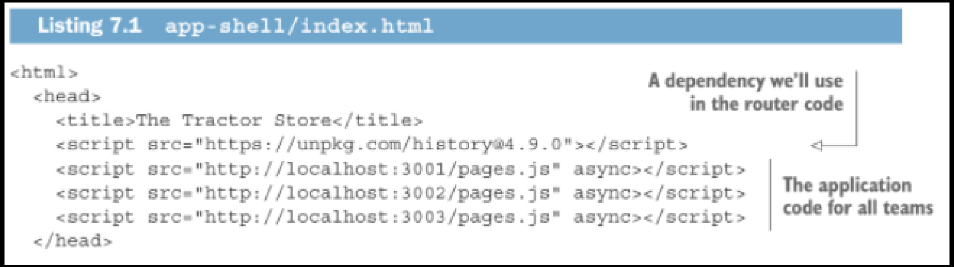




The flat routing approach requires that the app shell must know all URLs of the application.
When a team wants to change an existing or add a new URL, they also need to adjust and redeploy the app shell.

npm run 08_flat_routing

08_FLAT_ROUTING

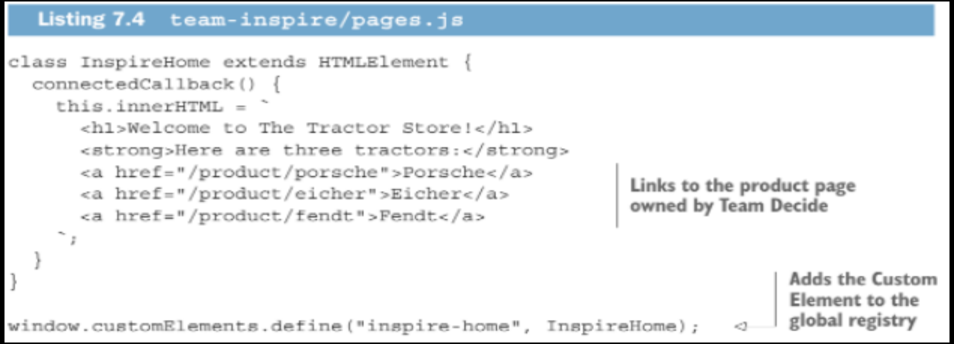


Listing 7.1 `app-shell/index.html`

```
<html>
  <head>
    <title>The Tractor Store</title>
    <script src="https://unpkg.com/history@4.9.0"></script>
    <script src="http://localhost:3001/pages.js" async></script>
    <script src="http://localhost:3002/pages.js" async></script>
    <script src="http://localhost:3003/pages.js" async></script>
  </head>
```

