Provide Your Feedback for a Chance to Win.
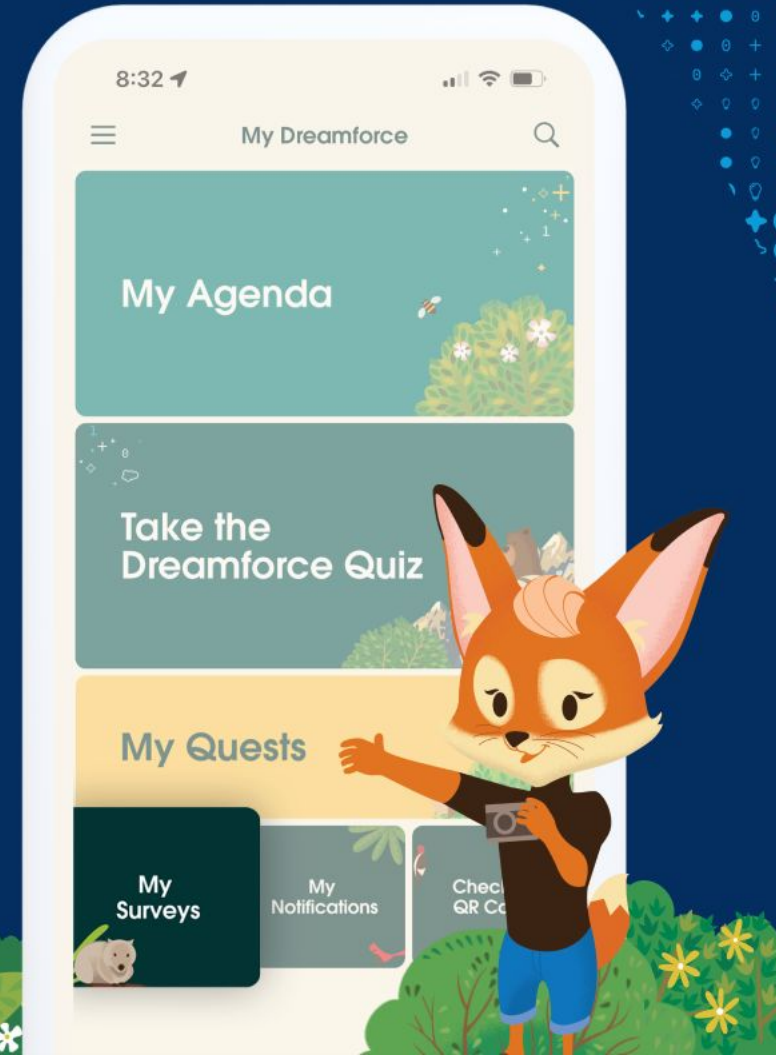
For every session survey submitted, you will be entered to win one of 15 passes to Dreamforce 2024.*

1 Open the Salesforce Events mobile app.

2 Navigate to My Dreamforce.

3 Select My Surveys.

4 For every quick session survey you take, you'll be entered to win.*

* Restrictions apply. See rules at sforce.co/session-survey-terms

Experience Delivery: Improve LWC Performance with SSR

Drew Lau, Sr. Director of Product
@drewzie
He/Him

Greg Whitworth, Sr. Director of Product
@gregwhitworth
He/Him

Forward Looking Statement

dreamforce
Thank you

Why Performance (Still) Matters

"

(Our site) is always a hot topic of discussion because of high average latencies (in the range of 3-5 seconds), whereas rest of (our) services are well below 1 second. We would like to fix it ASAP

Development Manager, anonymous customer

60% of sites do not pass Core Web Vitals

www.advancedwebranking.com/blog/core-web-vitals-study

57% of businesses expect more than half of their revenue will come from digital within 2 years

Salesforce State of Commerce, 2nd edition

salesforce

Reimagine How You Deliver B2C-like Experiences with the

#1 AI Digital Experience Platform

Partner Portals

Marketing Landing Pages

Commerce Storefronts

BYO sites

Self-Service Portal

3x conversion for sites < 1s

+1.11% conversion for every 100m decrease

24% lower abandonment rates

Salesforce Experience Services

Data Cloud   **AI Cloud**   **Hyperforce**

Our Performance Journey

Evolution Of Experience Cloud

salesforce

Visualforce 2013+

Pixel perfect experiences for external use cases (B2B, B2C)

