application.js:

Dieser Code enthält eine Sammlung von Import-Statements, um verschiedene Bibliotheken und Ressourcen in eine Webanwendung einzubinden. Hier ist eine Übersicht über den Code:

- Die erste Zeile ist ein Kommentar, der anzeigt, dass die Konsole nicht verwendet werden soll.
- Die nächsten Kommentare beschreiben, dass dieser Datei von Webpack automatisch kompiliert wird und dass sie dazu verwendet werden sollte, um auf die Anwendungslogik zu verweisen.
- Es werden verschiedene Bibliotheken und Ressourcen wie jQuery, Bootstrap, Chosen, jstree, underscore, d3 und Sentry importiert und in den globalen Namespace exportiert.
- Es werden CSS-Dateien für Chosen und jstree importiert.
- Es werden verschiedene Bibliotheken aus der jQuery-UI-Bibliothek importiert, um minimalen Mausinteraktionssupport zu ermöglichen. Dabei werden auch zugehörige CSS-Dateien importiert.
- Es werden Internationalisierungs-Ressourcen in der JSON-Datei "locales.json" importiert und mit der i18n-js-Bibliothek gespeichert.
- Es wird die Benutzersprache aus dem HTML-Tag "lang" extrahiert und als aktuelle Sprache für die i18n-js-Bibliothek festgelegt.
- Es werden Routen importiert und im globalen Namespace exportiert.

Zusammenfassend handelt es sich um eine Datei, die eine Sammlung von Ressourcen und Bibliotheken in eine Webanwendung importiert und diese in den globalen Namespace exportiert, um sie in der Anwendung zu nutzen.

Oder:

1. Import von JavaScript-Bibliotheken:
    - jQuery, jQuery-ujs, Bootstrap, Chosen, jstree, Underscore, D3, Sentry und Sorttable
2. Veröffentlichung von Bibliotheken im globalen Namensraum
3. Import von CSS-Bibliotheken:
    - Chosen und jstree
4. Import von jQuery-UI-Bibliotheken für minimale Mausinteraktionsunterstützung
5. Import von I18n-Bibliotheken für mehrsprachige Unterstützung der Anwendung
6. Einstellung der Anwendungssprache basierend auf der Sprache des Benutzers
7. Import von Routen für die Anwendung

routes.js.erb:

Dieser Code generiert JavaScript-Code, der es ermöglicht, Rails-Route-URLs in JavaScript zu nutzen. Es wird ein JavaScript-Modul mit dem Namen `JsRoutes` generiert, das Funktionen für jede in Rails definierte Route bereitstellt.

Der Code wird im Template einer Rails-Anwendung verwendet und verwendet die Ruby-Gem `js-routes`. Diese Gem generiert JavaScript-Code, der eine assoziative Datenstruktur mit allen verfügbaren Rails-Routen und deren URLs enthält.

Die Verwendung von `<%= JsRoutes.generate %>` im HTML-Code eines Rails-Views erzeugt den generierten JavaScript-Code, der dann in den Browser geladen wird. So können Rails-Routen im JavaScript-Code aufgerufen werden, ohne hartcodierte URLs zu verwenden.

sortable.js:

Dieser Code deaktiviert die Eslint-Regel `no-console` für das gesamte JavaScript-Modul, in dem er sich befindet. Die Eslint-Regel `no-console` erfordert normalerweise, dass alle Konsolenausgaben in JavaScript vermieden werden sollten.

Im JS-Teil des Codes wird das JavaScript-Modul `sortablejs` importiert und auf das globale Fensterobjekt `window` gesetzt. Dadurch kann `Sortable` im gesamten Skript verwendet werden. `sortablejs` ist eine Bibliothek zur Unterstützung von Drag-and-Drop-Interaktionen auf HTML-Elementen, z.B. zum Sortieren von Listen oder zum Ändern der Reihenfolge von Elementen.