Bringing Gutenberg Javascript outside of the Block Editor
Building a React WP-Admin App

— Interactivity
— Defined Extensibility
— Speed of Development
Getting Started

— New or existing codebase

^ This of course works great when you are starting a new plugin, but you can add it to an existing codebase as well.
Getting Started

— New or existing codebase
— REST API First
Getting Started

— New or existing codebase
— REST API First
— Don’t rewrite everything
WP Scripts

- Building
- Linting
- Testing
npm install @wordpress/scripts --save-dev
WP Scripts - Build/Start

- `npm run-script build`
- Builds code in `src/index.js` and outputs it to `build/index.js`
- Webpack
- Babel
- Dependency extraction
- No SCSS yet
Packages

- 60 odd packages
- Documented on developer.wordpress.org
Packages

- Components
- Data
- Plugins
Components Pre-Gutenberg

- Informal component library
- Limited set of components
Components Pre-Gutenberg

wpadmin.bracketspace.com
Components Pre-Gutenberg

- Informal component library
- Limited set of components
- Varying ease of use
  - submit_button()
  - Notices
  - Thickbox
Gutenberg Component Library

- Extensive
- Familiar patterns admin-wide
Gutenberg Component Library

— Speed of Development

— Dropzones, Dragging, Tab Panels, Modals, Popovers
Gutenberg Component Library

- Accessibility
- Modals/Popovers
- Focus Trapping
- Navigable Regions
Gutenberg Component Library

— Documented on developer.wordpress.org

— Usage Instructions
Storybook

https://wordpress.github.io/gutenberg/
Gotchas

— Most reliable in WP Admin
— Styles may be tied to the Block Editor
— CSS Selector Specificity
Data Package

— Brain of the Application
— Manages State
State – Application Data

- User generated content
  - Posts, comments, users, etc...
- Machine generated data
  - Logs
  - Emails
State – UI

- Current post
- Visible metaboxes
- Actions in progress
<table>
<thead>
<tr>
<th>Log Entry</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>[demo.test] Login Authentication Code</td>
<td>5:29 pm</td>
</tr>
<tr>
<td><a href="mailto:dev-email@flywheel.local">dev-email@flywheel.local</a></td>
<td></td>
</tr>
<tr>
<td>[demo.test] Scheduled malware scan found an error</td>
<td>1:44 pm</td>
</tr>
<tr>
<td><a href="mailto:dev-email@flywheel.local">dev-email@flywheel.local</a></td>
<td></td>
</tr>
<tr>
<td>[demo.test] Daily Security Digest</td>
<td>1:44 pm</td>
</tr>
<tr>
<td><a href="mailto:dev-email@flywheel.local">dev-email@flywheel.local</a></td>
<td></td>
</tr>
<tr>
<td>[demo.test] Scheduled malware scan found an error</td>
<td>1:28 am</td>
</tr>
<tr>
<td><a href="mailto:dev-email@flywheel.local">dev-email@flywheel.local</a></td>
<td></td>
</tr>
<tr>
<td>[Demo] Background Update Finished</td>
<td>12:31 am</td>
</tr>
<tr>
<td><a href="mailto:dev-email@flywheel.local">dev-email@flywheel.local</a></td>
<td></td>
</tr>
<tr>
<td>[Demo] Your site has updated to WordPress 5.3-alpha-46100</td>
<td>12:31 am</td>
</tr>
<tr>
<td><a href="mailto:dev-email@flywheel.local">dev-email@flywheel.local</a></td>
<td></td>
</tr>
<tr>
<td>[demo-test] Daily Security Digest</td>
<td>10:20 am</td>
</tr>
</tbody>
</table>
[demo.test] Daily Security Digest

To: dev-email@flywheel.local

Your Daily Security Digest for September 13, 2019

Daily Security Digest

The following is a summary of security related activity on your site:
{  
  // List of email ids for the list of emails
  list: [],
  // Object of ids to their full object
  byId: {},
  // List of ids of items that are being deleted
  deleting: [],
  // List of ids of items that are being fetched
  fetching: [],
  // Is the inbox currently being updated
  emptyingInbox: false,
  // Is the list being fetched
  isFetching: false,
  
