knowit

knowit

React, you’re doing it wrong

knowit

Agenda

- About me
- Definitions
- What are we doing wrong
- Alternative solution
- The tooling more or less in action
- Conclusion

Food recommendation – Consume with a grain of salt

knowit

Who to blame for this presentation?

Mikko Luhtasaari from Knowit Solutions

For the past five years

- 3 months of Java
- A bit over 4 years of everything JS/TS like Node.js, React, TestCafe etc etc.
- Currently Innovation Zone leader in Finland, one of our Javascript guild leaders

Conversation starters for beginners

- How’s your lovely dog doing? Are you planning on getting another dog?
- What was the biggest difference between EA NHL 2013 and 2023?
- How’s Marvel Snap?

knowit

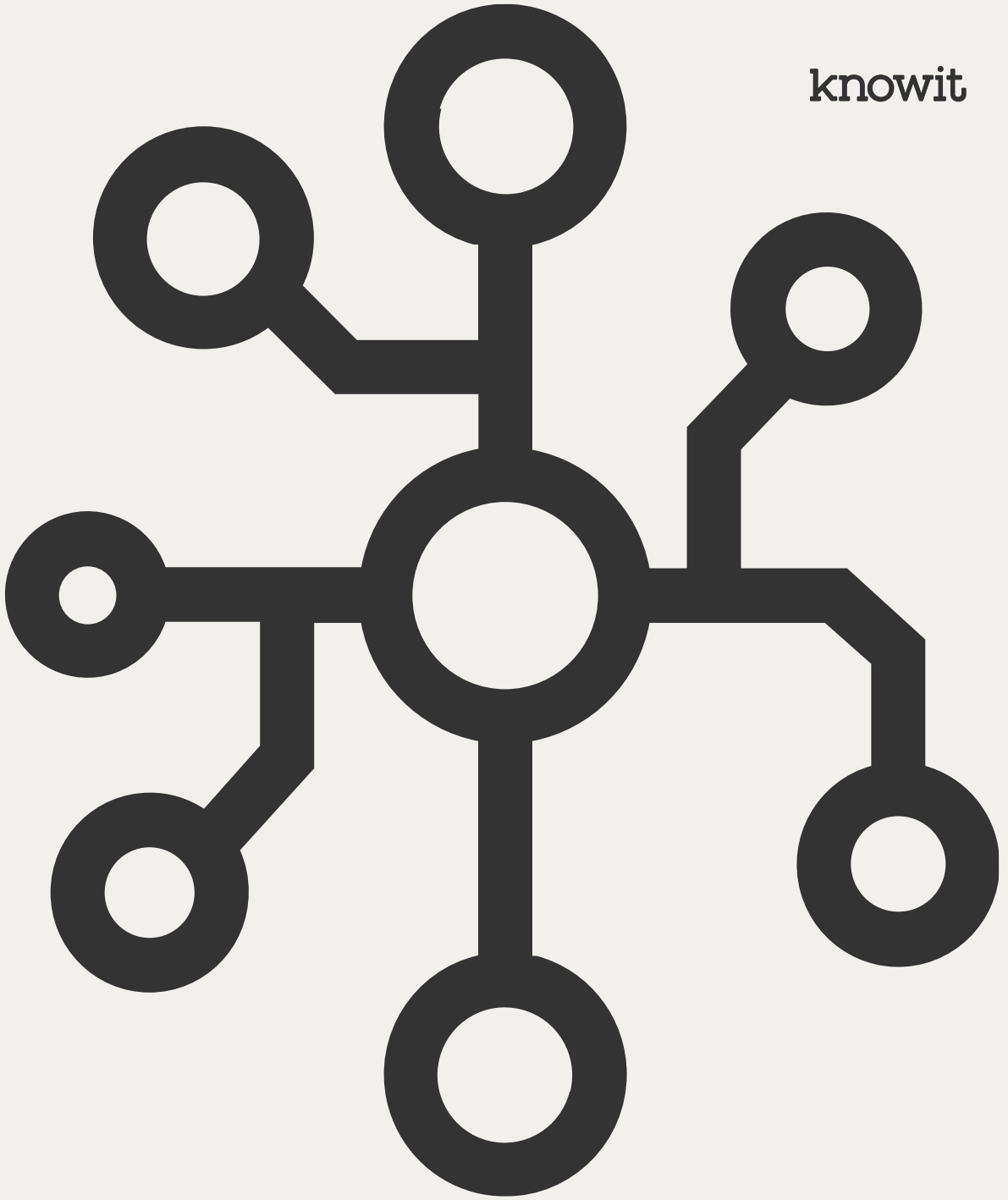
Component driven development

Components are the building blocks of the UIs, think of Lego bricks

How to be component driven?

