The adaptive interface system
for modern web experiences

Introducing FAST

- What are Web Components?
- What is FAST?
- DEMO: Using Fluent UI Web Components
- Overview: Building a FAST Web Component
- DEMO: Building a Todo App with FAST
- DEMO: Leveraging Adaptive UI
- Q&A

Rob Eisenberg
Principal UX Architecture and Tools Lead
WebXT – FAST, Edge Web UI, Web Platform & Developer Experiences
rob.eisenberg@microsoft.com

FAST

What are Web Components?

A collection of web standards that enable an HTML native component model.

Custom HTML Elements

Natively extend HTML with new element tags that respond to DOM lifecycle hooks and encapsulate component state and behavior.

Shadow DOM

Isolate rendering for components, allowing a component to hide its internal render details and prevent CSS styles from leaking in or out. Leverage slots to enable rich composition.

HTML Templates

Declare inert HTML that can be efficiently cloned in order to quickly "stamp out" HTML for use in web component Shadow DOM.

CSS Properties

Author styles with CSS Properties to enable parameterizing visual appearance. Because CSS Properties can pierce the Shadow DOM, they enable theming across components, libraries, and apps.

CSS Parts

Name parts of the Shadow DOM so that specific internal elements can be directly targeted by CSS selectors.

ES2015+

Leverage new languages features such as classes and static getters that are part of the web component programming model. With TypeScript and transpilers, use upcoming features such as decorators to enhance the developer experience.

Element Internals

Enable components to participate in Form validation and submission as well as other upcoming APIs such as custom pseudo-selectors.

Declarative Shadow DOM

Render Shadow DOM on the server and let the browser automatically hydrate it with no JavaScript involved.

Technology Agnostic

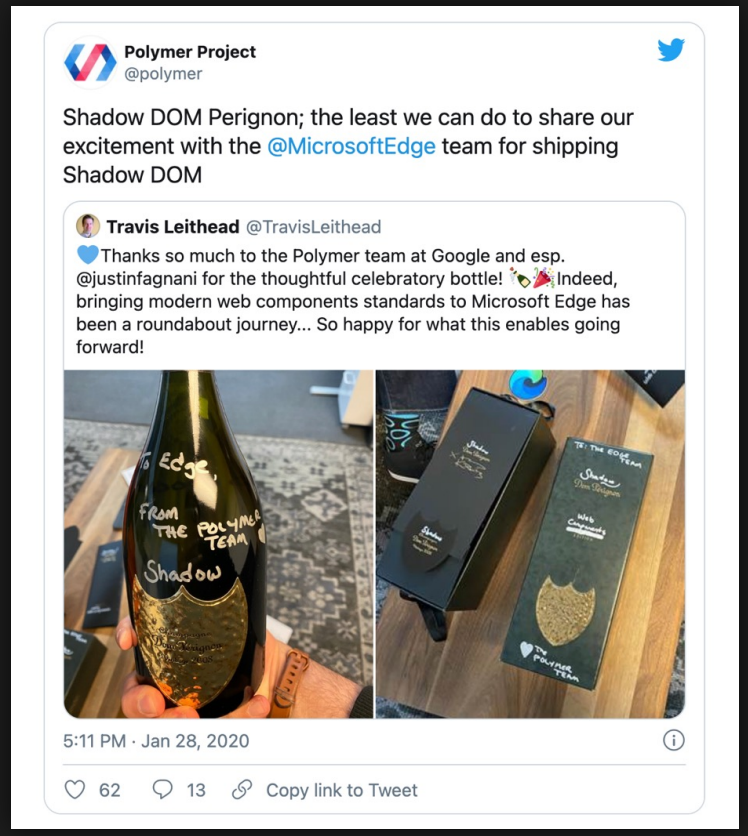
Integrate the same web components with all major frameworks including Angular, Aurelia, Blazor, Polymer, React, Svelte, Vue, etc. Or use without a framework or even without JavaScript.