Programmatically build pixel perfect sites and experiences

Connect to CRM data and Business Process Flows

Aura 2016+

Declarative low code enterprise-grade platform with fast time to value (B2B, B2C)

Build low-code sites with out of the box Templates & Components

Connect to CRM data and Business Process Flows

LWR 2021+

Quickly build connected, real-time, personalized C360 experiences at scale

(Pilot) Server-side Rendering with Experience Delivery for B2C perf and scale

LWR vs Aura Performance

Real world test results show improved performance across the board

Page	Aura LCP	LWR LCP		Aura FCP	LWR FCP	
Homepage	4,966 ms	2,502 ms	**2.0x Faster**	3,776 ms	1,870 ms	**2.0x Faster**
Search	7,270 ms	3715 ms	**2.0x Faster**	3,972 ms	2,155 ms	**1.8x Faster**
PDP	5,590 ms	2,876 ms	**1.9x Faster**	3,987 ms	1,718 ms	**2.3x Faster**
PLP	6,698 ms	2,846 ms	**2.4x Faster**	3,780 ms	2,054 ms	**1.8x Faster**

Note: Numbers are averages across 6 tests per page - Tests were run on a Chrome browser with cable connection in Virginia via https://www.webpagetest.org/
FCP = First Contentful Paint / LCP = Largest Contentful Paint
Tests were uncached, with guest users and not behind a CDN

Client-side Rendering 📱

Rendering is handled in the browser via JavaScript on user device.

Server-side Rendering ☁️

Rendering is handled on the "server" and HTML outputs are sent to the browser.

Comparing CSR and SSR

FCP = First Contentful Paint
LCP = Largest Contentful Paint
TTI = Time to Interactive

Client-side rendering 📱
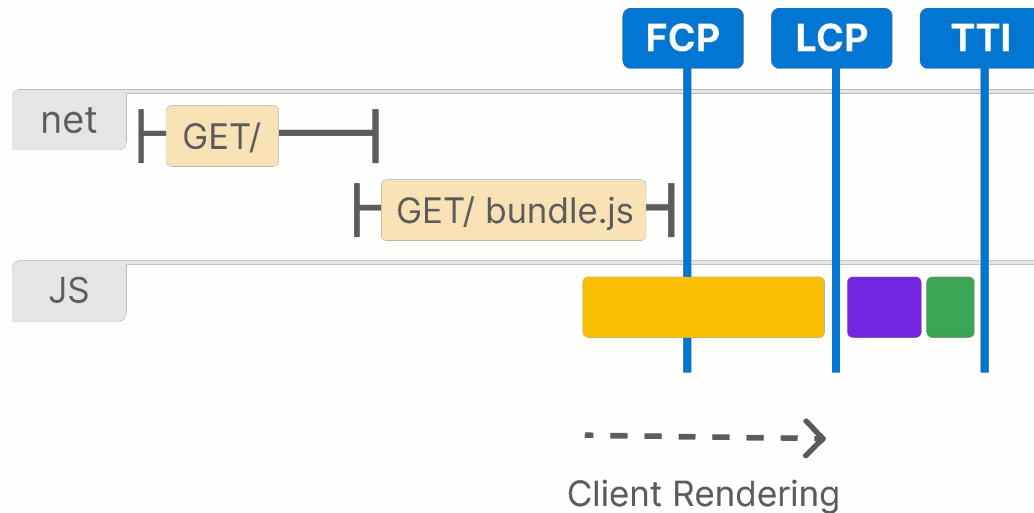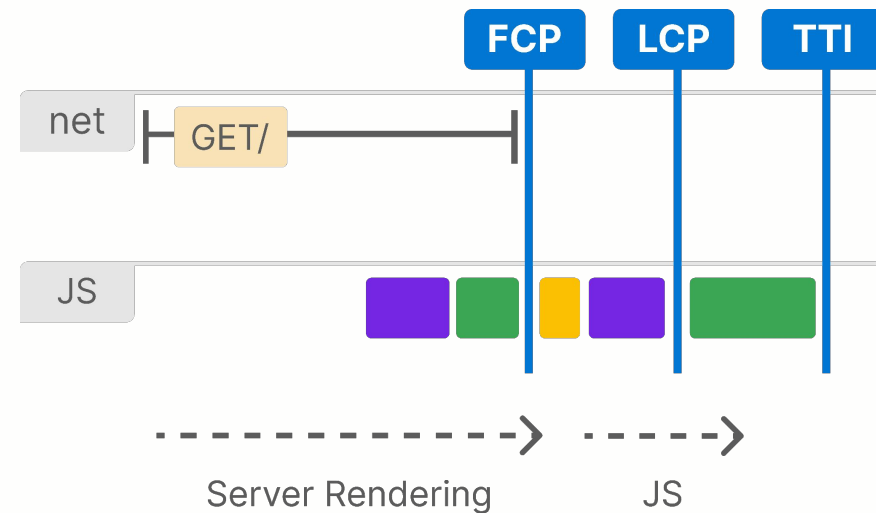
		FCP	LCP	TTI