A dependency we'll use in the router code

The application code for all teams

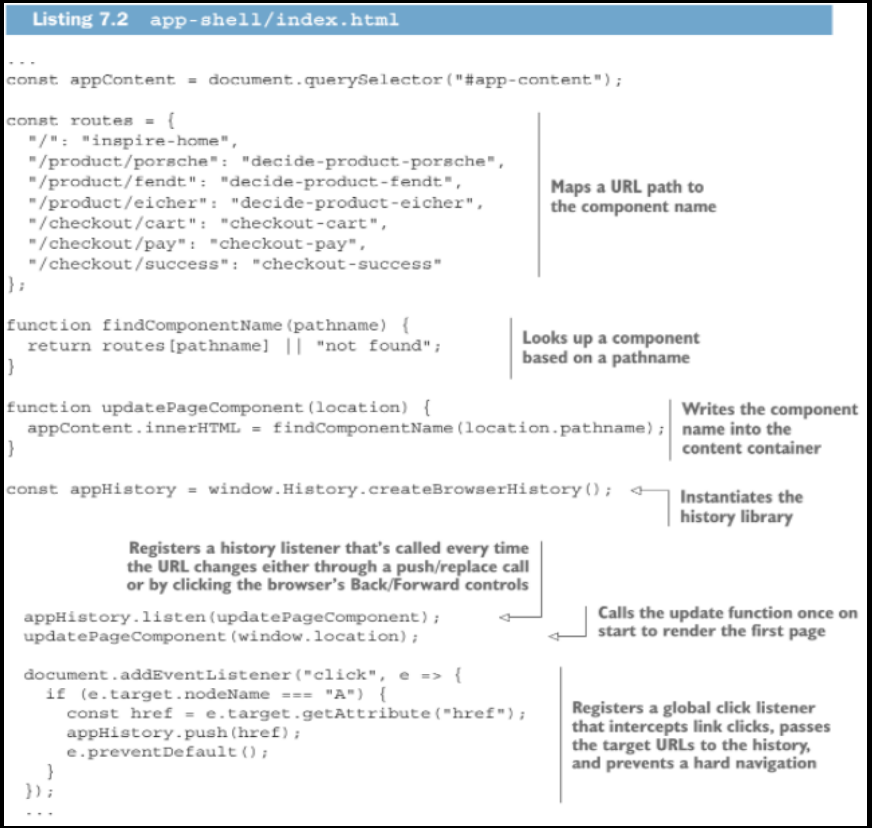


Listing 7.4 `team-inspire/pages.js`

```
class InspireHome extends HTMLElement {
  connectedCallback() {
    this.innerHTML = `
      <h1>Welcome to The Tractor Store!</h1>
      <strong>Here are three tractors:</strong>
      <a href="/product/porsche">Porsche</a>
      <a href="/product/eicher">Eicher</a>
      <a href="/product/fendt">Fendt</a>
    `;
  }
}

window.customElements.define("inspire-home", InspireHome);
```

Links to the product page owned by Team Decide

Adds the Custom Element to the global registry



Listing 7.2 `app-shell/index.html`

```
...
const appContent = document.querySelector("#app-content");

const routes = {
  "/": "inspire-home",
  "/product/porsche": "decide-product-porsche",
  "/product/fendt": "decide-product-fendt",
  "/product/eicher": "decide-product-eicher",
  "/checkout/cart": "checkout-cart",
  "/checkout/pay": "checkout-pay",
  "/checkout/success": "checkout-success"
};

function findComponentName(pathname) {
  return routes[pathname] || "not found";
}

function updatePageComponent(location) {
  appContent.innerHTML = findComponentName(location.pathname);
}

const appHistory = window.History.createBrowserHistory();

 appHistory.listen(updatePageComponent);
 updatePageComponent(window.location);

 document.addEventListener("click", e => {
   if (e.target.nodeName === "A") {
     const href = e.target.getAttribute("href");
     appHistory.push(href);
     e.preventDefault();
   }
 });
 ...
```

Maps a URL path to the component name

Looks up a component based on a pathname

Writes the component name into the content container

Instantiates the history library

Registers a history listener that's called every time the URL changes either through a push/replace call or by clicking the browser's Back/Forward controls

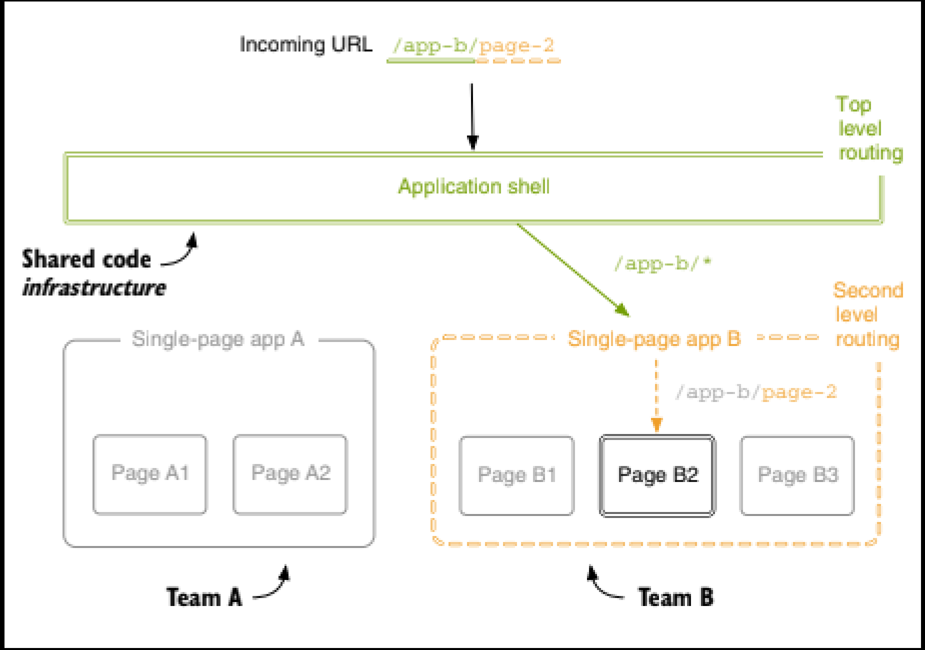
Calls the update function once on start to render the first page

