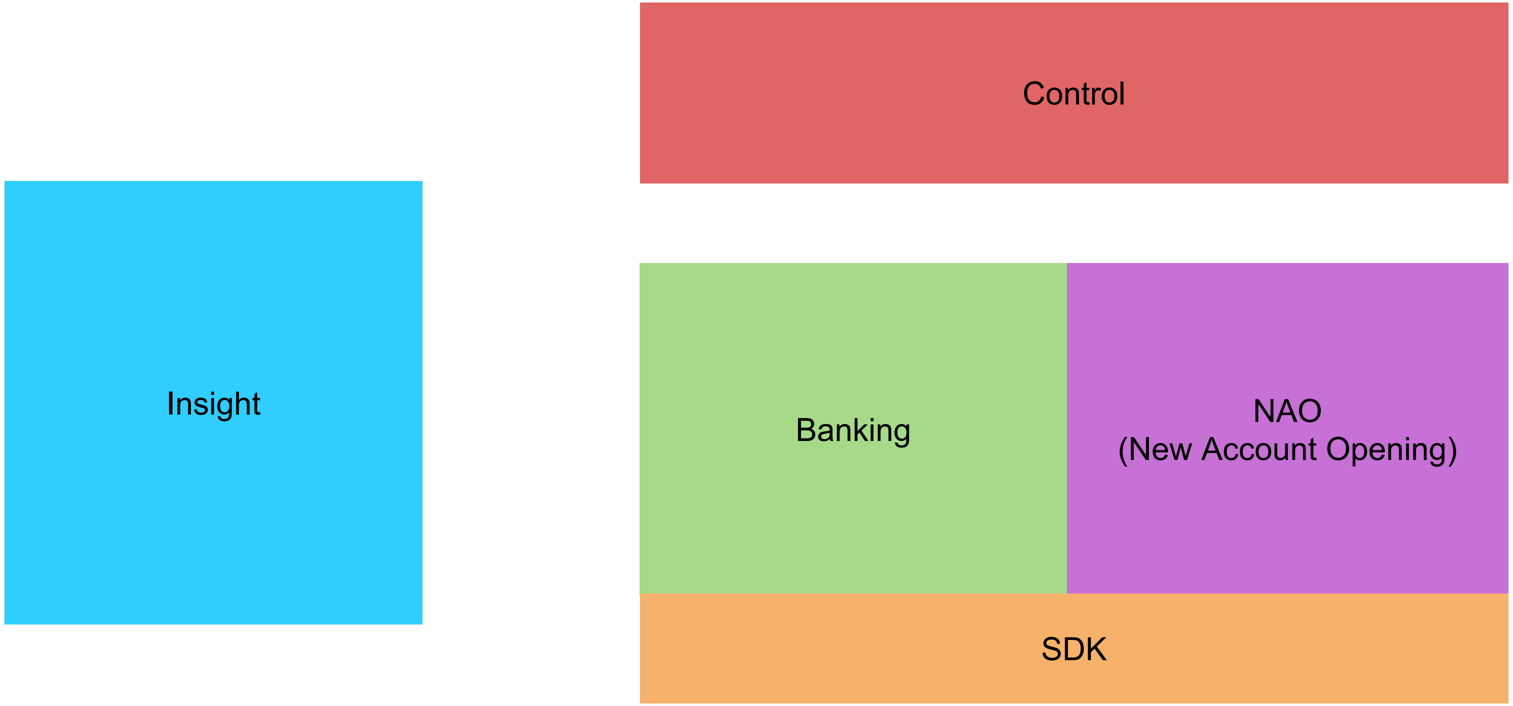
BANKING
TECHNOLOGY

Web Developer Training

BANKING
TECHNOLOGY

Web Architecture Overview

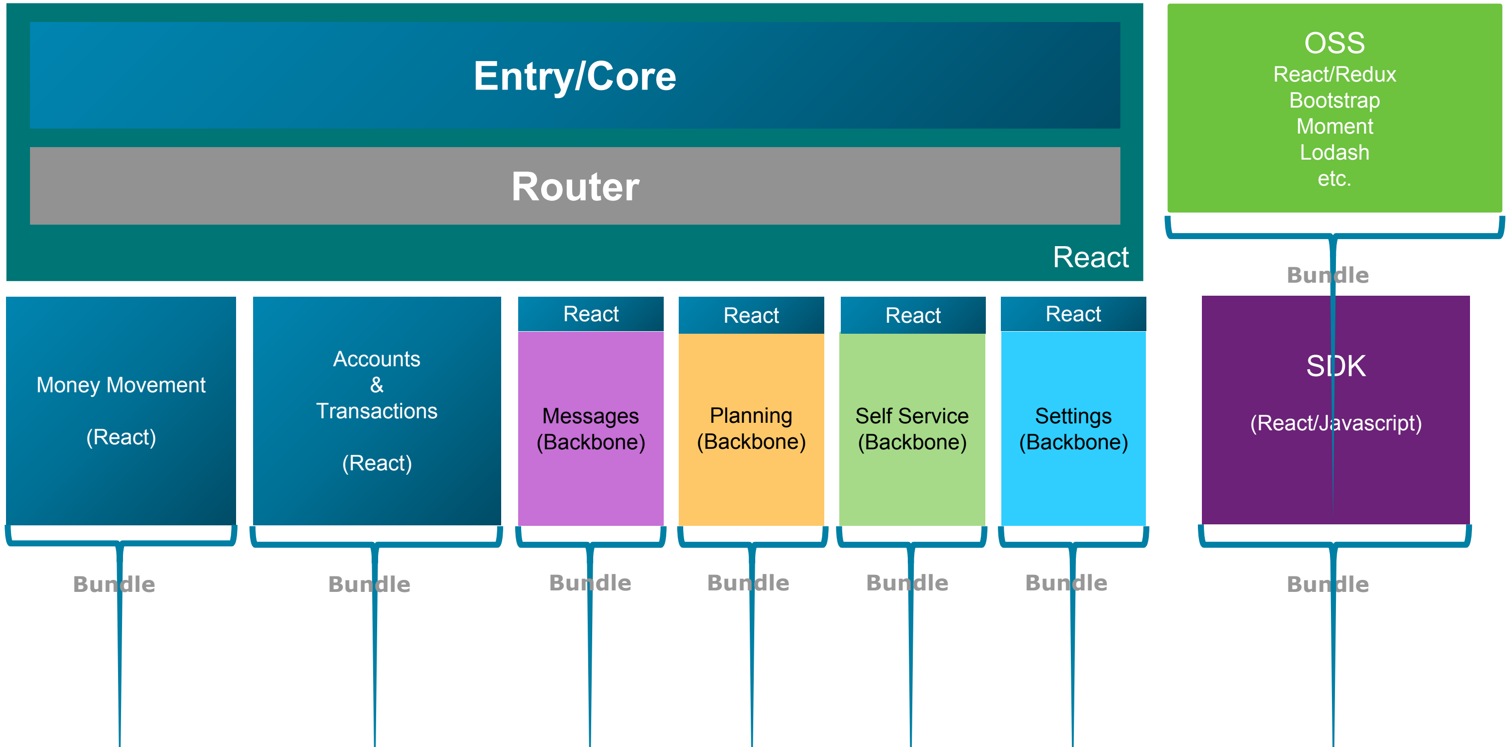
Web Apps



Web Philosophy

- Fast to market
  - Rapid Releases
  - On Demand Builds
  - Independently versioned modules
- Performance
  - Smallest Bundles
  - Dynamically Loaded Bundles
  - Progressive Loading
- Consistency
  - Shared Components
  - SDK
- Quality
  - Well Tested
  - Coding Standards and Practices
- Accessibility
- Customizable
  - Branding and Theming
  - Semantic Markup
  - Extension Framework
  - Component Configurations

Web UI for 4.0


Entry/Core
Router
React
OSS
React/Redux
Bootstrap
Moment
Lodash
etc.
Bundle
Money Movement
(React)
Accounts
&
Transactions
(React)
React
Messages
(Backbone)
React
Planning
(Backbone)
React
Self Service
(Backbone)
React
Settings
(Backbone)
SDK
(React/Javascript)
Bundle
Bundle
Bundle
Bundle
Bundle
Bundle
Bundle

Web-SDK Overview

- Private NPM library (with Typescript typings)
- Themed UI Elements/Widgets (React)
    - Forms, inputs, buttons, headers, typography, etc.
    - Page layouts
- Common Utilities & Services
    - Formatters (money, number)
    - Device specific checks & native app communication
    - Event/Notification services
- CSS-in-JS styling utilities (Emotion)
    - Media queries
    - Color manipulation
- Storybook
    - Documentation Portal
    - How to use SDK components