net ├─ GET/ ─┤

├─ GET/ bundle.js ─┤

JS

Client Rendering ----->

Server-side rendering ☁️

		FCP	LCP	TTI

net ├─ GET/ ──────────┤

JS

Server Rendering -----> JS ----->

CSR and SSR Considerations

	Client-side Rendering 📱	Server-side Rendering ☁️
Performance **Network**	😀 Faster TTFB	🤔 Slower TTFB → CDN caching critical
Performance **Rendering**	😢 Slower FCP/LCP/TTI for heavier JS payloads 😢 Slower on network/compute constrained devices	😀 Faster FCP/LCP/TTI 🤔 Susceptible to scale/load impacts from heavy traffic

TTFB = Time to First Byte
FCP = First Contentful Paint
LCP = Largest Contentful Paint
TTI = Time to Interactive

CSR and SSR Considerations

	Client-side Rendering 📱	Server-side Rendering ☁️
Performance **Network**	😃 Faster TTFB	🤔 Slower TTFB → CDN caching critical
Performance **Rendering**	😢 Slower FCP/LCP/TTI for heavier JS payloads 😢 Slower on network/compute constrained devices	😃 Faster FCP/LCP/TTI 🤔 Susceptible to scale/load impacts from heavy traffic
Developer Experience **Code Complexity**	🔍 Client-side observability 😢 Browser compatibility	🤔 Isomorphic code 🐞 SSR debugging
Developer Experience **Infrastructure**	🤔 Snapshots for SEO/crawlers	💰 Maintaining and budgeting for SSR infrastructure

TTFB = Time to First Byte
FCP = First Contentful Paint
LCP = Largest Contentful Paint
TTI = Time to Interactive

🤔 Dude, where's my server?
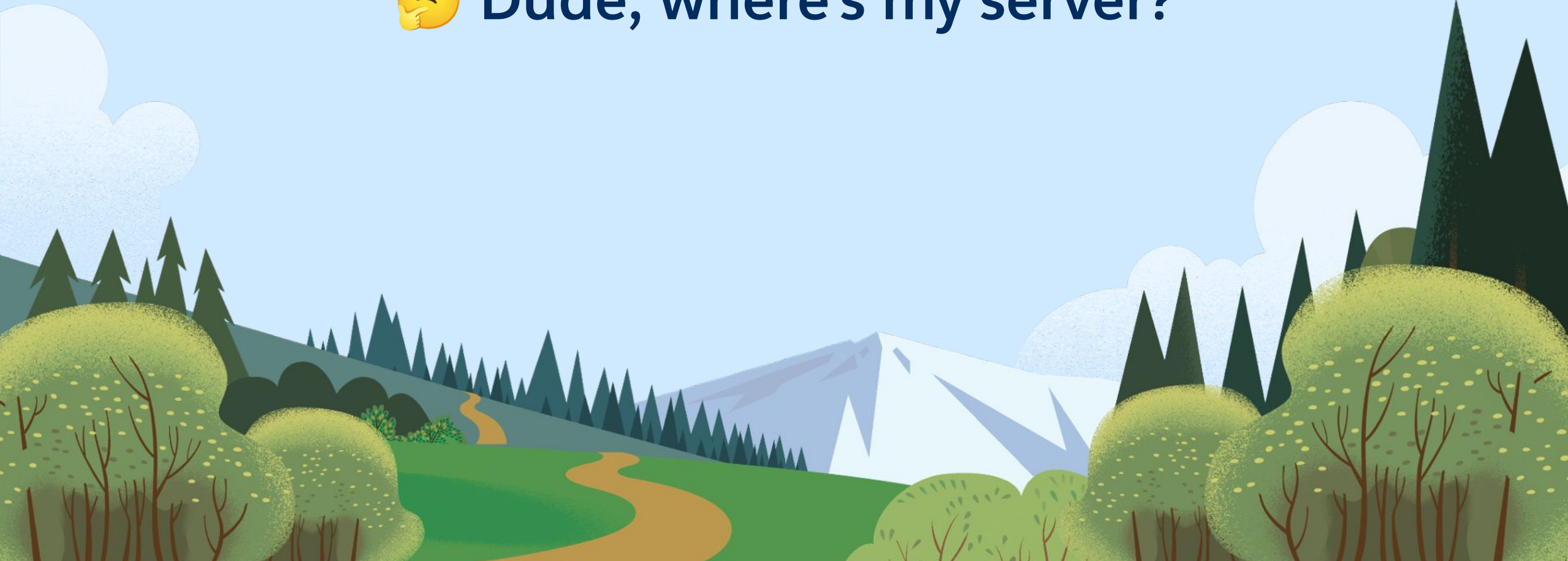
Introducing Experience Delivery

