Runtime I18n in Elm

Deceptively COMPLEX

Pre-build Phase

elm-i18n

Dynamically Loaded

elm-i18next


tachyons

Vertically centering things in css is easy!

English ▾

Vertically centering things in css is easy!

Back End

- Provide translations in JSON

Back End

- Provide translations in JSON
- Store selected language in `localStorage`

Back End

- Provide translations in JSON
- Store selected language in `localStorage`
- Explore type safety options

Vertically centering things in css is easy!

```
view : Model -> Html Msg
view model =
    main_ [ class Styles.main_ ]
        [ content ]


content : Html Msg
content =
    article [ class Styles.article ]
        [ div [ class Styles.articleContainer ]
            [ heading ]
        ]


heading : Html Msg
heading =
    h1 [ class Styles.heading ]
        [ text "Vertically centering things in css is easy!" ]
```

```elm
view : Model -> Html Msg
view model =
    main_ [ class Styles.main_ ]
        [ content ]


content : Html Msg
content =
    article [ class Styles.article ]
        [ div [ class Styles.articleContainer ]
            [ heading ]
        ]


heading : Html Msg
heading =
    h1 [ class Styles.heading ]
        [ text "Vertically centering things in css is easy!" ]
```

```
module Styles exposing (..)


main_ : String
main_ =
    [ "bg-dark-pink"
    , "overflow-container"
    , "pt3"
    , "sans-serif"
    , "vh-100"
    , "white"
    ]
        |> String.join " "
```

```elm
view : Model -> Html Msg
view model =
    main_ [ class Styles.main_ ]
        [ content ]


content : Html Msg
content =
    article [ class Styles.article ]
        [ div [ class Styles.articleContainer ]
            [ heading ]
        ]


heading : Html Msg
heading =
    h1 [ class Styles.heading ]
        [ text "Vertically centering things in css is easy!" ]
```

```elm
view : Model -> Html Msg
view model =
    main_ [ class Styles.main_ ]
        [ content ]


content : Html Msg
content =
    article [ class Styles.article ]
        [ div [ class Styles.articleContainer ]
            [ heading ]
        ]


heading : Html Msg
heading =
    h1 [ class Styles.heading ]
        [ text "Vertically centering things in css is easy!" ]
```

```elm
view : Model -> Html Msg
view model =
    main_ [ class Styles.main_ ]
        [ content ]


content : Html Msg
content =
    article [ class Styles.article ]
        [ div [ class Styles.articleContainer ]
            [ heading ]
        ]


heading : Html Msg
heading =
    h1 [ class Styles.heading ]
        [ text "Vertically centering things in css is easy!" ]
```

```
view : Model -> Html Msg
view model =
    main_ [ class Styles.main_ ]
        [ content ]


content : Html Msg
content =
    article [ class Styles.article ]
        [ div [ class Styles.articleContainer ]
            [ heading ]
        ]


heading : Html Msg
heading =
    h1 [ class Styles.heading ]
        [ text "Vertically centering things in css is easy!" ]
```

LANGUAGE Dropdown

Language Dropdown

- Show current language on menu

Language Dropdown

- Show current language on menu
- Reveal available languages

Language Dropdown

- Show current language on menu
- Reveal available languages
- Change language

Language Dropdown

- Show current language on menu
- Reveal available languages
- Change language
- Close on "blur"

```elm
module LanguageDropdown exposing (view)


view : Html msg
view =
    div [ class Styles.dropdownContainer ]
        [ currentSelection ]


currentSelection : Html msg
currentSelection =
    p [ class Styles.currentSelection ]
        [ span [] [ text "English" ]
        , span [ Styles.caret ] [ text "▾" ]
        ]
```

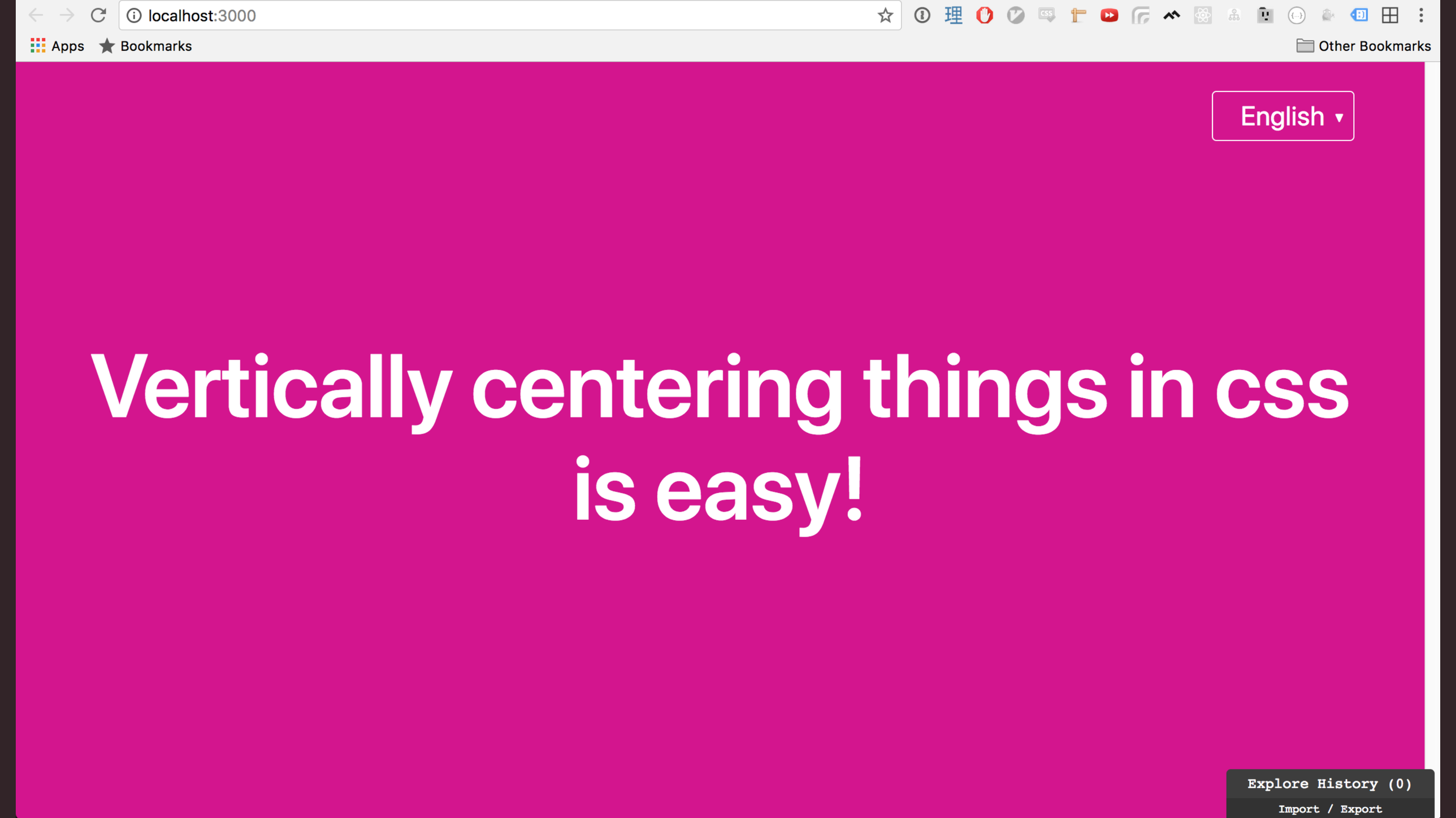
```elm
module LanguageDropdown exposing (view)


view : Html msg
view =
    div [ class Styles.dropdownContainer ]
        [ currentSelection ]


currentSelection : Html msg
currentSelection =
    p [ class Styles.currentSelection ]
        [ span [] [ text "English" ]
        , span [ Styles.caret ] [ text "▾" ]
        ]
```

```elm
module LanguageDropdown exposing (view)


view : Html msg
view =
    div [ class Styles.dropdownContainer ]
        [ currentSelection ]


currentSelection : Html msg
currentSelection =
    p [ class Styles.currentSelection ]
        [ span [] [ text "English" ]
        , span [ Styles.caret ] [ text "▾" ]
        ]
```

```elm
module LanguageDropdown exposing (view)


view : Html msg
view =
    div [ class Styles.dropdownContainer ]
        [ currentSelection ]


currentSelection : Html msg
currentSelection =
    p [ class Styles.currentSelection ]
        [ span [] [ text "English" ]
        , span [ Styles.caret ] [ text "▾" ]
        ]
```

```elm
module Main exposing (main)

import LanguageDropdown


view : Model -> Html Msg
view model =
    main_ [ class Styles.main_ ]
        [ LanguageDropdown.view
        , content
        ]
```

```elm
module Main exposing (main)

import LanguageDropdown


view : Model -> Html Msg
view model =
    main_ [ class Styles.main_ ]
        [ LanguageDropdown.view
        , content
        ]
```


