ReScript Code Examples

This document contains a collection of code examples in ReScript.

chat-script--chat-box

Rescript v11

Repo: https://github.com/Exegetech/chat-rescript

========== Start file package.json (part or full code)

```
{
  "name": "frontend",
  "scripts": {
    "res:build": "rescript",
    "res:clean": "rescript clean",
    "res:dev": "rescript build -w",
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "@rescript/core": "0.5.0",
    "@rescript/react": "0.11.0",
    "shared": "workspace:*",
    "daisyui": "3.9.2",
    "react": "18.2.0",
    "react-dom": "18.2.0",
    "rescript": "11.0.0-rc.4"
  },
  "devDependencies": {
    "@vitejs/plugin-react": "4.0.0",
    "autoprefixer": "10.4.15",
    "postcss": "8.4.28",
    "tailwindcss": "3.3.3",
    "vite": "4.4.9"
  }
}
```

========== End file

========== Start file Chat__Box.res

```
module Bubble = Chat__Bubble
module Input = Chat__Input

@react.component
let make = (
  ~chats: array<Message.ToClient.t>,
  ~currentUser: string,
  ~onSubmit: (string, string) => (),
) => {
  let bottomRef = React.useRef(Nullable.null)

  React.useEffect1(() => {
    switch Nullable.toOption(bottomRef.current) {
      | Some(dom) => dom->AppDom.scrollIntoView
      | None => ()
    }

    None
  }, [chats]);

  let usersColor = Util.makeUsersColorDict(currentUser, chats)

  let handleSubmit = (message) => {
    onSubmit(currentUser, message)
  }

  <div>
    <div className=`
      bg-slate-100
      p-4
      h-[40rem]
      overflow-y-scroll
      rounded-t-lg
    `>
      {Array.mapWithIndex(chats, (chat, idx) => {
        <Bubble
          key={Int.toString(idx)}
          usersColor
```

```
          currentUser
          chat
        />
      })->React.array}

      <div ref={ReactDOM.Ref.domRef(bottomRef)} />
    </div>

    <Input
      onSubmit={handleSubmit}
    />
  </div>
}
```

========== End file

========== Start file Chat__Box.resi

```
@react.component
let make: (
  ~chats: array<Message.ToClient.t>,
  ~currentUser: string,
  ~onSubmit: (string, string) => (),
) => Jsx.element
```

========== End file

brightid--bot

Rescript v10

Repo: https://github.com/ShenaniganDApp/brightid-discord-bot

========== Start file package.json (part or full code)

```
{
  "name": "root",
  "private": true,
  "devDependencies": {
    "patch-package": "^6.4.7"
  },
  "dependencies": {
    "@rescript/core": "^0.2.0",
    "brightid_sdk": "^1.0.1",
    "canvas": "^2.9.0",
    "concurrently": "^7.1.0",
    "dotenv": "^8.2.0",
    "find-up": "^6.3.0",
    "rescript": "^10.1.0-rc.5",
    "rescript-discordjs": "^0.3.0",
    "rescript-nodejs": "^14.3.1",
    "uuid": "^8.3.0"
  },
  "scripts": {
    "bot": "yarn workspace @brightidbot/bot",
    "web": "yarn workspace @brightidbot/web",
    "utils": "yarn workspace @brightidbot/utils",
    "scripts": "yarn workspace @brightidbot/scripts",
    "shared": "yarn workspace @brightidbot/shared",
    "re:build": "yarn shared re:build && yarn utils re:build && yarn bot re:build && yarn web re:build && yarn s
cripts re:build"
  },
  "workspaces": [
    "apps/*",
    "packages/*"
  ]
}
```

========== End file

========== Start file Bot.res

```
open Discord
open NodeFetch
open Shared

let {brightIdVerificationEndpoint} = module(Endpoints)
let {context} = module(Constants)

module type Command = {
  let data: SlashCommandBuilder.t
  let execute: Interaction.t => promise<unit>
}
```

```
module type Button = {
  let customId: string
  let execute: Interaction.t => promise<unit>
}

@val @scope("globalThis")
external fetch: (string, 'params) => promise<Response.t<JSON.t>> = "fetch"

Env.createEnv()

let envConfig = Env.getConfig()

@raises([Env.Error])
let envConfig = switch envConfig {
| Ok(envConfig) => envConfig
| Error(err) => err->Env.EnvError->raise
}

@raises([Env.Error])
let gistConfig = () => {
  let id = envConfig["gistId"]
  let name = "guildData.json"
  let token = envConfig["githubAccessToken"]
  Utils.Gist.makeGistConfig(~id, ~name, ~token)
}

let options: Client.clientOptions = {
  intents: ["GUILDS", "GUILD_MESSAGES", "GUILD_MEMBERS"],
  partials: ["GUILD_MEMBER"],
}

let client = Client.createDiscordClient(~options)

let commands: Collection.t<string, module(Command)> = Collection.make()
let buttons: Collection.t<string, module(Button)> = Collection.make()

// One by one is the only way I can find to do this atm. Hopefully we find a better way
let _ =
  commands
  ->Collection.set(Commands_Help.data->SlashCommandBuilder.getCommandName, module(Commands_Help))
  ->Collection.set(
    Commands_Verify.data->SlashCommandBuilder.getCommandName,
    module(Commands_Verify),
  )
  ->Collection.set(
    Commands_Invite.data->SlashCommandBuilder.getCommandName,
    module(Commands_Invite),
  )

let _ =
  buttons
  ->Collection.set(Buttons_Verify.customId, module(Buttons_Verify))
  ->Collection.set(Buttons_Sponsor.customId, module(Buttons_Sponsor))
  ->Collection.set(Buttons_PremiumSponsor.customId, module(Buttons_PremiumSponsor))

type missingFields = RoleID
let missingFields = guild => ()

let validateConfig = async (config, decoder) => {
  open Utils.Gist
  try {
    let brightIdGuilds = await ReadGist.content(~config, ~decoder)
    brightIdGuilds->Dict.get(config.id)->Option.map(missingFields)
  } catch {
  | _ => None
  }
}

let updateGistOnGuildCreate = async (guild, roleId, content) => {
  open Utils

  let guildId = guild->Guild.getGuildId

  let entry = {
    open Shared.BrightId.Gist
    {
      name: guild->Guild.getGuildName->Some,
      role: Some("Verified"),
      roleId: Some(roleId),
      inviteLink: None,
      sponsorshipAddress: None,
      sponsorshipAddressEth: None,
      usedSponsorships: None,
      assignedSponsorships: None,
      premiumSponsorshipsUsed: None,
      premiumExpirationTimestamp: None,
    }
```

```
  }

  await Gist.UpdateGist.addEntry(~content, ~config=gistConfig(), ~key=guildId, ~entry)
}

let rec fetchContextIds = async (~retry=5, ()) => {
  open Decode
  let requestTimeout = 60000
  let endpoint = `${brightIdVerificationEndpoint}/${context}`
  let params = {
    "method": "GET",
    "headers": {
      "Accept": "application/json",
      "Content-Type": "application/json",
    },
    "timestamp": requestTimeout,
  }
  let res = await fetch(endpoint, params)
  let json = await Response.json(res)
  switch (
    json->Json.decode(Decode_BrightId.Verifications.data),
    json->Json.decode(Decode_BrightId.Error.data),
  ) {
  | (Ok({data}), _) => Set.fromArray(data.contextIds)
  | (_, Ok(error)) =>
    let retry = retry - 1
    switch retry {
    | 0 => error->Exceptions.BrightIdError->raise
    | _ => await fetchContextIds(~retry, ())
    }
  | (Error(error), _) =>
    let retry = retry - 1
    switch retry {
    | 0 => error->Json.Decode.DecodeError->raise
    | _ => await fetchContextIds(~retry, ())
    }
  }
}

let assignRoleOnCreate = async (guild, role) => {
  let maybeMembers = switch await guild->Guild.getGuildMemberManager->GuildMemberManager.fetchAll {
  | exception _ => None
  | members => Some(members)
  }
  let contextIds = await fetchContextIds()

  let filterVerifiedMembers = (guildMember, contextIds) =>
    guildMember
    ->GuildMember.getGuildMemberId
    ->UUID.v5(envConfig["uuidNamespace"])
    ->Set.has(contextIds, _)

  let assignRoleToGuildMember = (guildMember, role) => {
    guildMember->GuildMember.getGuildMemberRoleManager->GuildMemberRoleManager.add(role, ())
  }

  let makeAddRolePromises = members => {
    Collection.filter(members, filterVerifiedMembers(_, contextIds))
    ->Collection.mapValues(assignRoleToGuildMember(_, role))
    ->Collection.values
  }

  let addRolePromises = Option.map(maybeMembers, makeAddRolePromises)

  switch addRolePromises {
  | None => 0
  | Some(promises) =>
    switch await Promise.all(promises) {
    | exception e => raise(e)
    | results => Array.length(results)
    }
  }
}

let onGuildCreate = async guild => {
  open Utils
  open Shared.Decode
  let roleManager = guild->Guild.getGuildRoleManager
  let guildId = guild->Guild.getGuildId
  let guildName = guild->Guild.getGuildName

  let id = envConfig["gistId"]
  let name = "guildData.json"
  let token = envConfig["githubAccessToken"]
  let config = Gist.makeGistConfig(~id, ~name, ~token)

  let role = await RoleManager.create(
```

```
    roleManager,
    {
      name: "Verified",
      color: "ORANGE",
      reason: "Create a role to mark verified users with BrightID",
    },
  )
  switch role {
  | exception e => Console.error2(`${guildName} : ${guildId}: `, e)
  | role =>
    let content = await Gist.ReadGist.content(~config, ~decoder=Decode_Gist.brightIdGuilds)

    switch await updateGistOnGuildCreate(guild, role->Role.getRoleId, content) {
    | exception e => Console.error2(`${guildName} : ${guildId}: `, e)
    | _ =>
      Console.log(`${guildName} : ${guildId}: Successfully added to the database`)

      switch await assignRoleOnCreate(guild, role) {
      | exception e => Console.error2(`${guildName} : ${guildId}: `, e)
      | verifiedMembersCount =>
        Console.log(
          `${guildName} : ${guildId}: Successfully assigned role to ${Int.toString(
              verifiedMembersCount,
            )} current members`,
        )
      }
    }
  }
}

let onInteraction = async (interaction: Interaction.t) => {
  let guildId = interaction->Interaction.getGuild->Guild.getGuildId
  let guildName = interaction->Interaction.getGuild->Guild.getGuildName
  let isCommand = interaction->Interaction.isCommand
  let isButton = interaction->Interaction.isButton
  let user = interaction->Interaction.getUser
  switch (isCommand, isButton) {
  | (true, false) => {
      let commandName = interaction->Interaction.getCommandName
      let command = commands->Collection.get(commandName)
      switch command->Nullable.toOption {
      | None => Console.error(`${guildName} : ${guildId}: Command ${commandName} not found `)
      | Some(module(Command)) =>
        switch await Command.execute(interaction) {
        | exception e =>
          switch e {
          | Exceptions.BrightIdError({errorMessage}) =>
            Console.error2(`${guildName} : ${guildId}: `, errorMessage)
          | Exceptions.VerifyCommandError(msg) => Console.error2(`${guildName} : ${guildId}: `, msg)
          | Exceptions.InviteCommandError(msg) => Console.error2(`${guildName} : ${guildId}: `, msg)
          | JsError(obj) => Console.error2(`${guildName} : ${guildId}: `, obj)
          | _ => Console.error2(`${guildName} : ${guildId}: `, e)
          }
        | _ =>
          Console.log(
            `${guildName} : ${guildId}: Successfully served the command ${commandName} for ${user->User.getUsern
ame}`,
          )
        }
      }
    }

  | (false, true) => {
      let buttonCustomId = interaction->Interaction.getCustomId

      let button = buttons->Collection.get(buttonCustomId)
      switch button->Nullable.toOption {
      | None => Console.error(`${guildName} : ${guildId}: Button ${buttonCustomId} not found `)
      | Some(module(Button)) =>
        switch await Button.execute(interaction) {
        | exception e =>
          switch e {
          | Exceptions.BrightIdError({errorMessage}) =>
            Console.error2(`${guildName} : ${guildId}: `, errorMessage)
          | Exceptions.PremiumSponsorButtonError(msg) =>
            Console.error2(`${guildName} : ${guildId}: `, msg)
          | Exceptions.SponsorButtonError(msg) => Console.error2(`${guildName} : ${guildId}: `, msg)
          | Exceptions.ButtonVerifyHandlerError(msg) =>
            Console.error2(`${guildName} : ${guildId}: `, msg)
          | JsError(obj) => Console.error2(`${guildName} : ${guildId}: `, obj)
          | _ => Console.error2(`${guildName} : ${guildId}: `, e)
          }
        | _ =>
          Console.log(
            `${guildName} : ${guildId}: Successfully served button press "${buttonCustomId}" for ${user->User.ge
tUsername}`,
          )
```

```
        }
      }
    }

  | (_, _) => Console.error("Bot.res: Unknown interaction")
  }
}

let onGuildDelete = async guild => {
  open Utils
  open Shared.Decode

  let guildId = Guild.getGuildId(guild)
  let guildName = Guild.getGuildName(guild)

  switch await Gist.ReadGist.content(~config=gistConfig(), ~decoder=Decode_Gist.brightIdGuilds) {
  | exception JsError(e) => Console.error2(`${guildName} : ${guildId}: `, e)
  | guilds =>
    switch guilds->Dict.get(guildId) {
    | Some(_) =>
      switch await Gist.UpdateGist.removeEntry(
        ~content=guilds,
        ~key=guildId,
        ~config=gistConfig(),
      ) {
      | _ => Console.log(`${guildName} : ${guildId}: Successfully removed guild data`)
      | exception JsError(e) => Console.error2(`${guildName} : ${guildId}: `, e)
      }

    | None => Console.error(`${guildName} : ${guildId}: Could not find guild data to delete`)
    }
  }
}

let onGuildMemberAdd = async guildMember => {
  open Utils
  open Services_VerificationInfo

  let guildName = guildMember->GuildMember.getGuild->Guild.getGuildName
  let guildId = guildMember->GuildMember.getGuild->Guild.getGuildId
  let _ = switch await getBrightIdVerification(guildMember) {
  | VerificationInfo({unique}) =>
    switch unique {
    | true =>
      switch await Gist.ReadGist.content(
        ~config=gistConfig(),
        ~decoder=Decode.Decode_Gist.brightIdGuilds,
      ) {
      | exception e => Console.error2(`${guildName} : ${guildId}: `, e)
      | guilds =>
        let guild = guildMember->GuildMember.getGuild
        let guildId = guild->Guild.getGuildId
        let brightIdGuild = guilds->Dict.get(guildId)
        switch brightIdGuild {
        | None => Console.error2(`${guildName} : ${guildId}: `, `Guild does not exist in Gist`)
        | Some({roleId: None}) =>
          Console.error2(`${guildName} : ${guildId}: `, `Guild does not have a saved roleId`)
        | Some({roleId: Some(roleId)}) =>
          let role =
            guild
            ->Guild.getGuildRoleManager
            ->RoleManager.getCache
            ->Collection.get(roleId)
            ->Nullable.toOption
          switch role {
          | None => Console.error2(`${guildName} : ${guildId}: `, `Role does not exist`)
          | Some(role) =>
            let guildMemberRoleManager = guildMember->GuildMember.getGuildMemberRoleManager
            let _ = switch await GuildMemberRoleManager.add(
              guildMemberRoleManager,
              role,
              ~reason="User is already verified by BrightID",
              (),
            ) {
            | exception e => Console.error2(`${guildName} : ${guildId}: `, e)
            | _ =>
              let uuid =
                guildMember->GuildMember.getGuildMemberId->UUID.v5(envConfig["uuidNamespace"])
              Console.log(`${guildName} : ${guildId} verified the user with contextId: ${uuid}`)
            }
          }
        }
      }
    | false =>
      Console.error2(
        `${guildName} : ${guildId}: `,
        `User ${guildMember->GuildMember.getDisplayName} is not unique`,
```

```
      )
    }
  | exception e =>
    switch e {
    | Exceptions.BrightIdError({errorMessage}) =>
      Console.error2(`${guildName} : ${guildId}: `, errorMessage)
    | JsError(obj) => Console.error2(`${guildName} : ${guildId}: `, obj)
    | _ => Console.error2(`${guildName} : ${guildId}: `, e)
    }
  }
}

let onRoleUpdate = async role => {
  open Utils
  let guildId = role->Role.getGuild->Guild.getGuildId
  let guildName = role->Role.getGuild->Guild.getGuildName
  try {
    let brightIdGuilds = await Gist.ReadGist.content(
      ~config=gistConfig(),
      ~decoder=Decode.Decode_Gist.brightIdGuilds,
    )
    switch brightIdGuilds->Dict.get(guildId) {
    | None => Console.error2(`${guildName} : ${guildId}: `, `Guild does not exist in Gist`)
    | Some(brightIdGuild) =>
      switch brightIdGuild.roleId {
      | None => Console.error2(`${guildName} : ${guildId}: `, `Guild does not have a saved roleId`)
      | Some(roleId) =>
        let isVerifiedRole = role->Role.getRoleId === roleId
        switch isVerifiedRole {
        | true =>
          let roleName = role->Role.getName
          let entry = {
            ...brightIdGuild,
            role: Some(roleName),
          }
          let _ = await Gist.UpdateGist.updateEntry(
            ~content=brightIdGuilds,
            ~entry,
            ~key=guildId,
            ~config=gistConfig(),
          )
          Console.log(`${guildName} : ${guildId} updated the role name to ${roleName}`)
        | false => ()
        }
      }
    }
  } catch {
  | e => Console.error2(`${guildName} : ${guildId}: `, e)
  }
}

let onGuildMemberUpdate = async (_, newMember) => {
  open Utils
  open Services_VerificationInfo
  let guild = newMember->GuildMember.getGuild
  let guildName = guild->Guild.getGuildName
  let guildId = guild->Guild.getGuildId
  try {
    let brightIdGuilds = await Gist.ReadGist.content(
      ~config=gistConfig(),
      ~decoder=Decode.Decode_Gist.brightIdGuilds,
    )
    switch brightIdGuilds->Dict.get(guildId) {
    | None => ()
    | Some({roleId: None}) => ()
    | Some({roleId: Some(roleId)}) =>
      let member =
        await guild
        ->Guild.getGuildMemberManager
        ->GuildMemberManager.fetchOne(newMember->GuildMember.getGuildMemberId)
      switch await getBrightIdVerification(member) {
      | VerificationInfo({unique}) =>
        let guildMemberRoleManager = member->GuildMember.getGuildMemberRoleManager
        let roles = guildMemberRoleManager->GuildMemberRoleManager.getCache
        let role =
          guild
          ->Guild.getGuildRoleManager
          ->RoleManager.getCache
          ->Collection.get(roleId)
          ->Nullable.toOption
        switch (role, roles->Collection.has(roleId), unique) {
        | (None, _, _) => ()
        | (Some(role), true, false) =>
          let _ = await GuildMemberRoleManager.removeRole(
            guildMemberRoleManager,
            role,
            ~reason="User is not verified by BrightID",
```

```
            (),
          )
        | (Some(role), false, true) =>
          let guildMemberRoleManager = member->GuildMember.getGuildMemberRoleManager
          let _ = await GuildMemberRoleManager.add(
            guildMemberRoleManager,
            role,
            ~reason="User is verified by BrightID",
            (),
          )
        | (_, _, _) => ()
        }
      | exception e =>
        switch e {
        | Exceptions.BrightIdError(_) =>
          let role =
            guild
            ->Guild.getGuildRoleManager
            ->RoleManager.getCache
            ->Collection.get(roleId)
            ->Nullable.toOption
          let guildMemberRoleManager = newMember->GuildMember.getGuildMemberRoleManager
          switch role {
          | None => ()
          | Some(role) =>
            let _ = switch await GuildMemberRoleManager.removeRole(
              guildMemberRoleManager,
              role,
              ~reason="User is not verified by BrightID",
              (),
            ) {
            | exception e =>
              switch e {
              | Exn.Error(obj) =>
                switch Exn.message(obj) {
                | Some(m) => Console.error2(`${guildName} : ${guildId}: `, m)
                | None => ()
                }
              | _ => ()
              }
            | _ => ()
            }
          }
        | Exn.Error(obj) =>
          switch Exn.message(obj) {
          | Some(m) => Console.error2(`${guildName} : ${guildId}: `, m)
          | None => ()
          }
        | _ => Console.error2(`${guildName} : ${guildId}: `, e)
        }
      }
    }
  } catch {
  | Exn.Error(obj) =>
    switch Exn.message(obj) {
    | Some(m) => Console.error2(`${guildName} : ${guildId}: `, m)
    | None => ()
    }
  | e => Console.error2(`${guildName} : ${guildId}: `, e)
  }
}

client->Client.on(
  #ready(
    () => {
      Console.log("Logged In")
    },
  ),
)

client->Client.on(#guildCreate(guild => guild->onGuildCreate->ignore))

client->Client.on(#interactionCreate(interaction => interaction->onInteraction->ignore))

client->Client.on(#guildDelete(guild => guild->onGuildDelete->ignore))

client->Client.on(#guildMemberAdd(member => member->onGuildMemberAdd->ignore))

client->Client.on(#roleUpdate((~oldRole as _, ~newRole) => newRole->onRoleUpdate->ignore))

client->Client.on(
  #guildMemberUpdate((~oldMember, ~newMember) => onGuildMemberUpdate(oldMember, newMember)->ignore),
)

client->Client.login(envConfig["discordApiToken"])->ignore
```

========== End file

fullstack--examples

Rescript v10

Repo: https://github.com/skonky/fullstack

========== Start file package.json (part or full code)

```
{
  "name": "rescript-web",
  "version": "0.0.0",
  "author": "skonky",
  "private": true,
  "license": "Apache-2.0",
  "dependencies": {
    "next": "10.2.3",
    "react": "17.0.1",
    "react-dom": "17.0.1"
  },
  "scripts": {
    "dev": "concurrently \"next dev -p 5000\" \"rescript build -w\"",
    "debug": "NODE_OPTIONS='--inspect' next",
    "build": "rescript && next build",
    "now-build": "rescript && next build",
    "export": "next export",
    "start": "next start -p $PORT",
    "res:build": "rescript",
    "res:clean": "rescript clean",
    "res:start": "rescript build -w"
  },
  "devDependencies": {
    "@rescript/react": "0.10.3",
    "autoprefixer": "10.1.0",
    "concurrently": "^7.6.0",
    "cssnano": "5.0.5",
    "daisyui": "^2.51.3",
    "gentype": "4.1",
    "next-transpile-modules": "7.1.2",
    "postcss": "8.2.15",
    "postcss-cli": "8.3.1",
    "rescript": "9.1",
    "tailwindcss": "^3.0.23"
  }
}
```