Consumer-grade LWR experiences on Hyperforce

B2C-grade Performance with Zero Stress

Optimize performance and automate scaling to free up teams

Autoscaling · Global CDN · Edge Optimizations · LWR SSR · Core Web Vitals

Improved SEO

Ensure that your content is always up to date with improved load times and static content

Query param support · Hydrated content is indexed

Perimeter Security & Compliance Tools Built for You

Start secure and stay secure

Custom Firewall Rules · Custom Rate Limiting · DDOS · Hyperforce · SOC2 · PCI · Data Residency

Up to **60%** faster page loads (LCP) for sites using Experience Delivery
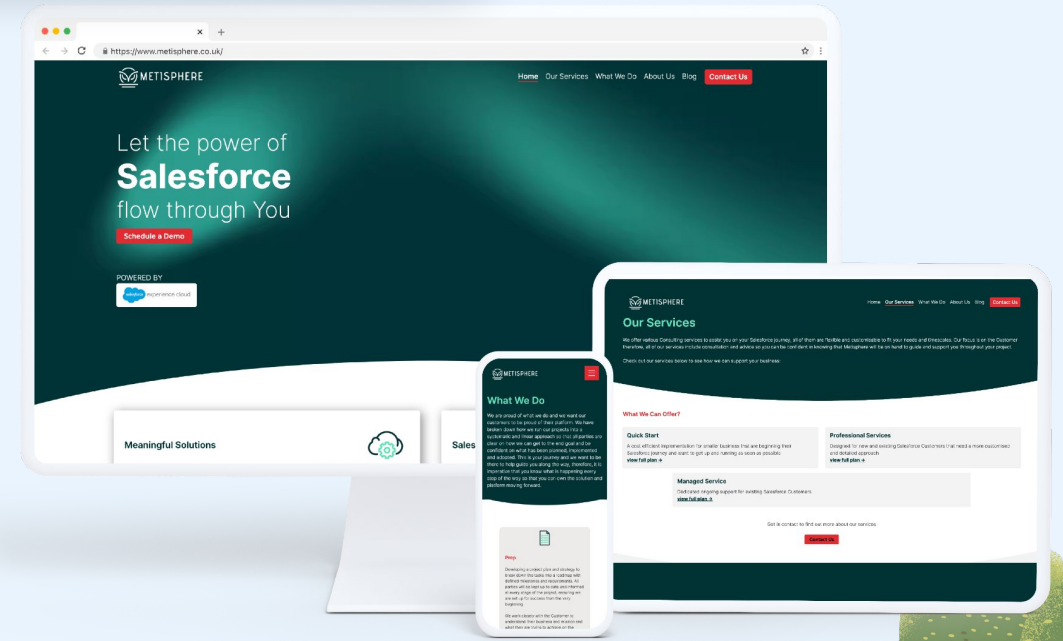
Experience Delivery unlocks consumer-grade sites