localhost:3000
Apps
Bookmarks
Other Bookmarks
English
Vertically centering things in css is easy!
Explore History (0)
Import / Export

Language

DROPDOWN LIST

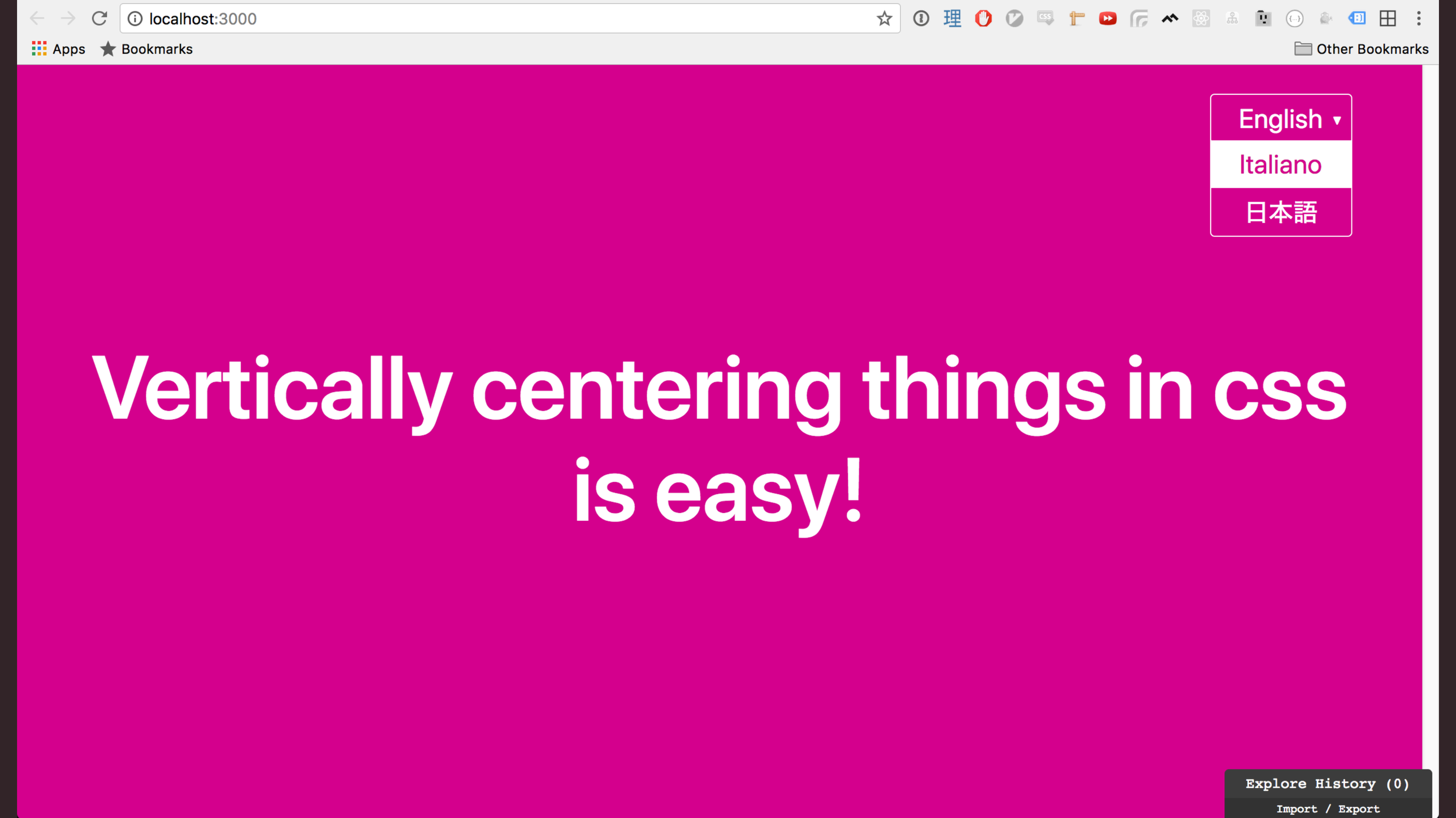
```elm
view : Html msg
view =
    div [ class Styles.dropdownContainer ]
        [ currentSelection
        , dropdownList
        ]

dropdownList : Html msg
dropdownList =
    let
        selectableLanguages =
            [ "Italiano", "日本語" ]
    in
        ul [ class Styles.dropdownList ]
            (List.map dropdownListItem selectableLanguages)

dropdownListItem : String -> Html msg
dropdownListItem language =
    li [ class Styles.dropdownListItem ]
        [ span [] [ text language ] ]
```

```elm
view : Html msg
view =
    div [ class Styles.dropdownContainer ]
        [ currentSelection
        , dropdownList
        ]

dropdownList : Html msg
dropdownList =
    let
        selectableLanguages =
            [ "Italiano", "日本語" ]
    in
        ul [ class Styles.dropdownList ]
            (List.map dropdownListItem selectableLanguages)

dropdownListItem : String -> Html msg
dropdownListItem language =
    li [ class Styles.dropdownListItem ]
        [ span [] [ text language ] ]
```

```elm
view : Html msg
view =
    div [ class Styles.dropdownContainer ]
        [ currentSelection
        , dropdownList
        ]

dropdownList : Html msg
dropdownList =
    let
        selectableLanguages =
            [ "Italiano", "日本語" ]
    in
        ul [ class Styles.dropdownList ]
            (List.map dropdownListItem selectableLanguages)

dropdownListItem : String -> Html msg
dropdownListItem language =
    li [ class Styles.dropdownListItem ]
        [ span [] [ text language ] ]
```

```
view : Html msg
view =
    div [ class Styles.dropdownContainer ]
        [ currentSelection
        , dropdownList
        ]

dropdownList : Html msg
dropdownList =
    let
        selectableLanguages =
            [ "Italiano", "日本語" ]
    in
        ul [ class Styles.dropdownList ]
            (List.map dropdownListItem selectableLanguages)

dropdownListItem : String -> Html msg
dropdownListItem language =
    li [ class Styles.dropdownListItem ]
        [ span [] [ text language ] ]
```

```elm
view : Html msg
view =
    div [ class Styles.dropdownContainer ]
        [ currentSelection
        , dropdownList
        ]

dropdownList : Html msg
dropdownList =
    let
        selectableLanguages =
            [ "Italiano", "日本語" ]
    in
        ul [ class Styles.dropdownList ]
            (List.map dropdownListItem selectableLanguages)

dropdownListItem : String -> Html msg
dropdownListItem language =
    li [ class Styles.dropdownListItem ]
        [ span [] [ text language ] ]
```