Extensions Overview

- Stylesheet Extensions
  - Plain CSS
  - Loaded after D3 styles
  - Easily override D3 styling/theming with own CSS
  - D3 markup uses semantic markup to help overrides
- Javascript Extensions
  - Add new components/bundles/features that aren’t part of the D3 product
  - Modify existing features
  - API of events, routing, navigation

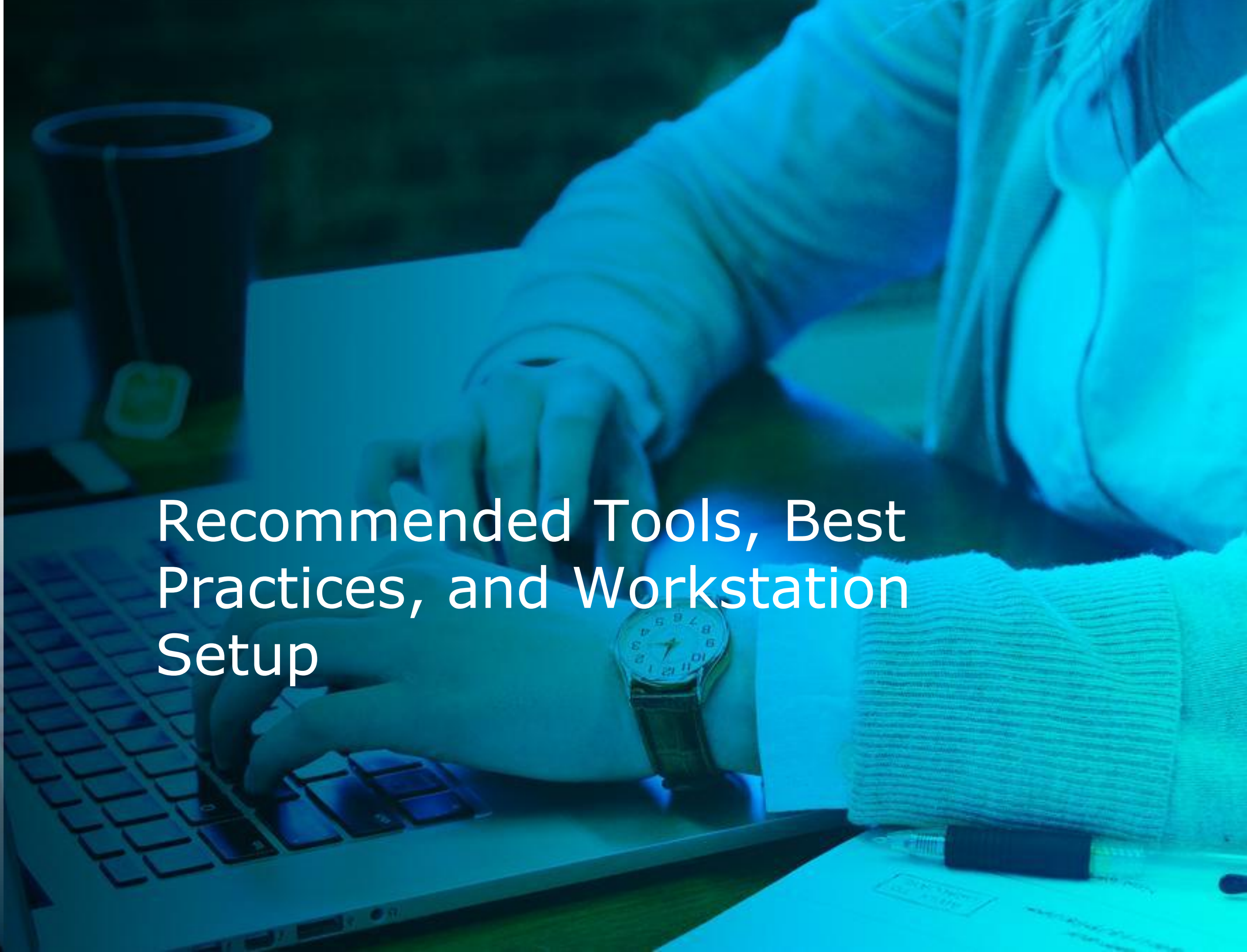
Long Term Goals

- Developer Portal (Storybook)
  - Internal/private website for developers
  - Access to live demos & documentation on SDK
- Jarvis Web
  - Static web builds (v3 Theme, Extensions, L10N?)
  - Improved web performance
  - Reproducible
  - On demand builds
- Independently loadable components that can be deployed separately
  - Smaller deliveries, contained changes
- Multiple feature component options configurable by client
  - Vertical/Horizontal Nav, Components for same function but different workflows, etc.