C360

Builders

Components

APIs

Publish →

Experience Delivery

Serverless Runtime

CDN & Perimeter Security

Deliver →

B2C-scale Experiences

60% faster load time (LCP)
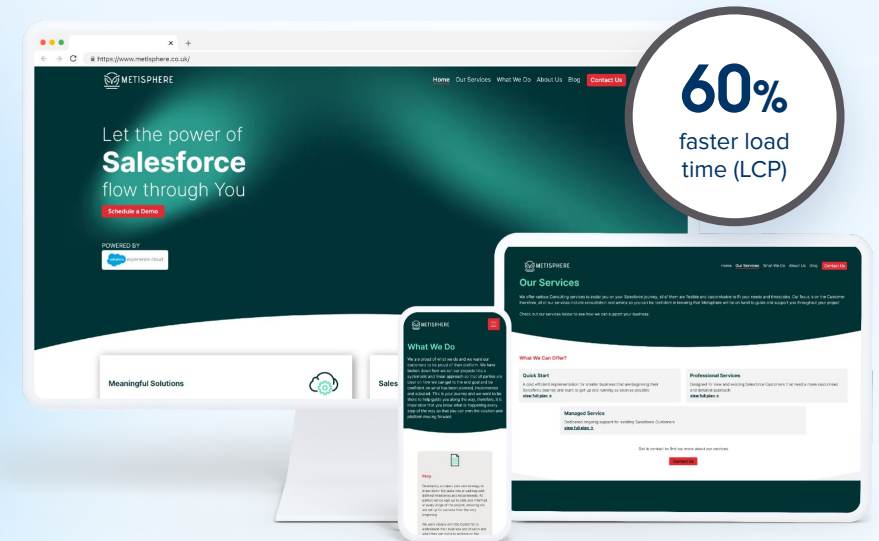
- B2C LWR Sites
- B2B LWR Sites
- B2E LWR Sites

Salesforce Platform

Hyperforce

Impact of SSR with Experience Delivery

Site	Baseline	With Experience Delivery	
Simple Guest Context	1,074 ms	472 ms	56% Faster
Microsite Guest Context	1,501 ms	1,343 ms	11% Faster
Pilot site Guest Context	1,220 ms	520 ms	57% Faster
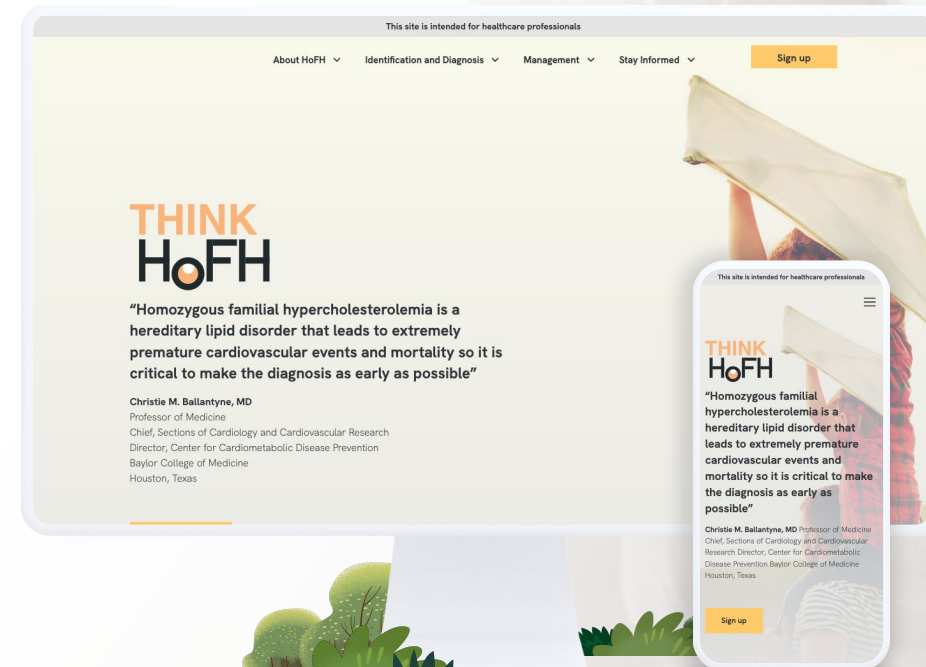
Adding SSR to LWCs

Preparing your components for Experience Delivery

How to ensure that your components can be SSR'd

Utilize Light DOM or Native Shadow

Both Light DOM and Native Shadow, via mixed-mode, will make it so that
your LWC's can be compiled in Experience Delivery

Make your components portable

DOM APIs are not available on the server. So leverage conditional checks
to avoid unnecessary exceptions that result SSR failures.

Test (ESLint, Manual Validation, Einstein)

Depending on how your site is built you may need to modify how your
CSS is written. Additionally, we need to ensure that the components are
SSR'ing correctly.

Light DOM: http://sfdc.co/light-dom
Mixed-mode: http://sfdc.co/mixed-mode

Choose a rendering paradigm

salesforce

Light DOM

Same DOM, same styles
The LWCs rendered in Light DOM are rendered without a Shadow DOM, taking advantage of reusing styles and expanded elements tree navigation/querying.