========== End file

========== Start file Examples.res

```
type todo = {
  id: int,
  title: string,
  isDone: bool,
}

type state = {newTodoValue: string, todos: array<todo>}

let initialState: state = {
  newTodoValue: "",
  todos: [
    {
      id: 1,
      title: "Initial todo",
      isDone: false,
    },
  ],
}

type actions =
  | CreateTodo
  | InputChanged(string)
  | RemoveTodo(int)
  | ClearCompleted
  | ToggleTodoStatus(int)

let reducer = (state, action) => {
  switch action {
  | CreateTodo => {
      newTodoValue: "",
      todos: Js.Array.concat(
        [
          {
            title: state.newTodoValue,
```

```
            isDone: false,
            id: state.todos->Js.Array2.length + 1,
          },
        ],
        state.todos,
      ),
    }
  | InputChanged(newTodoValue) => {
      ...state,
      newTodoValue: newTodoValue,
    }
  | RemoveTodo(id) => {
      ...state,
      todos: state.todos->Js.Array2.filter(t => t.id != id),
    }
  | ClearCompleted => {
      ...state,
      todos: state.todos->Js.Array2.filter(todo => !todo.isDone),
    }
  | ToggleTodoStatus(id) => {
      ...state,
      todos: state.todos->Js.Array2.map(todo => {
        if todo.id == id {
          {
            ...todo,
            isDone: !todo.isDone,
          }
        } else {
          todo
        }
      }),
    }
  }
}

let default = () => {
  open Js.Array2

  let (state, dispatch) = React.useReducer(reducer, initialState)
  let handleClearAllCompletedTodos = _ => dispatch(ClearCompleted)

  let noTodosCompleted = state.todos->Js.Array2.find(todo => todo.isDone)->Belt.Option.isNone

  let handleChange = event => {
    let value = ReactEvent.Form.target(event)["value"]
    dispatch(InputChanged(value))
  }

  let handleCreateTodo = _ => {
    dispatch(CreateTodo)
  }

  let handleDelete = (id, _) => {
    dispatch(RemoveTodo(id))
  }

  let handleToggleTodoStatus = (id, _) => {
    dispatch(ToggleTodoStatus(id))
  }

  <div className="p-6">
    <input
      value=state.newTodoValue
      onChange={handleChange}
      className="input input-bordered mr-5"
      placeholder="Type here"
    />
    <button onClick={handleCreateTodo} className="btn btn-primary mr-3">
      {"Create"->React.string}
    </button>
    <button
      disabled=noTodosCompleted onClick={handleClearAllCompletedTodos} className="btn btn-success">
      {"Clear completed"->React.string}
    </button>
    <div className="overflow-x-auto">
      <table className="table w-full mt-6">
        <thead>
          <tr>
            <th> {"#"->React.string} </th>
            <th> {"Name"->React.string} </th>
            <th className="text-right"> {"Controls"->React.string} </th>
          </tr>
        </thead>
        <tbody>
          {state.todos
          ->map(todo => {
            let todoClassName = todo.isDone ? "p-3 line-through opacity-50" : "p-3"
```

```
            <tr key={Belt.Int.toString(todo.id)}>
              <td> {Belt.Int.toString(todo.id)->React.string} </td>
              <td className=todoClassName> {todo.title->React.string} </td>
              <td className="text-right">
                <button onClick={handleToggleTodoStatus(todo.id)} className="btn ml-3">
                  {"toggle"->React.string}
                </button>
                <button onClick={handleDelete(todo.id)} className="btn btn-error ml-3">
                  {"delete"->React.string}
                </button>
              </td>
            </tr>
          })
          ->React.array}
        </tbody>
      </table>
    </div>
  </div>
}
```

========== End file

========== Start file Examples.resi

```
let default: unit => React.element
```

========== End file

brightid--sidebar

Rescript v10

Repo: https://github.com/ShenaniganDApp/brightid-discord-bot

========== Start file package.json (part or full code)

```
{
  "name": "root",
  "private": true,
  "devDependencies": {
    "patch-package": "^6.4.7"
  },
  "dependencies": {
    "@rescript/core": "^0.2.0",
    "brightid_sdk": "^1.0.1",
    "canvas": "^2.9.0",
    "concurrently": "^7.1.0",
    "dotenv": "^8.2.0",
    "find-up": "^6.3.0",
    "rescript": "^10.1.0-rc.5",
    "rescript-discordjs": "^0.3.0",
    "rescript-nodejs": "^14.3.1",
    "uuid": "^8.3.0"
  },
  "scripts": {
    "bot": "yarn workspace @brightidbot/bot",
    "web": "yarn workspace @brightidbot/web",
    "utils": "yarn workspace @brightidbot/utils",
    "scripts": "yarn workspace @brightidbot/scripts",
    "shared": "yarn workspace @brightidbot/shared",
    "re:build": "yarn shared re:build && yarn utils re:build && yarn bot re:build && yarn web re:build && yarn s
cripts re:build"
  },
  "workspaces": [
    "apps/*",
    "packages/*"
  ]
}
```

========== End file

========== Start file Sidebar.res

```
module ConnectButton = {
  @react.component @module("@rainbow-me/rainbowkit")
  external make: (
    ~children: React.element=?,
    ~style: ReactDOM.Style.t=?,
    ~className: string=?,
  ) => 'b = "ConnectButton"
}

@react.component
```

```
let make = (~isSidebarVisible, ~handleIsSidebarVisible, ~guilds, ~loadingGuilds) => {
  open ReactProSidebar

  let icon = ({id, icon}: Types.oauthGuild) => {
    switch icon {
    | None => "/assets/brightid_logo_white.png"
    | Some(icon) => `https://cdn.discordapp.com/icons/${id}/${icon}.png`
    }
  }

  let sidebarElements = {
    switch (guilds, loadingGuilds) {
    | (_, true) =>
      let intersection = guilds->Belt.Array.map((guild: Types.oauthGuild) => {
        <Menu iconShape="square" key={guild.id}>
          <MenuItem
            className="bg-extraDark"
            icon={<img
              className=" bg-extraDark rounded-lg border-1 border-white" src={guild->icon}
            />}>
            <Remix.Link
              className="font-semibold text-xl" to={`/guilds/${guild.id}`} prefetch={#intent}>
              {guild.name->React.string}
            </Remix.Link>
          </MenuItem>
        </Menu>
      })
      let loading = Belt.Array.range(0, 4)->Belt.Array.map(i => {
        <Menu iconShape="square" key={(i + 1)->Belt.Int.toString}>
          <MenuItem
            className="flex animate-pulse flex-row h-full bg-extraDark "
            icon={<img
              className=" bg-extraDark  rounded-lg" src="/assets/brightid_logo_white.png"
            />}>
            <div className="flex flex-col space-y-3">
              <div className="w-36 bg-gray-300 h-6 rounded-md " />
            </div>
          </MenuItem>
        </Menu>
      })
      intersection->Belt.Array.concat(loading)->React.array
    | ([], false) => <p className="text-white"> {"Couldn't Load Discord Servers"->React.string} </p>
    | (_, false) =>
      switch guilds->Belt.Array.length {
      | 0 => <p className="text-white"> {"No Guilds"->React.string} </p>
      | _ =>
        guilds
        ->Belt.Array.map((guild: Types.oauthGuild) => {
          <Menu iconShape="square" key={guild.id}>
            <MenuItem
              className="bg-extraDark"
              icon={<img
                className=" bg-extraDark rounded-lg border-1 border-white" src={guild->icon}
              />}>
              <Remix.Link
                className="font-semibold text-xl" to={`/guilds/${guild.id}`} prefetch={#intent}>
                {guild.name->React.string}
              </Remix.Link>
            </MenuItem>
          </Menu>
        })
        ->React.array
      }
    }
  }

  <ProSidebar
    className="bg-dark scrollbar-hide"
    breakPoint="md"
    onToggle={handleIsSidebarVisible}
    toggled={isSidebarVisible}>
    <SidebarHeader
      className="p-2 flex justify-around items-center top-0 sticky bg-dark z-10 scrollbar-hide">
      <InviteButton />
    </SidebarHeader>
    <SidebarContent className="scrollbar-hide">
      <Menu iconShape="square" key={0->Belt.Int.toString} />
      {sidebarElements}
    </SidebarContent>
    <SidebarFooter className="bg-extraDark bottom-0 sticky scrollbar-hide list-none">
      <Remix.Link to={""}>
        <MenuItem>
          <img src={"/assets/brightid_reversed.svg"} />
        </MenuItem>
      </Remix.Link>
    </SidebarFooter>
  </ProSidebar>
```

```
}
```

========== End file

chat-script--message-json

Rescript v11

Repo: https://github.com/Exegetech/chat-rescript

========== Start file package.json (part or full code)

```
{
  "name": "shared",
  "scripts": {
    "res:build": "rescript",
    "res:clean": "rescript clean",
    "res:dev": "rescript build -w"
  },
  "dependencies": {
    "@rescript/core": "0.5.0",
    "rescript": "11.0.0-rc.4"
  }
}
```

========== End file

========== Start file Message__JSON.res

```
@unboxed
type rec json =
  | @as(false) False
  | @as(true) True
  | @as(null) Null
  | String(string)
  | Number(float)
  | Object(Dict.t<json>)
  | Array(array<json>)

@val
@scope("JSON")
external parseExn: string => json = "parse"

@val
@scope("JSON")
external stringifyExn: json => string = "stringify"

let parse = (payload: string): result<json, string> => {
  try {
    payload
    -> parseExn
    -> Ok
  } catch {
    | Exn.Error(obj) => {
      switch Exn.message(obj) {
        | Some(msg) => Error(msg)
        | None => Error("Unknown error")
      }
    }
  }
}

let stringify = (payload: json): result<string, string> => {
  try {
    payload
    -> stringifyExn
    -> Ok
  } catch {
    | Exn.Error(obj) => {
      switch Exn.message(obj) {
        | Some(msg) => Error(msg)
        | None => Error("Unknown error")
      }
    }
  }
}
```

========== End file

========== Start file Message__JSON.resi

```
@unboxed
type rec json =
```

```
  | @as(false) False
  | @as(true) True
  | @as(null) Null
  | String(string)
  | Number(float)
  | Object(Dict.t<json>)
  | Array(array<json>)

let parse: (string) => result<json, string>
let stringify: (json) => result<string, string>
```

========== End file

chat-script--chat-input

Rescript v11

Repo: https://github.com/Exegetech/chat-rescript

========== Start file package.json (part or full code)

```
{
  "name": "frontend",
  "scripts": {
    "res:build": "rescript",
    "res:clean": "rescript clean",
    "res:dev": "rescript build -w",
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "@rescript/core": "0.5.0",
    "@rescript/react": "0.11.0",
    "shared": "workspace:*",
    "daisyui": "3.9.2",
    "react": "18.2.0",
    "react-dom": "18.2.0",
    "rescript": "11.0.0-rc.4"
  },
  "devDependencies": {
    "@vitejs/plugin-react": "4.0.0",
    "autoprefixer": "10.4.15",
    "postcss": "8.4.28",
    "tailwindcss": "3.3.3",
    "vite": "4.4.9"
  }
}
```

========== End file

========== Start file Chat__Input.res

```
@react.component
let make = (~onSubmit: (string) => unit) => {
  let (text, setText) = React.useState(() => "")

  let handleInputChange = (event) => {
    let value = ReactEvent.Form.currentTarget(event)["value"]
    setText((_) => value)
  }

  let handleButtonClick = (_) => {
    switch text {
      | "" => ()
      | text => {
        onSubmit(text)
        setText((_) => "")
      }
    }
  }

  let handleKeyDown = (e) => {
    let key = ReactEvent.Keyboard.key(e)

    switch (text, key) {
      | (text, "Enter") if text !== "" => {
        ReactEvent.Keyboard.preventDefault(e)

        onSubmit(text)
        setText((_) => "")
      }
      | _ => ()
    }
```

```
  }

  <div className=`
    bg-slate-400
    p-2
    flex
    rounded-b-lg
  `>
    <input
      type_="text"
      placeholder="Type here"
      className="input input-bordered mr-1 w-full"
      value={text}
      onChange={handleInputChange}
      onKeyDown={handleKeyDown}
    />

    <button
      className="btn btn-square"
      onClick={handleButtonClick}
    >
      <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" viewBox="0 0 16 16"><path fill="currentColo
r" d="M15.854.146a.5.5 0 0 1 .11.541l-5.819 14.547a.75.75 0 0 1-1.329.124l-3.178-4.995L.643 7.184a.75.75 0 0 1 .1
24-1.33L15.314.037a.5.5 0 0 1 .54.11ZM6.636 10.07l2.761 4.338L14.13 2.576L6.636 10.07Zm6.787-8.201L1.591 6.60214
.339 2.76l7.494-7.493Z"/></svg>
    </button>
  </div>
}
```

========== End file

========== Start file Chat__Input.resi

```
@react.component
let make: (~onSubmit: (string) => unit) => Jsx.element
```

========== End file

brightid--reactprosidebar

Rescript v10

Repo: https://github.com/ShenaniganDApp/brightid-discord-bot

========== Start file package.json (part or full code)

```
{
  "name": "root",
  "private": true,
  "devDependencies": {
    "patch-package": "^6.4.7"
  },
  "dependencies": {
    "@rescript/core": "^0.2.0",
    "brightid_sdk": "^1.0.1",
    "canvas": "^2.9.0",
    "concurrently": "^7.1.0",
    "dotenv": "^8.2.0",
    "find-up": "^6.3.0",
    "rescript": "^10.1.0-rc.5",
    "rescript-discordjs": "^0.3.0",
    "rescript-nodejs": "^14.3.1",
    "uuid": "^8.3.0"
  },
  "scripts": {
    "bot": "yarn workspace @brightidbot/bot",
    "web": "yarn workspace @brightidbot/web",
    "utils": "yarn workspace @brightidbot/utils",
    "scripts": "yarn workspace @brightidbot/scripts",
    "shared": "yarn workspace @brightidbot/shared",
    "re:build": "yarn shared re:build && yarn utils re:build && yarn bot re:build && yarn web re:build && yarn s
cripts re:build"
  },
  "workspaces": [
    "apps/*",
    "packages/*"
  ]
}
```

========== End file

========== Start file ReactProSidebar.res

```
module ProSidebar = {
  @react.component @module("react-pro-sidebar")
  external make: (
    ~children: React.element,
    ~className: string=?,
    ~breakPoint: string,
    ~onToggle: bool => unit,
    ~collapsed: bool=?,
    ~toggled: bool=?,
  ) => React.element = "ProSidebar"
}

module Menu = {
  @react.component @module("react-pro-sidebar")
  external make: (~children: React.element=?, ~iconShape: string) => React.element = "Menu"
}
module MenuItem = {
  @react.component @module("react-pro-sidebar")
  external make: (
    ~children: React.element=?,
    ~className: string=?,
    ~icon: React.element=?,
  ) => React.element = "MenuItem"
}
module SidebarHeader = {
  @react.component @module("react-pro-sidebar")
  external make: (~children: React.element=?, ~className: string=?) => React.element =
    "SidebarHeader"
}
module SidebarContent = {
  @react.component @module("react-pro-sidebar")
  external make: (~children: React.element, ~className: string=?) => React.element =
    "SidebarContent"
}
module SidebarFooter = {
  @react.component @module("react-pro-sidebar")
  external make: (~children: React.element=?, ~className: string=?) => React.element =
    "SidebarFooter"
}
```

========== End file

catala-dsfr--form

Rescript v10

Repo: https://github.com/CatalaLang/catala-dsfr

========== Start file package.json (part or full code)

```
{
  "name": "catala-dsfr",
  "version": "0.1.1",
  "repository": "https://github.com/CatalaLang/catala-dsfr",
  "scripts": {
    "clean": "rescript clean -with-deps",
    "build": "yarn run pre && vite build",
    "deploy": "yarn run build --base=/demos/catala/ && rsync -rv --delete-before dist/*",
    "serve": "vite preview",
    "dev": "yarn run pre && vite",
    "re:build": "rescript build -with-deps",
    "re:watch": "yarn run pre && rescript build -w -with-deps",
    "assets": "rsync -r node_modules/@catala-lang/catala-web-assets/assets/* assets",
    "postinstall": "copy-dsfr-to-public",
    "pre": "yarn run re:build && only-include-used-icons && yarn run assets"
  },
  "keywords": [
    "rescript"
  ],
  "author": "Emile Rolley <emile.rolley@tuta.io>",
  "license": "Apache-2.0",
  "dependencies": {
    "@catala-lang/catala-explain": "^0.2.2",
    "@catala-lang/catala-web-assets": "^0.8.9",
    "@catala-lang/french-law": "^0.8.3-b.3",
    "@catala-lang/rescript-catala": "^0.8.1-b.0",
    "@codegouvfr/react-dsfr": "^0.78.2",
    "@rescript/core": "^0.5.0",
    "@rescript/react": "^0.11.0",
    "@rjsf/core": "^5.1.0",
    "@rjsf/utils": "^5.1.0",
    "@rjsf/validator-ajv8": "^5.1.0",
    "file-saver": "^2.0.5",
    "react": "^18.2.0",
```

```
    "react-dom": "^18.2.0",
    "react-loader-spinner": "^5.4.5",
    "rescript-docx": "^0.1.5",
    "tslib": "^2.6.2"
  },
  "devDependencies": {
    "@jihchi/vite-plugin-rescript": "^5.1.0",
    "@originjs/vite-plugin-commonjs": "^1.0.3",
    "@vitejs/plugin-react": "^3.1.0",
    "jsdom": "^21.1.0",
    "rescript": "^10.1.4",
    "tailwindcss": "^3.2.6",
    "vite": "^4.4.9"
  }
}
```

========== End file

========== Start file Form.res

```
/*
  Binding for the React component [JSONSchemaForm.default] of the package
  [react-jsonschema-form].

  The component is capable of building HTML forms out of a JSON schema.
*/
module RjsfFormDsfrLazy = {
  @react.component @module("./RjsfFormDsfrLazy.tsx")
  external make: (
    ~onChange: Js.Dict.t<Js.Json.t> => unit=?,
    ~onSubmit: Js.Dict.t<Js.Json.t> => unit=?,
    ~onError: _ => unit=?,
    ~schema: Js.Json.t,
    ~uiSchema: Js.Json.t=?,
    ~formData: Js.Json.t=?,
  ) => React.element = "default"
}

// Function to download or import a JSON object
@val external downloadJSONstring: string => unit = "downloadJSONstring"
%%raw(`
const downloadJSONstring = (data) => {
  const blob = new Blob([data],{type:'application/json'});
  const href = URL.createObjectURL(blob);
  const link = document.createElement('a');
  link.href = href;
  link.download = "data.json";
  document.body.appendChild(link);
  link.click();
  document.body.removeChild(link);
};
`)

// Function to read a file and get its contents as string
@val external readFileAsJSON: (Js.Json.t, Js.Json.t => 'a) => unit = "readFileAsJSON"
%%raw(`
const readFileAsJSON = (file, callback) => {
  var reader = new FileReader();
  var contents = ""
  reader.onload = function(evt) {
    contents = evt.target.result;
    var json;
    try {
      json = JSON.parse(contents)
    } catch (error) {
      console.log(error)
      json = null;
    }
    callback(json);
  };
  reader.readAsText(file);
};
`)

/*
  Builds a React component from provided information.
*/
module Make = (
  FormInfos: {
    let webAssets: WebAssets.t
    let name: string
    let resultLabel: string
    let formDataPostProcessing: option<Js.Json.t => Js.Json.t>
    let computeAndPrintResult: Js.Json.t => React.element
    let url: string
  },
```

```
) => {
  @react.component
  let make = () => {
    let currentPath = Nav.getCurrentURL().path
    let (formData, setFormData) = React.useState(_ => FormInfos.webAssets.initialData)
    let (eventsOpt, setEventsOpt) = React.useState(_ => None)

    React.useEffect2(() => {
      setEventsOpt(_ => {
        let events = {
          try {CatalaFrenchLaw.retrieveEventsSerialized()->CatalaRuntime.deserializedEvents} catch {
          | _ => []
          }
        }
        if 0 == events->Belt.Array.size {
          None
        } else {
          Some(events)
        }
      })
      None
    }, (formData, setEventsOpt))

    let (uploadedFile, setUploadedFile) = React.useState(_ => {
      Js.Json.object_(Js.Dict.empty())
    })

    let fileChangeHandler = (_event: ReactEvent.Form.t) => {
      setUploadedFile(%raw(`_event.target.files[0]`))
    }

    let retrieveFileContents = _ => {
      if %raw(`uploadedFile instanceof File`) {
        readFileAsJSON(uploadedFile, form_data => setFormData(_ => Some(form_data)))
      }
    }

    let form_footer = {
      let priority = "tertiary"
      <Dsfr.ButtonsGroup
        inlineLayoutWhen="always"
        className="text-left"
        buttonsEquisized=true
        buttonsSize="medium"
        alignment="center"
        buttons=[
          {
            children: `RÃ©initialiser le formulaire`->React.string,
            onClick: _ => {
              Console.debug("Resetting form data")
              setFormData(_ => FormInfos.webAssets.initialData)
            },
            iconId: "fr-icon-refresh-line",
            priority,
          },
          {
            children: `Exporter les donnÃ©es au format JSON`->React.string,
            onClick: _ => {
              let data_str = Js.Json.stringify(formData->Belt.Option.getWithDefault(Js.Json.null))
              downloadJSONstring(data_str)
            },
            iconId: "fr-icon-upload-line",
            priority,
          },
          {
            children: <>
              <input
                className="hidden w-100" id="file-upload" type_="file" onChange={fileChangeHandler}
              />
              <label htmlFor="file-upload" className="cursor-pointer">
                {`Importer les donnÃ©es au format JSON `->React.string}
              </label>
              <p />
            </>,
            onClick: retrieveFileContents,
            iconId: "fr-icon-download-line",
            priority,
          },
          {
            children: {"Code source du programme"->React.string},
            onClick: {_ => currentPath->List.concat(list{`sources`})->Nav.goToPath},
            iconId: "fr-icon-code-s-slash-line",
            priority,
          },
        ]
      />
    }
```

```
    let form_result =
      <Dsfr.CallOut>
        {switch formData {
        | None => `En attente de la confirmation du formulaire...`->React.string
        | Some(formData) =>
          try {
            <div className="flex flex-col">
              <div>
                {FormInfos.resultLabel->React.string}
                {": "->React.string}
                {FormInfos.computeAndPrintResult(formData)}
              </div>
              <Dsfr.Button
                onClick={_ => {
                  let doc = CatalaExplain.generate(
                    // NOTE(@EmileRolley): we assume that the events exist,
                    // because we have a result.
                    ~events=eventsOpt->Option.getExn,
                    ~userInputs=formData,
                    ~schema=FormInfos.webAssets.schema,
                    ~opts={
                      title: `Calcul des ${FormInfos.name}`,
                      // Contains an explicatory text about the computation and the catala program etc...
                      description: `Explication du dÃ©tail des Ã©tapes de calcul Ã©tablissant l'Ã©ligibilitÃ© et
 le montant des ${FormInfos.name} pour votre demande`,
                      creator: `catala-dsfr`,
                      keysToIgnore: FormInfos.webAssets.keysToIgnore,
                      selectedOutput: FormInfos.webAssets.selectedOutput,
                      sourcesURL: `${Constants.host}/${FormInfos.url}/sources`,
                    },
                  )

                  doc
                  ->Docx.Packer.toBlob
                  ->Promise.thenResolve(blob => {
                    FileSaver.saveAs(
                      blob,
                      `explication-decision-${FormInfos.name->String.replaceRegExp(
                          %re("/\s/g"),
                          "_",
                        )}.docx`,
                    )
                  })
                  ->ignore
                }}
                iconPosition="left"
                iconId="fr-icon-newspaper-line"
                priority="secondary">
                {`TÃ©lÃ©charger une explication du calcul`->React.string}
              </Dsfr.Button>
            </div>
          } catch {
          | err =>
            <>
              <Lang.String english="Computation error: " french={`Erreur de calcul : `} />
              {err
              ->Js.Exn.asJsExn
              ->Belt.Option.map(Js.Exn.message)
              ->Belt.Option.getWithDefault(Some(""))
              ->Belt.Option.getWithDefault("unknwon error, please retry the computation")
              ->React.string}
            </>
          }
        }}
      </Dsfr.CallOut>

    <>
      <div className="fr-container--fluid">
        <div className="fr-grid-row fr-grid-row--gutters fr-grid-row--center">
          <Dsfr.Notice
            title={`Les donnÃ©es collectÃ©es par ce formulaire ne sont envoyÃ©es nulle part, et sont gÃ©rÃ©es un
iquement par votre navigateur internet. \
            Les donnÃ©es sont traitÃ©es localement par un programme Javascript qui a Ã©tÃ© transmis avec le rest
e de ce site Internet. \
            Ainsi, ce site ne collecte aucune donnÃ©e de ses utilisateurs.`}
            isClosable=true
          />
          <div className="fr-col">
            <React.Suspense fallback={Spinners.loader}>
              <RjsfFormDsfrLazy
                schema={FormInfos.webAssets.schema}
                uiSchema={FormInfos.webAssets.uiSchema}
                formData={formData->Belt.Option.getWithDefault(Js.Json.null)}
                onSubmit={t => {
                  setFormData(_ => {
                    let formData = t->Js.Dict.get("formData")
```

```
                    switch (FormInfos.formDataPostProcessing, formData) {
                    | (Some(f), Some(formData)) => {
                        let newFormData = f(formData)
                        Some(newFormData)
                      }
                    | _ => formData
                    }
                  })
                }}
              />
            </React.Suspense>
          </div>
          <div
            className="w-full fr-m-1w border-2 border-solid rounded-full border-[var(--border-default-grey)]"
          />
          <div className="fr-col"> form_result </div>
        </div>
        form_footer
      </div>
    </>
  }
}
```