Related Standards Bodies and Community Groups: W3C, WHATWG, TC39, Web Components CG, Open UI CG, Open Web Components, Web Component Community Protocols

FAST

Who is betting on Web Components?

Major technology organizations investing in the future of web components.



Microsoft

Google

Adobe

SAP

salesforce

UBISOFT

SPACEX

ING

redhat

Massachusetts Institute of Technology

The adaptive interface system for modern web experiences.

The system

FAST Element & FAST Foundation

Used to build performant, memory-efficient, standards-compliant Web Components and Design Systems. FAST Elements work in every major browser and can be used in combination with any front-end framework or even without a framework.

FluentUI & FAST Web Components

Libraries of Web Components that support Adaptive UI and multiple design systems. Each of these libraries also delivers on all the basics such as accessibility, performance, and internationalization.

Utilities, Plugins & Helpers

Libraries that can be used with the FAST system or à la carte and include support for animation, color, dependency injection, routing, and more.

Tools

Prebuilt tools that support design-to-code and component development, constructed out of a suite of libraries that can be used to build your own tools or extend others.

FAST

FAST Packages

@microsoft/fast-element

Declarative
Use modern JavaScript and TypeScript to declare web components along with their attributes, properties, templates, and styles.

High Performance & Low Memory
Render using a modern reactivity system that features low memory usage and highly efficient, incremental and batched DOM updates.

Interoperable & Composable
Compose with web components built using other libraries and even with non-web component frameworks (React, Blazor, Angular, Vue, etc.)

@microsoft/fast-foundation

Foundation Components
Almost 40 foundation "style-less" components that form a fully accessible base library, ready to create your own custom design system.

Design Tokens & Adaptive UI
First class support for both static and dynamic design tokens integrated with an adaptive UI system that automatically enables accessible component themes.

Dependency Injection
Build up components, systems, and services by declaring dependencies and easily separate interfaces from implementations.

Common UI Helpers
APIs for creating anchored popups, supporting high contrast, handling RTL, creating form elements, and more.

@fluentui/web-components
@microsoft/fast-components

- Accordion
- AccordionItem
- Anchor
- AnchoredRegion
- Badge
- Breadcrumb
- BreadcrumbItem
- Button
- Card
- Checkbox
- Combobox
- DataGrid
- Dialog
- Divider
- Flipper
- HorizontalScroll
- Listbox
- ListboxOption
- Menu
- MenuItem
- NumberField
- Progress
- ProgressRing
- Radio
- RadioGroup
- Select
- Skeleton
- Slider
- SliderLabel
- Switch
- Tabs
- Tab
- TabPanel
- TextArea
- TextField
- Tooltip
- TreeView
- TreeViewItem

- Avatar*
- Calendar*
- ColorSwatch*
- DateTimePicker*
- Disclosure*
- PresenceBadge*
- Toolbar*

@microsoft/fast-router ALPHA

Route Patterns
Static routes, route parameters, splat routes, and optional parameters.

Navigation
Redirect routes, ignored routes, and fallback routes. History event queuing, navigation contributors, navigation lifecycle, child routes, and child routers.

UI Composition
Component and template rendering. Layout composition and screen transitions.

Data Scenarios
Typed route parameters and integrated data loading. Dependency injection integration. Route and query string generation. Document title manipulation.

Customization
Componentized architecture where any core router service can be replaced with a custom implementation. Routing lifecycle integration. Routing events.