Third party integration
With light DOM, third-party tools can traverse the DOM, enabling standard browser query APIs like querySelector and querySelectorAll, without traversing the shadow root.

Accessibility
Enables referencing elements IDs cross components using Light DOM as they are not encapsulated.

Native Shadow DOM

Encapsulated Development
Develop within the boundaries you want. Make changes to a subset of components without breaking other parts of your application or customer environments.

Improved Custom Styling
Easily apply custom themes to your application. Native Shadow components will create bespoke pages with the now available `:slotted` and `::part` apis.

Light DOM

myCmp.html

```html
<template lwc:render-mode="light">
    <p>Hello World</p>
</template>
```

myCmp.js

```js
export default class myCmp
extends LightningElement {
    static renderMode = "light";
}
```

Native Shadow DOM via mixed-mode

myCmp.html

```html
<template>
    <p>Hello World</p>
</template>
```

myCmp.js

```javascript
export default class myCmp
extends LightningElement {
    static shadowSupportMode = "any";
}
```

Choose a rendering paradigm

salesforce

	Shadow DOM	Light DOM
Security	Strong component encapsulation protects components from unauthorized access	Weak encapsulation makes components open to unauthorized access
Portability	Highly portable with access controlled through public APIs	Susceptible to breaking changes caused by component authors or consumers
Styling	Requires CSS custom properties to override styles	Easy to override styles
Third-party library and tool integration	Limited compatibility with third-party libraries or tools requiring DOM traversal or event retargeting	Simple integration with third-party libraries and tools

Making your components portable

work outside of a browser context
(eg: on a server)

The new SSR environment variable

```javascript
export default class App extends LightningElement {

    connectedCallback() {

        if (!import.meta.env.SSR) {

            window.addEventListener('error', (evt) => {

                console.error(`⚠️ Uncaught error: ${evt.message}`);

            });

        }

    }

}
```

Move async methods to rendered callback

```javascript
export default class App extends LightningElement {
    @api bookId;
    @wire(getBook, { id: '$bookId'})
    book;
}
```

Move async methods to rendered callback

```javascript
export default class App extends LightningElement {
    renderedCallback() {
        @api bookId;
        @wire(getBook, { id: '$bookId'})
        book;
    }
}
```

LWC ESLint tools for automated guidance

no-restricted-browser-globals-during-ssr

Browser APIs must not be accessed when SSR is being done. This rule prevents usage of browser APIs like `DOMParser`, `DocumentFragment` etc. in connectedCallback (and in methods called from connectedCallback or anywhere when SSR is being done).

no-unsupported-ssr-properties

Browser APIs must not be accessed when SSR is being done. This rule prevents usage of browser APIs like `querySelector`, `dispatchEvent`, on this in `connectedCallback` (and in methods called from connectedCallback or anywhere when SSR is being done).

Working with data

the getPageData method

Using getPageData hook to SSR data

salesforce

myCmp.js

```js
export default class Cmp extends LightningElement {

    @api label;

}

export async function getPageData() {

    // Returns Greg

  static label = fetch(url)

    return { props: { label } };

}
```

Using getPageData hook to SSR data

myCmp.html

```html
<template>
    <label>{label}</label>
</template>
```

Using getPageData hook to SSR data

Output from SSR

```
<label>Greg</label>
```

Takeaways

How to ensure that your components can be Server Side Rendered

Utilize Light DOM or Native Shadow

Both Light DOM and Native Shadow, via mixed-mode, will make it so that
your LWC's can be compiled in Experience Delivery

Make your components portable

DOM APIs are not available on the server. So leverage conditional checks
to avoid unnecessary exceptions that result SSR failures.

Test

Depending on how your site is built you may need to modify how your
CSS is written. Additionally, we need to ensure that the components are
SSR'ing correctly.