========== End file

brightid--discorserver

Rescript v10

Repo: https://github.com/ShenaniganDApp/brightid-discord-bot

========== Start file package.json (part or full code)

```
{
  "name": "root",
  "private": true,
  "devDependencies": {
    "patch-package": "^6.4.7"
  },
  "dependencies": {
    "@rescript/core": "^0.2.0",
    "brightid_sdk": "^1.0.1",
    "canvas": "^2.9.0",
    "concurrently": "^7.1.0",
    "dotenv": "^8.2.0",
    "find-up": "^6.3.0",
    "rescript": "^10.1.0-rc.5",
    "rescript-discordjs": "^0.3.0",
    "rescript-nodejs": "^14.3.1",
    "uuid": "^8.3.0"
  },
  "scripts": {
    "bot": "yarn workspace @brightidbot/bot",
    "web": "yarn workspace @brightidbot/web",
    "utils": "yarn workspace @brightidbot/utils",
    "scripts": "yarn workspace @brightidbot/scripts",
    "shared": "yarn workspace @brightidbot/shared",
    "re:build": "yarn shared re:build && yarn utils re:build && yarn bot re:build && yarn web re:build && yarn s
cripts re:build"
  },
  "workspaces": [
    "apps/*",
    "packages/*"
  ]
}
```

========== End file

========== Start file DiscordServer.res

```
open Promise

exception DiscordRateLimited

let botToken = Remix.process["env"]["DISCORD_API_TOKEN"]

// let options: Client.clientOptions = {
//   intents: ["GUILDS"],
// }

// let client = Client.createDiscordClient(~options)

let mapGuildOauthRecord = decodedGuilds => {
  decodedGuilds->Belt.Option.map(guilds =>
```

```
    guilds->Js.Array2.map(guild => {
      let guild = guild->Js.Json.decodeObject->Belt.Option.getUnsafe

      (
        {
          id: guild
          ->Js.Dict.get("id")
          ->Belt.Option.flatMap(Js.Json.decodeString)
          ->Belt.Option.getExn,
          name: guild
          ->Js.Dict.get("name")
          ->Belt.Option.flatMap(Js.Json.decodeString)
          ->Belt.Option.getExn,
          icon: guild->Js.Dict.get("icon")->Belt.Option.flatMap(Js.Json.decodeString),
        }: Types.oauthGuild
      )
    })
  )
}
let mapGuildRecord = decodedGuild => {
  switch decodedGuild {
  | None => None
  | Some(guild) =>
    Some(
      (
        {
          id: guild
          ->Js.Dict.get("id")
          ->Belt.Option.flatMap(Js.Json.decodeString)
          ->Belt.Option.getExn,
          name: guild
          ->Js.Dict.get("name")
          ->Belt.Option.flatMap(Js.Json.decodeString)
          ->Belt.Option.getExn,
          icon: guild->Js.Dict.get("icon")->Belt.Option.flatMap(Js.Json.decodeString),
          roles: guild
          ->Js.Dict.get("roles")
          ->Belt.Option.flatMap(Js.Json.decodeArray)
          ->Belt.Option.getExn,
          owner_id: guild
          ->Js.Dict.get("owner_id")
          ->Belt.Option.flatMap(Js.Json.decodeString)
          ->Belt.Option.getExn,
        }: Types.guild
      ),
    )
  }
}
let mapGuildMemberRecord = decodedGuildMember => {
  switch decodedGuildMember {
  | None => None
  | Some(guildMember) =>
    Some(
      (
        {
          roles: guildMember
          ->Js.Dict.get("roles")
          ->Belt.Option.flatMap(Js.Json.decodeArray)
          ->Belt.Option.map(roles =>
            roles->Js.Array2.map(role => role->Js.Json.decodeString->Belt.Option.getExn)
          )
          ->Belt.Option.getExn,
        }: Types.guildMember
      ),
    )
  }
}

let mapRoleRecord = decodedRoles => {
  decodedRoles->Belt.Option.map(roles =>
    roles->Js.Array2.map(role => {
      let role = role->Js.Json.decodeObject->Belt.Option.getUnsafe

      (
        {
          id: role
          ->Js.Dict.get("id")
          ->Belt.Option.flatMap(Js.Json.decodeString)
          ->Belt.Option.getExn,
          name: role
          ->Js.Dict.get("name")
          ->Belt.Option.flatMap(Js.Json.decodeString)
          ->Belt.Option.getExn,
          permissions: role
          ->Js.Dict.get("permissions")
          ->Belt.Option.flatMap(Js.Json.decodeNumber)
          ->Belt.Option.getExn,
```

```
        }: Types.role
      )
    })
  )
}

let sleep = _ms => %raw(` new Promise((resolve) => setTimeout(resolve, _ms))`)

//fetch all bot and user guilds
let rec fetchBotGuilds = (~after=0, ~allGuilds=[], ()): Promise.t<array<Types.oauthGuild>> => {
  open Webapi.Fetch

  let headers = HeadersInit.make({
    "Authorization": `Bot ${botToken}`,
  })
  let init = RequestInit.make(~method_=Get, ~headers, ())

  `https://discord.com/api/users/@me/guilds?after=${after->Belt.Int.toString}`
  ->Request.makeWithInit(init)
  ->fetchWithRequest
  ->then(res => res->Response.json)
  ->then(json => {
    switch json->Js.Json.test(Js.Json.Array) {
    | false => {
        let rateLimit = json->Js.Json.decodeObject->Belt.Option.getUnsafe

        let retry_after =
          rateLimit->Js.Dict.get("retry_after")->Belt.Option.flatMap(Js.Json.decodeNumber)

        let retry_after = switch retry_after {
        | None => DiscordRateLimited->raise
        | Some(retry_after) => retry_after->Belt.Float.toInt + 100
        }

        Js.log(
          `Discord Rate Limited: Retrying fetch for guilds after: ${after->Belt.Int.toString} in ${retry_after->
Belt.Int.toString}ms`,
        )
        sleep(retry_after)->then(_ => fetchBotGuilds(~after, ~allGuilds, ()))
      }

    | true => {
        let guilds = json->Js.Json.decodeArray->mapGuildOauthRecord->Belt.Option.getUnsafe
        switch guilds->Belt.Array.length <= 1 {
        | true => allGuilds->Belt.Array.concat(guilds)->resolve
        | false => {
            let last = guilds->Js.Array2.length - 1
            let after = guilds[last].id->Belt.Int.fromString->Belt.Option.getUnsafe
            let allGuilds = allGuilds->Belt.Array.concat(guilds)
            fetchBotGuilds(~after, ~allGuilds, ())
          }
        }
      }
    }
  })
  ->catch(e => {
    switch e {
    | DiscordRateLimited => e->raise
    | _ => allGuilds->resolve
    }
  })
}

type guildsCursor = {guilds: array<Types.oauthGuild>, after: option<string>}
//fetch first 1000 guilds
let rec fetchBotGuildsLimit = (~after): Promise.t<guildsCursor> => {
  open Webapi.Fetch

  let headers = HeadersInit.make({
    "Authorization": `Bot ${botToken}`,
  })
  let init = RequestInit.make(~method_=Get, ~headers, ())
  switch after {
  | Some(after) =>
    `https://discord.com/api/users/@me/guilds?after=${after}`
    ->Request.makeWithInit(init)
    ->fetchWithRequest
    ->then(res => res->Response.json)
    ->then(json => {
      switch json->Js.Json.test(Js.Json.Array) {
      | false => {
          let rateLimit = json->Js.Json.decodeObject->Belt.Option.getUnsafe

          let retry_after =
            rateLimit->Js.Dict.get("retry_after")->Belt.Option.flatMap(Js.Json.decodeNumber)

          let retry_after = switch retry_after {
```

```
          | None => DiscordRateLimited->raise
          | Some(retry_after) => retry_after->Belt.Float.toInt + 100
          }

          Js.log(
            `Discord Rate Limited: Retrying fetch for guilds after: ${after} in ${retry_after->Belt.Int.toString
}ms`,
          )
          sleep(retry_after)->then(_ => fetchBotGuildsLimit(~after=Some(after)))
        }

      | true => {
          let guilds = json->Js.Json.decodeArray->mapGuildOauthRecord->Belt.Option.getUnsafe
          let last = guilds->Js.Array2.length - 1
          let after = guilds[last].id->Some
          {guilds, after}->resolve
        }
      }
    })
    ->catch(e => {
      switch e {
      | DiscordRateLimited => e->raise
      | _ => {guilds: [], after: Some(after)}->resolve
      }
    })
  | None => {guilds: [], after}->resolve
  }
}

let rec fetchUserGuilds = (user: RemixAuth.User.t) => {
  open Webapi.Fetch
  let headers = HeadersInit.make({
    "Authorization": `Bearer ${user->RemixAuth.User.getAccessToken}`,
  })
  let init = RequestInit.make(~method_=Get, ~headers, ())
  "https://discord.com/api/users/@me/guilds"
  ->Request.makeWithInit(init)
  ->fetchWithRequest
  ->then(res => res->Response.json)
  ->then(json =>
    switch json->Js.Json.test(Js.Json.Array) {
    | false => {
        let rateLimit = json->Js.Json.decodeObject->Belt.Option.getUnsafe

        let retry_after =
          rateLimit->Js.Dict.get("retry_after")->Belt.Option.flatMap(Js.Json.decodeNumber)

        let retry_after = switch retry_after {
        | None => DiscordRateLimited->raise
        | Some(retry_after) => retry_after->Belt.Float.toInt + 100
        }
        Js.log(
          `Discord Rate Limited: Retrying fetch user guilds in ${retry_after->Belt.Int.toString}ms`,
        )
        sleep(retry_after)->then(_ => fetchUserGuilds(user))
      }

    | true => json->Js.Json.decodeArray->mapGuildOauthRecord->Belt.Option.getUnsafe->resolve
    }
  )
  ->catch(e => {
    switch e {
    | DiscordRateLimited => e->raise
    | _ => []->resolve
    }
  })
}

let fetchDiscordGuildFromId = (~guildId) => {
  open Webapi.Fetch
  let headers = HeadersInit.make({
    "Authorization": `Bot ${botToken}`,
  })
  let init = RequestInit.make(~method_=Get, ~headers, ())

  `https://discord.com/api/guilds/${guildId}`
  ->Request.makeWithInit(init)
  ->fetchWithRequest
  ->then(res => res->Response.json)
  ->then(json => json->Js.Json.decodeObject->mapGuildRecord->Js.Nullable.fromOption->resolve)
}

let fetchGuildMemberFromId = (~guildId, ~userId) => {
  open Webapi.Fetch
  let headers = HeadersInit.make({
    "Authorization": `Bot ${botToken}`,
  })
```

```
  let init = RequestInit.make(~method_=Get, ~headers, ())

  `https://discord.com/api/guilds/${guildId}/members/${userId}`
  ->Request.makeWithInit(init)
  ->fetchWithRequest
  ->then(res => res->Response.json)
  ->then(json => {
    json->Js.Json.decodeObject->mapGuildMemberRecord->Js.Nullable.fromOption->resolve
  })
}

let rec fetchGuildRoles = (~guildId) => {
  open Webapi.Fetch
  let headers = HeadersInit.make({
    "Authorization": `Bot ${botToken}`,
  })

  let init = RequestInit.make(~method_=Get, ~headers, ())

  `https://discord.com/api/guilds/${guildId}/roles`
  ->Request.makeWithInit(init)
  ->fetchWithRequest
  ->then(res => res->Response.json)
  ->then(json =>
    switch json->Js.Json.test(Js.Json.Array) {
    | false => {
        let rateLimit = json->Js.Json.decodeObject->Belt.Option.getUnsafe

        let retry_after =
          rateLimit->Js.Dict.get("retry_after")->Belt.Option.flatMap(Js.Json.decodeNumber)

        let retry_after = switch retry_after {
        | None => DiscordRateLimited->raise
        | Some(retry_after) => retry_after->Belt.Float.toInt + 100
        }
        Js.log(
          `Discord Rate Limited: Retrying fetch guild: ${guildId} roles in ${retry_after->Belt.Int.toString}ms`,
        )
        sleep(retry_after)->then(_ => fetchGuildRoles(~guildId))
      }

    | true => json->Js.Json.decodeArray->mapRoleRecord->Belt.Option.getUnsafe->resolve
    }
  )
  ->catch(e => {
    switch e {
    | DiscordRateLimited => e->raise
    | _ => []->resolve
    }
  })
}

let memberIsAdmin = (~guildRoles: array<Types.role>, ~memberRoles) => {
  let adminPerm = %raw(`0x0000000000000008`)

  let memberRoles = guildRoles->Js.Array2.filter(role => memberRoles->Js.Array2.includes(role.id))
  memberRoles->Js.Array2.some(role => {
    %raw(`(role.permissions & adminPerm)`) === adminPerm
  })
}
```

========== End file

catala-dsfr--spinners

Rescript v10

Repo: https://github.com/CatalaLang/catala-dsfr

========== Start file package.json (part or full code)

```
{
  "name": "catala-dsfr",
  "version": "0.1.1",
  "repository": "https://github.com/CatalaLang/catala-dsfr",
  "scripts": {
    "clean": "rescript clean -with-deps",
    "build": "yarn run pre && vite build",
    "deploy": "yarn run build --base=/demos/catala/ && rsync -rv --delete-before dist/*",
    "serve": "vite preview",
    "dev": "yarn run pre && vite",
    "re:build": "rescript build -with-deps",
    "re:watch": "yarn run pre && rescript build -w -with-deps",
    "assets": "rsync -r node_modules/@catala-lang/catala-web-assets/assets/* assets",
```

```
    "postinstall": "copy-dsfr-to-public",
    "pre": "yarn run re:build && only-include-used-icons && yarn run assets"
  },
  "keywords": [
    "rescript"
  ],
  "author": "Emile Rolley <emile.rolley@tuta.io>",
  "license": "Apache-2.0",
  "dependencies": {
    "@catala-lang/catala-explain": "^0.2.2",
    "@catala-lang/catala-web-assets": "^0.8.9",
    "@catala-lang/french-law": "^0.8.3-b.3",
    "@catala-lang/rescript-catala": "^0.8.1-b.0",
    "@codegouvfr/react-dsfr": "^0.78.2",
    "@rescript/core": "^0.5.0",
    "@rescript/react": "^0.11.0",
    "@rjsf/core": "^5.1.0",
    "@rjsf/utils": "^5.1.0",
    "@rjsf/validator-ajv8": "^5.1.0",
    "file-saver": "^2.0.5",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-loader-spinner": "^5.4.5",
    "rescript-docx": "^0.1.5",
    "tslib": "^2.6.2"
  },
  "devDependencies": {
    "@jihchi/vite-plugin-rescript": "^5.1.0",
    "@originjs/vite-plugin-commonjs": "^1.0.3",
    "@vitejs/plugin-react": "^3.1.0",
    "jsdom": "^21.1.0",
    "rescript": "^10.1.4",
    "tailwindcss": "^3.2.6",
    "vite": "^4.4.9"
  }
}
```

========== End file

========== Start file Spinners.res

```
module Oval = {
  @react.component @module("react-loader-spinner")
  external make: (
    ~height: int=?,
    ~width: int=?,
    ~color: string=?,
    ~visible: bool=?,
    ~secondaryColor: string=?,
    ~strokeWidth: int=?,
    ~strokeWidthSecondary: int=?,
    ~radius: int=?,
    ~wrapperClassName: string=?,
  ) => React.element = "Oval"
}

module ThreeDots = {
  @react.component @module("react-loader-spinner")
  external make: (
    ~height: int=?,
    ~width: int=?,
    ~color: string=?,
    ~secondaryColor: string=?,
    ~visible: bool=?,
    ~radius: int=?,
    ~wrapperClassName: string=?,
  ) => React.element = "ThreeDots"
}

let loader =
  <Oval
    height=25
    width=25
    strokeWidth=5
    color="#518fff"
    secondaryColor="#98b4ff"
    wrapperClassName="justifyCenter"
  />
```

========== End file

chat-script--server

Rescript v11

Repo: https://github.com/Exegetech/chat-rescript

========== Start file package.json (part or full code)

```
{
  "name": "backend",
  "scripts": {
    "res:build": "rescript",
    "res:clean": "rescript clean",
    "res:dev": "rescript build -w",
    "dev": "NODE_ENV=dev nodemon src/Server.bs.mjs",
    "build": "node scripts/build.js"
  },
  "dependencies": {
    "@fastify/cors": "8.4.0",
    "@fastify/static": "6.11.2",
    "@fastify/websocket": "8.2.0",
    "@rescript/core": "0.5.0",
    "fastify": "4.24.1",
    "rescript": "11.0.0-rc.4",
    "shared": "workspace:*"
  },
  "devDependencies": {
    "esbuild": "0.19.5",
    "nodemon": "3.0.1"
  }
}
```

========== End file

========== Start file Server.res

```
open Fastify

let env = Dict.get(Node.Process.env, "NODE_ENV")

@val external importMetaUrl: string = "import.meta.url"

let dirname = importMetaUrl
  -> Node.Url.fileURLToPath
  -> Node.Path.dirname

let fastify = create({ logger: true })

switch env {
  | None => {
    fastify->registerStatic(fastifyStatic, {
      root: Node.Path.join(dirname, "public"),
    })
  }
  | _ => ()
}

fastify->register(fastifyCors)
fastify->register(fastifyWebsocket)

fastify->addHook(PreValidation, async (request, reply) => {
  let path = request.routeOptions.url
  let username = Dict.get(request.query, "username")

  switch (path, username) {
    | ("/chat", None) => reply
      ->HTTP.code(Forbidden)
      ->HTTP.send("Connection rejected")
    | _ => ()
  }
})

fastify->httpGet("/chat", async (_request, reply) => {
  let payload = Chat.getChatHistory()
    -> Message.ToClient.serializeMany

  switch payload {
    | Error(error) => fastify.log->Log.logError(error)
    | Ok(payload) => reply
      ->HTTP.code(Okay)
      ->HTTP.send(payload)
  }
})

fastify->register(async (fastify) => {
  fastify->socketGet("/room", (connection, request) => {
    let username = request.query
      -> Dict.get("username")
      -> Option.getExn
```

```
    Chat.handleClient(~username, ~socket=connection.socket, ~onError=(errMsg) => {
      fastify.log->Log.logError(errMsg)
    })
  })
})

let start = async () => {
  try {
    await fastify->listen({ port: 3000 })
  } catch {
    | Exn.Error(obj) =>
      switch Exn.message(obj) {
        | Some(m) => fastify.log->Log.logError(m)
    | None => ()
      }

      Node.Process.exit(1)
  }
}

let _ = await start()
```

========== End file

brightid--authserver

Rescript v10

Repo: https://github.com/ShenaniganDApp/brightid-discord-bot

========== Start file package.json (part or full code)

```
{
  "name": "root",
  "private": true,
  "devDependencies": {
    "patch-package": "^6.4.7"
  },
  "dependencies": {
    "@rescript/core": "^0.2.0",
    "brightid_sdk": "^1.0.1",
    "canvas": "^2.9.0",
    "concurrently": "^7.1.0",
    "dotenv": "^8.2.0",
    "find-up": "^6.3.0",
    "rescript": "^10.1.0-rc.5",
    "rescript-discordjs": "^0.3.0",
    "rescript-nodejs": "^14.3.1",
    "uuid": "^8.3.0"
  },
  "scripts": {
    "bot": "yarn workspace @brightidbot/bot",
    "web": "yarn workspace @brightidbot/web",
    "utils": "yarn workspace @brightidbot/utils",
    "scripts": "yarn workspace @brightidbot/scripts",
    "shared": "yarn workspace @brightidbot/shared",
    "re:build": "yarn shared re:build && yarn utils re:build && yarn bot re:build && yarn web re:build && yarn s
cripts re:build"
  },
  "workspaces": [
    "apps/*",
    "packages/*"
  ]
}
```

========== End file

========== Start file AuthServer.res

```
let clientID = Remix.process["env"]["DISCORD_CLIENT_ID"]
let clientSecret = Remix.process["env"]["DISCORD_CLIENT_SECRET"]
let baseUrl = Remix.process["env"]["BASE_URL"]
let uuidNamespace = Remix.process["env"]["UUID_NAMESPACE"]

let cookieOptions = Remix.CreateCookieOptions.make(
  ~sameSite=#lax,
  ~path="/",
  ~httpOnly=true,
  ~secrets=[uuidNamespace],
  ~secure=Remix.process["env"]["NODE_ENV"] === "production",
  (),
)

let cookie = Remix.createCookieWithOptions("__session", cookieOptions)
```

```
let sessionStorage =
  cookie
  ->Remix.CreateCookieSessionStorageOptions.make(~cookie=_)
  ->Remix.createCookieSessionStorageWithOptions(~options=_)

let authenticator = sessionStorage->RemixAuth.Authenticator.make

let discordStrategy = RemixAuth.DiscordStrategy.CreateDiscordStategyOptions.make(
  ~clientID,
  ~clientSecret,
  ~callbackURL=baseUrl ++ "/auth/discord/callback",
  ~scope=["identify", "guilds", "guilds.join"],
  (),
)->RemixAuth.DiscordStrategy.make(({accessToken, profile}) => {
  {"accessToken": accessToken, "profile": profile}->Promise.resolve
})

authenticator->RemixAuth.Authenticator.use(discordStrategy)
```

========== End file

catala-dsfr--router

Rescript v10

Repo: https://github.com/CatalaLang/catala-dsfr

========== Start file package.json (part or full code)

```
{
  "name": "catala-dsfr",
  "version": "0.1.1",
  "repository": "https://github.com/CatalaLang/catala-dsfr",
  "scripts": {
    "clean": "rescript clean -with-deps",
    "build": "yarn run pre && vite build",
    "deploy": "yarn run build --base=/demos/catala/ && rsync -rv --delete-before dist/*",
    "serve": "vite preview",
    "dev": "yarn run pre && vite",
    "re:build": "rescript build -with-deps",
    "re:watch": "yarn run pre && rescript build -w -with-deps",
    "assets": "rsync -r node_modules/@catala-lang/catala-web-assets/assets/* assets",
    "postinstall": "copy-dsfr-to-public",
    "pre": "yarn run re:build && only-include-used-icons && yarn run assets"
  },
  "keywords": [
    "rescript"
  ],
  "author": "Emile Rolley <emile.rolley@tuta.io>",
  "license": "Apache-2.0",
  "dependencies": {
    "@catala-lang/catala-explain": "^0.2.2",
    "@catala-lang/catala-web-assets": "^0.8.9",
    "@catala-lang/french-law": "^0.8.3-b.3",
    "@catala-lang/rescript-catala": "^0.8.1-b.0",
    "@codegouvfr/react-dsfr": "^0.78.2",
    "@rescript/core": "^0.5.0",
    "@rescript/react": "^0.11.0",
    "@rjsf/core": "^5.1.0",
    "@rjsf/utils": "^5.1.0",
    "@rjsf/validator-ajv8": "^5.1.0",
    "file-saver": "^2.0.5",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-loader-spinner": "^5.4.5",
    "rescript-docx": "^0.1.5",
    "tslib": "^2.6.2"
  },
  "devDependencies": {
    "@jihchi/vite-plugin-rescript": "^5.1.0",
    "@originjs/vite-plugin-commonjs": "^1.0.3",
    "@vitejs/plugin-react": "^3.1.0",
    "jsdom": "^21.1.0",
    "rescript": "^10.1.4",
    "tailwindcss": "^3.2.6",
    "vite": "^4.4.9"
  }
}
```

========== End file

========== Start file Router.res

```
@react.component
let make = () => {
  switch Nav.getCurrentURL().path {
  | list{route} if route == AllocationsFamiliales.FormInfos.url => <AllocationsFamiliales />
  | list{route} if route == AidesLogement.FormInfos.url => <AidesLogement />
  | list{route, "sources"} if route == AllocationsFamiliales.FormInfos.url =>
    <SourceCode
      html={WebAssets.allocationsFamilialesAssets.html}
      simulatorUrl={AllocationsFamiliales.FormInfos.url}
    />
  | list{route, "sources"} if route == AidesLogement.FormInfos.url =>
    <SourceCode
      html={WebAssets.aidesLogementAssets.html} simulatorUrl={AidesLogement.FormInfos.url}
    />
  | _ => <Home />
  }
}

module Link = {
  @react.component
  let make = (~href: string, ~children) => {
    <a
      href={href}
      onClick={evt => {
        evt->ReactEvent.Mouse.preventDefault
        href->Nav.goTo
      }}>
      children
    </a>
  }
}
```