- Build one component at a time
- Combine components
- Assemble pages
- Integrate pages into your project

See: https://www.componentdriven.org/



knowit

Two different kinds of components

Ad-hoc components

- Made with a specific use case in mind
- Typically lives in the projects /components folder
- Can have internal state, Redux connections, data fetching, etc.

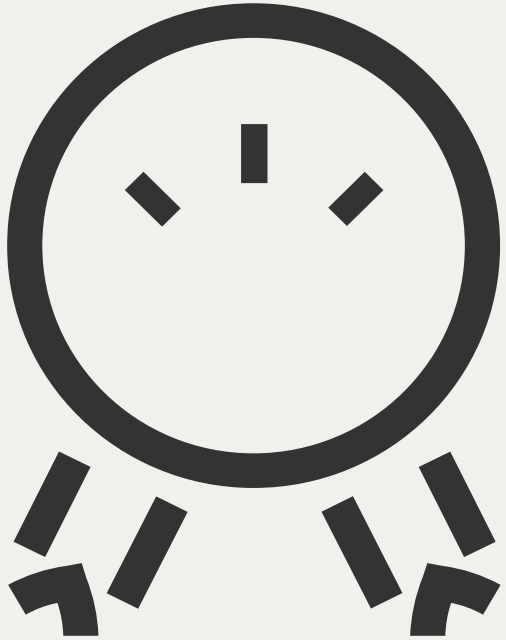
Reusable components

- Customizable components for general use cases
- Typically live in a component library or a design system
- No internal state, not connected to Redux, doesn’t fetch data, doesn’t contain business logic

knowit

What does the React docs teach you?

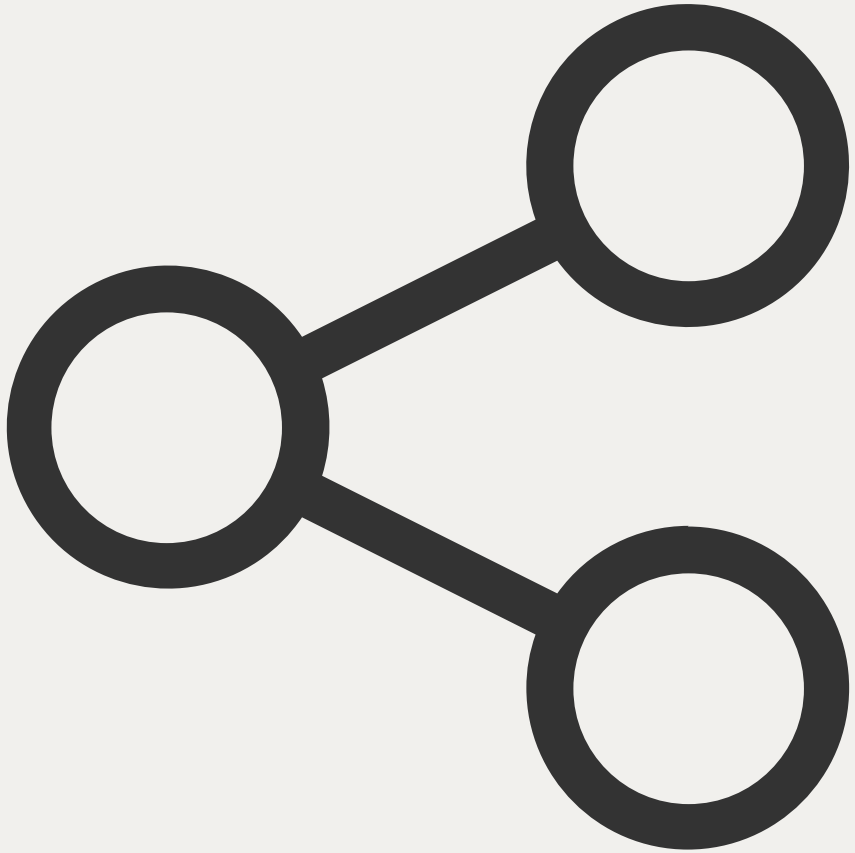
- React teaches you the correct things
  - / Pure components
  - / Lifting the state up
  - / Using reducers on app level
  - / Separation of concerns
- Why do we tend to lean towards ad-hoc components?
  - / Examples are simplified
  - / Tooling like Redux or Context makes it convenient
  - / They are fast to build for a single application