Light DOM: http://sfdc.co/light-dom
Mixed-mode: http://sfdc.co/mixed-mode

EXPLORER

JOKES-APP
> .sfdx
> config
> node_modules
∨ scripts
  JS start.js
∨ src
  > assets
  > content
  > layouts
  ∨ modules
    ∨ dad
      ∨ app
        # app.css
        <> app.html
        TS app.ts   2
      ∨ data
        TS data.ts
      > joke
      > menu
      > rating
      > error
      > wires
    > raml
    > serviceWorkers
    > types
  .eslintrc.cjs
  .gitignore
  {} lerna.json
  ☘ LICENSE

> OUTLINE
> TIMELINE
> RUNNING TASKS

TS app.ts   2   ✕   TS data.ts

src > modules > dad > app > TS app.ts > ⚡ App > ۞ initialize

```typescript
1   import { LightningElement } from "lwc";
2   import Navigo from "navigo";
3
4   import { getMetadata, getJokes, getJokeById } from "dad/data";
5
6   export default class App extends LightningElement {
7     router = new Navigo("/");
8
9     jokeIds = [];
10    currentJokeIndex = 0;
11    currentJoke: any = null;
12    metadata: any = {};
13
14    constructor() {
15      super();
16
17      this.initialize();
18
19      this.router.on("/jokes/:id", async (url: any) => {
20        this.currentJoke = await getJokeById(url.data.id);
21      });
22    }
23
24    connectedCallback(): void {
25      window.addEventListener("error", (evt) => {
26        console.error(`⚠ Uncaught error: ${evt.message}`);
27      });
28    }
29
30    async initialize() {
31      this.metadata = await getMetadata();
32      this.jokeIds = await getJokes();
33
34      const [, , jokeId] = document.location.pathname.split("/");
35      this.currentJoke =
36        jokeId && jokeId.length >= 24
37          ? await getJokeById(jokeId)
38          : await getJokeById(this.jokeIds[this.currentJokeIndex]);
39    }
```
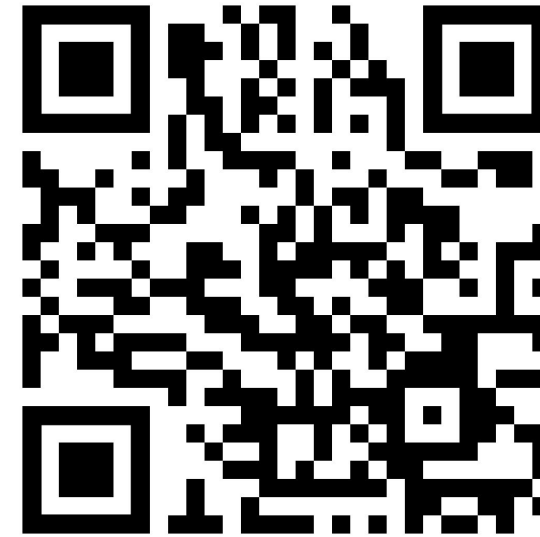
ssr-prep ⊗ 2 △ 0   Ln 31, Col 30   Spaces: 2   UTF-8   LF   {} TypeScript   Prettier

Join the Experience Delivery Open Pilot!

Pilot Benefits

- Improved page load performance
- Improved SEO through data freshness
- Ability to provide early feedback to influence Salesforce roadmap

Learn more and apply here

sfdc.co/df23-experience-delivery

Thank you

WELCOME TRAILBLAZERS

POWELL & HYDE Sts.

DREAMFORCE

Provide Your Feedback for a Chance to Win.

For every session survey submitted, you will be entered to win one of 15 passes to Dreamforce 2024.*