localhost:3000
Apps
Bookmarks
Other Bookmarks
English
Italiano
日本語
Vertically centering things in css is easy!
Explore History (0)
Import / Export

SHOW/HIDE
Available Languages

Show/Hide Available Languages

- Flag to `showAvailableLanguages` in model

Show/Hide Available Languages

- Flag to `showAvailableLanguages` in model
- Toggle dropdown list visibility (`ShowAvailableLanguages`)

Show/Hide Available Languages

- Flag to `showAvailableLanguages` in model
- Toggle dropdown list visibility (`ShowAvailableLanguages`)
- Hide dropdown list on "blur" (`CloseAvailableLanguages`)

```
type alias Model =
    { showAvailableLanguages : Bool }


init : ( Model, Cmd Msg )
init =
    ( { showAvailableLanguages = False }, Cmd.none )
```

```
type Msg
    = CloseAvailableLanguages
    | ShowAvailableLanguages
```

```
update : Msg -> Model -> ( Model, Cmd Msg )
update msg model =
    case msg of
        CloseAvailableLanguages ->
            ( { model | showAvailableLanguages = False }
            , Cmd.none
            )

        ShowAvailableLanguages ->
            ( { model
                | showAvailableLanguages =
                    not model.showAvailableLanguages
              }
            , Cmd.none
            )
```

```
update : Msg -> Model -> ( Model, Cmd Msg )
update msg model =
    case msg of
        CloseAvailableLanguages ->
            ( { model | showAvailableLanguages = False }
            , Cmd.none
            )

        ShowAvailableLanguages ->
            ( { model
                | showAvailableLanguages =
                    not model.showAvailableLanguages
              }
            , Cmd.none
            )
```

```
update : Msg -> Model -> ( Model, Cmd Msg )
update msg model =
    case msg of
        CloseAvailableLanguages ->
            ( { model | showAvailableLanguages = False }
            , Cmd.none
            )


        ShowAvailableLanguages ->
            ( { model
                | showAvailableLanguages =
                    not model.showAvailableLanguages
              }
            , Cmd.none
            )
```

```elm
update : Msg -> Model -> ( Model, Cmd Msg )
update msg model =
    case msg of
        CloseAvailableLanguages ->
            ( { model | showAvailableLanguages = False }
            , Cmd.none
            )

        ShowAvailableLanguages ->
            ( { model
                | showAvailableLanguages =
                    not model.showAvailableLanguages
              }
            , Cmd.none
            )
```

```elm
view : Model -> Html msg
view { showAvailableLanguages } =
    div [ class Styles.dropdownContainer ]
        [ currentSelection showAvailableLanguages
        , dropdownList showAvailableLanguages
        ]

currentSelection : Bool -> Html Msg
currentSelection showAvailableLanguages =
    p
        [ class (Styles.currentSelection showAvailableLanguages)
        , onClick ShowAvailableLanguages
        ]
        [ span [] [ text "English" ]
        , span [ class Styles.caret ] [ text "▾" ]
        ]
```

```elm
view : Model -> Html msg
view { showAvailableLanguages } =
    div [ class Styles.dropdownContainer ]
        [ currentSelection showAvailableLanguages
        , dropdownList showAvailableLanguages
        ]

currentSelection : Bool -> Html Msg
currentSelection showAvailableLanguages =
    p
        [ class (Styles.currentSelection showAvailableLanguages)
        , onClick ShowAvailableLanguages
        ]
        [ span [] [ text "English" ]
        , span [ class Styles.caret ] [ text "▾" ]
        ]
```

```
view : Model -> Html msg
view { showAvailableLanguages } =
    div [ class Styles.dropdownContainer ]
        [ currentSelection showAvailableLanguages
        , dropdownList showAvailableLanguages
        ]


currentSelection : Bool -> Html Msg
currentSelection showAvailableLanguages =
    p
        [ class (Styles.currentSelection showAvailableLanguages)
        , onClick ShowAvailableLanguages
        ]
        [ span [] [ text "English" ]
        , span [ class Styles.caret ] [ text "▾" ]
        ]
```

```elm
view : Model -> Html msg
view { showAvailableLanguages } =
    div [ class Styles.dropdownContainer ]
        [ currentSelection showAvailableLanguages
        , dropdownList showAvailableLanguages
        ]


dropdownList : Bool -> Html msg
dropdownList showAvailableLanguages =
    let
        selectableLanguages =
            [ "Italiano", "日本語" ]

    in
        ul [ class (Styles.dropdownList showAvailableLanguages) ]
            (List.map dropdownListItem selectableLanguages)
```

```elm
dropdownList : Bool -> String
dropdownList showAvailableLanguages =
    let
        displayClasses =
            if showAvailableLanguages then
                [ "flex", "flex-column" ]
            else
                [ "dn" ]
    in
        [ "absolute"
        , "b--white"
        , -- ...
        ]
            ++ displayClasses
            |> String.join " "
```

```elm
dropdownList : Bool -> String
dropdownList showAvailableLanguages =
    let
        displayClasses =
            if showAvailableLanguages then
                [ "flex", "flex-column" ]
            else
                [ "dn" ]
    in
        [ "absolute"
        , "b--white"
        , -- ...
        ]
            ++ displayClasses
            |> String.join " "
```

```elm
dropdownList : Bool -> String
dropdownList showAvailableLanguages =
    let
        displayClasses =
            if showAvailableLanguages then
                [ "flex", "flex-column" ]
            else
                [ "dn" ]
    in
        [ "absolute"
        , "b--white"
        , -- ...
        ]
            ++ displayClasses
            |> String.join " "
```

```elm
dropdownList : Bool -> String
dropdownList showAvailableLanguages =
    let
        displayClasses =
            if showAvailableLanguages then
                [ "flex", "flex-column" ]
            else
                [ "dn" ]
    in
        [ "absolute"
        , "b--white"
        , -- ...
        ]
            ++ displayClasses
            |> String.join " "
```

Subscribe to MOUSE CLICKS

```
elm-package install -y elm-lang/mouse
```

```
import Mouse


subscriptions : Model -> Sub Msg
subscriptions model =
    if model.showAvailableLanguages then
        Mouse.clicks (\_ -> CloseAvailableLanguages)
    else
        Sub.none
```

English ▾

Vertically centering things in css is easy!