========== End file

res-x--htmx

Rescript v11

Repo: https://github.com/zth/res-x

========== Start file package.json (part or full code)

```
{
  "name": "rescript-x",
  "version": "0.1.0-alpha.7",
  "scripts": {
    "res:build": "rescript",
    "res:clean": "rescript clean",
    "res:dev": "rescript build -w"
  },
  "keywords": [
    "rescript"
  ],
  "files": [
    "README.md",
    "CHANGELOG.md",
    "rescript.json",
    "src/**/*",
    "res-x-vite-plugin.mjs"
  ],
  "author": "Gabriel Nordeborn",
  "license": "MIT",
  "peerDependencies": {
    "rescript": ">=11.0.0-rc.5",
    "@rescript/core": ">=0.5.0",
    "vite": ">=4.4.11",
    "rescript-bun": ">=0.1.0"
  },
  "devDependencies": {
    "@rescript/core": "^0.5.0",
    "fast-glob": "^3.3.1",
    "rescript": "11.0.0-rc.5",
    "rescript-bun": "0.1.0"
  },
  "dependencies": {
    "fast-glob": "^3.3.1"
  }
}
```

========== End file

========== Start file Htmx.res

```
type hxSwap =
```

```
  | @as("outerHTML") OuterHTML
  | @as("innerHTML") InnerHTML
  | @as("beforebegin") BeforeBegin
  | @as("afterbegin") AfterBegin
  | @as("beforeend") BeforeEnd
  | @as("afterend") AfterEnd
  | @as("delete") Delete
  | @as("none") None

type topOrBottom = | @as("top") Top | @as("bottom") Bottom

type modifier =
  | Swap(string)
  | Settle(string)
  | Transition
  | Scroll(topOrBottom)
  | ScrollWithSelector(string, topOrBottom)
  | Show(topOrBottom)
  | ShowWithSelector(string, topOrBottom)

module Swap = {
  type t = string
  let make = (swap: hxSwap, ~modifier=?) =>
    [
      Some((swap :> string)),
      modifier->Option.map(modifier =>
        switch modifier {
        | Swap(s) => `swap:${s}`
        | Transition => `transition:true`
        | Settle(s) => `settle:${s}`
        | Scroll(topOrBottom) => `scroll:${(topOrBottom :> string)}`
        | ScrollWithSelector(s, topOrBottom) => `scroll:${s}:${(topOrBottom :> string)}`
        | Show(topOrBottom) => `show:${(topOrBottom :> string)}`
        | ShowWithSelector(s, topOrBottom) => `show:${s}:${(topOrBottom :> string)}`
        }
      ),
    ]
    ->Array.keepSome
    ->Array.joinWith(" ")
}

type hxTarget =
  | CssSelector(string)
  | This
  | Closest({cssSelector: string})
  | Find({cssSelector: string})
  | Next({cssSelector: string})
  | Previous({cssSelector: string})

module Target = {
  type t = string
  let make = (target: hxTarget) =>
    switch target {
    | Closest({cssSelector}) => `closest ${cssSelector}`
    | This => "this"
    | CssSelector(cssSelector) => cssSelector
    | Find({cssSelector}) => `find ${cssSelector}`
    | Next({cssSelector}) => `next ${cssSelector}`
    | Previous({cssSelector}) => `previous ${cssSelector}`
    }
}

@unboxed type hxUrl = | @as(true) True | @as(false) False | URL(string)

type hxParams = IncludeAll | IncludeNone | Not(array<string>) | Only(array<string>)

module Params = {
  type t = string
  let make = (p: hxParams) => {
    switch p {
    | IncludeAll => "*"
    | IncludeNone => "none"
    | Not(list) => `not ${list->Array.joinWith(",")}`
    | Only(list) => list->Array.joinWith(",")
    }
  }
}

type hxEncoding = MultipartFormData

module Encoding = {
  type t = string
  let make = encoding =>
    switch encoding {
    | MultipartFormData => "multipart/form-data"
    }
}
```

```
type hxIndicator = Selector(string) | Closest(string)

module Indicator = {
  type t = string
  let make = (hxIndicator: hxIndicator) =>
    switch hxIndicator {
    | Selector(s) => s
    | Closest(s) => `closest ${s}`
    }
}

module Headers: {
  type t
  let make: Dict.t<string> => t
} = {
  type t = string
  let make = (dict: Dict.t<string>) => dict->JSON.stringifyAny->Option.getWithDefault("{}")
}

type hxSyncStrategyQueueModifier =
  | /** queue the first request to show up while a request is in flight */
  @as("first")
  First
  | /** queue the last request to show up while a request is in flight */
  @as("last")
  Last
  | /** queue all requests that show up while a request is in flight */
  @as("all")
  All

type hxSyncStrategy =
  Drop | Abort | Replace | Queue | QueueWithModifier(hxSyncStrategyQueueModifier)

type hxSync = Selector(string) | SelectorAndStrategy(string, hxSyncStrategy)

module Sync = {
  type t = string
  let strategyToString = (s: hxSyncStrategy) =>
    switch s {
    | Drop => "drop"
    | Abort => "abort"
    | Replace => "replace"
    | Queue => "queue"
    | QueueWithModifier(modifier) =>
      `queue ${switch modifier {
        | First => "first"
        | Last => "last"
        | All => "all"
        }}`
    }

  let make = (c: hxSync) =>
    switch c {
    | Selector(s) => s
    | SelectorAndStrategy(s, strategy) => `${s}:${strategy->strategyToString}`
    }
}

type hxVals =
  | /** A JSON value. */ Json(Js.Json.t)
  | /** Raw string. This needs to be valid, parseable JSON. */ JsonUnsafe(string)
  | /** WARNING: This might introduce security issues. Avoid unless really needed.

  Stringified JS that will be evaluated. */
  RawJavaScript(string)

module Vals = {
  type t = string
  let make = vals =>
    switch vals {
    | Json(json) => JSON.stringify(json)
    | JsonUnsafe(s) => s
    | RawJavaScript(s) => `js:${s}`
    }
}

type hxInheritedAttributes =
  | @as("hx-swap") Swap
  | @as("hx-boost") Boost
  | @as("hx-push-url") PushUrl
  | @as("hx-replace-url") ReplaceUrl
  | @as("hx-select") Select
  | @as("hx-select-oob") SelectOob
  | @as("hx-params") Params
  | @as("hx-prompt") Prompt
  | @as("hx-validate") Validate
```

```
  | @as("hx-confirm") Confirm
  | @as("hx-disable") Disable
  | @as("hx-encoding") Encoding
  | @as("hx-indicator") Indicator
  | @as("hx-history") History
  | @as("hx-history-elt") HistoryElt
  | @as("hx-include") Include
  | @as("hx-headers") Headers
  | @as("hx-sync") Sync
  | @as("hx-vals") Vals
  | @as("hx-preserve") Preserve

type hxDisinherit = All | Attributes(array<hxInheritedAttributes>)

module Disinherit = {
  type t = string
  let make = d =>
    switch d {
    | All => "*"
    | Attributes(attrs) => attrs->Array.map(a => (a :> string))->Array.joinWith(" ")
    }
}

// Missing:
//  request, ext
type htmxProps = {
  /** https://htmx.org/attributes/hx-get/ */
  @as("hx-get")
  hxGet?: Handlers.hxGet,
  @as("data-hx-get")
  rawHxGet?: string,
  /** https://htmx.org/attributes/hx-post/ */
  @as("hx-post")
  hxPost?: Handlers.hxPost,
  @as("data-hx-post")
  rawHxPost?: string,
  /** https://htmx.org/attributes/hx-put/ */
  @as("hx-put")
  hxPut?: Handlers.hxPut,
  @as("data-hx-put")
  rawHxPut?: string,
  /** https://htmx.org/attributes/hx-delete/ */
  @as("hx-delete")
  hxDelete?: Handlers.hxDelete,
  @as("data-hx-delete")
  rawHxDelete?: string,
  /** https://htmx.org/attributes/hx-patch/ */
  @as("hx-patch")
  hxPatch?: Handlers.hxPatch,
  @as("data-hx-patch")
  rawHxPatch?: string,
  /** https://htmx.org/attributes/hx-swap/ */
  @as("hx-swap")
  hxSwap?: Swap.t,
  @as("data-hx-swap")
  rawHxSwap?: string,
  /** https://htmx.org/docs/#boosting */
  @as("hx-boost")
  hxBoost?: bool,
  /** https://htmx.org/attributes/hx-push-url/ */
  @as("hx-push-url")
  hxPushUrl?: hxUrl,
  @as("data-hx-push-url")
  rawHxPushUrl?: string,
  /** https://htmx.org/attributes/hx-replace-url/ */
  @as("hx-replace-url")
  hxReplaceUrl?: hxUrl,
  @as("data-hx-replace-url")
  rawHxReplaceUrl?: string,
  /** https://htmx.org/attributes/hx-select/ */
  @as("hx-select")
  hxSelect?: string,
  @as("data-hx-select")
  rawHxSelect?: string,
  /** https://htmx.org/attributes/hx-select-oob/ */
  @as("hx-select-oob")
  hxSelectOob?: string,
  @as("data-hx-select-oob")
  rawHxSelectOob?: string,
  /** https://htmx.org/attributes/hx-params/ */
  @as("hx-params")
  hxParams?: Params.t,
  @as("data-hx-params")
  rawHxParams?: string,
  /** https://htmx.org/attributes/hx-prompt/ */
  @as("hx-prompt")
  hxPrompt?: string,
```

```
  @as("data-hx-prompt")
  rawHxPrompt?: string,
  /** https://htmx.org/attributes/hx-validate/ */
  @as("hx-validate")
  hxValidate?: bool,
  /** https://htmx.org/attributes/hx-confirm/ */
  @as("hx-confirm")
  hxConfirm?: string,
  /** https://htmx.org/attributes/hx-disable/ */
  @as("hx-disable")
  hxDisable?: bool,
  /** https://htmx.org/attributes/hx-encoding/ */
  @as("hx-encoding")
  hxEncoding?: Encoding.t,
  @as("data-hx-encoding")
  rawHxEncoding?: string,
  /** https://htmx.org/attributes/hx-indicator/ */
  @as("hx-indicator")
  hxIndicator?: Indicator.t,
  @as("data-hx-indicator")
  rawHxIndicator?: string,
  /** https://htmx.org/attributes/hx-history/ */
  @as("hx-history")
  hxHistory?: bool,
  /** https://htmx.org/attributes/hx-history-elt/ */
  @as("hx-history-elt")
  hxHistoryElt?: bool,
  /** https://htmx.org/attributes/hx-include/ */
  @as("hx-include")
  hxInclude?: string,
  /** https://htmx.org/attributes/hx-headers/ */
  @as("hx-headers")
  hxHeaders?: Headers.t,
  @as("data-hx-headers")
  rawHxHeaders?: string,
  /** https://htmx.org/attributes/hx-sync/ */
  @as("hx-sync")
  hxSync?: Sync.t,
  @as("data-hx-sync")
  rawHxSync?: string,
  /** https://htmx.org/attributes/hx-vals/ */
  @as("hx-vals")
  hxVals?: Vals.t,
  @as("data-hx-vals")
  rawHxVals?: string,
  /** https://htmx.org/attributes/hx-preserve/ */
  @as("hx-preserve")
  hxPreserve?: bool,
  /** https://htmx.org/attributes/hx-disinherit/ */
  @as("hx-disinherit")
  hxDisinherit?: Disinherit.t,
  @as("data-hx-disinherit")
  rawHxDisinherit?: string,
  /** https://htmx.org/attributes/hx-target/  TODO */
  @as("hx-target")
  hxTarget?: Target.t,
  @as("data-hx-target")
  rawHxTarget?: string,
}
```

========== End file

========== Start file

```
type hxSwap =
  | @as("outerHTML") OuterHTML
  | @as("innerHTML") InnerHTML
  | @as("beforebegin") BeforeBegin
  | @as("afterbegin") AfterBegin
  | @as("beforeend") BeforeEnd
  | @as("afterend") AfterEnd
  | @as("delete") Delete
  | @as("none") None

type topOrBottom = | @as("top") Top | @as("bottom") Bottom

type modifier =
  | Swap(string)
  | Settle(string)
  | Transition
  | Scroll(topOrBottom)
  | ScrollWithSelector(string, topOrBottom)
  | Show(topOrBottom)
  | ShowWithSelector(string, topOrBottom)

module Swap: {
```

```
  type t
  let make: (hxSwap, ~modifier: modifier=?) => t
}

type hxTarget =
  | CssSelector(string)
  | This
  | Closest({cssSelector: string})
  | Find({cssSelector: string})
  | Next({cssSelector: string})
  | Previous({cssSelector: string})

module Target: {
  type t
  let make: hxTarget => t
}

@unboxed type hxUrl = | @as(true) True | @as(false) False | URL(string)

type hxParams = IncludeAll | IncludeNone | Not(array<string>) | Only(array<string>)

module Params: {
  type t
  let make: hxParams => t
}

type hxEncoding = MultipartFormData

module Encoding: {
  type t
  let make: hxEncoding => t
}

type hxIndicator = Selector(string) | Closest(string)

module Indicator: {
  type t
  let make: hxIndicator => t
}

module Headers: {
  type t
  let make: RescriptCore.Dict.t<string> => t
}

type hxSyncStrategyQueueModifier =
  | /** queue the first request to show up while a request is in flight */
  @as("first")
  First
  | /** queue the last request to show up while a request is in flight */
  @as("last")
  Last
  | /** queue all requests that show up while a request is in flight */
  @as("all")
  All

type hxSyncStrategy =
  | Drop
  | Abort
  | Replace
  | Queue
  | QueueWithModifier(hxSyncStrategyQueueModifier)

type hxSync = Selector(string) | SelectorAndStrategy(string, hxSyncStrategy)

module Sync: {
  type t
  let make: hxSync => t
}

type hxVals = Json(Js.Json.t) | JsonUnsafe(string) | RawJavaScript(string)

module Vals: {
  type t
  let make: hxVals => t
}

type hxInheritedAttributes =
  | Swap
  | Boost
  | PushUrl
  | ReplaceUrl
  | Select
  | SelectOob
  | Params
  | Prompt
  | Validate
```

```
  | Confirm
  | Disable
  | Encoding
  | Indicator
  | History
  | HistoryElt
  | Include
  | Headers
  | Sync
  | Vals
  | Preserve

type hxDisinherit = All | Attributes(array<hxInheritedAttributes>)

module Disinherit: {
  type t
  let make: hxDisinherit => t
}

type htmxProps = {
  /** https://htmx.org/attributes/hx-get/ */
  @as("hx-get")
  hxGet?: Handlers.hxGet,
  @as("data-hx-get")
  rawHxGet?: string,
  /** https://htmx.org/attributes/hx-post/ */
  @as("hx-post")
  hxPost?: Handlers.hxPost,
  @as("data-hx-post")
  rawHxPost?: string,
  /** https://htmx.org/attributes/hx-put/ */
  @as("hx-put")
  hxPut?: Handlers.hxPut,
  @as("data-hx-put")
  rawHxPut?: string,
  /** https://htmx.org/attributes/hx-delete/ */
  @as("hx-delete")
  hxDelete?: Handlers.hxDelete,
  @as("data-hx-delete")
  rawHxDelete?: string,
  /** https://htmx.org/attributes/hx-patch/ */
  @as("hx-patch")
  hxPatch?: Handlers.hxPatch,
  @as("data-hx-patch")
  rawHxPatch?: string,
  /** https://htmx.org/attributes/hx-swap/ */
  @as("hx-swap")
  hxSwap?: Swap.t,
  @as("data-hx-swap")
  rawHxSwap?: string,
  /** https://htmx.org/docs/#boosting */
  @as("hx-boost")
  hxBoost?: bool,
  /** https://htmx.org/attributes/hx-push-url/ */
  @as("hx-push-url")
  hxPushUrl?: hxUrl,
  @as("data-hx-push-url")
  rawHxPushUrl?: string,
  /** https://htmx.org/attributes/hx-replace-url/ */
  @as("hx-replace-url")
  hxReplaceUrl?: hxUrl,
  @as("data-hx-replace-url")
  rawHxReplaceUrl?: string,
  /** https://htmx.org/attributes/hx-select/ */
  @as("hx-select")
  hxSelect?: string,
  @as("data-hx-select")
  rawHxSelect?: string,
  /** https://htmx.org/attributes/hx-select-oob/ */
  @as("hx-select-oob")
  hxSelectOob?: string,
  @as("data-hx-select-oob")
  rawHxSelectOob?: string,
  /** https://htmx.org/attributes/hx-params/ */
  @as("hx-params")
  hxParams?: Params.t,
  @as("data-hx-params")
  rawHxParams?: string,
  /** https://htmx.org/attributes/hx-prompt/ */
  @as("hx-prompt")
  hxPrompt?: string,
  @as("data-hx-prompt")
  rawHxPrompt?: string,
  /** https://htmx.org/attributes/hx-validate/ */
  @as("hx-validate")
  hxValidate?: bool,
  /** https://htmx.org/attributes/hx-confirm/ */
```

```
  @as("hx-confirm")
  hxConfirm?: string,
  /** https://htmx.org/attributes/hx-disable/ */
  @as("hx-disable")
  hxDisable?: bool,
  /** https://htmx.org/attributes/hx-encoding/ */
  @as("hx-encoding")
  hxEncoding?: Encoding.t,
  @as("data-hx-encoding")
  rawHxEncoding?: string,
  /** https://htmx.org/attributes/hx-indicator/ */
  @as("hx-indicator")
  hxIndicator?: Indicator.t,
  @as("data-hx-indicator")
  rawHxIndicator?: string,
  /** https://htmx.org/attributes/hx-history/ */
  @as("hx-history")
  hxHistory?: bool,
  /** https://htmx.org/attributes/hx-history-elt/ */
  @as("hx-history-elt")
  hxHistoryElt?: bool,
  /** https://htmx.org/attributes/hx-include/ */
  @as("hx-include")
  hxInclude?: string,
  /** https://htmx.org/attributes/hx-headers/ */
  @as("hx-headers")
  hxHeaders?: Headers.t,
  @as("data-hx-headers")
  rawHxHeaders?: string,
  /** https://htmx.org/attributes/hx-sync/ */
  @as("hx-sync")
  hxSync?: Sync.t,
  @as("data-hx-sync")
  rawHxSync?: string,
  /** https://htmx.org/attributes/hx-vals/ */
  @as("hx-vals")
  hxVals?: Vals.t,
  @as("data-hx-vals")
  rawHxVals?: string,
  /** https://htmx.org/attributes/hx-preserve/ */
  @as("hx-preserve")
  hxPreserve?: bool,
  /** https://htmx.org/attributes/hx-disinherit/ */
  @as("hx-disinherit")
  hxDisinherit?: Disinherit.t,
  @as("data-hx-disinherit")
  rawHxDisinherit?: string,
  /** https://htmx.org/attributes/hx-target/  TODO */
  @as("hx-target")
  hxTarget?: Target.t,
  @as("data-hx-target")
  rawHxTarget?: string,
}
```

========== End file

brightid--deploycommands

Rescript v10

Repo: https://github.com/ShenaniganDApp/brightid-discord-bot

========== Start file package.json (part or full code)

```
{
  "name": "root",
  "private": true,
  "devDependencies": {
    "patch-package": "^6.4.7"
  },
  "dependencies": {
    "@rescript/core": "^0.2.0",
    "brightid_sdk": "^1.0.1",
    "canvas": "^2.9.0",
    "concurrently": "^7.1.0",
    "dotenv": "^8.2.0",
    "find-up": "^6.3.0",
    "rescript": "^10.1.0-rc.5",
    "rescript-discordjs": "^0.3.0",
    "rescript-nodejs": "^14.3.1",
    "uuid": "^8.3.0"
  },
  "scripts": {
    "bot": "yarn workspace @brightidbot/bot",
```

```
    "web": "yarn workspace @brightidbot/web",
    "utils": "yarn workspace @brightidbot/utils",
    "scripts": "yarn workspace @brightidbot/scripts",
    "shared": "yarn workspace @brightidbot/shared",
    "re:build": "yarn shared re:build && yarn utils re:build && yarn bot re:build && yarn web re:build && yarn s
cripts re:build"
  },
  "workspaces": [
    "apps/*",
    "packages/*"
  ]
}
```

========== End file

========== Start file DeployCommands.res

```
open Promise
open Discord

exception DeployCommandsError(string)
module Rest = {
  type t
  @module("@discordjs/rest") @new external make: {"version": int} => t = "REST"
  @send external setToken: (t, string) => t = "setToken"
  @send
  external put: (t, string, {"body": array<SlashCommandBuilder.json>}) => promise<unit> = "put"
  @send
  external delete: (t, string) => promise<unit> = "delete"
}

module Routes = {
  type t
  @module("discord-api-types/v9") @scope("Routes")
  external applicationCommands: (~clientId: string) => string = "applicationCommands"
  @module("discord-api-types/v9") @scope("Routes")
  external applicationCommand: (~clientId: string, ~commandId: string) => string =
    "applicationCommand"
}

Env.createEnv()

let envConfig = Env.getConfig()
let envConfig = switch envConfig {
| Ok(config) => config
| Error(err) => err->Env.EnvError->raise
}

let token = envConfig["discordApiToken"]
let clientId = envConfig["discordClientId"]

// @TODO: Shouldn't need to hardcode each command, instaed loop through files
let helpCommand = Commands_Help.data->SlashCommandBuilder.toJSON
let verifyCommand = Commands_Verify.data->SlashCommandBuilder.toJSON
let inviteCommand = Commands_Invite.data->SlashCommandBuilder.toJSON

let commands = [helpCommand, verifyCommand, inviteCommand]

let rest = Rest.make({"version": 9})->Rest.setToken(token)

rest
->Rest.put(Routes.applicationCommands(~clientId), {"body": commands})
->thenResolve(() => Console.log("Successfully registered application commands."))
->catch(e => {
  switch e {
  | DeployCommandsError(msg) => Console.error("Deploy Commands Error:" ++ msg)
  | Exn.Error(obj) =>
    switch Exn.message(obj) {
    | Some(msg) => Console.error("Deploy Commands Error: " ++ msg)
    | None => Console.error("Must be some non-error value")
    }
  | _ => Console.error("Some unknown error")
  }
  resolve()
})
->ignore

// delete guilds command
// rest
// ->Rest.delete(Routes.applicationCommand(~clientId, ~commandId="981007485634748511"))
// ->then(_ => {
//   Console.log("Successfully deleted guilds command.")->resolve
// })
// ->catch(e => {
//   Console.log(e)
//   resolve()
```

```
// })
// ->ignore
```

========== End file

catala-dsfr--dsfr

Rescript v10

Repo: https://github.com/CatalaLang/catala-dsfr

========== Start file package.json (part or full code)

```
{
  "name": "catala-dsfr",
  "version": "0.1.1",
  "repository": "https://github.com/CatalaLang/catala-dsfr",
  "scripts": {
    "clean": "rescript clean -with-deps",
    "build": "yarn run pre && vite build",
    "deploy": "yarn run build --base=/demos/catala/ && rsync -rv --delete-before dist/*",
    "serve": "vite preview",
    "dev": "yarn run pre && vite",
    "re:build": "rescript build -with-deps",
    "re:watch": "yarn run pre && rescript build -w -with-deps",
    "assets": "rsync -r node_modules/@catala-lang/catala-web-assets/assets/* assets",
    "postinstall": "copy-dsfr-to-public",
    "pre": "yarn run re:build && only-include-used-icons && yarn run assets"
  },
  "keywords": [
    "rescript"
  ],
  "author": "Emile Rolley <emile.rolley@tuta.io>",
  "license": "Apache-2.0",
  "dependencies": {
    "@catala-lang/catala-explain": "^0.2.2",
    "@catala-lang/catala-web-assets": "^0.8.9",
    "@catala-lang/french-law": "^0.8.3-b.3",
    "@catala-lang/rescript-catala": "^0.8.1-b.0",
    "@codegouvfr/react-dsfr": "^0.78.2",
    "@rescript/core": "^0.5.0",
    "@rescript/react": "^0.11.0",
    "@rjsf/core": "^5.1.0",
    "@rjsf/utils": "^5.1.0",
    "@rjsf/validator-ajv8": "^5.1.0",
    "file-saver": "^2.0.5",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-loader-spinner": "^5.4.5",
    "rescript-docx": "^0.1.5",
    "tslib": "^2.6.2"
  },
  "devDependencies": {
    "@jihchi/vite-plugin-rescript": "^5.1.0",
    "@originjs/vite-plugin-commonjs": "^1.0.3",
    "@vitejs/plugin-react": "^3.1.0",
    "jsdom": "^21.1.0",
    "rescript": "^10.1.4",
    "tailwindcss": "^3.2.6",
    "vite": "^4.4.9"
  }
}
```