}
Data Store

- Layer on top of state
- With a defined API
- Integrates with React
Actions

- Select an email to view
- Delete an email
- Refresh the inbox
- Receive the emails
Actions

dispatch('wpmd/emails').fetchEmails();
dispatch('wpmd/emails').deleteEmail( emailId );
dispatch('wpmd/app').viewEmail( emailId );
Actions

{
  type: 'DELETE_EMAIL',
  emailId: 5,
}

Reducer

— Given an action just took place, what is the new application state
<table>
<thead>
<tr>
<th>Action</th>
<th>Old State</th>
<th>New State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetch Items</td>
<td>isFetching: false</td>
<td>isFetching: true</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive Items</td>
<td>list: []</td>
<td>list: [1, 2]</td>
</tr>
<tr>
<td>Finish Fetch</td>
<td>isFetching: true</td>
<td>isFetching: false</td>
</tr>
</tbody>
</table>
Selectors

- Hide "raw" state
- Answers a question based on the raw state
- Transforms state into richer values
function isFetching(state) {
    return state.isFetching;
}

select('wpmd/emails').isFetching();
function getEmails(state) {
    const ids = state.list, byId = state.byId;
    const emails = new Array(ids.length);
    let index = -1;

    while (++index < ids.length) {
        emails[index] = byId[ids[index]].email;
    }

    return emails;
}
select('wpmd/emails').get_emails();

[{
  id: 1, subject: 'Hello World'
},
{
  id: 2, subject: 'Password Reset'
},
]
Multiple Data Stores

- Multiple stores for an application
- Different store for each context
Application UI Store

— What is reusable in other contexts?
— What is specific to this interface?
Benefits over Redux?

— Forces you to define an API, adding semantics to a cryptic state shape
— Easier for 3rd party developers to use
Benefits over Redux?

- Forces you to define an API, adding semantics to a cryptic state shape
- Easier for 3rd party developers to use
- Resolvers
- Async actions
- Testability
Plugins Package

- Define code for inclusion in another application
- 3rd party developers
- Modular 1st party code
<PluginArea />

- Render in application root
- Once per-application
- Not where plugins appear
registerPlugin()

— Unique slug for the plugin
— Render function
registerPlugin('my-plugin', {
  render() {
    // Todo
  }
});
Slot Fill

- `<Slot name="Sidebar" />`
- `<Fill name="Sidebar" />`
- Unique slot name per pluggable area
Create the Slot Fill

```javascript
import {createSlotFill} from '@wordpress/components';

const { Fill, Slot } = createSlotFill( 'Sidebar' );

export { Fill as SidebarFill, Slot as SidebarSlot };```
Register a Fill

```javascript
registerPlugin('my-plugin', {
    render() {
        return (.DeserializeObject
            <SidebarFill>
                Hello World!
            </SidebarFill>
        );
    }
});
```
Display the Fills

function EmailSidebar( { email } ) {
    return (  
    <div className="wmd-email-sidebar">
        <Headers email={ email } />
        <SidebarSlot />
    </div>
    );
}
Passing data

- fillProps are passed to fills
- Fill must be a function
Passing data

<SidebarSlot fillProps={{ email }} />

render() {
  return (  
    <SidebarFill>
      {(props) => (  
        <div>{props.email.subject}</div>
      )}
    </SidebarFill>
  )
}
Sharing Components

- Export components in your entry file
- Specify library and libraryTarget
Sharing Components

```javascript
export {SidebarFill} // src/index.js

output: { // webpack.config.js
  ...defaultConfig.output,
  library: [ 'wpMailDebugger', '[name]' ],
  libraryTarget: 'this',
},

const {SidebarFill} = wpMailDebugger.entry;
```
Demo
Questions

Slides & Plugin: timothybjacobs.com