Registers a global click listener that intercepts link clicks, passes the target URLs to the history, and prevents a hard navigation

npm run 08_flat_routing

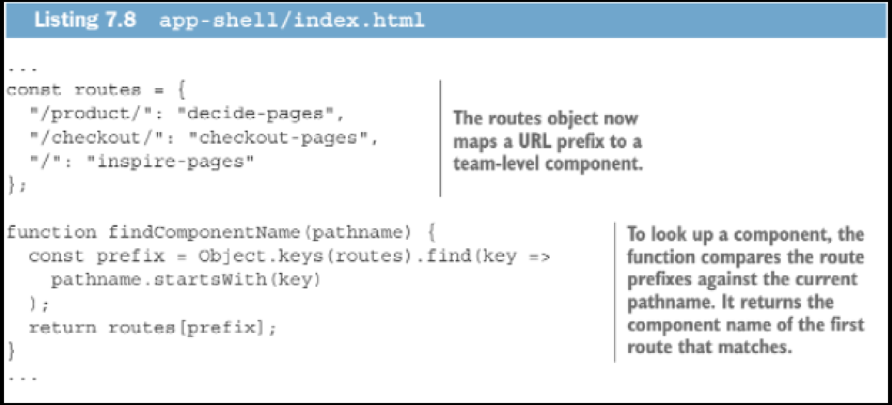
09_TWO_LEVEL_ROUTING

- CDN
- TEAM-DECIDE
- TEAM-INSPIRE
- TEAM-CHECKOUT
- APP-SHELL



npm run 09_two_level_routing

09_TWO_LEVEL_ROUTING

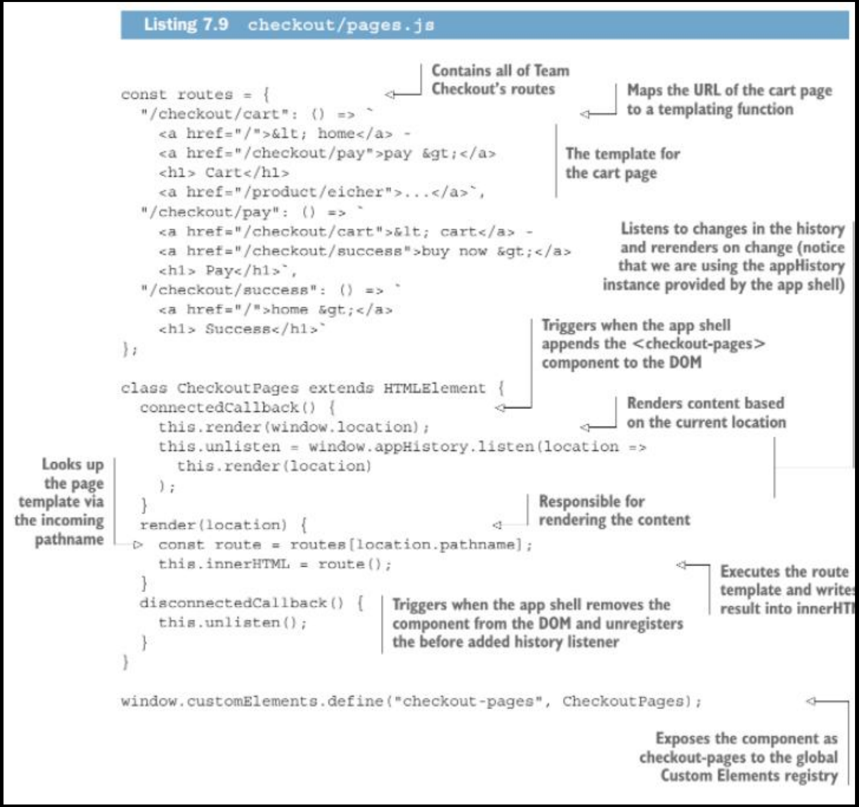


Listing 7.8 `app-shell/index.html`

```
...
const routes = {
  "/product/": "decide-pages",
  "/checkout/": "checkout-pages",
  "/": "inspire-pages"
};

function findComponentName(pathname) {
  const prefix = Object.keys(routes).find(key =>
    pathname.startsWith(key)
  );
  return routes[prefix];
}
...
```

The routes object now maps a URL prefix to a team-level component.