========== End file

========== Start file Dsfr.res

```
type linkProps = {"href": string, "title": string}

module Spa = {
  type startReactDsfrParams<'props> = {
    defaultColorScheme: [#light | #dark | #system],
    verbose?: bool,
    @as("Link") link: 'props => React.element,
    useLang?: unit => [#fr | #en],
  }

  @module("@codegouvfr/react-dsfr/spa")
  external startReactDsfr: startReactDsfrParams<'props> => unit = "startReactDsfr"
}

module Badge = {
  type severity = [#success | #info | #warning | #error]
  type tag = [#span | #div | #p]
```

```
  @react.component @module("@codegouvfr/react-dsfr/Badge")
  external make: (
    ~className: string=?,
    ~children: React.element,
    ~noIcon: bool=?,
    ~small: bool=?,
    ~severity: severity=?,
    @as("as") ~as_: tag=?,
  ) => React.element = "default"
}

module Breadcrumb = {
  type segment = {label: string, linkProps: linkProps}
  @react.component @module("@codegouvfr/react-dsfr/Breadcrumb")
  external make: (
    ~id: string=?,
    ~className: string=?,
    ~homeLinkProps: linkProps=?,
    ~segments: array<segment>,
    ~currentPageLabel: string=?,
  ) => React.element = "default"
}

module Button = {
  type options = {
    disabled?: bool,
    iconId?: string,
    iconPosition?: string,
    onClick: JsxEvent.Mouse.t => unit,
    priority?: string,
    size?: string,
    children: React.element,
  }

  @react.component @module("@codegouvfr/react-dsfr/Button")
  external make: (
    ~children: React.element,
    ~disabled: bool=?,
    ~iconId: string=?,
    ~iconPosition: string=?,
    ~onClick: JsxEvent.Mouse.t => unit,
    ~priority: string=?,
    ~size: string=?,
  ) => React.element = "default"
}

module ButtonsGroup = {
  @react.component @module("@codegouvfr/react-dsfr/ButtonsGroup")
  external make: (
    ~alignment: string=?,
    ~buttonsSize: string=?,
    ~buttonsIconPosition: string=?,
    ~buttonsEquisized: bool=?,
    ~buttons: array<Button.options>,
    ~inlineLayoutWhen: string=?,
    ~className: string=?,
  ) => React.element = "default"
}

module CallOut = {
  @react.component @module("@codegouvfr/react-dsfr/CallOut")
  external make: (~title: string=?, ~children: React.element, ~iconId: string=?) => React.element =
    "default"
}

module Card = {
  @react.component @module("@codegouvfr/react-dsfr/Card")
  external make: (
    ~title: string,
    ~desc: string,
    ~linkProps: linkProps,
    ~enlargeLink: bool=?,
    ~size: string=?,
  ) => React.element = "default"
}

module Header = {
  @react.component @module("@codegouvfr/react-dsfr/Header")
  external make: (
    ~brandTop: React.element=?,
    ~homeLinkProps: linkProps,
    ~serviceTagline: string,
    ~operatorLogo: {"alt": string, "imgUrl": string, "orientation": string}=?,
    ~serviceTitle: React.element,
  ) => React.element = "default"
}
```

```
module Footer = {
  @react.component @module("@codegouvfr/react-dsfr/Footer")
  external make: (
    ~accessibility: string,
    ~brandTop: React.element=?,
    ~contentDescription: React.element=?,
    ~homeLinkProps: linkProps=?,
    ~bottomItems: array<'button>=?,
    ~license: React.element=?,
  ) => React.element = "default"
}

module Display = {
  @module("@codegouvfr/react-dsfr/Display")
  external headerFooterDisplayItem: 'button = "headerFooterDisplayItem"
}

module Notice = {
  @react.component @module("@codegouvfr/react-dsfr/Notice")
  external make: (~title: string, ~isClosable: bool=?) => React.element = "default"
}
```

========== End file

brightid--remixauth

Rescript v10

Repo: https://github.com/ShenaniganDApp/brightid-discord-bot

========== Start file package.json (part or full code)

```
{
  "name": "root",
  "private": true,
  "devDependencies": {
    "patch-package": "^6.4.7"
  },
  "dependencies": {
    "@rescript/core": "^0.2.0",
    "brightid_sdk": "^1.0.1",
    "canvas": "^2.9.0",
    "concurrently": "^7.1.0",
    "dotenv": "^8.2.0",
    "find-up": "^6.3.0",
    "rescript": "^10.1.0-rc.5",
    "rescript-discordjs": "^0.3.0",
    "rescript-nodejs": "^14.3.1",
    "uuid": "^8.3.0"
  },
  "scripts": {
    "bot": "yarn workspace @brightidbot/bot",
    "web": "yarn workspace @brightidbot/web",
    "utils": "yarn workspace @brightidbot/utils",
    "scripts": "yarn workspace @brightidbot/scripts",
    "shared": "yarn workspace @brightidbot/shared",
    "re:build": "yarn shared re:build && yarn utils re:build && yarn bot re:build && yarn web re:build && yarn s
cripts re:build"
  },
  "workspaces": [
    "apps/*",
    "packages/*"
  ]
}
```

========== End file

========== Start file RemixAuth.res

```
module User = {
  type t
  type profile
  @get external getAccessToken: t => string = "accessToken"
  @get external getProfile: t => profile = "profile"
  @get external getId: profile => string = "id"
}

module DiscordStrategy = {
  type t
  module CreateDiscordStategyOptions = {
    type t
    @obj
    external make: (
      ~clientID: string,
```

```
      ~clientSecret: string,
      ~callbackURL: string,
      // Provide all the scopes you want as an array
      ~scope: array<string>,
      unit,
    ) => t = ""
  }

  // module CreateVerifyFunctionOptions = {
  //   type t
  //   @obj
  //   external make: (
  //     ~accessToken: string,
  //     ~refreshToken: string,
  //     ~extraParams: 'a,
  //     ~profile: 'b,
  //     unit,
  //   ) => t = ""
  // }
  type verifyFunctionParams<'a, 'b> = {
    accessToken: string,
    refreshToken: string,
    extraParams: 'a,
    profile: 'b,
  }

  @module("remix-auth-socials") @new
  external make: (
    CreateDiscordStategyOptions.t,
    verifyFunctionParams<'a, 'b> => Js.Promise.t<'a>,
  ) => t = "DiscordStrategy"
}

// module SocialsProvider = {
//   type t = [#Discord]

// }

module CreateAuthenticateOptions = {
  type t

  @obj external make: (~successRedirect: string=?, ~failureRedirect: string=?, unit) => t = ""
}

module Authenticator = {
  type t
  @module("remix-auth") @new external make: Remix.SessionStorage.t => t = "Authenticator"
  @send external use: (t, DiscordStrategy.t) => unit = "use"
  @send
  external authenticate: (t, string, Webapi.Fetch.Request.t) => Js.Promise.t<User.t> =
    "authenticate"

  @send
  external authenticateWithOptions: (
    t,
    string,
    Webapi.Fetch.Request.t,
    ~options: CreateAuthenticateOptions.t,
  ) => Js.Promise.t<User.t> = "authenticate"
  @send
  external isAuthenticated: (t, Webapi.Fetch.Request.t) => Js.Promise.t<Js.Nullable.t<User.t>> =
    "isAuthenticated"
  @send
  external isAuthenticatedWithOptions: (
    t,
    Webapi.Fetch.Request.t,
    ~options: CreateAuthenticateOptions.t,
  ) => Js.Promise.t<Js.Nullable.t<User.t>> = "isAuthenticated"

  @send
  external logout: (t, Webapi.Fetch.Request.t, ~options: 'option) => Js.Promise.t<unit> = "logout"
}
```

========== End file

catala-dsfr--sourcecode

Rescript v10

Repo: https://github.com/CatalaLang/catala-dsfr

========== Start file package.json (part or full code)

```
{
```

```
  "name": "catala-dsfr",
  "version": "0.1.1",
  "repository": "https://github.com/CatalaLang/catala-dsfr",
  "scripts": {
    "clean": "rescript clean -with-deps",
    "build": "yarn run pre && vite build",
    "deploy": "yarn run build --base=/demos/catala/ && rsync -rv --delete-before dist/*",
    "serve": "vite preview",
    "dev": "yarn run pre && vite",
    "re:build": "rescript build -with-deps",
    "re:watch": "yarn run pre && rescript build -w -with-deps",
    "assets": "rsync -r node_modules/@catala-lang/catala-web-assets/assets/* assets",
    "postinstall": "copy-dsfr-to-public",
    "pre": "yarn run re:build && only-include-used-icons && yarn run assets"
  },
  "keywords": [
    "rescript"
  ],
  "author": "Emile Rolley <emile.rolley@tuta.io>",
  "license": "Apache-2.0",
  "dependencies": {
    "@catala-lang/catala-explain": "^0.2.2",
    "@catala-lang/catala-web-assets": "^0.8.9",
    "@catala-lang/french-law": "^0.8.3-b.3",
    "@catala-lang/rescript-catala": "^0.8.1-b.0",
    "@codegouvfr/react-dsfr": "^0.78.2",
    "@rescript/core": "^0.5.0",
    "@rescript/react": "^0.11.0",
    "@rjsf/core": "^5.1.0",
    "@rjsf/utils": "^5.1.0",
    "@rjsf/validator-ajv8": "^5.1.0",
    "file-saver": "^2.0.5",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-loader-spinner": "^5.4.5",
    "rescript-docx": "^0.1.5",
    "tslib": "^2.6.2"
  },
  "devDependencies": {
    "@jihchi/vite-plugin-rescript": "^5.1.0",
    "@originjs/vite-plugin-commonjs": "^1.0.3",
    "@vitejs/plugin-react": "^3.1.0",
    "jsdom": "^21.1.0",
    "rescript": "^10.1.4",
    "tailwindcss": "^3.2.6",
    "vite": "^4.4.9"
  }
}
```

========== End file

========== Start file SourceCode.res

```
%%raw(`import  "../css/catala-code.css"`)
%%raw(`import  "../css/syntax-highlighting.css"`)

/*
[scrollToAndHighlightLineNum(parentElem, ids)] scrolls into the corresponding
Catala code line of [ids] inside the [parentElem] DOM element and highlight the
line numbers.
*/

module HtmlSourceCodeLazy = {
  @react.component @module("../components/HtmlSourceCodeLazy.tsx")
  external make: (~html: string, ~hash: string) => React.element = "default"
}

@react.component
let make = (~html: option<string>, ~simulatorUrl: string) => {
  let {hash} = Nav.getCurrentURL()

  switch html {
  | Some(html) =>
    <div className="fr-container">
      <Button.RightAlign
        props={
          iconId: "fr-icon-equalizer-line",
          iconPosition: "left",
          priority: "tertiary",
          size: "medium",
          onClick: {_ => `/${simulatorUrl}`->Nav.goTo},
          children: {"AccÃ©der au simulateur"->React.string},
        }
      />
      <React.Suspense fallback={Spinners.loader}>
        <HtmlSourceCodeLazy html hash />
```

```
      </React.Suspense>
    </div>
  | None =>
    ()
    <div>
      <p> {"No source code available for this snippet."->React.string} </p>
    </div>
  }
}
```

========== End file

res-x--devs

Rescript v11

Repo: https://github.com/zth/res-x

========== Start file package.json (part or full code)

```
{
  "name": "rescript-x",
  "version": "0.1.0-alpha.7",
  "scripts": {
    "res:build": "rescript",
    "res:clean": "rescript clean",
    "res:dev": "rescript build -w"
  },
  "keywords": [
    "rescript"
  ],
  "files": [
    "README.md",
    "CHANGELOG.md",
    "rescript.json",
    "src/**/*",
    "res-x-vite-plugin.mjs"
  ],
  "author": "Gabriel Nordeborn",
  "license": "MIT",
  "peerDependencies": {
    "rescript": ">=11.0.0-rc.5",
    "@rescript/core": ">=0.5.0",
    "vite": ">=4.4.11",
    "rescript-bun": ">=0.1.0"
  },
  "devDependencies": {
    "@rescript/core": "^0.5.0",
    "fast-glob": "^3.3.1",
    "rescript": "11.0.0-rc.5",
    "rescript-bun": "0.1.0"
  },
  "dependencies": {
    "fast-glob": "^3.3.1"
  }
}
```

========== End file

========== Start file Dev.res

```
module React = ResX__React
module ReactDOM = ResX__ReactDOM

let getScript = (~port) =>
  `(() => {
let hasMadeInitialConnection = false;
let timeout;
let socket = null;
let reconnectInterval;
let debugging = true;

let debug = (...msg) => {
  if (debugging) {
    console.log(...msg)
  }
}

function reload() {
  clearTimeout(timeout)
  timeout = setTimeout(() => {
    window.location.reload()
  }, 200)
}
```

```
function connect() {
  clearInterval(reconnectInterval)
  reconnectInterval = setInterval(() => {
    if (socket == null || socket.readyState === 2 || socket.readyState === 3) {
      bootSocket()
    } else if (socket != null && (socket.readyState === 0 || socket.readyState === 1)) {
      clearInterval(reconnectInterval)
    }
  }, 200)
}

function updateContent() {
  fetch(document.location.href).then(async res => {
    let text = await res.text()
    try {
      let domParser = new DOMParser()
      let fromDom = document.documentElement
      let toDom = domParser.parseFromString(text, "text/html").querySelector("html")
      morphdom(fromDom, toDom, {
        onBeforeElUpdated: function(fromEl, toEl) {
          if (fromEl.isEqualNode(toEl)) {
            return false;
          }

          if (fromEl.tagName === 'INPUT') {
            if (fromEl.type === 'checkbox' || fromEl.type === 'radio') {
              toEl.checked = fromEl.checked;
            } else {
              toEl.value = fromEl.value;
            }
          } else if (fromEl.tagName === 'TEXTAREA') {
            toEl.value = fromEl.value;
          }

          if (fromEl.tagName === 'SELECT') {
            toEl.selectedIndex = fromEl.selectedIndex;
          }

          return true;
        }
      })
      debug("[dev] Content reloaded.")
    } catch(e) {
      console.warn("[dev] Error morphing DOM. Doing full reload.")
      console.error(e)
      document.documentElement.innerHTML = text

    }
  })
}

function bootSocket() {
  socket = new WebSocket("ws://localhost:${(port + 1)->Int.toString}")
  socket.addEventListener("close", event => {
    debug("[dev] Server restarting")
    if (event.isTrusted) {
      socket = null
      connect()
    }
  })
  socket.addEventListener("open", event => {
    debug("[dev] Server connection opened.")
    if (hasMadeInitialConnection) {
      updateContent()
    }
    hasMadeInitialConnection = true
  })
}

bootSocket()
})()
`


@react.component
let make = (~port=4444) => {
  if BunUtils.isDev {
    [
      <script dangerouslySetInnerHTML={{"__html": getScript(~port)}} />,
      <script src="https://unpkg.com/morphdom/dist/morphdom-umd.js" />,
      <script type_="module" src="http://localhost:9000/@vite/client" />,
    ]->H.array
  } else {
    H.null
  }
}
```

========== End file

brightid--root

Rescript v10

Repo: https://github.com/ShenaniganDApp/brightid-discord-bot

========== Start file package.json (part or full code)

```
{
  "name": "root",
  "private": true,
  "devDependencies": {
    "patch-package": "^6.4.7"
  },
  "dependencies": {
    "@rescript/core": "^0.2.0",
    "brightid_sdk": "^1.0.1",
    "canvas": "^2.9.0",
    "concurrently": "^7.1.0",
    "dotenv": "^8.2.0",
    "find-up": "^6.3.0",
    "rescript": "^10.1.0-rc.5",
    "rescript-discordjs": "^0.3.0",
    "rescript-nodejs": "^14.3.1",
    "uuid": "^8.3.0"
  },
  "scripts": {
    "bot": "yarn workspace @brightidbot/bot",
    "web": "yarn workspace @brightidbot/web",
    "utils": "yarn workspace @brightidbot/utils",
    "scripts": "yarn workspace @brightidbot/scripts",
    "shared": "yarn workspace @brightidbot/shared",
    "re:build": "yarn shared re:build && yarn utils re:build && yarn bot re:build && yarn web re:build && yarn s
cripts re:build"
  },
  "workspaces": [
    "apps/*",
    "packages/*"
  ]
}
```

========== End file

========== Start file Root.res

```
%%raw(`import rainbowKit from "@rainbow-me/rainbowkit/styles.css"`)
%%raw(`import proSidebar from "react-pro-sidebar/dist/css/styles.css"`)
%%raw(`
import {
  getDefaultWallets,
} from "@rainbow-me/rainbowkit";
import { createClient, configureChains } from "wagmi"
import { mainnet} from 'wagmi/chains'
import { alchemyProvider } from 'wagmi/providers/alchemy'
import { publicProvider } from 'wagmi/providers/public'
import { jsonRpcProvider } from '@wagmi/core/providers/jsonRpc'

`)

module LodashMerge = {
  @module("lodash.merge") external merge: ('a, 'b) => 'a = "default"
}

@live
let meta = () =>
  {
    "charset": "utf-8",
    "title": "Bright ID Discord Command Center",
    "viewport": "width=device-width,initial-scale=1.0, maximum-scale=1.0, user-scalable=no",
  }

@live
let links = () => {
  [
    {
      "rel": "stylesheet",
      "href": %raw(`require("./styles/app.css")`),
    },
    {
      "rel": "stylesheet",
      "href": %raw(`rainbowKit`),
    },
```

```
    {
      "rel": "stylesheet",
      "href": %raw(`proSidebar`),
    },
  ]
}

let _idChain = {
  "id": 74,
  "name": "ID Chain",
  "nativeCurrency": {"name": "Eidi", "symbol": "EIDI", "decimals": 18},
  "rpcUrls": {
    "default": {
      "http": "https://idchain.one/rpc",
    },
  },
  "blockExplorers": [
    {
      "name": "Blockscout",
      "url": "https://explorer.idchain.one/",
    },
  ],
}

type loaderData = {maybeUser: option<RemixAuth.User.t>, rateLimited: bool}

@live
let loader: Remix.loaderFunction<loaderData> = ({request}) => {
  open DiscordServer
  open Promise

  AuthServer.authenticator
  ->RemixAuth.Authenticator.isAuthenticated(request)
  ->then(user => {
    {maybeUser: user->Js.Nullable.toOption, rateLimited: false}->resolve
  })
  ->catch(error => {
    switch error {
    | DiscordRateLimited => {maybeUser: None, rateLimited: true}->resolve
    | _ => {maybeUser: None, rateLimited: false}->resolve
    }
  })
}

let myTheme = LodashMerge.merge(
  RainbowKit.Themes.darkTheme(),
  {"colors": {"accentColor": "#ed7a5c"}},
)

let chainConfig = %raw(`configureChains(
    [mainnet, _idChain],
    [
      alchemyProvider({
        apiKey: "Klcw92W_rTgV55TL0zq972TFXTI1FieU",
        stallTimeout: 5_000
      }),
      jsonRpcProvider({
        rpc: (chain)  => ({ http: chain.rpcUrls.default.http })
      }),

    ]

  )`)

let _defaultWallets = %raw(`getDefaultWallets({
    appName: "Bright ID Discord Command Center",
    chains: chainConfig.chains,
  })`)

let wagmiClient = %raw(`createClient({
    autoConnect: true,
    connectors: _defaultWallets.connectors,
    provider: chainConfig.provider,
  })`)

type state = {
  userGuilds: array<Types.oauthGuild>,
  botGuilds: array<Types.oauthGuild>,
  after: option<string>,
  loadingGuilds: bool,
  wagmiClient: option<Wagmi.client>,
  chains: option<array<Wagmi.chain>>,
}

let state = {
  userGuilds: [],
  botGuilds: [],
```

```
  after: Some("0"),
  loadingGuilds: true,
  wagmiClient: Some(wagmiClient),
  chains: Some(chainConfig["chains"]),
}

type actions =
  | AddBotGuilds(array<Types.oauthGuild>)
  | UserGuilds(array<Types.oauthGuild>)
  | SetAfter(option<string>)
  | SetLoadingGuilds(bool)
  | SetWagmiClient(option<Wagmi.client>)
  | SetChains(option<array<Wagmi.chain>>)

let reducer = (state, action) =>
  switch action {
  | AddBotGuilds(newBotGuilds) => {
      ...state,
      botGuilds: state.botGuilds->Belt.Array.concat(newBotGuilds),
    }
  | UserGuilds(userGuilds) => {...state, userGuilds}
  | SetAfter(after) => {...state, after}
  | SetLoadingGuilds(loadingGuilds) => {...state, loadingGuilds}
  | SetWagmiClient(wagmiClient) => {...state, wagmiClient}
  | SetChains(chains) => {...state, chains}
  }

@live @react.component
let default = () => {
  open RainbowKit
  let {maybeUser, rateLimited} = Remix.useLoaderData()
  let (isSidebarVisible, setIsSidebarVisible) = React.useState(_ => false)

  let fetcher = Remix.useFetcher()

  let (state, dispatch) = React.useReducer(reducer, state)

  React.useEffect1(() => {
    open Remix
    switch state.after {
    | None => ()
    | Some(after) =>
      switch fetcher->Fetcher._type {
      | "init" =>
        fetcher->Fetcher.load(~href=`/Root_FetchGuilds?after=${after}`)
        SetLoadingGuilds(true)->dispatch

      | "done" =>
        switch fetcher->Remix.Fetcher.data->Js.Nullable.toOption {
        | None =>
          SetLoadingGuilds(false)->dispatch
          None->SetAfter->dispatch
        | Some(data) =>
          switch data["userGuilds"] {
          | [] => ()
          | _ => data["userGuilds"]->UserGuilds->dispatch
          }
          switch data["botGuilds"] {
          | [] => None->SetAfter->dispatch
          | _ => data["botGuilds"]->AddBotGuilds->dispatch
          }
          if state.after === data["after"] {
            None->SetAfter->dispatch
            SetLoadingGuilds(false)->dispatch
          } else {
            data["after"]->SetAfter->dispatch
            fetcher->Fetcher.load(~href=`/Root_FetchGuilds?after=${data["after"]}`)
          }
        }
      | _ => ()
      }
    }

    None
  }, [fetcher])

  let guilds =
    state.userGuilds->Js.Array2.filter(userGuild =>
      state.botGuilds->Js.Array2.findIndex(botGuild => botGuild.id === userGuild.id) !== -1
    )

  let handleIsSidebarVisible = value => {
    setIsSidebarVisible(_prev => value)
  }

  <html>
    <head>
```

```
      <meta charSet="utf-8" />
      <meta name="viewport" content="width=device-width,initial-scale=1" />
      <Remix.Meta />
      <Remix.Links />
    </head>
    <body className="h-screen w-screen bg-dark">
      {switch (state.wagmiClient, state.chains) {
      | (Some(client), Some(chains)) =>
        <Wagmi.WagmiConfig client={client}>
          <RainbowKitProvider chains={chains} theme={myTheme}>
            <div className="flex h-screen w-screen">
              {switch maybeUser {
              | None => <> </>
              | Some(_) =>
                <Sidebar
                  isSidebarVisible handleIsSidebarVisible guilds loadingGuilds={state.loadingGuilds}
                />
              }}
              <Remix.Outlet
                context={{
                  "isSidebarVisible": isSidebarVisible,
                  "handleIsSidebarVisible": handleIsSidebarVisible,
                  "rateLimited": rateLimited,
                  "guilds": guilds,
                }}
              />
            </div>
          </RainbowKitProvider>
        </Wagmi.WagmiConfig>
      | _ => <> </>
      }}
      <Remix.ScrollRestoration />
      <Remix.Scripts />
      {if Remix.process["env"]["NODE_ENV"] === "development" {
        <Remix.LiveReload />
      } else {
        React.null
      }}
    </body>
  </html>
}

%%raw(`
export function ErrorBoundary({ error }) {
  console.error(error);
  return (
    <html>
      <head>
        <title>Oh no!</title>
        <React$1.Meta />
        <React$1.Links />
      </head>
      <body>
        <p className="text-center">Something went wrong!</p>
        <p className="text-center">BrightID command center is still in Beta. Try reloading the page!</p>
        <React$1.Scripts />
      </body>
    </html>
  );
}`)
```