Language SWITCHING

elm-i18next

```
elm-package install -y ChristophP/elm-i18next
elm-package install -y elm-lang/http
```

Translations

- public/
  - locale/
    - translations.en.json
    - translations.it.json
    - translations.ja.json

```
{
  "verticallyCenteringInCssIsEasy": "Vertically centering things in css is easy!"
}

{
  "verticallyCenteringInCssIsEasy": "Centrare verticalmente con css è facile!"
}

{
  "verticallyCenteringInCssIsEasy": "CSSで垂直センタリングは簡単だよ！"
}
```

```elm
module Translations exposing (Lang(..), getLnFromCode)


type Lang
    = En
    | It
    | Ja


getLnFromCode : String -> Lang
getLnFromCode code =
    case code of
        "en" ->
            En

        "it" ->
            It

        "ja" ->
            Ja

        _ ->
            En
```

```
type Lang
    = En
    | It
    | Ja
```

```
getLnFromCode : String -> Lang
getLnFromCode code =
    case code of
        "en" ->
            En

        "it" ->
            It

        "ja" ->
            Ja

        _ ->
            En
```

```
import Http exposing (Error)
import I18Next exposing (Translations)
import Translations exposing (Lang)


type Msg
    = ChangeLanguage Lang
    | CloseAvailableLanguages
    | FetchTranslations (Result Error Translations)
    | ShowAvailableLanguages
```

```elm
import Http exposing (Error)
import I18Next exposing (Translations)
import Translations exposing (Lang)


type Msg
    = ChangeLanguage Lang
    | CloseAvailableLanguages
    | FetchTranslations (Result Error Translations)
    | ShowAvailableLanguages
```

```elm
import Http exposing (Error)
import I18Next exposing (Translations)
import Translations exposing (Lang)



type Msg
    = ChangeLanguage Lang
    | CloseAvailableLanguages
    | FetchTranslations (Result Error Translations)
    | ShowAvailableLanguages
```

```elm
fetchTranslations : Lang -> Cmd Msg
fetchTranslations language =
    language
        |> toTranslationsUrl
        |> I18Next.fetchTranslations FetchTranslations


toTranslationsUrl : Lang -> String
toTranslationsUrl language =
    let
        translationLanguage =
            language
                |> toString
                |> String.toLower
    in
        "/locale/translations." ++ translationLanguage ++ ".json"
```

```
fetchTranslations : Lang -> Cmd Msg
fetchTranslations language =
    language
        |> toTranslationsUrl
        |> I18Next.fetchTranslations FetchTranslations


toTranslationsUrl : Lang -> String
toTranslationsUrl language =
    let
        translationLanguage =
            language
                |> toString
                |> String.toLower
    in
        "/locale/translations." ++ translationLanguage ++ ".json"
```

```elm
fetchTranslations : Lang -> Cmd Msg
fetchTranslations language =
    language
        |> toTranslationsUrl
        |> I18Next.fetchTranslations FetchTranslations


toTranslationsUrl : Lang -> String
toTranslationsUrl language =
    let
        translationLanguage =
            language
                |> toString
                |> String.toLower
    in
        "/locale/translations." ++ translationLanguage ++ ".json"
```

```elm
fetchTranslations : Lang -> Cmd Msg
fetchTranslations language =
    language
        |> toTranslationsUrl
        |> I18Next.fetchTranslations FetchTranslations


toTranslationsUrl : Lang -> String
toTranslationsUrl language =
    let
        translationLanguage =
            language
                |> toString
                |> String.toLower
    in
        "/locale/translations." ++ translationLanguage ++ ".json"
```

```elm
import I18Next exposing (Translations)


type alias Model =
    { currentLanguage : Lang
    , showAvailableLanguages : Bool
    , translations : Translations
    }


init : ( Model, Cmd Msg )
init =
    ( { currentLanguage = En
      , showAvailableLanguages = False
      , translations = I18Next.initialTranslations
      }
    , fetchTranslations En
    )
```

```elm
import I18Next exposing (Translations)


type alias Model =
    { currentLanguage : Lang
    , showAvailableLanguages : Bool
    , translations : Translations
    }


init : ( Model, Cmd Msg )
init =
    ( { currentLanguage = En
      , showAvailableLanguages = False
      , translations = I18Next.initialTranslations
      }
    , fetchTranslations En
    )
```

```elm
import I18Next exposing (Translations)


type alias Model =
    { currentLanguage : Lang
    , showAvailableLanguages : Bool
    , translations : Translations
    }


init : ( Model, Cmd Msg )
init =
    ( { currentLanguage = En
      , showAvailableLanguages = False
      , translations = I18Next.initialTranslations
      }
    , fetchTranslations En
    )
```

```elm
import I18Next exposing (Translations)


type alias Model =
    { currentLanguage : Lang
    , showAvailableLanguages : Bool
    , translations : Translations
    }


init : ( Model, Cmd Msg )
init =
    ( { currentLanguage = En
      , showAvailableLanguages = False
      , translations = I18Next.initialTranslations
      }
    , fetchTranslations En
    )
```

```elm
update : Msg -> Model -> ( Model, Cmd Msg )
update msg model =
    case msg of
        -- ...
        ChangeLanguage language ->
            ( { model | currentLanguage = language }
            , fetchTranslations language
            )

        FetchTranslations (Ok translations) ->
            ( { model | translations = translations }, Cmd.none )

        FetchTranslations (Err msg) ->
            ( model, Cmd.none )
```

```elm
update : Msg -> Model -> ( Model, Cmd Msg )
update msg model =
    case msg of
        -- ...
        ChangeLanguage language ->
            ( { model | currentLanguage = language }
            , fetchTranslations language
            )

        FetchTranslations (Ok translations) ->
            ( { model | translations = translations }, Cmd.none )

        FetchTranslations (Err msg) ->
            ( model, Cmd.none )
```

```elm
update : Msg -> Model -> ( Model, Cmd Msg )
update msg model =
    case msg of
        -- ...
        ChangeLanguage language ->
            ( { model | currentLanguage = language }
            , fetchTranslations language
            )

        FetchTranslations (Ok translations) ->
            ( { model | translations = translations }, Cmd.none )

        FetchTranslations (Err msg) ->
            ( model, Cmd.none )
```