FAST

DEMO

Using Fluent UI Web Components

FAST

Overview: Building a Web Component

```
import { attr, css, customElement, FASTElement, html } from '@microsoft/fast-element';
import { neutralForegroundRest } from '@fluentui/web-components';


const template = html<HelloWorld>`
  <h1>Hello ${x => x.name}!</h1>
`;



const styles = css`
  :host {
    contain: content;
    display: block;
  }

  h1 {
    color: ${neutralForegroundRest};
  }
`;


@customElement({
  name: 'hello-world',
  template,
  styles
})
export class HelloWorld extends FASTElement {
  @attr name: string = 'World';
}
```

3. Describe rendering using a template with bindings and directives.

4. Define styles.

5. Associate the template, styles, and element tag name with the custom element class.

1. Create a `class` that inherits from `FASTElement`.

2. Declare HTML attributes and observable properties.

CONTENT

```
<a>${…}</a>
```

ATTRIBUTES

```
<a href=${…}></a>
```

PROPERTIES

```
<input :value=${…}>
```

BOOLEAN ATTRIBUTES

```
<input ?disabled=${…}>
```

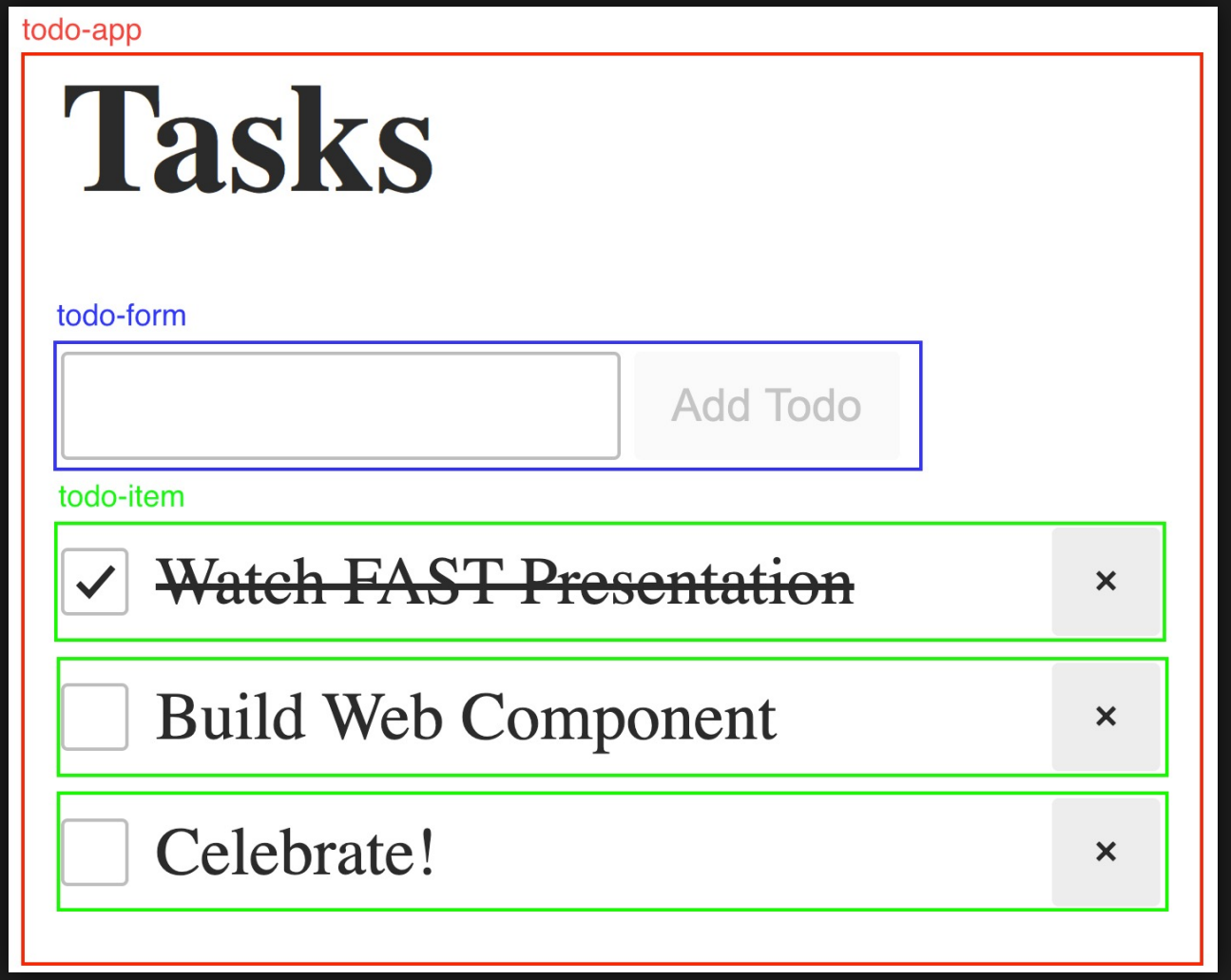
EVENTS

```
<button @click=${…}></button>
```