========== End file

res-x--bunutils

Rescript v11

Repo: https://github.com/zth/res-x

========== Start file package.json (part or full code)

```
{
  "name": "rescript-x",
  "version": "0.1.0-alpha.7",
  "scripts": {
    "res:build": "rescript",
    "res:clean": "rescript clean",
    "res:dev": "rescript build -w"
  },
  "keywords": [
    "rescript"
  ],
  "files": [
    "README.md",
```

```
    "CHANGELOG.md",
    "rescript.json",
    "src/**/*",
    "res-x-vite-plugin.mjs"
  ],
  "author": "Gabriel Nordeborn",
  "license": "MIT",
  "peerDependencies": {
    "rescript": ">=11.0.0-rc.5",
    "@rescript/core": ">=0.5.0",
    "vite": ">=4.4.11",
    "rescript-bun": ">=0.1.0"
  },
  "devDependencies": {
    "@rescript/core": "^0.5.0",
    "fast-glob": "^3.3.1",
    "rescript": "11.0.0-rc.5",
    "rescript-bun": "0.1.0"
  },
  "dependencies": {
    "fast-glob": "^3.3.1"
  }
}
```

========== End file

========== Start file BunUtils.res

```
external process: 'process = "process"

let isDev = process["env"]["NODE_ENV"] !== "production"

type globConfig = {
  dot?: bool,
  cwd?: string,
}

@module("fast-glob")
external glob: (array<string>, globConfig) => promise<array<string>> = "glob"

let loadStaticFiles = async (~root=?) => {
  await glob(
    switch isDev {
    | true => ["public/**/*", "assets/**/*"]
    | false => ["dist/**/*"]
    },
    {
      dot: true,
      cwd: switch root {
      | None => process["cwd"]()
      | Some(cwd) => cwd
      },
    },
  )
}

let staticFiles = ref(None)

let serveStaticFile = async request => {
  open Bun

  let staticFiles = switch staticFiles.contents {
  | None =>
    let files = await loadStaticFiles()
    let files =
      files
      ->Array.map(f => {
        (
          switch isDev {
          | true if f->String.startsWith("public/") => f->String.sliceToEnd(~start=7)
          | false if f->String.startsWith("dist/") => f->String.sliceToEnd(~start=5)
          | _ => f
          },
          f,
        )
      })
      ->Map.fromArray
    staticFiles := Some(files)
    files
  | Some(s) => s
  }

  let url = request->Request.url->URL.make
  let pathname = url->URL.pathname

  let path = pathname->String.split("/")->Array.filter(p => p !== "")
```

```
  let joined = path->Array.joinWith("/")

  switch staticFiles->Map.get(joined) {
  | None => None
  | Some(fileLoc) =>
    let bunFile = Bun.file("./" ++ fileLoc)

    Some(
      switch bunFile->BunFile.size {
      | 0. => Response.make("", ~options={status: 404})
      | _ => Response.makeFromFile(bunFile)
      },
    )
  }
}

let runDevServer = (~port) => {
  let _devServer = Bun.serveWithWebSocket({
    port: port + 1,
    development: true,
    websocket: {
      open_: _v => {
        ()
      },
    },
    fetch: async (request, server) => {
      open Bun

      if server->Server.upgrade(request) {
        Response.defer
      } else {
        Response.make("", ~options={status: 404})
      }
    },
  })
}

module URLSearchParams = {
  let copy = search =>
    URLSearchParams.makeWithInit(
      search
      ->URLSearchParams.entries
      ->Dict.fromIterator
      ->Object,
    )
}
```

========== End file

========== Start file BunUtils.resi

```
let serveStaticFile: Request.t => promise<option<Response.t>>

let runDevServer: (~port: int) => unit

let isDev: bool

module URLSearchParams: {
  let copy: URLSearchParams.t => URLSearchParams.t
}
```

========== End file

brightid--brightid

Rescript v10

Repo: https://github.com/ShenaniganDApp/brightid-discord-bot

========== Start file package.json (part or full code)

```
{
  "name": "root",
  "private": true,
  "devDependencies": {
    "patch-package": "^6.4.7"
  },
  "dependencies": {
    "@rescript/core": "^0.2.0",
    "brightid_sdk": "^1.0.1",
    "canvas": "^2.9.0",
    "concurrently": "^7.1.0",
    "dotenv": "^8.2.0",
    "find-up": "^6.3.0",
```

```
    "rescript": "^10.1.0-rc.5",
    "rescript-discordjs": "^0.3.0",
    "rescript-nodejs": "^14.3.1",
    "uuid": "^8.3.0"
  },
  "scripts": {
    "bot": "yarn workspace @brightidbot/bot",
    "web": "yarn workspace @brightidbot/web",
    "utils": "yarn workspace @brightidbot/utils",
    "scripts": "yarn workspace @brightidbot/scripts",
    "shared": "yarn workspace @brightidbot/shared",
    "re:build": "yarn shared re:build && yarn utils re:build && yarn bot re:build && yarn web re:build && yarn s
cripts re:build"
  },
  "workspaces": [
    "apps/*",
    "packages/*"
  ]
}
```

========== End file

========== Start file BrightId.res

```
@module("brightid_sdk")
external verifyContextId: (
  ~context: string,
  ~contextId: string,
  ~nodeUrl: string=?,
  unit,
) => Js.Promise.t<Js.Json.t> = "verifyContextId"

@module("brightid_sdk")
external generateDeeplink: (
  ~context: string,
  ~contextId: string,
  ~nodeUrl: string=?,
  unit,
) => string = "generateDeeplink"
```

========== End file

res-x--resxclient

Rescript v11

Repo: https://github.com/zth/res-x

========== Start file package.json (part or full code)

```
{
  "name": "rescript-x",
  "version": "0.1.0-alpha.7",
  "scripts": {
    "res:build": "rescript",
    "res:clean": "rescript clean",
    "res:dev": "rescript build -w"
  },
  "keywords": [
    "rescript"
  ],
  "files": [
    "README.md",
    "CHANGELOG.md",
    "rescript.json",
    "src/**/*",
    "res-x-vite-plugin.mjs"
  ],
  "author": "Gabriel Nordeborn",
  "license": "MIT",
  "peerDependencies": {
    "rescript": ">=11.0.0-rc.5",
    "@rescript/core": ">=0.5.0",
    "vite": ">=4.4.11",
    "rescript-bun": ">=0.1.0"
  },
  "devDependencies": {
    "@rescript/core": "^0.5.0",
    "fast-glob": "^3.3.1",
    "rescript": "11.0.0-rc.5",
    "rescript-bun": "0.1.0"
  },
  "dependencies": {
    "fast-glob": "^3.3.1"
```

```
  }
}
```

========== End file

========== Start file ResXClient.res

========== End file ========== Start file ResXClient.res

```
@send
external addEventListener: (Dom.document, string, 'event => unit, ~capturePhase: bool=?) => unit =
  "addEventListener"
external document: Dom.document = "document"

external querySelector: string => Null.t<'element> = "document.querySelector"

type validity = {
  badInput: bool,
  patternMismatch: bool,
  rangeOverflow: bool,
  rangeUnderflow: bool,
  stepMismatch: bool,
  tooLong: bool,
  tooShort: bool,
  typeMismatch: bool,
  valueMissing: bool,
  valid: bool,
}

type attr = {value: string}
type classList = {toggle: string => unit, add: string => unit, remove: string => unit}
type element = {
  "attributes": {"resx-onclick": option<attr>, "resx-validity-message": option<attr>},
  "classList": classList,
  "validity": option<validity>,
  "setCustomValidity": string => unit,
  "remove": unit => unit,
}
type event = {"target": element}

external parseActions: string => array<Client.Actions.action> = "JSON.parse"
external parseValidityMessage: string => Client.ValidityMessage.config = "JSON.parse"

(
  () => {
    let getTarget = (target: Client.Actions.target, this: element): Null.t<element> => {
      switch target {
      | This => Value(this)
      | CssSelector({selector}) => querySelector(selector)
      }
    }

    let handleAction = (action: Client.Actions.action, this) => {
      let target = switch action {
      | ToggleClass({target})
      | RemoveClass({target})
      | AddClass({target})
      | RemoveElement({target}) =>
        getTarget(target, this)
      }

      switch target {
      | Null => ()
      | Value(target) =>
        switch action {
        | ToggleClass({className}) => target["classList"].toggle(className)
        | RemoveClass({className}) => target["classList"].remove(className)
        | AddClass({className}) => target["classList"].add(className)
        | RemoveElement(_) => target["remove"]()
        }
      }
    }

    document->addEventListener("click", (event: event) => {
      let this = event["target"]
      let actions = switch this["attributes"]["resx-onclick"] {
      | None => []
      | Some({value}) => parseActions(value)
      }

      actions->Array.forEach(action => handleAction(action, this))
    })

    document->addEventListener("invalid", ~capturePhase=true, (event: event) => {
```

```
      let this = event["target"]
      switch (this["validity"], this["attributes"]["resx-validity-message"]) {
      | (Some({valid: false} as validity), Some({value})) =>
        let validityMessages = parseValidityMessage(value)
        let messageToSet = switch validity {
        | {badInput: true} => validityMessages.badInput
        | {patternMismatch: true} => validityMessages.patternMismatch
        | {rangeOverflow: true} => validityMessages.rangeOverflow
        | {rangeUnderflow: true} => validityMessages.rangeUnderflow
        | {stepMismatch: true} => validityMessages.stepMismatch
        | {tooLong: true} => validityMessages.tooLong
        | {tooShort: true} => validityMessages.tooShort
        | {typeMismatch: true} => validityMessages.typeMismatch
        | {valueMissing: true} => validityMessages.valueMissing
        | _ => None
        }
        switch messageToSet {
        | None => ()
        | Some(messageToSet) => this["setCustomValidity"](messageToSet)
        }
      | _ => ()
      }
    })

    document->addEventListener("change", (event: event) => {
      let this = event["target"]
      switch this["attributes"]["resx-validity-message"] {
      | Some(_) => this["setCustomValidity"]("")
      | None => ()
      }
    })
  }
)()
```

========== End file

res-x--handlers

Rescript v11

Repo: https://github.com/zth/res-x

========== Start file package.json (part or full code)

```
{
  "name": "rescript-x",
  "version": "0.1.0-alpha.7",
  "scripts": {
    "res:build": "rescript",
    "res:clean": "rescript clean",
    "res:dev": "rescript build -w"
  },
  "keywords": [
    "rescript"
  ],
  "files": [
    "README.md",
    "CHANGELOG.md",
    "rescript.json",
    "src/**/*",
    "res-x-vite-plugin.mjs"
  ],
  "author": "Gabriel Nordeborn",
  "license": "MIT",
  "peerDependencies": {
    "rescript": ">=11.0.0-rc.5",
    "@rescript/core": ">=0.5.0",
    "vite": ">=4.4.11",
    "rescript-bun": ">=0.1.0"
  },
  "devDependencies": {
    "@rescript/core": "^0.5.0",
    "fast-glob": "^3.3.1",
    "rescript": "11.0.0-rc.5",
    "rescript-bun": "0.1.0"
  },
  "dependencies": {
    "fast-glob": "^3.3.1"
  }
}
```

========== End file

========== Start file Handlers.res

```
type htmxHandlerConfig<'ctx> = {
  request: Request.t,
  context: 'ctx,
  headers: Headers.t,
  requestController: RequestController.t,
}

type htmxHandler<'ctx> = htmxHandlerConfig<'ctx> => promise<Jsx.element>

type renderConfig<'ctx> = {
  request: Request.t,
  headers: Headers.t,
  context: 'ctx,
  path: list<string>,
  url: URL.t,
  requestController: RequestController.t,
}

type t<'ctx> = {
  handlers: array<(method, string, htmxHandler<'ctx>)>,
  requestToContext: Request.t => promise<'ctx>,
  asyncLocalStorage: AsyncHooks.AsyncLocalStorage.t<renderConfig<'ctx>>,
}

type hxGet = string
type hxPost = string
type hxPut = string
type hxPatch = string
type hxDelete = string

let make = (~requestToContext) => {
  handlers: [],
  requestToContext,
  asyncLocalStorage: AsyncHooks.AsyncLocalStorage.make(),
}

let useContext = t => t.asyncLocalStorage->AsyncHooks.AsyncLocalStorage.getStoreUnsafe

let defaultRenderTitle = segments => segments->Array.joinWith(" | ")

let renderWithDocType = async (
  el,
  ~requestController: RequestController.t,
  ~renderTitle=defaultRenderTitle,
) => {
  let (content, appendToHead) = await Promise.all2((
    H.renderToString(el),
    requestController->RequestController.getAppendedHeadContent,
  ))

  // TODO: Escape? Hyperons has something

  let appendToHead = switch (appendToHead, requestController->RequestController.getTitleSegments) {
  | (appendToHead, []) => appendToHead
  | (Some(appendToHead), titleSegments) =>
    let titleElement = `<title>${renderTitle(titleSegments)}</title>`
    Some(appendToHead ++ titleElement)
  | (None, titleSegments) => Some(`<title>${renderTitle(titleSegments)}</title>`)
  }

  let content = switch appendToHead {
  | None => content
  | Some(appendToHead) => content->String.replace("</head>", appendToHead ++ "</head>")
  }

  requestController->RequestController.getDocHeader ++ content
}
let defaultHeaders = [("Content-Type", "text/html")]

type handleRequestConfig<'ctx> = {
  request: Request.t,
  server: Bun.Server.t,
  render: renderConfig<'ctx> => promise<Jsx.element>,
  setupHeaders?: unit => Headers.t,
  renderTitle?: array<string> => string,
  experimental_stream?: bool,
}

let handleRequest = async (t, {request, render, ?experimental_stream} as config) => {
  let stream = experimental_stream->Option.getWithDefault(false)

  let url = request->Request.url->URL.make
  let pathname = url->URL.pathname
  let targetHandler = t.handlers->Array.findMap(((handlerType, path, handler)) =>
    if handlerType === request->Request.method && path === pathname {
      Some(handler)
    } else {
```

```
      None
    }
  )

  let ctx = await t.requestToContext(request)
  let requestController = RequestController.make()

  let headers = switch config.setupHeaders {
  | Some(setupHeaders) => setupHeaders()
  | None => Headers.make(~init=FromArray(defaultHeaders))
  }
  let renderConfig = {
    context: ctx,
    headers,
    request,
    path: pathname
    ->String.split("/")
    ->Array.filter(s => s->String.trim !== "")
    ->List.fromArray,
    url,
    requestController,
  }

  await t.asyncLocalStorage->AsyncHooks.AsyncLocalStorage.run(renderConfig, async _token => {
    let content = switch targetHandler {
    | None => await render(renderConfig)
    | Some(handler) =>
      await handler({
        request,
        context: ctx,
        headers,
        requestController,
      })
    }

    if stream {
      let {readable, writable} = TransformStream.make({
        transform: (chunk, controller) => {
          controller->TransformStream.Controller.enqueue(chunk)
        },
      })
      let writer = writable->WritableStream.getWriter
      let textEncoder = TextEncoder.make()

      H.renderToStream(content, ~onChunk=chunk => {
        let encoded = textEncoder->TextEncoder.encode(chunk)
        writer->WritableStream.WritableStreamDefaultWriter.write(encoded)->Promise.done
      })
      ->Promise.thenResolve(_ => {
        writer->WritableStream.WritableStreamDefaultWriter.close
      })
      ->Promise.done

      Response.makeFromReadableStream(
        readable,
        ~options={
          status: 200,
          headers: FromArray([("Content-Type", "text/html")]),
        },
      )
    } else {
      let content = await renderWithDocType(
        content,
        ~requestController,
        ~renderTitle=?config.renderTitle,
      )
      switch (
        requestController->RequestController.getCurrentRedirect,
        requestController->RequestController.getCurrentStatus,
      ) {
      | (Some(url, status), _) => Response.makeRedirect(url, ~status?)
      | (None, status) => Response.makeWithHeaders(content, ~options={headers, status})
      }
    }
  })
}

let hxGet = (t, path, ~handler) => {
  t.handlers->Array.push((GET, path, handler))
  path
}
let makeHxGetIdentifier = path => {
  path
}
let implementHxGetIdentifier = (t, path, ~handler) => {
  let _: hxGet = hxGet(t, path, ~handler)
}
```

```
let hxPost = (t, path, ~handler) => {
  t.handlers->Array.push((POST, path, handler))
  path
}
let makeHxPostIdentifier = path => {
  path
}
let implementHxPostIdentifier = (t, path, ~handler) => {
  let _: hxPost = hxPost(t, path, ~handler)
}

let hxPut = (t, path, ~handler) => {
  t.handlers->Array.push((PUT, path, handler))
  path
}
let makeHxPutIdentifier = path => {
  path
}
let implementHxPutIdentifier = (t, path, ~handler) => {
  let _: hxPut = hxPut(t, path, ~handler)
}

let hxDelete = (t, path, ~handler) => {
  t.handlers->Array.push((DELETE, path, handler))
  path
}
let makeHxDeleteIdentifier = path => {
  path
}
let implementHxDeleteIdentifier = (t, path, ~handler) => {
  let _: hxDelete = hxDelete(t, path, ~handler)
}

let hxPatch = (t, path, ~handler) => {
  t.handlers->Array.push((PATCH, path, handler))
  path
}
let makeHxPatchIdentifier = path => {
  path
}
let implementHxPatchIdentifier = (t, path, ~handler) => {
  let _: hxPatch = hxPatch(t, path, ~handler)
}

module Internal = {
  let getHandlers = t => t.handlers
}
```

========== End file

========== Start file Handlers.resi

```
type htmxHandlerConfig<'ctx> = {
  request: Request.t,
  context: 'ctx,
  headers: Headers.t,
  requestController: RequestController.t,
}

type htmxHandler<'ctx> = htmxHandlerConfig<'ctx> => promise<Jsx.element>

type t<'ctx>

type hxGet
type hxPost
type hxPut
type hxPatch
type hxDelete

let make: (~requestToContext: Request.t => promise<'ctx>) => t<'ctx>

let hxGet: (t<'ctx>, string, ~handler: htmxHandler<'ctx>) => hxGet
let makeHxGetIdentifier: string => hxGet
let implementHxGetIdentifier: (t<'ctx>, hxGet, ~handler: htmxHandler<'ctx>) => unit

let hxPost: (t<'ctx>, string, ~handler: htmxHandler<'ctx>) => hxPost
let makeHxPostIdentifier: string => hxPost
let implementHxPostIdentifier: (t<'ctx>, hxPost, ~handler: htmxHandler<'ctx>) => unit

let hxPut: (t<'ctx>, string, ~handler: htmxHandler<'ctx>) => hxPut
let makeHxPutIdentifier: string => hxPut
let implementHxPutIdentifier: (t<'ctx>, hxPut, ~handler: htmxHandler<'ctx>) => unit

let hxDelete: (t<'ctx>, string, ~handler: htmxHandler<'ctx>) => hxDelete
let makeHxDeleteIdentifier: string => hxDelete
```

```
let implementHxDeleteIdentifier: (t<'ctx>, hxDelete, ~handler: htmxHandler<'ctx>) => unit

let hxPatch: (t<'ctx>, string, ~handler: htmxHandler<'ctx>) => hxPatch
let makeHxPatchIdentifier: string => hxPatch
let implementHxPatchIdentifier: (t<'ctx>, hxPatch, ~handler: htmxHandler<'ctx>) => unit

type renderConfig<'ctx> = {
  request: Request.t,
  headers: Headers.t,
  context: 'ctx,
  path: list<string>,
  url: URL.t,
  requestController: RequestController.t,
}

let useContext: t<'ctx> => renderConfig<'ctx>

type handleRequestConfig<'ctx> = {
  request: Request.t,
  server: Bun.Server.t,
  render: renderConfig<'ctx> => promise<Jsx.element>,
  setupHeaders?: unit => Headers.t,
  renderTitle?: array<string> => string,
  experimental_stream?: bool,
}

let handleRequest: (t<'ctx>, handleRequestConfig<'ctx>) => promise<Response.t>

module Internal: {
  let getHandlers: t<'ctx> => array<(method, string, htmxHandler<'ctx>)>
}
```

========== End file

brightid--buttonssponsor

Rescript v10

Repo: https://github.com/ShenaniganDApp/brightid-discord-bot

========== Start file package.json (part or full code)

```
{
  "name": "root",
  "private": true,
  "devDependencies": {
    "patch-package": "^6.4.7"
  },
  "dependencies": {
    "@rescript/core": "^0.2.0",
    "brightid_sdk": "^1.0.1",
    "canvas": "^2.9.0",
    "concurrently": "^7.1.0",
    "dotenv": "^8.2.0",
    "find-up": "^6.3.0",
    "rescript": "^10.1.0-rc.5",
    "rescript-discordjs": "^0.3.0",
    "rescript-nodejs": "^14.3.1",
    "uuid": "^8.3.0"
  },
  "scripts": {
    "bot": "yarn workspace @brightidbot/bot",
    "web": "yarn workspace @brightidbot/web",
    "utils": "yarn workspace @brightidbot/utils",
    "scripts": "yarn workspace @brightidbot/scripts",
    "shared": "yarn workspace @brightidbot/shared",
    "re:build": "yarn shared re:build && yarn utils re:build && yarn bot re:build && yarn web re:build && yarn s
cripts re:build"
  },
  "workspaces": [
    "apps/*",
    "packages/*"
  ]
}
```