To look up a component, the function compares the route prefixes against the current pathname. It returns the component name of the first route that matches.



Listing 7.9 `checkout/pages.js`

```
const routes = {
  "/checkout/cart": () => `
    <a href="/">&lt; home</a> -
    <a href="/checkout/pay">pay &gt;</a>
    <h1> Cart</h1>
    <a href="/product/eicher">...</a>`,
  "/checkout/pay": () => `
    <a href="/checkout/cart">&lt; cart</a> -
    <a href="/checkout/success">buy now &gt;</a>
    <h1> Pay</h1>`,
  "/checkout/success": () => `
    <a href="/">home &gt;</a>
    <h1> Success</h1>`
};

class CheckoutPages extends HTMLElement {
  connectedCallback() {
    this.render(window.location);
    this.unlisten = window.appHistory.listen(location =>
      this.render(location)
    );
  }
  render(location) {
    const route = routes[location.pathname];
    this.innerHTML = route();
  }
  disconnectedCallback() {
    this.unlisten();
  }
}

window.customElements.define("checkout-pages", CheckoutPages);
```

Contains all of Team Checkout's routes

Maps the URL of the cart page to a templating function

The template for the cart page

Listens to changes in the history and rerenders on change (notice that we are using the appHistory instance provided by the app shell)

Triggers when the app shell appends the <checkout-pages> component to the DOM

Renders content based on the current location

Responsible for rendering the content

Looks up the page template via the incoming pathname

Executes the route template and writes result into innerHT

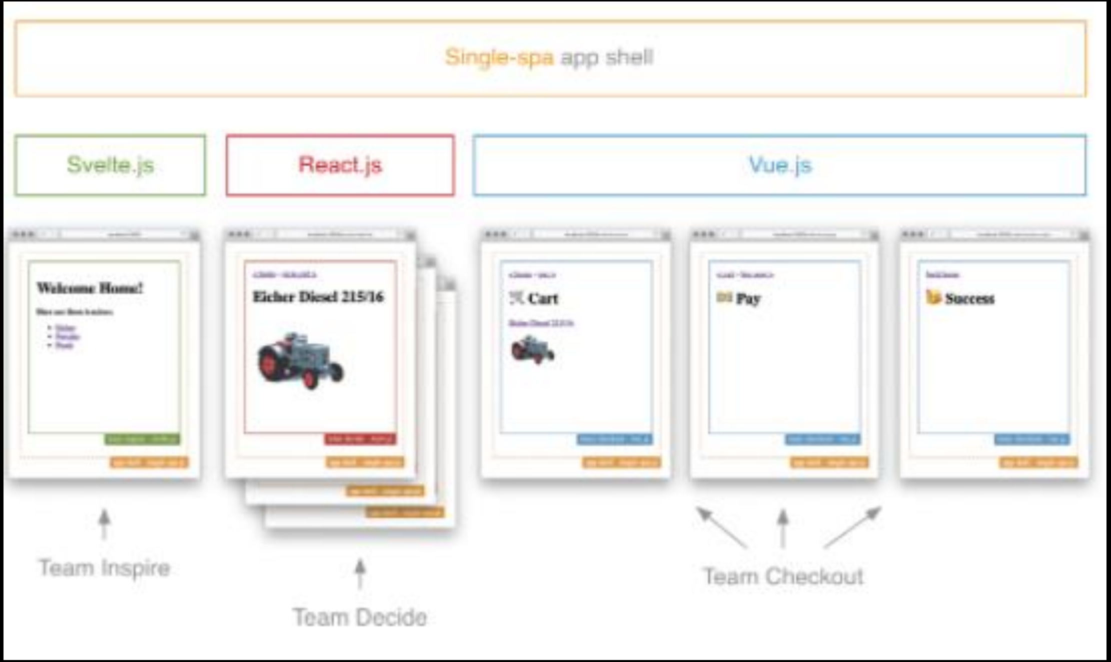
Triggers when the app shell removes the component from the DOM and unregisters the before added history listener