knowit

What’s wrong with ad-hoc components?

- Testability
  - Some boilerplate is needed in order to mock the hooks
- Separation of concerns
  - Does your component need to know where in the Redux state the variables are stored?
- Can you share information about your components?
  - Give me a link that shows your primary button as disabled
  - Prove that your component handles errors as specified
  - How does your component react to an empty object or array
- Components are hard to reason about
  - useState, useEffect, useInterval, useReducer, useContext etc etc.



knowit

Reusable components to the rescue!

- Separation of concerns
  - This is how the component looks like
  - This is what is required from the application the component is integrated into
- Developer experience (as always, a personal preference)
  - No need to worry about Redux when developing components
  - Bite sized development
  - Can reach different component states easily
- UX reviews
  - How are your UX designers approving your visual changes?
  - Do your UX designers need access to your version control?

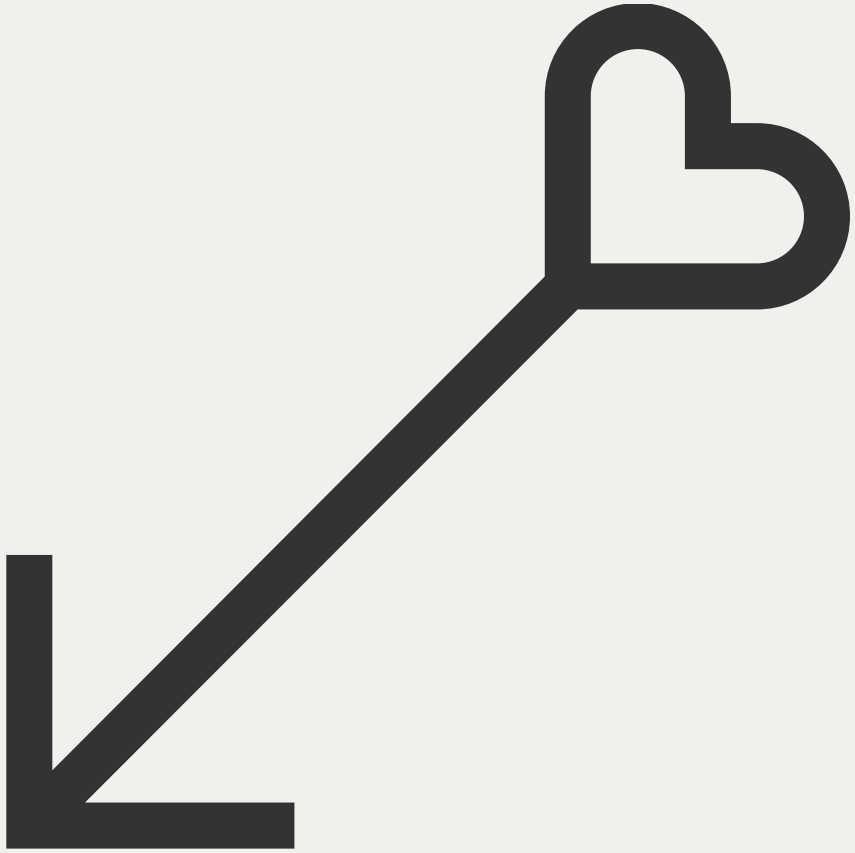
knowit

As always there’s no silver bullets..