========== End file

========== Start file Buttons_Sponsor.res

```
open Discord
open Shared
open NodeFetch
open Exceptions
```

```
let {brightIdAppDeeplink, brightIdLinkVerificationEndpoint} = module(Endpoints)

let {makeCanvasFromUri, createMessageAttachmentFromCanvas, makeBeforeSponsorActionRow} = module(
  Commands_Verify
)

@val @scope("globalThis")
external fetch: (string, 'params) => promise<Response.t<JSON.t>> = "fetch"

let sleep: int => promise<unit> = _ms => %raw(` new Promise((resolve) => setTimeout(resolve, _ms))`)

Env.createEnv()

let envConfig = switch Env.getConfig() {
| Ok(config) => config
| Error(err) => err->Env.EnvError->raise
}

let sponsorRequestSubmittedMessageOptions = async () => {
  let nowInSeconds = Math.round(Date.now() /. 1000.)
  let fifteenMinutesAfter = 15. *. 60. +. nowInSeconds
  let content = `You sponsor request has been submitted! \n\n Make sure you have scanned the QR code above in th
e BrightID mobile app to confirm your sponsor and link Discord to BrightID. \n This process will timeout <t:${fi
fteenMinutesAfter->Float.toString}:R>.\n\nPlease be patient as the BrightID nodes sync your request \n`
  {
    "content": content,
    "ephemeral": true,
  }
}

let noWriteToGistMessage = async interaction => {
  let options = {
    "content": "It seems like I can't write to my database at the moment. Please try again or contact the Bright
ID support.",
    "ephemeral": true,
  }

  await Interaction.followUp(interaction, ~options, ())
}

let makeAfterSponsorActionRow = label => {
  let verifyButton =
    MessageButton.make()
    ->MessageButton.setCustomId("verify")
    ->MessageButton.setLabel(label)
    ->MessageButton.setStyle("PRIMARY")

  MessageActionRow.make()->MessageActionRow.addComponents([verifyButton])
}

type sponsorship = Sponsorship(BrightId.Sponsorships.t)
let checkSponsor = async uuid => {
  open Shared.Decode
  let endpoint = `https://app.brightid.org/node/v5/sponsorships/${uuid}`
  let params = {
    "method": "GET",
    "headers": {
      "Accept": "application/json",
      "Content-Type": "application/json",
    },
    "timeout": 60000,
  }
  let res = await fetch(endpoint, params)
  let json = await Response.json(res)

  switch (
    json->Json.decode(Decode_BrightId.Sponsorships.data),
    json->Json.decode(Decode_BrightId.Error.data),
  ) {
  | (Ok({data}), _) => Sponsorship(data)
  | (_, Ok(error)) => error->Exceptions.BrightIdError->raise
  | (Error(err), _) => err->Json.Decode.DecodeError->raise
  }
}

let gistConfig = () =>
  Utils.Gist.makeGistConfig(
    ~id=envConfig["gistId"],
    ~name="guildData.json",
    ~token=envConfig["githubAccessToken"],
  )

let execute = async interaction => {
  open Utils
  open Shared.Decode
  let guild = interaction->Interaction.getGuild
  let guildId = guild->Guild.getGuildId
```

```
  let member = interaction->Interaction.getGuildMember
  let memberId = member->GuildMember.getGuildMemberId
  let uuid = memberId->UUID.v5(envConfig["uuidNamespace"])
  switch await Interaction.deferReply(interaction, ~options={"ephemeral": true}, ()) {
  | exception e => e->raise
  | _ =>
    switch await Gist.ReadGist.content(~config=gistConfig(), ~decoder=Decode_Gist.brightIdGuilds) {
    | exception e => e->raise
    | guilds =>
      switch guilds->Dict.get(guildId) {
      | None =>
        let _ = await noWriteToGistMessage(interaction)
        SponsorButtonError(
          `Buttons_Sponsor: Guild with guildId: ${guildId} not found in gist`,
        )->raise
      | Some(guildData) =>
        open Services_Sponsor
        let _ = switch await handleSponsor(interaction, uuid, Helpers.fifteenMinutesFromNow()) {
        | exception e => e->raise
        | SponsorshipUsed =>
          let usedSponsorships =
            guildData.usedSponsorships->Option.getWithDefault(
              Ethers.BigNumber.zero->Ethers.BigNumber.toString,
            )
          let usedSponsorships =
            usedSponsorships
            ->Ethers.BigNumber.fromString
            ->Ethers.BigNumber.addWithString("1")
            ->Ethers.BigNumber.toString

          let updateUsedSponsorships = await Utils.Gist.UpdateGist.updateEntry(
            ~config=gistConfig(),
            ~content=guilds,
            ~key=guildId,
            ~entry={...guildData, usedSponsorships: Some(usedSponsorships)},
          )
          switch updateUsedSponsorships {
          | Ok(_) =>
            let options = await successfulSponsorMessageOptions(uuid)
            let _ = await Interaction.followUp(interaction, ~options, ())
          | Error(err) =>
            Console.error2("Buttons Sponsor: Error updating used sponsorships", err)
            let _ = await noWriteToGistMessage(interaction)
          }

        | NoUnusedSponsorships =>
          let _ = await Interaction.followUp(
            interaction,
            ~options=noUnusedSponsorshipsOptions(),
            (),
          )

        | RetriedCommandDuring =>
          let options = {
            "content": "Your request is still processing. Maybe you haven't scanned the QR code yet?\n\n If you
have already scanned the code, please wait a few minutes for BrightID nodes to sync your sponsorship request",
            "ephemeral": true,
          }
          let _ = await Interaction.followUp(interaction, ~options, ())
        | TimedOut =>
          let options = await unsuccessfulSponsorMessageOptions(uuid)
          let _ = await Interaction.editReply(interaction, ~options, ())
        }
      }
    }
  }
}

let customId = "before-sponsor"
```

========== End file

res-x--react

Rescript v11

Repo: https://github.com/zth/res-x

========== Start file package.json (part or full code)

```
{
  "name": "rescript-x",
  "version": "0.1.0-alpha.7",
  "scripts": {
```

```
    "res:build": "rescript",
    "res:clean": "rescript clean",
    "res:dev": "rescript build -w"
  },
  "keywords": [
    "rescript"
  ],
  "files": [
    "README.md",
    "CHANGELOG.md",
    "rescript.json",
    "src/**/*",
    "res-x-vite-plugin.mjs"
  ],
  "author": "Gabriel Nordeborn",
  "license": "MIT",
  "peerDependencies": {
    "rescript": ">=11.0.0-rc.5",
    "@rescript/core": ">=0.5.0",
    "vite": ">=4.4.11",
    "rescript-bun": ">=0.1.0"
  },
  "devDependencies": {
    "@rescript/core": "^0.5.0",
    "fast-glob": "^3.3.1",
    "rescript": "11.0.0-rc.5",
    "rescript-bun": "0.1.0"
  },
  "dependencies": {
    "fast-glob": "^3.3.1"
  }
}
```

========== End file

========== Start file ResX__React.res

```
type element = Jsx.element

type component<'props> = Jsx.component<'props>
type componentLike<'props, 'return> = Jsx.componentLike<'props, 'return>

type fragmentProps = {children?: element}

@module("./vendor/hyperons.js") external jsxFragment: component<fragmentProps> = "Fragment"

@module("./vendor/hyperons.js")
external jsx: (component<'props>, 'props) => Jsx.element = "h"

@module("./vendor/hyperons.js")
external jsxs: (component<'props>, 'props) => element = "h"

@val external null: Jsx.element = "null"

external float: float => Jsx.element = "%identity"
external int: int => Jsx.element = "%identity"
external string: string => Jsx.element = "%identity"
external array: array<Jsx.element> => Jsx.element = "%identity"
```

========== End file

catala-dsfr--app

Rescript v10

Repo: https://github.com/CatalaLang/catala-dsfr

========== Start file package.json (part or full code)

```
{
  "name": "catala-dsfr",
  "version": "0.1.1",
  "repository": "https://github.com/CatalaLang/catala-dsfr",
  "scripts": {
    "clean": "rescript clean -with-deps",
    "build": "yarn run pre && vite build",
    "deploy": "yarn run build --base=/demos/catala/ && rsync -rv --delete-before dist/*",
    "serve": "vite preview",
    "dev": "yarn run pre && vite",
    "re:build": "rescript build -with-deps",
    "re:watch": "yarn run pre && rescript build -w -with-deps",
    "assets": "rsync -r node_modules/@catala-lang/catala-web-assets/assets/* assets",
    "postinstall": "copy-dsfr-to-public",
    "pre": "yarn run re:build && only-include-used-icons && yarn run assets"
```

```
  },
  "keywords": [
    "rescript"
  ],
  "author": "Emile Rolley <emile.rolley@tuta.io>",
  "license": "Apache-2.0",
  "dependencies": {
    "@catala-lang/catala-explain": "^0.2.2",
    "@catala-lang/catala-web-assets": "^0.8.9",
    "@catala-lang/french-law": "^0.8.3-b.3",
    "@catala-lang/rescript-catala": "^0.8.1-b.0",
    "@codegouvfr/react-dsfr": "^0.78.2",
    "@rescript/core": "^0.5.0",
    "@rescript/react": "^0.11.0",
    "@rjsf/core": "^5.1.0",
    "@rjsf/utils": "^5.1.0",
    "@rjsf/validator-ajv8": "^5.1.0",
    "file-saver": "^2.0.5",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-loader-spinner": "^5.4.5",
    "rescript-docx": "^0.1.5",
    "tslib": "^2.6.2"
  },
  "devDependencies": {
    "@jihchi/vite-plugin-rescript": "^5.1.0",
    "@originjs/vite-plugin-commonjs": "^1.0.3",
    "@vitejs/plugin-react": "^3.1.0",
    "jsdom": "^21.1.0",
    "rescript": "^10.1.4",
    "tailwindcss": "^3.2.6",
    "vite": "^4.4.9"
  }
}
```

========== End file

========== Start file App.res

```
%%raw(`import "./css/index.css";`)

Dsfr.Spa.startReactDsfr({
  defaultColorScheme: #system,
  link: Router.Link.make,
  useLang: () => #fr,
})

module App = {
  @react.component
  let make = () => {
    <>
      <Header />
      <main role="main" className="fr-h-10w fr-p-2w">
        <Router />
      </main>
      <Footer />
    </>
  }
}

ReactDOM.Client.createRoot(
  ReactDOM.querySelector("#app-root")->Belt.Option.getExn,
)->ReactDOM.Client.Root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```

========== End file

brightid--servicessponsor

Rescript v10

Repo: https://github.com/ShenaniganDApp/brightid-discord-bot

========== Start file package.json (part or full code)

```
{
  "name": "root",
  "private": true,
  "devDependencies": {
    "patch-package": "^6.4.7"
  },
```

```
  "dependencies": {
    "@rescript/core": "^0.2.0",
    "brightid_sdk": "^1.0.1",
    "canvas": "^2.9.0",
    "concurrently": "^7.1.0",
    "dotenv": "^8.2.0",
    "find-up": "^6.3.0",
    "rescript": "^10.1.0-rc.5",
    "rescript-discordjs": "^0.3.0",
    "rescript-nodejs": "^14.3.1",
    "uuid": "^8.3.0"
  },
  "scripts": {
    "bot": "yarn workspace @brightidbot/bot",
    "web": "yarn workspace @brightidbot/web",
    "utils": "yarn workspace @brightidbot/utils",
    "scripts": "yarn workspace @brightidbot/scripts",
    "shared": "yarn workspace @brightidbot/shared",
    "re:build": "yarn shared re:build && yarn utils re:build && yarn bot re:build && yarn web re:build && yarn s
cripts re:build"
  },
  "workspaces": [
    "apps/*",
    "packages/*"
  ]
}
```

========== End file

========== Start file Services_Sponsor.res

```
open Discord
open NodeFetch

let {brightIdVerificationEndpoint, brightIdAppDeeplink, brightIdLinkVerificationEndpoint} = module(
  Endpoints
)

let {makeCanvasFromUri, createMessageAttachmentFromCanvas, makeBeforeSponsorActionRow} = module(
  Commands_Verify
)

@val @scope("globalThis")
external fetch: (string, 'params) => promise<Response.t<JSON.t>> = "fetch"

let sleep: int => promise<unit> = _ms => %raw(` new Promise((resolve) => setTimeout(resolve, _ms))`)

Env.createEnv()

let envConfig = switch Env.getConfig() {
| Ok(config) => config
| Error(err) => err->Env.EnvError->raise
}

exception RetryAsync(string)
let rec retry = async (fn, n) => {
  try {
    let _ = await sleep(1000)
    await fn()
  } catch {
  | _ =>
    if n > 0 {
      await retry(fn, n - 1)
    }
  }
  RetryAsync(j`Failed $fn retrying $n times`)->raise
}

let noUnusedSponsorshipsOptions = () =>
  {
    "content": "There are no sponsorhips available in the Discord pool. Please try again later.",
    "ephemeral": true,
  }

let unsuccessfulSponsorMessageOptions = async uuid => {
  let verifyUrl = `${brightIdLinkVerificationEndpoint}/${uuid}`
  let row = makeBeforeSponsorActionRow("Retry Sponsor", verifyUrl)
  {
    "content": "Your sponsor request failed. \n\n This is often due to the BrightID App not being linked to Disc
ord. Please scan the previous QR code in the BrightID mobile app then retry your sponsorship request.\n\n",
    "ephemeral": true,
    "components": [row],
  }
}
let sponsorRequestSubmittedMessageOptions = async () => {
  let nowInSeconds = Math.round(Date.now() /. 1000.)
```

```
  let fifteenMinutesAfter = 15. *. 60. +. nowInSeconds
  let content = `You sponsor request has been submitted! \n\n Make sure you have scanned this QR code in the Bri
ghtID mobile app to confirm your sponsor and link Discord to BrightID. \n This process will timeout <t:${fifteen
MinutesAfter->Float.toString}:R>.\n\n`
  {
    "content": content,
    "ephemeral": true,
  }
}

let makeAfterSponsorActionRow = label => {
  let verifyButton =
    MessageButton.make()
    ->MessageButton.setCustomId("verify")
    ->MessageButton.setLabel(label)
    ->MessageButton.setStyle("PRIMARY")

  MessageActionRow.make()->MessageActionRow.addComponents([verifyButton])
}

let successfulSponsorMessageOptions = async uuid => {
  let uri = `${brightIdAppDeeplink}/${uuid}`
  let canvas = await makeCanvasFromUri(uri)
  let attachment = await createMessageAttachmentFromCanvas(canvas)
  let row = makeAfterSponsorActionRow("Assign BrightID Verified Role")
  {
    "content": "You have succesfully been sponsored \n\n If you are verified in BrightID you are all done. Click
 the button below to assign your role.\n\n",
    "files": [attachment],
    "ephemeral": true,
    "components": [row],
  }
}

exception HandleSponsorError(string)
type sponsor = SponsorSuccess(Shared.BrightId.Sponsorships.sponsor)
type handleSponsor =
  | SponsorshipUsed
  | RetriedCommandDuring
  | NoUnusedSponsorships
  | TimedOut

type sponsorship = Sponsorship(Shared.BrightId.Sponsorships.t)
let checkSponsor = async uuid => {
  open Shared.Decode
  let endpoint = `https://app.brightid.org/node/v5/sponsorships/${uuid}`
  let params = {
    "method": "GET",
    "headers": {
      "Accept": "application/json",
      "Content-Type": "application/json",
    },
    "timeout": 60000,
  }
  let res = await fetch(endpoint, params)
  let json = await Response.json(res)

  switch (
    json->Json.decode(Decode_BrightId.Sponsorships.data),
    json->Json.decode(Decode_BrightId.Error.data),
  ) {
  | (Ok({data}), _) => Sponsorship(data)
  | (_, Ok(error)) => error->Exceptions.BrightIdError->raise
  | (Error(err), _) => err->Json.Decode.DecodeError->raise
  }
}

@raises([HandleSponsorError, Exn.Error, Json.Decode.DecodeError])
let rec handleSponsor = async (
  ~maybeHash=None,
  ~maybeLogMessage=None,
  interaction,
  uuid,
  endTimeInSeconds,
) => {
  open Shared.BrightId
  open Shared.Decode

  let guildId = interaction->Interaction.getGuild->Guild.getGuildId
  let secondsBetweenAttempts = 15 //Probably won't need this if whe are using our own node
  // 10 second buffer for Webhook expiry
  let hasWebhookExpired = endTimeInSeconds - Helpers.nowInSeconds() < 10
  switch hasWebhookExpired {
  | true =>
    if maybeLogMessage->Option.isSome {
      let _ = await CustomMessages.editSponsorshipMessage(
        maybeLogMessage->Option.getExn,
```

```
      interaction,
      CustomMessages.Status.Failed,
      uuid,
      maybeHash,
    )
  }
  TimedOut
| _ =>
  try {
    let json = await sponsor(
      ~key=envConfig["sponsorshipKey"],
      ~context="Discord",
      ~contextId=uuid,
    )
    switch json->Json.decode(Decode_BrightId.Sponsorships.sponsor) {
    | Ok({hash}) =>
      let options = await sponsorRequestSubmittedMessageOptions()
      let _ = await Interaction.editReply(interaction, ~options, ())
      Console.log2(
        `A sponsor request has been submitted`,
        {"guild": guildId, "contextId": uuid, "hash": hash},
      )
      let maybeLogMessage = await CustomMessages.sponsorshipRequested(
        interaction,
        uuid,
        Some(hash),
      )
      await handleSponsor(
        interaction,
        uuid,
        ~maybeHash=Some(hash),
        ~maybeLogMessage,
        endTimeInSeconds,
      )
    | Error(err) => Json.Decode.DecodeError(err)->raise
    }
  } catch {
  | Exn.Error(error) =>
    try {
      let brightIdError =
        JSON.stringifyAny(error)
        ->Option.map(JSON.parseExn)
        ->Option.map(Json.decode(_, Decode_BrightId.Error.data))

      switch brightIdError {
      | None =>
        HandleSponsorError(
          "Handle Sponsor Error: There was a problem JSON parsing the error from sponsor()",
        )->raise
      | Some(Error(err)) => err->Json.Decode.DecodeError->raise
      | Some(Ok({errorNum})) =>
        switch (errorNum, maybeHash) {
        //No Sponsorships in the Discord App
        | (38, _) =>
          if maybeLogMessage->Option.isSome {
            let _ = await CustomMessages.editSponsorshipMessage(
              maybeLogMessage->Option.getExn,
              interaction,
              CustomMessages.Status.Error(
                "No Sponsorships available in the BrightID Discord App",
              ),
              uuid,
              maybeHash,
            )
          }
          NoUnusedSponsorships
        //Sponsorship already assigned
        | (_, None) => RetriedCommandDuring
        | (39, Some(hash)) =>
          let Sponsorship({spendRequested, appHasAuthorized}) = await checkSponsor(uuid)
          if spendRequested && appHasAuthorized {
            if maybeLogMessage->Option.isSome {
              let _ =
                maybeLogMessage->Option.map(async logMessage =>
                  await CustomMessages.editSponsorshipMessage(
                    logMessage,
                    interaction,
                    CustomMessages.Status.Successful,
                    uuid,
                    Some(hash),
                  )
                )
            }
            let options = successfulSponsorMessageOptions(uuid)
            let _ = await Interaction.editReply(interaction, ~options, ())
            SponsorshipUsed
          } else {
```

```
        let _ = await sleep(secondsBetweenAttempts * 1000)
        await handleSponsor(~maybeHash, ~maybeLogMessage, interaction, uuid, endTimeInSeconds)
      }
    //App authorized before
    | (45, Some(hash)) =>
      let Sponsorship({spendRequested, appHasAuthorized}) = await checkSponsor(uuid)
      if spendRequested && appHasAuthorized {
        if maybeLogMessage->Option.isSome {
          let _ = await CustomMessages.editSponsorshipMessage(
            maybeLogMessage->Option.getExn,
            interaction,
            CustomMessages.Status.Successful,
            uuid,
            Some(hash),
          )
        }
        let options = successfulSponsorMessageOptions(uuid)
        let _ = {await Interaction.editReply(interaction, ~options, ())}
        SponsorshipUsed
      } else {
        let _ = await sleep(secondsBetweenAttempts * 1000)
        await handleSponsor(~maybeHash, ~maybeLogMessage, interaction, uuid, endTimeInSeconds)
      }

    // Spend Request Submitted
    | (46, Some(hash)) =>
      let Sponsorship({spendRequested, appHasAuthorized}) = await checkSponsor(uuid)
      if spendRequested && appHasAuthorized {
        if maybeLogMessage->Option.isSome {
          let _ = await CustomMessages.editSponsorshipMessage(
            maybeLogMessage->Option.getExn,
            interaction,
            CustomMessages.Status.Successful,
            uuid,
            Some(hash),
          )
        }
        let options = await successfulSponsorMessageOptions(uuid)
        let _ = await interaction->Interaction.editReply(~options, ())
        SponsorshipUsed
      } else {
        let _ = await sleep(secondsBetweenAttempts * 1000)
        await handleSponsor(~maybeHash, ~maybeLogMessage, interaction, uuid, endTimeInSeconds)
      }

    // Sponsored Request Recently
    | (47, Some(_)) =>
      let Sponsorship({spendRequested, appHasAuthorized}) = await checkSponsor(uuid)
      if spendRequested && appHasAuthorized {
        if maybeLogMessage->Option.isSome {
          let _ = await CustomMessages.editSponsorshipMessage(
            maybeLogMessage->Option.getExn,
            interaction,
            CustomMessages.Status.Successful,
            uuid,
            maybeHash,
          )
        }
        let options = successfulSponsorMessageOptions(uuid)
        let _ = await Interaction.editReply(interaction, ~options, ())
        SponsorshipUsed
      } else {
        let _ = await sleep(secondsBetweenAttempts * 1000)
        await handleSponsor(~maybeHash, ~maybeLogMessage, interaction, uuid, endTimeInSeconds)
      }

    | _ =>
      let _ = await sleep(secondsBetweenAttempts * 1000)
      await handleSponsor(~maybeHash, ~maybeLogMessage, interaction, uuid, endTimeInSeconds)
    }
  }
} catch {
| Exceptions.BrightIdError(_) =>
  let _ = await sleep(secondsBetweenAttempts * 1000)
  await handleSponsor(interaction, uuid, ~maybeHash, ~maybeLogMessage, endTimeInSeconds)
| Json.Decode.DecodeError(msg) =>
  if msg->String.includes("503 Service Temporarily Unavailable") {
    let _ = await sleep(secondsBetweenAttempts * 1000)
    await handleSponsor(~maybeHash, ~maybeLogMessage, interaction, uuid, endTimeInSeconds)
  } else {
    HandleSponsorError(msg)->raise
  }
| Exn.Error(obj) =>
  switch Exn.name(obj) {
  | Some("FetchError") =>
    let _ = await sleep(3000)
    await handleSponsor(~maybeHash, ~maybeLogMessage, interaction, uuid, endTimeInSeconds)
```

```
        | _ =>
          switch Exn.message(obj) {
          | Some(msg) =>
            if maybeLogMessage->Option.isSome {
              let _ = await CustomMessages.editSponsorshipMessage(
                maybeLogMessage->Option.getExn,
                interaction,
                CustomMessages.Status.Error(msg),
                uuid,
                maybeHash,
              )
            }
            HandleSponsorError(msg)->raise
          | None =>
            Console.error(obj)
            if maybeLogMessage->Option.isSome {
              let _ = await CustomMessages.editSponsorshipMessage(
                maybeLogMessage->Option.getExn,
                interaction,
                CustomMessages.Status.Error("Something went wrong"),
                uuid,
                maybeHash,
              )
            }
            HandleSponsorError("Handle Sponsor: Unknown Error")->raise
          }
        }
      }
    }
  }
}
```

========== End file

brightid--commandsinvite

Rescript v10

Repo: https://github.com/ShenaniganDApp/brightid-discord-bot

========== Start file package.json (part or full code)

```
{
  "name": "root",
  "private": true,
  "devDependencies": {
    "patch-package": "^6.4.7"
  },
  "dependencies": {
    "@rescript/core": "^0.2.0",
    "brightid_sdk": "^1.0.1",
    "canvas": "^2.9.0",
    "concurrently": "^7.1.0",
    "dotenv": "^8.2.0",
    "find-up": "^6.3.0",
    "rescript": "^10.1.0-rc.5",
    "rescript-discordjs": "^0.3.0",
    "rescript-nodejs": "^14.3.1",
    "uuid": "^8.3.0"
  },
  "scripts": {
    "bot": "yarn workspace @brightidbot/bot",
    "web": "yarn workspace @brightidbot/web",
    "utils": "yarn workspace @brightidbot/utils",
    "scripts": "yarn workspace @brightidbot/scripts",
    "shared": "yarn workspace @brightidbot/shared",
    "re:build": "yarn shared re:build && yarn utils re:build && yarn bot re:build && yarn web re:build && yarn s
cripts re:build"
  },
  "workspaces": [
    "apps/*",
    "packages/*"
  ]
}
```

========== End file

========== Start file Commands_Invite.res

```
open Discord
open Promise
open Exceptions

@module("../updateOrReadGist.mjs")
```

```
external updateGist: (string, 'a) => promise<unit> = "updateGist"

let urlRe = %re(
  "/(https?:\/\/(?:www\.|(?!www))[a-zA-Z0-9][a-zA-Z0-9-]+[a-zA-Z0-9]\.[^\s]{2,}|www\.[a-zA-Z0-9][a-zA-Z0-9-]+[a-
zA-Z0-9]\.[^\s]{2,}|https?:\/\/(?:www\.|(?!www))[a-zA-Z0-9]+\.[^\s]{2,}|www\.[a-zA-Z0-9]+\.[^\s]{2,})/"
)

let execute = (interaction: Interaction.t) => {
  let guild = interaction->Interaction.getGuild
  let member = interaction->Interaction.getGuildMember
  let isAdmin = member->GuildMember.getPermissions->Permissions.has(Permissions.Flags.administrator)
  let commandOptions = interaction->Interaction.getOptions
  interaction
  ->Interaction.deferReply(~options={"ephemeral": true}, ())
  ->then(_ => {
    switch isAdmin {
    | false =>
      interaction
      ->Interaction.editReply(
        ~options={"content": "Only administrators can change the invite link"},
        (),
      )
      ->ignore
      InviteCommandError("Commands_Invite: User does not have Administrator permissions")->raise
    | true => {
        let inviteLink = commandOptions->CommandInteractionOptionResolver.getString("invite")
        switch inviteLink->Nullable.toOption {
        | None =>
          interaction
          ->Interaction.editReply(
            ~options={"content": "I didn't receive an invite link. (For some unexplained reason)"},
            (),
          )
          ->ignore
          InviteCommandError("Commands_Invite: Invite Link returned null or undefined")->reject
        | Some(inviteLink) =>
          switch urlRe->RegExp.test(inviteLink) {
          | false => {
              interaction
              ->Interaction.editReply(
                ~options={"content": "The invite link is not a valid URL"},
                (),
              )
              ->ignore
              InviteCommandError("Commands_Invite: Invite Link is not a valid URL")->reject
            }

          | true => {
              updateGist(
                guild->Guild.getGuildId,
                {
                  "inviteLink": inviteLink,
                },
              )->ignore

              interaction
              ->Interaction.editReply(
                ~options={
                  "content": `Successfully update server invite link to ${inviteLink}`,
                  "ephemeral": true,
                },
                (),
              )
              ->ignore
              resolve()
            }
          }
        }
      }
    }->catch(e => {
      switch e {
      | InviteCommandError(msg) => Console.error(msg)
      | Exn.Error(obj) =>
        switch Exn.message(obj) {
        | Some(msg) => Console.error(msg)
        | None => Console.error("Must be some non-error value")
        }
      | _ => Console.error("Some unknown error")
      }
      resolve()
    })
  })
}

let data =
  SlashCommandBuilder.make()
  ->SlashCommandBuilder.setName("invite")
```