1 Open the Salesforce Events mobile app.

2 Navigate to My Dreamforce.

3 Select My Surveys.

4 For every quick session survey you take, you'll be entered to win.*

* Restrictions apply. See rules at sforce.co/session-survey-terms
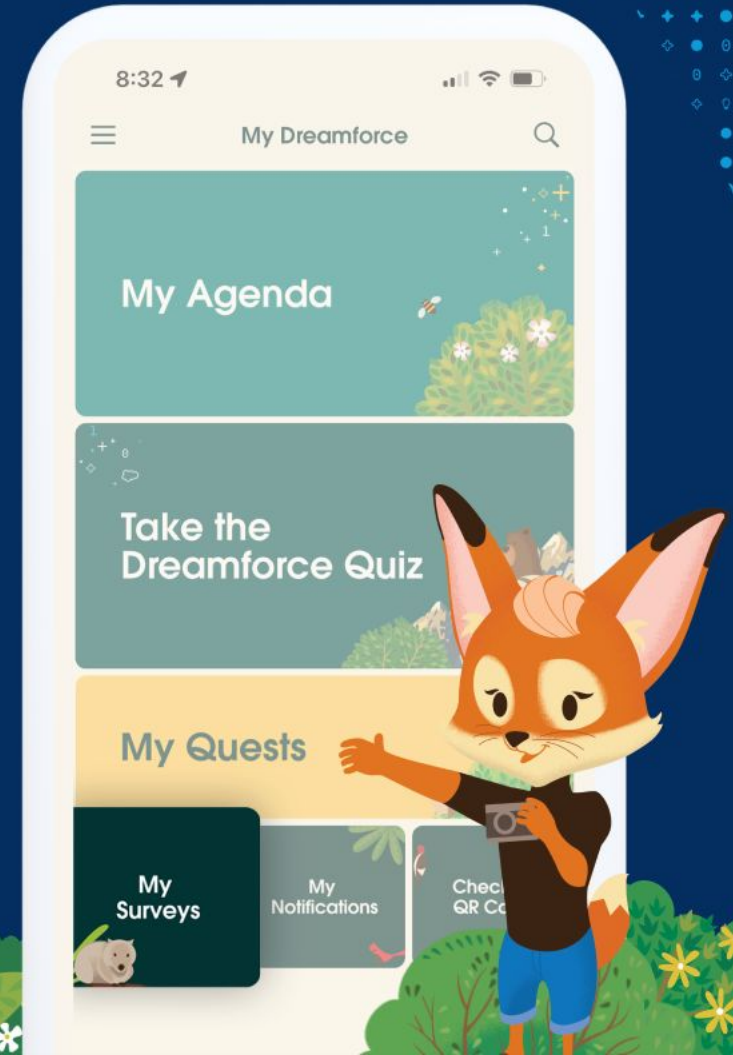
Visit our Demo Booth Stations in Trailblazer Forest & BrAInforest

PLATFORM PARK

EINSTEIN LOOKOUT

AI LAUNCH ZONE

Lightning App Builder

Compose apps with drag-and-drop tools

Lightning Experience Builder and CMS

Compose consumer grade experiences with clicks

Salesforce Mobile Experiences

Build custom mobile experiences with code and low code

Lightning Web Components

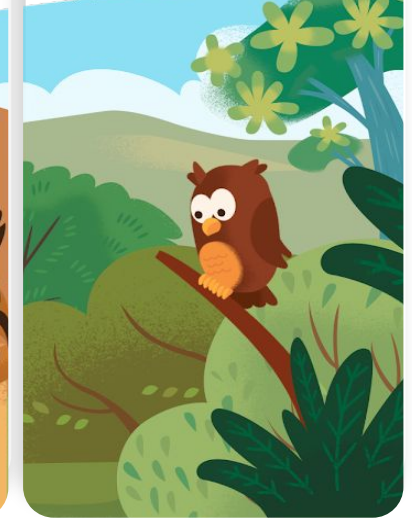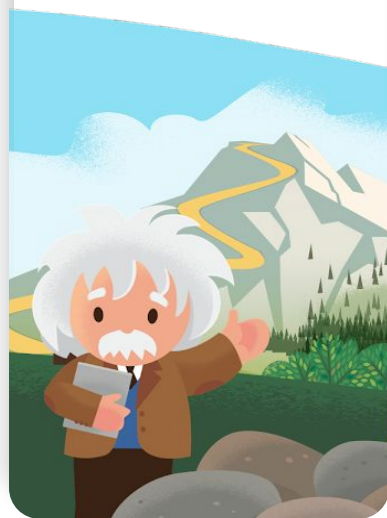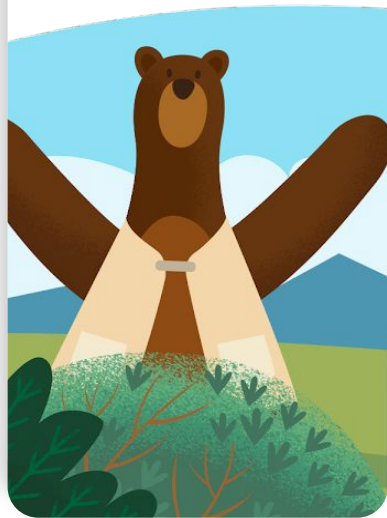Build reusable UI components with modern web standards

Prompt Builder

Design trusted AI prompts

Prompt Builder

Build, test, and deploy grounded AI prompt

Watch keynotes showcasing these products live at Moscone West 3rd fl and Salesforce+

IT Keynote on Tues Sept 12 at 2:30PM & **Admin Keynote** on Wed Sept 13 at 11:00AM