- Extra tooling is required
  - Somewhere to develop the components
  - How to distribute the library
- Slower to get started
  - Can’t just jump straight in to developing new features
  - Setup the component development environment
- Requires a new process
  - Create new components or customize existing ones
  - Making breaking changes
  - Deprecating old code

knowit

Storybook

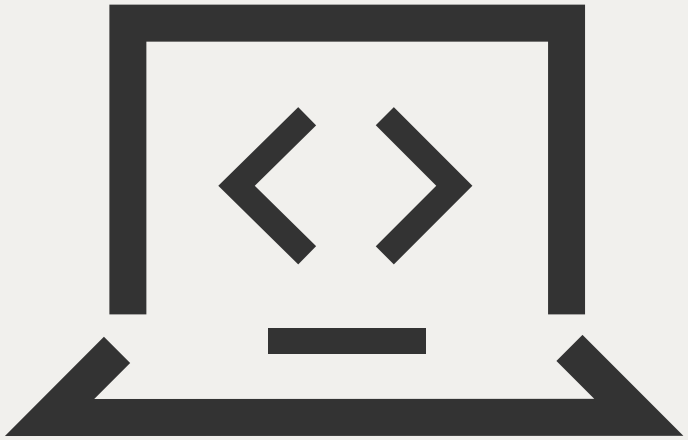


- A tool for building components and pages in isolation
  - Work only on a single component at a time
- The best features
  - Document components
  - Large variety of possible tests to run easily
  - Lots of addons to use (pro and con)
  - Publish an online playground of your components
- A story captures the rendered state of a component
  - Many stories per component for different component states

knowit

Storybook testing

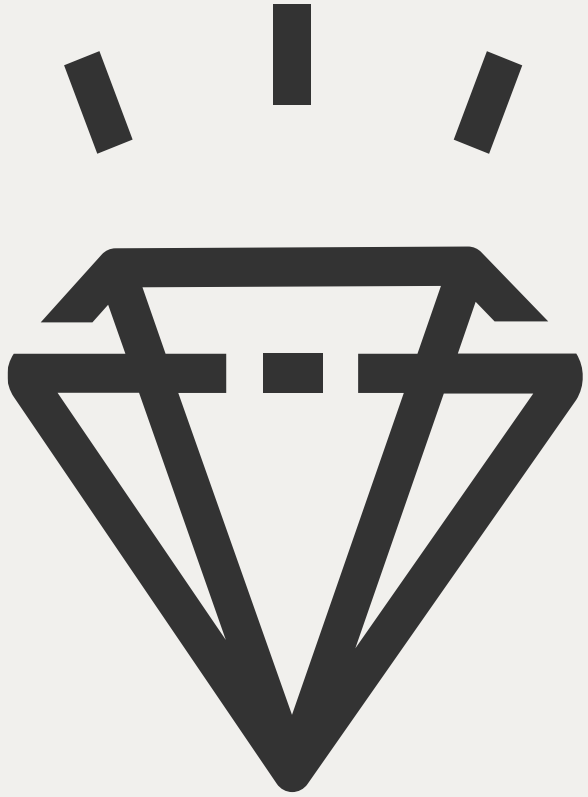
- Test runner
  - Automatically catch broken stories
- Visual tests (Chromatic)
  - Capture images of your components and diff the changes in a separate environment
- Accessibility tests (a11y addon)
  - Automatically audit components against some accessibility violations
- Interaction tests (interactions addon)
  - Simulate user interactions on components
- Coverage tests, Snapshot tests



knowit

Storybook in action

- Materials available at: https://github.com/knowit-finland-javascript-guild/tamperejs-march-2023 or https://bit.ly/3jQ62X6



knowit

Recommendations

- For one front end project without need to ever reuse components
  - / Ad hoc components stronk
- For one front end project but the future is looking bright
  - / Try it out and see how you like it
- For multiple front end projects
  - / Be component driven
  - / Share components across projects
  - / Properly test your components since it just became way easier

knowit

Disclaimer

- Don’t change your whole process just because some rando told you to
  - Not all projects will benefit from this
- I’ve been wrong about trivial, minor and major things
  - Oh so terribly wrong
- Components:

knowit

Further reading

- https://www.componentdriven.org/
- https://bradfrost.com/blog/post/atomic-web-design/
- https://storybook.js.org/
- https://bit.dev/
- https://monorepo.tools/
- https://www.nngroup.com/articles/design-systems-101/

knowit

Makers of a sustainable future