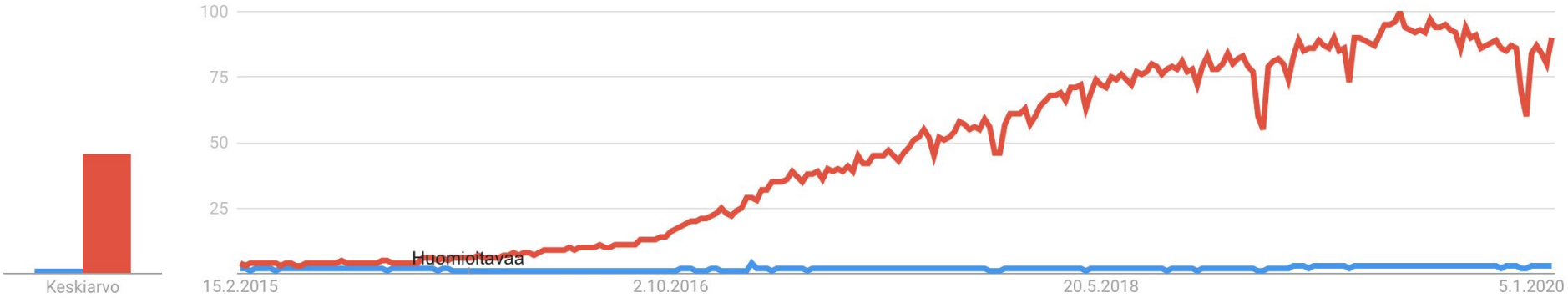
Intro to Web Components

•••

Jaakko Juvonen
Luoto Company

Google Trends - Web Components vs Vue.js



→ Still not a big player, though..

Google Trends - jQuery

→ Better web platform make libraries less useful

What are Web Components

- A set of standard web APIs
- Allows extending HTML with new elements
- Allows bundling structure, behavior and styles into isolated component
- Isn’t opinionated - use vanilla APIs or any framework to implement
- Use inside any framework - they behave like standard HTML elements

Why to use it?

- All the benefits of modeling app using semantic components - maintainability, code sharing, ...
- Well-defined and standards-based API → interoperable components, no lock-in to any framework
- Big players already adopted: YouTube, GitHub, Salesforce
- Already about 5% of global page loads use web components in some way

Browser APIs for Web Components

- Custom elements
  - Define new elements. Combine js and structure.
- Shadow DOM v1
  - Sandbox CSS and DOM
- HTML templates
  - Reusable HTML chunks
- HTML imports (deprecated)
  - Easy sharing
  - ***HTML modules*** may be the future

Basic implementation ideas

- Component's public API is it's
  - Attributes and properties
  - Events
  - Slots
  - CSS custom properties
- Data down, events up
- Example

Frameworks / tools for building Web Components

- Plain web APIs
- LitElement (Polymer / Google)
  - Lightweight, very React-like
  - Automatic property and event bindings
  - No build step, good for simple use cases
- Haunted
  - Build Web Components like React functional components and **hooks**
- Stencil (Ionic)
  - “Compiler for Web components”
  - Lot of compile-time goodies like TypeScript, JSX and exports to popular frontend framework components
  - Good for serious Web components stuff like publishing components and building whole apps
- Vue, Angular (probably also other frontend frameworks)
  - Provide relatively easy way to wrap components as Web components

Using inside frameworks

- Detailed info about framework interoperability: Custom Elements Everywhere
- Most important aspects for framework interoperability:
  - Can do both property and attribute binding
  - Can listen native DOM events
- React is not well-behaving
- Preact, Vue, Angular and many others work well

Libraries and resources

- Polyfills for Custom elements and Shadow DOM
- Ionic web components - high-quality ui components
- Material UI web components (WIP)
- Wired Elements - for wireframing
- open-wc - **very** high quality guides and project templates

Styling web components

- Shadow DOM encapsulates component styles
  - Inheritable styles leak in (on purpose)
- Two ways to reach out from Shadow DOM
  - :host, :host-context()
  - ::slotted()
- CSS Custom properties
  - Good for theming, not for sharing styles
- Linking to shared stylesheets from Shadow DOM
- In the future: Constructable style sheets

Summary (opinionated)

- *“Docker container of web"* - write once, use everywhere
- Set of standards, pick the ones you need
- Already good browser support especially with polyfills
- Developer-friendliness with web frameworks varies
- Not a big thing yet, ecosystem is still forming
- Already good for e.g. style guides, highly specialized widgets and micro-frontends
- **Eventually platform will win**

Demo time!

Questions?

Thank you!