Web Technologies

- Typescript
  - Better developer experience and easier refactoring
- Migrating from BackboneJS to React/Redux
  - Components and Unidirectional data flow
  - Simpler/faster development
- CSS-in-JS - Emotion
  - Simplified component specific styling
  - Isolated styling for components
- Layout - Bootstrap
  - Row/Column based layouts
  - Semantic spacing (padding, margin, etc)
  - Flex based

D3 BANKING TECHNOLOGY



Recommended Tools, Best Practices, and Workstation Setup

Workstation Setup, Recommended Tools, and Best Practices

- Editors
- Build Tools
- Plugins
- Code Standards
- Testing

Editor

- VS Code
    - Lightweight
    - Fast
    - Typescript integration
    - Lots of plugins for customization

- Recommended Plugins
    - TSLint and/or ESLint
        - Assists with code quality and consistency
    - vscode-icons
        - Adds icons to the files/folders in the explorer for better/quick indicator of what a file is

Build Tools

- Prerequisites
  - Node (LTS Recommended)
  - Yarn or NPM

- Bundler/Dev Server
  - Webpack
- Transpiler
  - Babel
    - Ensures code can be run across all browsers
- All-In-One
  - Create React App

Code Quality and Consistency

- ESLint / TSLint
  - Code linters help maintain code consistency and quality
  - Fully customizable to support developer preferences
  - D3 publishes our configuration for each
    - @d3banking/tslint-config
    - @d3banking/eslint-config
- Linting should be run as part of CI process before code gets committed into main codebase

Code Quality and Consistency

- ES6 / ES2015 / ESNext
  - The next iterations of the Javascript language that are used within the application.
  - Since the application needs to support IE11, a tool called Babel is used to transpile the code into code that can be recognized in the older browsers.
  - Prefer native javascript over libraries like Lodash, Underscore, jQuery, etc.
    - Less 3rd party dependencies keep the bundle size down and can help with performance

Testing

- Testing framework
  - Runner - Jest
  - React Components - Enzyme
- Test the behavior of components and utilities with various input scenarios
- Good tests helps ensure code does not get broken accidentally, have more confidence
- Tests should be run as part of CI process before code gets committed into main codebase

Code Reviews

- Benefits
  - Finding bugs in the code
  - Ensuring code quality
  - Teaching and sharing knowledge
  - Shared accountability
- Process
  - Make code changes in feature branch
  - Open Pull Request in GitHub
  - 2 other developers review code
  - Make any fixes from code review
  - Ensure tests to pass (Travis or Jenkins)
  - Merge code into main codestream