DYNAMIC Values

```elm
module Language exposing (availableLanguages, langToString)

import Translations exposing (Lang(En, It, Ja))


availableLanguages : List Lang
availableLanguages =
    [ En, It, Ja ]


langToString : Lang -> String
langToString language =
    case language of
        En ->
            "English"

        It ->
            "Italiano"

        Ja ->
            "日本語"
```

```elm
module Language exposing (availableLanguages, langToString)

import Translations exposing (Lang(En, It, Ja))


availableLanguages : List Lang
availableLanguages =
    [ En, It, Ja ]


langToString : Lang -> String
langToString language =
    case language of
        En ->
            "English"

        It ->
            "Italiano"

        Ja ->
            "日本語"
```

```elm
module Language exposing (availableLanguages, langToString)

import Translations exposing (Lang(En, It, Ja))


availableLanguages : List Lang
availableLanguages =
    [ En, It, Ja ]


langToString : Lang -> String
langToString language =
    case language of
        En ->
            "English"

        It ->
            "Italiano"

        Ja ->
            "日本語"
```

```elm
module Language exposing (availableLanguages, langToString)

import Translations exposing (Lang(En, It, Ja))


availableLanguages : List Lang
availableLanguages =
    [ En, It, Ja ]


langToString : Lang -> String
langToString language =
    case language of
        En ->
            "English"

        It ->
            "Italiano"

        Ja ->
            "日本語"
```

```elm
module LanguageDropdown exposing (view)

view : Model -> Html Msg
view { currentLanguage, showAvailableLanguages } =
    let
        selectableLanguages =
            Language.availableLanguages
                |> List.filter (\language -> language /= currentLanguage)
    in
        div [ class Styles.dropdownContainer ]
            [ currentSelection currentLanguage showAvailableLanguages
            , dropdownList showAvailableLanguages selectableLanguages
            ]
```

```elm
module LanguageDropdown exposing (view)

view : Model -> Html Msg
view { currentLanguage, showAvailableLanguages } =
    let
        selectableLanguages =
            Language.availableLanguages
                |> List.filter (\language -> language /= currentLanguage)
    in
        div [ class Styles.dropdownContainer ]
            [ currentSelection currentLanguage showAvailableLanguages
            , dropdownList showAvailableLanguages selectableLanguages
            ]
```

```elm
module LanguageDropdown exposing (view)

view : Model -> Html Msg
view { currentLanguage, showAvailableLanguages } =
    let
        selectableLanguages =
            Language.availableLanguages
                |> List.filter (\language -> language /= currentLanguage)
    in
        div [ class Styles.dropdownContainer ]
            [ currentSelection currentLanguage showAvailableLanguages
            , dropdownList showAvailableLanguages selectableLanguages
            ]
```

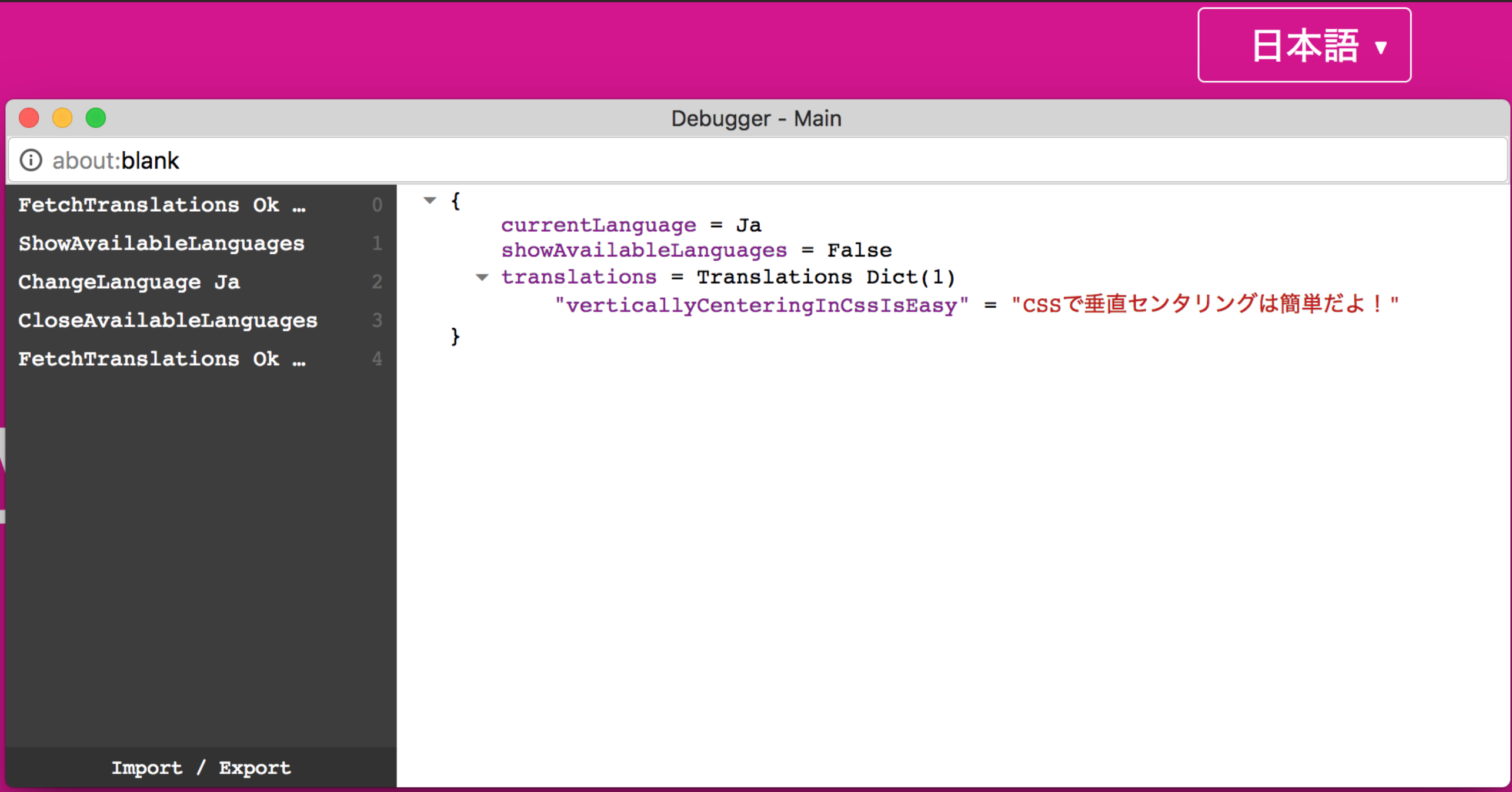
```
currentSelection : Lang -> Bool -> Html Msg
currentSelection currentLanguage showAvailableLanguages =
    p
        [ class (Styles.currentSelection showAvailableLanguages)
        , onClick ShowAvailableLanguages
        ]
        [ span []
            [ text (Language.langToString currentLanguage) ]
        , span [ class Styles.caret ]
            [ text "▾" ]
        ]
```

```
currentSelection : Lang -> Bool -> Html Msg
currentSelection currentLanguage showAvailableLanguages =
    p
        [ class (Styles.currentSelection showAvailableLanguages)
        , onClick ShowAvailableLanguages
        ]
        [ span []
            [ text (Language.langToString currentLanguage) ]
        , span [ class Styles.caret ]
            [ text "▾" ]
        ]
```

```
dropdownList : Bool -> List Lang -> Html Msg
dropdownList showAvailableLanguages selectableLanguages =
    ul [ class (Styles.dropdownList showAvailableLanguages) ]
        (List.map dropdownListItem selectableLanguages)


dropdownListItem : Lang -> Html Msg
dropdownListItem language =
    li
        [ class Styles.dropdownListItem
        , onClick (ChangeLanguage language)
        ]
        [ span []
            [ text (Language.langToString language) ]
        ]
```

```
dropdownList : Bool -> List Lang -> Html Msg
dropdownList showAvailableLanguages selectableLanguages =
    ul [ class (Styles.dropdownList showAvailableLanguages) ]
        (List.map dropdownListItem selectableLanguages)


dropdownListItem : Lang -> Html Msg
dropdownListItem language =
    li
        [ class Styles.dropdownListItem
        , onClick (ChangeLanguage language)
        ]
        [ span []
            [ text (Language.langToString language) ]
        ]
```

```elm
dropdownList : Bool -> List Lang -> Html Msg
dropdownList showAvailableLanguages selectableLanguages =
    ul [ class (Styles.dropdownList showAvailableLanguages) ]
        (List.map dropdownListItem selectableLanguages)


dropdownListItem : Lang -> Html Msg
dropdownListItem language =
    li
        [ class Styles.dropdownListItem
        , onClick (ChangeLanguage language)
        ]
        [ span []
            [ text (Language.langToString language) ]
        ]
```

