Salesforce and the Web Platform

Leo Balter, Sr. Product Manager
@leobalter
He/Him
Salesforce <3 Open Web
LWC was introduced over 3 years ago
Over 22 million LWC components created
Why did we build LWC?
Lightning Web Components (LWC)

Every JavaScript developer can code on Salesforce

Enhanced productivity with web standards
Use the modern language of the web: ES6+, Custom Elements, classes, modules and imports

Engineered for performance
More code executed by the browser instead of JavaScript abstractions for a blazing fast experience

Compatible and easy to use
Runs side-by-side with existing Lightning components and can be composed with clicks or code
Work with the web platform, not against it
Push the web forward for our Salesforce customers & their users
We do this together!
We do this together!
How do we work together?

Addressing browser breaking changes

Tracking Browser Bugs

Testing early for responsible migration as standards become recommendations

Driving Salesforce innovation into the web platform
Browser breaking changes
Browser breaking changes

- Over 400 cases created in one week
- “Reported By” over one thousand customers on Known Issue
LWC: quick and effective migration plan:

```javascript
window.alert('foo');

LightningAlert.open({
  message: 'foo',
  variant: 'headerless',
}).then((result) => {
  console.log('alert', result);
});
```
Lightning Base Components available today
Browser breaking changes
alert()/confirm()/prompt()
Preparing Your Components for the Removal of alert()/confirm()/prompt()

GREG WHITWORTH

In early 2021, the Web Hypertext Application Technology Working Group (WHATWG) changed the HTML specification to deprecate support for the alert(), confirm(), and prompt() APIs when used in a third-party context.
Performance impact
When specific Accessibility Feature is enabled in Chromium 96

- Basic flows - such as tab switching went from sub-second to over 6 seconds
- Targeted fixes to address the largest issues landed in Chrome 99 following Salesforce collaboration
- Chromium team is working towards a complete long-term solution
Tracking
Browser Bugs
No more workarounds, please

- ✅ `:host::part(foo)` (chromium)
- ✅ `:host::part(foo)` (WebKit)
- ✅ `debugger` statement is ignored in iframes removed from the document
- 🚨 `Object.getOwnPropertyDescriptors` causes window to delete itself in detached iframe

...
Report, track, and describe impact for bugs in the Web Platform.
Testing early for responsible migration as standards become recommendations
Mixed Shadow mode  

Platform: Dev Preview

OSS

Shadow DOM support in 2017

- Chrome: Yes
- Firefox: No
- Safari: Yes
- Edge: No
- Internet Explorer: No
Mixed Shadow mode

OSS
Platform: Dev Preview

- LWC
- Synthetic Shadow DOM
- Web Platform
Mixed Shadow mode

OSS

Platform: Dev Preview

Shadow DOM support in 2017

- Chrome: Yes
- Firefox: No
- Safari: Yes
- Edge (Chromium): No
- Edge (Core): No

Shadow DOM support in 2022

- Chrome: Yes
- Firefox: Yes
- Safari: Yes
- Edge (Chromium): Yes
- Edge (Core): No
- Edge (Core): No
Mixed Shadow mode

Platform: Dev Preview

OSS

Shadow DOM support in 2017

- Chrome: Yes
- Firefox: No
- Edge (Legacy): Yes
- Edge (Chromium): No

Shadow DOM support in 2022

- Chrome: Yes
- Firefox: Yes
- Edge (Legacy): Yes
- Edge (Chromium): Yes
- Opera: No
Mixed Shadow mode OSS Platform: Dev Preview

LWC

Synthetic Shadow DOM

Native Shadow DOM

Web Platform
Mixed Shadow mode

- LWC
- Synthetic Shadow DOM
- Native Shadow DOM
- Web Platform

OS (OSS) Platform: Dev Preview
Mixed Shadow mode

Enablement

Indicates support for Native and synthetic shadow.

import { LightningElement } from 'lwc';

export default class extends LightningElement {
  static shadowSupportMode = 'any';
}

Native shadow DOM:

Synthetic shadow DOM:
Mixed Shadow mode  
Differences between Native and Synthetic Shadow DOM

No global style leakage (eg. loaded via static resources)

Subtle invocation timing changes of `connectedCallback` and `disconnectedCallback`

Minor changes in DOM APIs
Native Shadow Compatibility Challenge

- No compromise to Accessibility
- Web Standards Solutions
- AOM is not yet available
Enable adoption of Native Shadow DOM
Salesforce is working with Igalia to ship ID Ref and Cross-root ARIA delegation
Driving Salesforce innovation into the web platform
ShadowRealms

A new way of evaluating code at runtime within its own JavaScript global scope.

It's a lightweight, smart, and clean alternative for iframes.
Extensible Web Applications

A platform web application is composed by multiple components and parts from different origins.
Salesforce Components

The fundamental platform parts reused across the customers.
Custom Components

Tailored directly by the customers and composed with content specific to their needs.
Customers can take advantage of extensions for their specific contents and benefit.
Improved Integrity and Security

LWS ensures integrity and security to the Platform in real time.

ShadowRealms empowers this assurance at much faster fashion!
Salesforce is sponsoring Igalia to ship ShadowRealms in Web Browsers.
Up to 13x faster on initialization!
Up to 8x faster using LWS' membranes framework
In the spirit of pushing the web forward...
LWC IE11 Support Ends Jan 1, 2023 (Spring ‘23 Release)
We are not browser implementers, but we are part of the open web.

It's also our responsibility to move the web forward!