FAST

DEMO

Building a Todo App with FAST



FAST

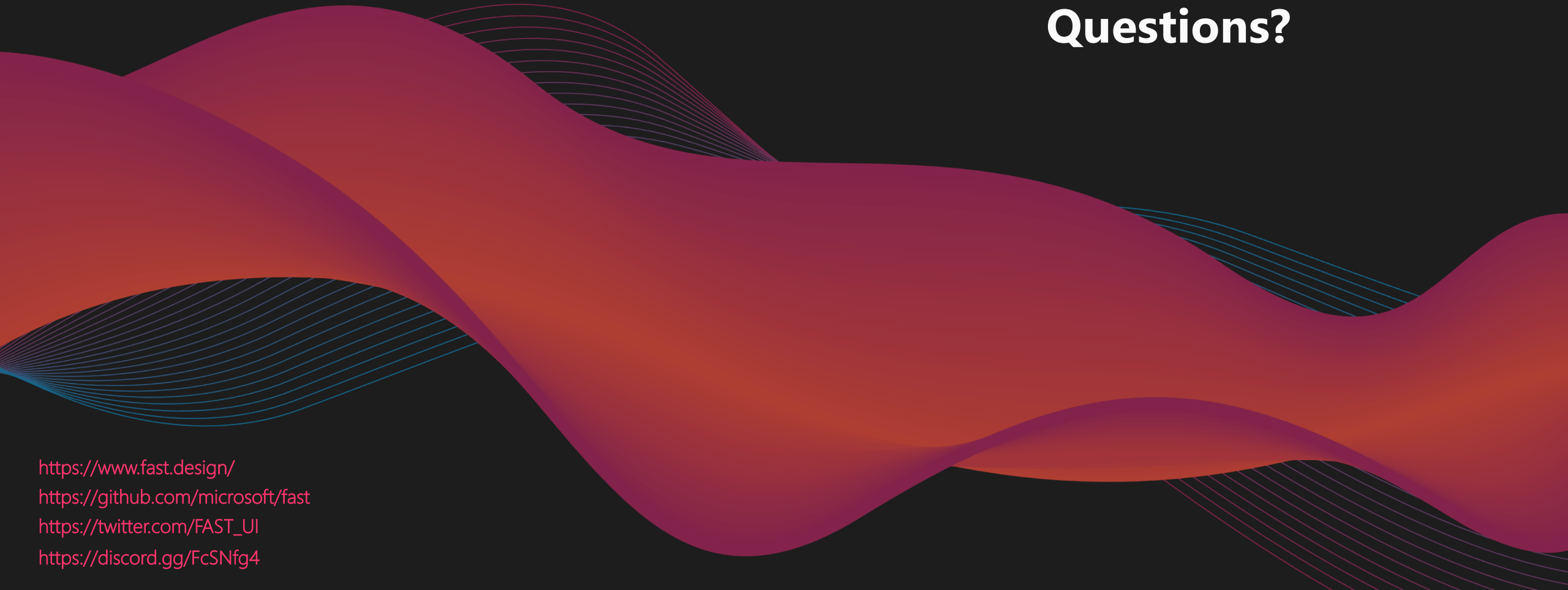
DEMO

Leveraging Adaptive UI

FAST

Thank you!

Questions?



https://www.fast.design/
https://github.com/microsoft/fast
https://twitter.com/FAST_UI
https://discord.gg/FcSNfg4