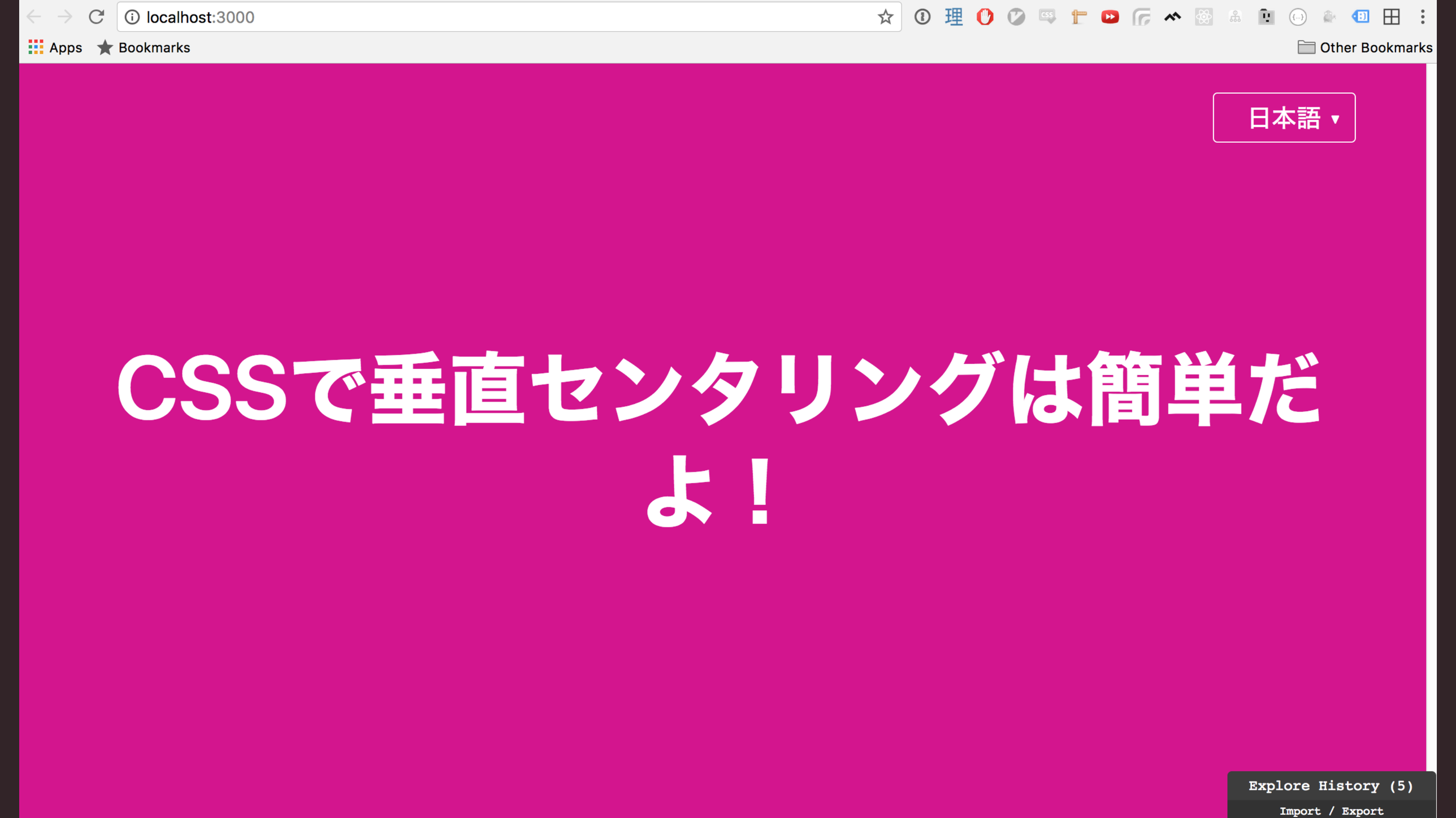

日本語
Debugger - Main
about:blank
FetchTranslations Ok … 0
ShowAvailableLanguages 1
ChangeLanguage Ja 2
CloseAvailableLanguages 3
FetchTranslations Ok … 4
{
currentLanguage = Ja
showAvailableLanguages = False
translations = Translations Dict(1)
"verticallyCenteringInCssIsEasy" = "CSSで垂直センタリングは簡単だよ！"
}
Import / Export

```elm
view : Model -> Html Msg
view model =
    main_ [ class Styles.main_ ]
        [ LanguageDropdown.view
        , content model.translations
        ]


content : Translations -> Html Msg
content translations =
    article [ class Styles.article ]
        [ div [ class Styles.articleContainer ]
            [ heading translations ]
        ]

heading : Translations -> Html Msg
heading translations =
    h1 [ class Styles.heading ]
        [ text (I18Next.t translations "verticallyCenteringInCssIsEasy")]
```

```elm
view : Model -> Html Msg
view model =
    main_ [ class Styles.main_ ]
        [ LanguageDropdown.view
        , content model.translations
        ]


content : Translations -> Html Msg
content translations =
    article [ class Styles.article ]
        [ div [ class Styles.articleContainer ]
            [ heading translations ]
        ]

heading : Translations -> Html Msg
heading translations =
    h1 [ class Styles.heading ]
        [ text (I18Next.t translations "verticallyCenteringInCssIsEasy")]
```

```elm
view : Model -> Html Msg
view model =
    main_ [ class Styles.main_ ]
        [ LanguageDropdown.view
        , content model.translations
        ]


content : Translations -> Html Msg
content translations =
    article [ class Styles.article ]
        [ div [ class Styles.articleContainer ]
            [ heading translations ]
        ]

heading : Translations -> Html Msg
heading translations =
    h1 [ class Styles.heading ]
        [ text (I18Next.t translations "verticallyCenteringInCssIsEasy")]
```

```elm
view : Model -> Html Msg
view model =
    main_ [ class Styles.main_ ]
        [ LanguageDropdown.view
        , content model.translations
        ]


content : Translations -> Html Msg
content translations =
    article [ class Styles.article ]
        [ div [ class Styles.articleContainer ]
            [ heading translations ]
        ]

heading : Translations -> Html Msg
heading translations =
    h1 [ class Styles.heading ]
        [ text (I18Next.t translations "verticallyCenteringInCssIsEasy")]
```