Exposes the component as checkout-pages to the global Custom Elements registry

npm run 09_two_level_routing

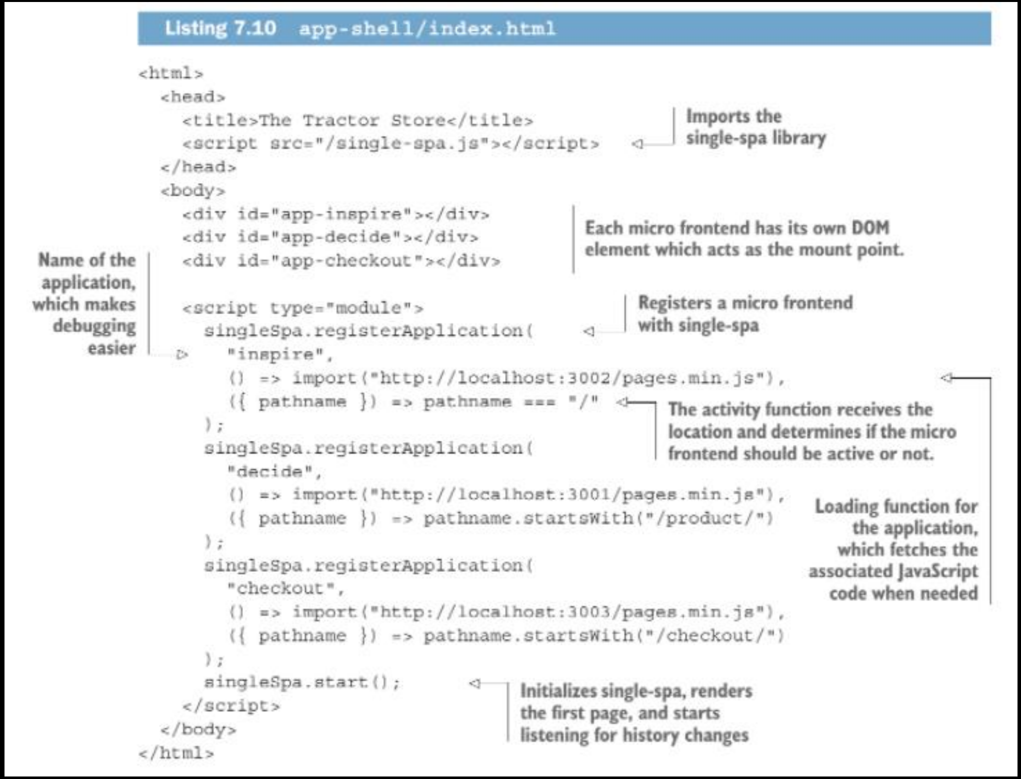
10_SINGLE_SPA

- CDN
- TEAM-DECIDE
- TEAM-INSPIRE
- TEAM-CHECKOUT
- APP-SHELL



npm run 10_single_spa

10_SINGLE_SPA



Listing 7.10 app-shell/index.html

```
<html>
  <head>
    <title>The Tractor Store</title>
    <script src="/single-spa.js"></script>
  </head>
  <body>
    <div id="app-inspire"></div>
    <div id="app-decide"></div>
    <div id="app-checkout"></div>

    <script type="module">
      singleSpa.registerApplication(
        "inspire",
        () => import("http://localhost:3002/pages.min.js"),
        ({ pathname }) => pathname === "/"
      );
      singleSpa.registerApplication(
        "decide",
        () => import("http://localhost:3001/pages.min.js"),
        ({ pathname }) => pathname.startsWith("/product/")
      );
      singleSpa.registerApplication(
        "checkout",
        () => import("http://localhost:3003/pages.min.js"),
        ({ pathname }) => pathname.startsWith("/checkout/")
      );
      singleSpa.start();
    </script>
  </body>
</html>
```

Imports the single-spa library

Each micro frontend has its own DOM element which acts as the mount point.

Name of the application, which makes debugging easier

Registers a micro frontend with single-spa

The activity function receives the location and determines if the micro frontend should be active or not.

Loading function for the application, which fetches the associated JavaScript code when needed

Initializes single-spa, renders the first page, and starts listening for history changes

npm run 10_single_spa