Workstation setup

https://github.com/LodoSoftware/web-training/wiki/Environment-Setup

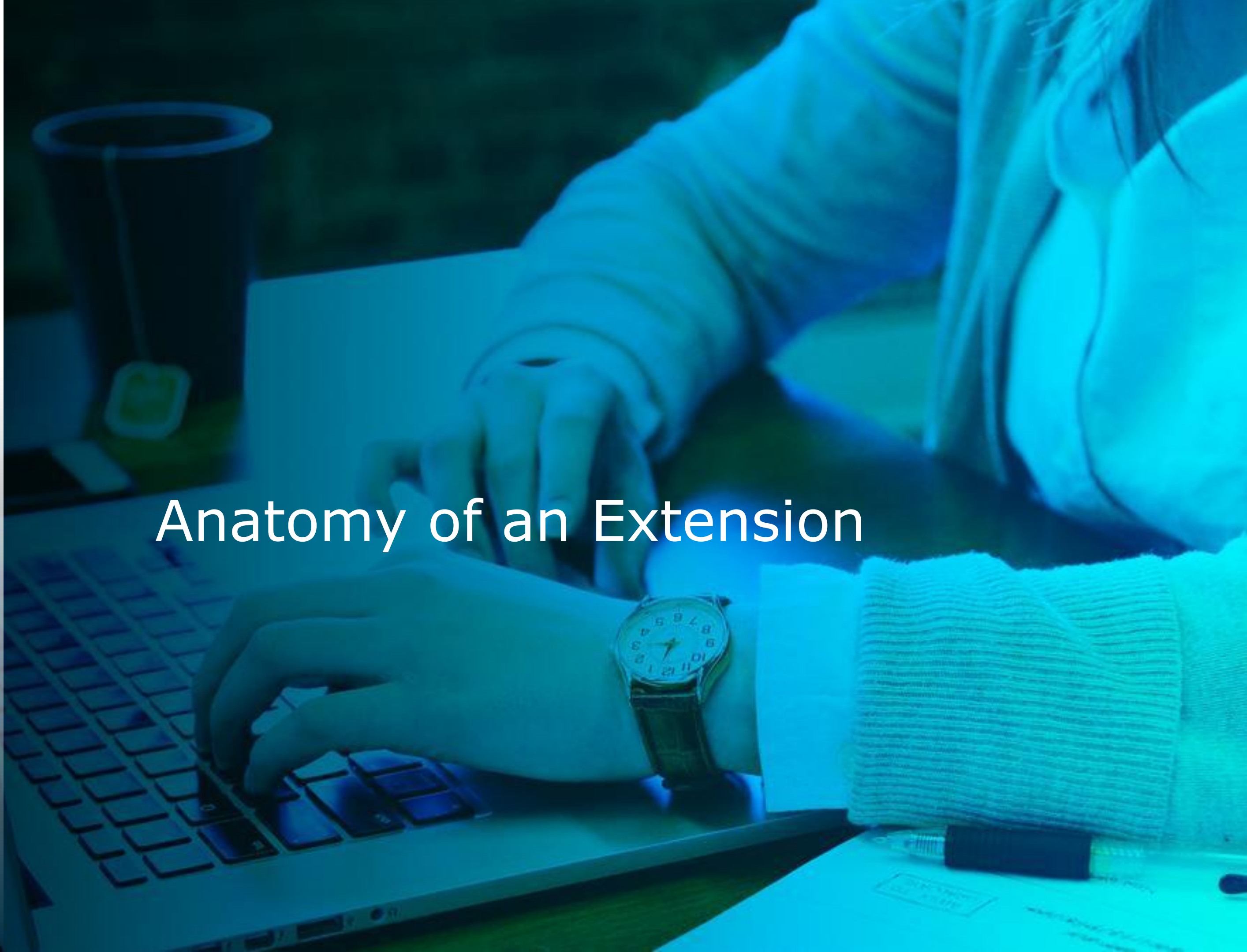
BANKING
TECHNOLOGY

SDK Documentation Walkthrough

SDK Documentation Walkthrough

- Storybook Demo
    - High level documentation
    - Changelog
    - Utils (typedoc)
    - Components
        - View all options
        - Customize props

BANKING TECHNOLOGY



Anatomy of an Extension

Anatomy of an Extension

- Types of extensions (Module vs Modification)
- Extension API Deep Dive
- When to use an extension
    - Branding & Theming
    - Localization

Module Extensions

- Build time
- Add, replace, customize any part of web UI
- Examples
  - Building a customized view (a new main or subnav item)
  - Replacing a section of the D3 web UI with the client's own content

- Advantages
  - Most performant option
  - Does not rely on D3 markup or styling
- Disadvantages
  - Module must be packaged and provided to D3 to be bundled with the application (can be an NPM module)

Modification Extensions

- Run time
- Add, replace, customize any part of web UI
- Examples
  - Styling updates/changes to existing UI
  - Adding to or modifying the existing D3 web UI components

- Advantages
  - Built and deployed completely independent of D3
- Disadvantages
  - Can be reliant on D3 markup or styling (depending on what is being done)

Module vs Modification Extensions

- You can achieve the same result with both types of extensions
- The entire D3 UI is built using Module extensions

Extension API Deep Dive

- Storybook
- Walkthrough Docs
    - Extensions
    - Services
    - Store
    - Styles
    - Examples

When to use an Extension

- Do I need to change colors, fonts or images?
  - Try Branding & Theming - Demo

- Do I need to change some text?
  - Try Localization - Demo

- Create an extension
  - CSS extensions are an easy way to achieve styling or layout changes
  - JS extensions unlock the full capabilities and allow for complete customization of the UI

BANKING
TECHNOLOGY

Extensions Workshop

Extensions Workshop

- Setup extension project
- How to enable an extension
- Create a variety of extensions
    - Add custom styling
    - Modify elements on existing page
    - Adding a widget on existing page
    - Replace an existing module
    - Add additional module with navigation