localhost:3000
Apps
Bookmarks
Other Bookmarks
日本語
CSSで垂直センタリングは簡単だよ！
Explore History (5)
Import / Export

DETECT

User Language

Detect User Language

- `navigator.language`
- `navigator.userLanguage` (IE)

```
import "tachyons"
import { Main } from "./Main.elm"

const appContainer = document.getElementById("root")

if (appContainer) {
  Main.embed(appContainer, { language: getLanguage() })
}

function getLanguage() {
  return navigator.language || navigator.userLanguage
}
```

```
import "tachyons"
import { Main } from "./Main.elm"

const appContainer = document.getElementById("root")

if (appContainer) {
  Main.embed(appContainer, { language: getLanguage() })
}

function getLanguage() {
  return navigator.language || navigator.userLanguage
}
```

```
import "tachyons"
import { Main } from "./Main.elm"

const appContainer = document.getElementById("root")

if (appContainer) {
  Main.embed(appContainer, { language: getLanguage() })
}

function getLanguage() {
  return navigator.language || navigator.userLanguage
}
```

```
import "tachyons"
import { Main } from "./Main.elm"

const appContainer = document.getElementById("root")

if (appContainer) {
  Main.embed(appContainer, { language: getLanguage() })
}

function getLanguage() {
  return navigator.language || navigator.userLanguage
}
```

```
import Json.Decode as Decode exposing (Value)


type alias Flags =
    { language : Value }
```

```elm
init : Flags -> ( Model, Cmd Msg )
init flags =
    let
        language =
            flags.language
                |> Decode.decodeValue Decode.string
                |> Language.langFromFlag
    in
        ( { currentLanguage = language
          , showAvailableLanguages = False
          , translations = I18Next.initialTranslations
          }
        , Cmd.fetchTranslations language
        )
```

```elm
init : Flags -> ( Model, Cmd Msg )
init flags =
    let
        language =
            flags.language
                |> Decode.decodeValue Decode.string
                |> Language.langFromFlag
    in
        ( { currentLanguage = language
          , showAvailableLanguages = False
          , translations = I18Next.initialTranslations
          }
        , Cmd.fetchTranslations language
        )
```

```elm
module Language exposing (..)


langFromFlag : Result String String -> Lang
langFromFlag language =
    case language of
        Ok language ->
            Translations.getLnFromCode language

        Err _ ->
            En
```

STORE

Language Preference

```
port module Main exposing (..)


port storeLanguageInLocalStorage : String -> Cmd msg


storeLanguage : Lang -> Cmd msg
storeLanguage language =
    language
        |> toString
        |> String.toLower
        |> storeLanguageInLocalStorage
```

```
port module Main exposing (..)


port storeLanguageInLocalStorage : String -> Cmd msg


storeLanguage : Lang -> Cmd msg
storeLanguage language =
    language
        |> toString
        |> String.toLower
        |> storeLanguageInLocalStorage
```

```
port module Main exposing (..)


port storeLanguageInLocalStorage : String -> Cmd msg


storeLanguage : Lang -> Cmd msg
storeLanguage language =
    language
        |> toString
        |> String.toLower
        |> storeLanguageInLocalStorage
```

```
if (appContainer) {
  const app = Main.embed(appContainer, { language: getLanguage() })

  app.ports.storeLanguageInLocalStorage.subscribe((language) => {
    localStorage.setItem("elm-i18n-example-language", language)
  })
}

function getLanguage() {
  return localStorage.getItem("elm-i18n-example-language") ||
    navigator.language ||
    navigator.userLanguage
}
```

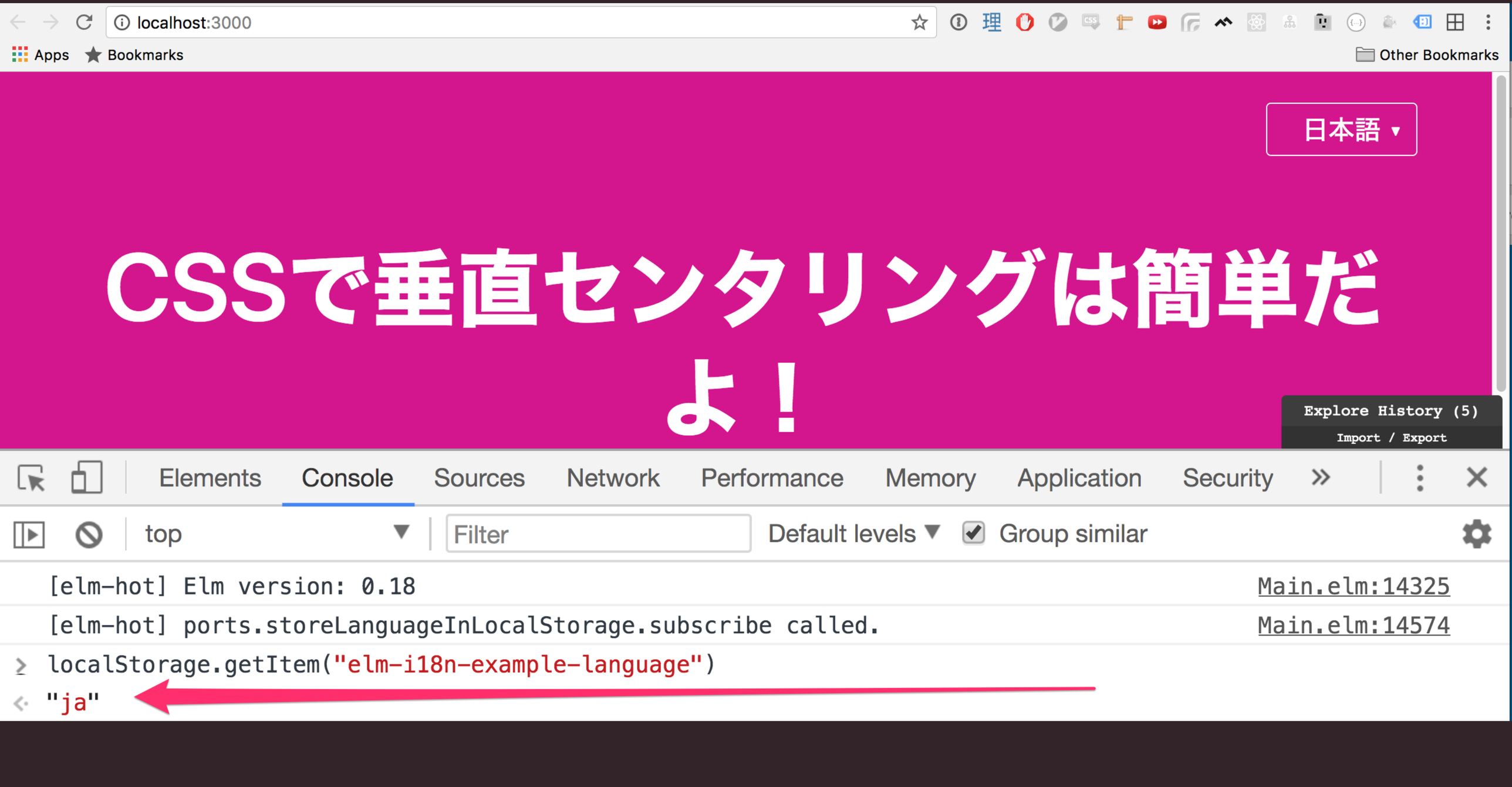
```
if (appContainer) {
  const app = Main.embed(appContainer, { language: getLanguage() })

  app.ports.storeLanguageInLocalStorage.subscribe((language) => {
    localStorage.setItem("elm-i18n-example-language", language)
  })
}

function getLanguage() {
  return localStorage.getItem("elm-i18n-example-language") ||
    navigator.language ||
    navigator.userLanguage
}
```

```
if (appContainer) {
  const app = Main.embed(appContainer, { language: getLanguage() })

  app.ports.storeLanguageInLocalStorage.subscribe((language) => {
    localStorage.setItem("elm-i18n-example-language", language)
  })
}

function getLanguage() {
  return localStorage.getItem("elm-i18n-example-language") ||
    navigator.language ||
    navigator.userLanguage
}
```

```
if (appContainer) {
  const app = Main.embed(appContainer, { language: getLanguage() })

  app.ports.storeLanguageInLocalStorage.subscribe((language) => {
    localStorage.setItem("elm-i18n-example-language", language)
  })
}

function getLanguage() {
  return localStorage.getItem("elm-i18n-example-language") ||
    navigator.language ||
    navigator.userLanguage
}
```

```elm
update : Msg -> Model -> ( Model, Cmd Msg )
update msg model =
    case msg of
        -- ...
        ChangeLanguage language ->
            ( { model | currentLanguage = language }
            , Cmd.batch
                [ fetchTranslations language
                , storeLanguage language
                ]
            )
```

```elm
update : Msg -> Model -> ( Model, Cmd Msg )
update msg model =
    case msg of
        -- ...

        ChangeLanguage language ->
            ( { model | currentLanguage = language }
            , Cmd.batch
                [ fetchTranslations language
                , storeLanguage language
                ]
            )
```


localhost:3000
Apps
Bookmarks
Other Bookmarks
日本語
CSSで垂直センタリングは簡単だよ！
Explore History (5)
Import / Export
Elements
Console
Sources
Network
Performance
Memory
Application
Security
top
Filter
Default levels
Group similar
[elm-hot] Elm version: 0.18
Main.elm:14325
[elm-hot] ports.storeLanguageInLocalStorage.subscribe called.
Main.elm:14574
localStorage.getItem("elm-i18n-example-language")
"ja"

LINGERING

ISSUES

Lingering Issues

- Translation key visible on refresh

English ▾

Vertically centering things in css is easy!

Lingering Issues

- Translation key visible on refresh
- `I18Next.t translations "thisKeyDoesNotExist"`

Lingering Issues

- Translation key visible on refresh
- `I18Next.t translations "thisKeyDoesNotExist"`
- Missed translations and typos ignored

TYPE-SAFE Translations

```
npm install -g elm-i18n-gen
```

```
elm-i18n-gen public/locale src/Translations.elm
```

```elm
module Translations exposing (..)

-- ...

verticallyCenteringInCssIsEasy : Lang -> String
verticallyCenteringInCssIsEasy lang =
    case lang of
        En ->
            "Vertically centering things in css is easy!"

        It ->
            "Centrare verticalmente con css è facile!"

        Ja ->
            "CSSで垂直センタリングは簡単だよ！"
```

```elm
module Translations exposing (..)


type Lang
    = En
    | It
    | Ja


getLnFromCode : String -> Lang
getLnFromCode code =
    -- ...


verticallyCenteringInCssIsEasy : Lang -> String
verticallyCenteringInCssIsEasy lang =
    case lang of
        En ->
            "Vertically centering things in css is easy!"

        It ->
            "Centrare verticalmente con css è facile!"

        Ja ->
            "CSSで垂直センタリングは簡単だよ！"
```

```elm
module Translations exposing (..)

-- ...

verticallyCenteringInCssIsEasy : Lang -> String
verticallyCenteringInCssIsEasy lang =
    case lang of
        En ->
            "Vertically centering things in css is easy!"

        It ->
            "Centrare verticalmente con css è facile!"

        Ja ->
            "CSSで垂直センタリングは簡単だよ！"
```

```elm
view : Model -> Html Msg
view model =
    -- ...

content : Translations -> Html Msg
content translations =
    -- ...

heading : Translations -> Html Msg
heading translations =
    h1 [ class Styles.heading ]
        [ text (I18Next.t translations "verticallyCenteringInCssIsEasy")]
```

```elm
view : Model -> Html Msg
view model =
    -- ...

content : Lang -> Html Msg
content language =
    -- ...

heading : Lang -> Html Msg
heading language =
    h1 [ class Styles.heading ]
        [ text (Translations.verticallyCenteringInCssIsEasy language) ]
```

Type-Safe Translations

- No need to fetch translations

Type-Safe Translations

- No need to fetch translations
- No translation key visible

Type-Safe Translations

- No need to fetch translations
- No translation key visible
- Compiler errors if translation not provided

CONCLUSION

Introduce run-time language switching #11

Open **niklas** wants to merge 6 commits into `iosphere:master` from `niklas:switchable-locale`

niklas commented on 25 Oct 2017

This incorporates all present translations into one build as described in #2 . Instead of a symbolic link to the compile-time language, a switch is generated for every translation module. This switch contains all the functions of the original(s), but they take a Language as an extra (preceeding) argument, delegating the translation itself to the corresponding module for that language.

Migration of apps is needed, instructions in README.md.

NOTE: This breaks the currently implemented behaviour of "one language per build".
NOTE2: This might still contain some old behavior. I will clean that up if you want that PR :-)

👍 2

Links

- https://github.com/paulfioravanti/elm-i18n-example
- https://paulfioravanti.com/blog/2018/05/11/runtime-language-switching-in-elm/

Thanks!

@paulfioravanti