```
  ->SlashCommandBuilder.setDescription("Add an invite link to be displayed for this server")
  ->SlashCommandBuilder.addStringOption(option => {
    open SlashCommandStringOption
    option
    ->setName("invite")
    ->setDescription("Enter an invite link to this server")
    ->setRequired(true)
  })
```

========== End file

brightid--entryclient

Rescript v10

Repo: https://github.com/ShenaniganDApp/brightid-discord-bot

========== Start file package.json (part or full code)

```
{
  "name": "root",
  "private": true,
  "devDependencies": {
    "patch-package": "^6.4.7"
  },
  "dependencies": {
    "@rescript/core": "^0.2.0",
    "brightid_sdk": "^1.0.1",
    "canvas": "^2.9.0",
    "concurrently": "^7.1.0",
    "dotenv": "^8.2.0",
    "find-up": "^6.3.0",
    "rescript": "^10.1.0-rc.5",
    "rescript-discordjs": "^0.3.0",
    "rescript-nodejs": "^14.3.1",
    "uuid": "^8.3.0"
  },
  "scripts": {
    "bot": "yarn workspace @brightidbot/bot",
    "web": "yarn workspace @brightidbot/web",
    "utils": "yarn workspace @brightidbot/utils",
    "scripts": "yarn workspace @brightidbot/scripts",
    "shared": "yarn workspace @brightidbot/shared",
    "re:build": "yarn shared re:build && yarn utils re:build && yarn bot re:build && yarn web re:build && yarn s
cripts re:build"
  },
  "workspaces": [
    "apps/*",
    "packages/*"
  ]
}
```

========== End file

========== Start file entry.client.res

```
@val external document: Dom.element = "document"

module ReactDOM = {
  @module("react-dom/client")
  external hydrateRoot: (Dom.element, React.element) => unit = "hydrateRoot"
}

@module("react") external startTransition: (unit => unit) => unit = "startTransition"

let hydrate = () =>
  startTransition(() => {
    ReactDOM.hydrateRoot(
      document,
      <React.StrictMode>
        <Remix.RemixBrowser />
      </React.StrictMode>,
    )
  })

%%raw(`
if (window.requestIdleCallback) {
    window.requestIdleCallback(hydrate);
 }else {
  // Safari doesn't support requestIdleCallback
  // https://caniuse.com/requestidlecallback
  window.setTimeout(hydrate, 1);
  }`)
```

========== End file

chat-script--app

Rescript v11

Repo: https://github.com/Exegetech/chat-rescript

========== Start file package.json (part or full code)

```
{
  "name": "frontend",
  "scripts": {
    "res:build": "rescript",
    "res:clean": "rescript clean",
    "res:dev": "rescript build -w",
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "@rescript/core": "0.5.0",
    "@rescript/react": "0.11.0",
    "shared": "workspace:*",
    "daisyui": "3.9.2",
    "react": "18.2.0",
    "react-dom": "18.2.0",
    "rescript": "11.0.0-rc.4"
  },
  "devDependencies": {
    "@vitejs/plugin-react": "4.0.0",
    "autoprefixer": "10.4.15",
    "postcss": "8.4.28",
    "tailwindcss": "3.3.3",
    "vite": "4.4.9"
  }
}
```

========== End file

// ========== Start App.res file module ChatBox = Chat__Box

@react.component let make = () => { let (username, setUsername) = React.useState(() => "") let (chats, setChats) = React.useState(() => []) let socket = React.useRef(None)

React.useEffect1(() => { let run = async () => { switch username { | "" => () | username => { let chatHistory = await Util.fetchChatHistory(username) switch chatHistory { | Error(error) => { Console.error(error) } | Ok(chatHistory) => setChats((_prev) => chatHistory) }

```
      let url = `ws://localhost:3000/room?username=${username}`
      let ws = WebSocket.create(url)

      ws->WebSocket.set_onOpen(() => {
        Console.log("Connected to websocket")
      })

      ws->WebSocket.set_onMessage((event) => {
        let payload = Message.ToClient.deserializeOne(event.data)
        switch payload {
          | Error(error) => {
            Console.error(error)
          }
          | Ok(payload) => {
            setChats((prev) => {
              let newArr = Array.copy(prev)
              Array.push(newArr, payload)

              newArr
            })
          }
        }
      })

      socket.current = Some(ws)
    }
  }
}

let _ = run()

Some(() => {
  switch socket.current {
    | None => ()
```

```
    | Some(ws) => WebSocket.close(ws)
  }
})
```

}, [username])

let handleUsernameSubmit = (username) => { setUsername((_prev) => username) }

let handleChatSubmit = (from, message) => { switch socket.current { | None => () | Some(ws) => { open Message

```
    let payload = ToServer.create(~from, ~message)
      -> Message.ToServer.serialize

    switch payload {
      | Error(errMsg) => Console.error(errMsg)
      | Ok(payload) => ws->WebSocket.send(payload)
    }
  }
}
```

}

container mx-auto h-screen w-1/3 flex flex-col>
{switch username { | "" => ( ) | username => ( ) }}
} // ========== End file

// ========== Start file App.resi @react.component let make: unit => Jsx.element // ========== End file

brightid--env

Rescript v10

Repo: https://github.com/ShenaniganDApp/brightid-discord-bot

========== Start file package.json (part or full code)

```
{
  "name": "root",
  "private": true,
  "devDependencies": {
    "patch-package": "^6.4.7"
  },
  "dependencies": {
    "@rescript/core": "^0.2.0",
    "brightid_sdk": "^1.0.1",
    "canvas": "^2.9.0",
    "concurrently": "^7.1.0",
    "dotenv": "^8.2.0",
    "find-up": "^6.3.0",
    "rescript": "^10.1.0-rc.5",
    "rescript-discordjs": "^0.3.0",
    "rescript-nodejs": "^14.3.1",
    "uuid": "^8.3.0"
  },
  "scripts": {
    "bot": "yarn workspace @brightidbot/bot",
    "web": "yarn workspace @brightidbot/web",
    "utils": "yarn workspace @brightidbot/utils",
    "scripts": "yarn workspace @brightidbot/scripts",
    "shared": "yarn workspace @brightidbot/shared",
    "re:build": "yarn shared re:build && yarn utils re:build && yarn bot re:build && yarn web re:build && yarn s
cripts re:build"
  },
  "workspaces": [
    "apps/*",
    "packages/*"
  ]
}
```

========== End file

========== Start file Env.res

```
exception EnvError(string)
@module("find-up") external findUpSync: (string, 'options) => string = "findUpSync"
@module("dotenv") external createEnv: {"path": string} => unit = "config"

let nodeEnv = Node.Process.process["env"]

let createEnv = () => {
  let path = switch nodeEnv->Dict.get("ENV_FILE") {
  | None => ".env.local"->findUpSync()
```

```
  | Some(envFile) => envFile->findUpSync()
  }
  createEnv({"path": path})
}

let env = name =>
  switch Dict.get(nodeEnv, name) {
  | Some(value) => Ok(value)
  | None => Error(`Environment variable ${name} is missing`)
  }

let getConfig = () =>
  switch (
    env("DISCORD_API_TOKEN"),
    env("DISCORD_CLIENT_ID"),
    env("UUID_NAMESPACE"),
    env("GIST_ID"),
    env("GITHUB_ACCESS_TOKEN"),
    env("SPONSORSHIP_KEY"),
    env("SPONSORSHIPS_WHITELIST"),
    env("DISCORD_LOG_CHANNEL_ID"),
  ) {
  // Got all vars
  | (
      Ok(discordApiToken),
      Ok(discordClientId),
      Ok(uuidNamespace),
      Ok(gistId),
      Ok(githubAccessToken),
      Ok(sponsorshipKey),
      Ok(sponsorshipsWhitelist),
      Ok(discordLogChannelId),
    ) =>
    Ok({
      "discordApiToken": discordApiToken,
      "discordClientId": discordClientId,
      "uuidNamespace": uuidNamespace,
      "gistId": gistId,
      "githubAccessToken": githubAccessToken,
      "sponsorshipKey": sponsorshipKey,
      "sponsorshipsWhitelist": sponsorshipsWhitelist,
      "discordLogChannelId": discordLogChannelId,
    })
  // Did not get one or more vars, return the first error
  | (Error(_) as err, _, _, _, _, _, _, _)
  | (_, Error(_) as err, _, _, _, _, _, _)
  | (_, _, Error(_) as err, _, _, _, _, _)
  | (_, _, _, Error(_) as err, _, _, _, _)
  | (_, _, _, _, Error(_) as err, _, _, _)
  | (_, _, _, _, _, Error(_) as err, _, _)
  | (_, _, _, _, _, _, Error(_) as err, _)
  | (_, _, _, _, _, _, _, Error(_) as err) => err
  }
```

========== End file

catala-dsfr--allocation

Rescript v10

Repo: https://github.com/CatalaLang/catala-dsfr

========== Start file package.json (part or full code)

```
{
  "name": "catala-dsfr",
  "version": "0.1.1",
  "repository": "https://github.com/CatalaLang/catala-dsfr",
  "scripts": {
    "clean": "rescript clean -with-deps",
    "build": "yarn run pre && vite build",
    "deploy": "yarn run build --base=/demos/catala/ && rsync -rv --delete-before dist/*",
    "serve": "vite preview",
    "dev": "yarn run pre && vite",
    "re:build": "rescript build -with-deps",
    "re:watch": "yarn run pre && rescript build -w -with-deps",
    "assets": "rsync -r node_modules/@catala-lang/catala-web-assets/assets/* assets",
    "postinstall": "copy-dsfr-to-public",
    "pre": "yarn run re:build && only-include-used-icons && yarn run assets"
  },
  "keywords": [
    "rescript"
  ],
  "author": "Emile Rolley <emile.rolley@tuta.io>",
```

```
  "license": "Apache-2.0",
  "dependencies": {
    "@catala-lang/catala-explain": "^0.2.2",
    "@catala-lang/catala-web-assets": "^0.8.9",
    "@catala-lang/french-law": "^0.8.3-b.3",
    "@catala-lang/rescript-catala": "^0.8.1-b.0",
    "@codegouvfr/react-dsfr": "^0.78.2",
    "@rescript/core": "^0.5.0",
    "@rescript/react": "^0.11.0",
    "@rjsf/core": "^5.1.0",
    "@rjsf/utils": "^5.1.0",
    "@rjsf/validator-ajv8": "^5.1.0",
    "file-saver": "^2.0.5",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-loader-spinner": "^5.4.5",
    "rescript-docx": "^0.1.5",
    "tslib": "^2.6.2"
  },
  "devDependencies": {
    "@jihchi/vite-plugin-rescript": "^5.1.0",
    "@originjs/vite-plugin-commonjs": "^1.0.3",
    "@vitejs/plugin-react": "^3.1.0",
    "jsdom": "^21.1.0",
    "rescript": "^10.1.4",
    "tailwindcss": "^3.2.6",
    "vite": "^4.4.9"
  }
}
```

========== End file

========== Start file AllocationsFamiliales.res

```
module FormInfos = {
  let webAssets = WebAssets.allocationsFamilialesAssets
  let name = `allocations familiales`
  let resultLabel = `Montant mensuel des ${name}`
  let url = "allocations-familiales"

  // This function automatically assigns numerical ID to kids so we don't
  // have to ask the question in the form
  let formDataPostProcessing = %raw(`
      function (data) {
          var i = 0;
          for (var enfant of data.iEnfantsIn) {
              enfant.dIdentifiant = i;
              i++;
          }
          return data;
      }
  `)

  let computeAndPrintResult = (input: Js.Json.t): React.element => <>
    <span className="font-mono font-bold text-[var(--text-active-blue-france)]">
      {input->CatalaFrenchLaw.computeAllocationsFamiliales->Belt.Float.toString->React.string}
    </span>
    {React.string(` â,¬`)}
  </>
}

module Form = Form.Make(FormInfos)

@react.component
let make = () => {
  React.useEffect0(() => {
    // Reset the log when the page is loaded.
    CatalaFrenchLaw.resetLog()
    None
  })

  <div className="fr-container">
    <PageComponents.Title>
      {"Calcul des allocations familiales"->React.string}
    </PageComponents.Title>
    <Form />
  </div>
}
```

========== End file

res-x--client

Rescript v11

Repo: https://github.com/zth/res-x

========== Start file package.json (part or full code)

```
{
  "name": "rescript-x",
  "version": "0.1.0-alpha.7",
  "scripts": {
    "res:build": "rescript",
    "res:clean": "rescript clean",
    "res:dev": "rescript build -w"
  },
  "keywords": [
    "rescript"
  ],
  "files": [
    "README.md",
    "CHANGELOG.md",
    "rescript.json",
    "src/**/*",
    "res-x-vite-plugin.mjs"
  ],
  "author": "Gabriel Nordeborn",
  "license": "MIT",
  "peerDependencies": {
    "rescript": ">=11.0.0-rc.5",
    "@rescript/core": ">=0.5.0",
    "vite": ">=4.4.11",
    "rescript-bun": ">=0.1.0"
  },
  "devDependencies": {
    "@rescript/core": "^0.5.0",
    "fast-glob": "^3.3.1",
    "rescript": "11.0.0-rc.5",
    "rescript-bun": "0.1.0"
  },
  "dependencies": {
    "fast-glob": "^3.3.1"
  }
}
```

========== End file

========== Start file Client.res

```
module Actions: {
  type t

  @tag("kind")
  type target = This | CssSelector({selector: string})

  @tag("kind")
  type action =
    | ToggleClass({target: target, className: string})
    | RemoveClass({target: target, className: string})
    | AddClass({target: target, className: string})
    | RemoveElement({target: target})

  let make: array<action> => t
} = {
  type t = string

  @tag("kind")
  type target = This | CssSelector({selector: string})

  @tag("kind")
  type action =
    | ToggleClass({target: target, className: string})
    | RemoveClass({target: target, className: string})
    | AddClass({target: target, className: string})
    | RemoveElement({target: target})

  external stringifyActions: array<action> => string = "JSON.stringify"

  let make = actions => stringifyActions(actions)
}

module ValidityMessage: {
  type config = {
    badInput?: string,
    patternMismatch?: string,
    rangeOverflow?: string,
    rangeUnderflow?: string,
    stepMismatch?: string,
    tooLong?: string,
    tooShort?: string,
```

```
    typeMismatch?: string,
    valueMissing?: string,
  }

  type t

  let make: config => t
} = {
  type config = {
    badInput?: string,
    patternMismatch?: string,
    rangeOverflow?: string,
    rangeUnderflow?: string,
    stepMismatch?: string,
    tooLong?: string,
    tooShort?: string,
    typeMismatch?: string,
    valueMissing?: string,
  }

  type t = string

  external stringifyConfig: config => string = "JSON.stringify"

  let make = config => stringifyConfig(config)
}
```

========== End file

res-x--staticexporter

Rescript v11

Repo: https://github.com/zth/res-x

========== Start file package.json (part or full code)

```
{
  "name": "rescript-x",
  "version": "0.1.0-alpha.7",
  "scripts": {
    "res:build": "rescript",
    "res:clean": "rescript clean",
    "res:dev": "rescript build -w"
  },
  "keywords": [
    "rescript"
  ],
  "files": [
    "README.md",
    "CHANGELOG.md",
    "rescript.json",
    "src/**/*",
    "res-x-vite-plugin.mjs"
  ],
  "author": "Gabriel Nordeborn",
  "license": "MIT",
  "peerDependencies": {
    "rescript": ">=11.0.0-rc.5",
    "@rescript/core": ">=0.5.0",
    "vite": ">=4.4.11",
    "rescript-bun": ">=0.1.0"
  },
  "devDependencies": {
    "@rescript/core": "^0.5.0",
    "fast-glob": "^3.3.1",
    "rescript": "11.0.0-rc.5",
    "rescript-bun": "0.1.0"
  },
  "dependencies": {
    "fast-glob": "^3.3.1"
  }
}
```

========== End file

========== Start file StaticExporter.res

```
open Bun

external process: 'a = "process"
external fetch: string => promise<Response.t> = "fetch"

let debugging = true
```

```
let debug = s =>
  if debugging {
    Console.log2("[debug]", s)
  }

let log = s => Console.log2("[info]", s)

let run = async (server: Server.t, ~urls: array<string>) => {
  let serverUrl = `http://${server->Server.hostname}:${server->Server.port->Int.toString}`
  log(`Exporting ${urls->Array.length->Int.toString} URLs.`)

  let _ = await Promise.all(
    urls->Array.map(async url => {
      log(`[export] ${url} - Exporting...`)
      let res = await fetch(serverUrl ++ url)

      switch res->Response.status {
      | 200 =>
        let structure =
          url
          ->String.split("/")
          ->Array.filter(p => p !== "")
          ->Array.toReversed

        let (sliceStart, fileName) = switch structure->Array.get(0) {
        | None | Some("") => (0, "index.html")
        | Some(f) => (1, f ++ ".html")
        }

        structure->Array.push("dist")

        let dirStructure = structure->Array.sliceToEnd(~start=sliceStart)->Array.toReversed

        switch dirStructure {
        | [] => ()
        | dirStructure =>
          await Fs.mkdir(dirStructure->Array.joinWith("/"), ~options={recursive: true})
        }

        dirStructure->Array.push(fileName)
        let filePath = dirStructure->Array.joinWith("/")

        await Fs.writeFile(filePath, await res->Response.text)
        log(`[export] ${url} - Wrote ${filePath}.`)

      | otherStatus => Console.error(url ++ " gave status " ++ otherStatus->Int.toString)
      }
    }),
  )

  log("Done.")

  server->Server.stop(~closeActiveConnections=true)
  process["exit"](0)
}
```

========== End file

brightid--entryresserver

Rescript v10

Repo: https://github.com/ShenaniganDApp/brightid-discord-bot

========== Start file package.json (part or full code)

```
{
  "name": "root",
  "private": true,
  "devDependencies": {
    "patch-package": "^6.4.7"
  },
  "dependencies": {
    "@rescript/core": "^0.2.0",
    "brightid_sdk": "^1.0.1",
    "canvas": "^2.9.0",
    "concurrently": "^7.1.0",
    "dotenv": "^8.2.0",
    "find-up": "^6.3.0",
    "rescript": "^10.1.0-rc.5",
    "rescript-discordjs": "^0.3.0",
    "rescript-nodejs": "^14.3.1",
    "uuid": "^8.3.0"
```

```
  },
  "scripts": {
    "bot": "yarn workspace @brightidbot/bot",
    "web": "yarn workspace @brightidbot/web",
    "utils": "yarn workspace @brightidbot/utils",
    "scripts": "yarn workspace @brightidbot/scripts",
    "shared": "yarn workspace @brightidbot/shared",
    "re:build": "yarn shared re:build && yarn utils re:build && yarn bot re:build && yarn web re:build && yarn s
cripts re:build"
  },
  "workspaces": [
    "apps/*",
    "packages/*"
  ]
}
```

========== End file

========== Start file entryRes.server.res

```
module ResponseInit = {
  type t

  external make: {..} => t = "%identity"
}

module BodyInit = {
  open Webapi.Fetch
  external makeWithPipeapleStream: NodeJs.Stream.PassThrough.t<
    NodeJs.Buffer.t,
    NodeJs.Buffer.t,
  > => BodyInit.t = "%identity"
}

@module("isbot") external isbot: string => bool = "default"

module ReactDOMServer = {
  type pipe = NodeJs.Stream.PassThrough.t<
    NodeJs.Buffer.t,
    NodeJs.Buffer.t,
  > => NodeJs.Stream.writable<NodeJs.Buffer.t>
  type abort = unit => unit

  type pipeableStream = {
    abort: abort,
    pipe: pipe,
  }

  @get external pipe: pipeableStream => pipe = "pipe"
  @get external abort: pipeableStream => abort = "abort"

  @module("react-dom/server")
  external renderToPipeableStream: (React.element, 'options) => pipeableStream =
    "renderToPipeableStream"
}

// TODO: Swap out for Webapi.Fetch.Response when it supports construction
// See https://github.com/tinymce/rescript-webapi/issues/63
@new
external makeResponse: (Webapi.Fetch.BodyInit.t, ResponseInit.t) => Webapi.Fetch.Response.t =
  "Response"

type onAllReady = {
  onAllReady: unit => unit,
  onShellError: exn => unit,
  onError: exn => unit,
}
type onShellReady = {
  onShellReady: unit => unit,
  onShellError: exn => unit,
  onError: exn => unit,
}
type ready = AllReady(onAllReady) | ShellReady(onShellReady)

@live
let default = (request, responseStatusCode, responseHeaders, remixContext) => {
  open Webapi
  let abortDelay = 5000

  let maybeCallbackName =
    request
    ->Fetch.Request.headers
    ->Fetch.Headers.get("User-Agent")
    ->Belt.Option.map(isbot)
    ->Belt.Option.map(onAllReady => onAllReady ? "onAllReady" : "onShellReady")
```

```
  Promise.make((resolve, reject) => {
    let onAllReadyOptions = pipe => {
      let callbackFn = () => {
        let body = NodeJs.Stream.PassThrough.make()

        request->Fetch.Request.headers->Fetch.Headers.set("Content-Type", "text/html")

        let response = BodyInit.makeWithPipeapleStream(body)->makeResponse(
          ResponseInit.make({
            "status": responseStatusCode,
            "headers": responseHeaders,
          }),
        )

        resolve(. response)
        pipe(body)->ignore
      }
      {
        onAllReady: callbackFn,
        onShellError: err => reject(. err),
        onError: err => Js.Console.error(err),
      }
    }
    let onShellReadyOptions = pipe => {
      let callbackFn = () => {
        let body = NodeJs.Stream.PassThrough.make()

        request->Fetch.Request.headers->Fetch.Headers.set("Content-Type", "text/html")

        let response = BodyInit.makeWithPipeapleStream(body)->makeResponse(
          ResponseInit.make({
            "status": responseStatusCode,
            "headers": responseHeaders,
          }),
        )

        resolve(. response)
        pipe(body)->ignore
      }
      {
        onShellReady: callbackFn,
        onShellError: err => reject(. err),
        onError: err => Js.Console.error(err),
      }
    }

    // This is hacky because we can't access the return in params in rescript
    open ReactDOMServer
    if maybeCallbackName->Belt.Option.getWithDefault("") === "onAllReady" {
      let allStream = renderToPipeableStream(
        <Remix.RemixServer context={remixContext} url={request->Fetch.Request.url} />,
        onAllReadyOptions(%raw(`allStream`)->pipe),
      )
      let _ = NodeJs.Timers.setTimeout(allStream.abort, abortDelay)
    } else if maybeCallbackName->Belt.Option.getWithDefault("") === "onShellReady" {
      let {abort, pipe} = renderToPipeableStream(
        <Remix.RemixServer context={remixContext} url={request->Fetch.Request.url} />,
        onShellReadyOptions(%raw(`pipe`)),
      )

      let _ = NodeJs.Timers.setTimeout(abort, abortDelay)
    }
  })
}
```

========== End file

chat-script--message-toclient

Rescript v11

Repo: https://github.com/Exegetech/chat-rescript

========== Start file package.json (part or full code)

```
{
  "name": "shared",
  "scripts": {
    "res:build": "rescript",
    "res:clean": "rescript clean",
    "res:dev": "rescript build -w"
  },
  "dependencies": {
    "@rescript/core": "0.5.0",
```

```
    "rescript": "11.0.0-rc.4"
  }
}
```

========== End file

========== Start file Message__ToClient.res

```
open Message__JSON

type t = {
  from: string,
  message: string,
  timestamp: float,
}

let getServerUsername = () => "server"

let create = (~from, ~message) => {
  let timestamp = Date.now()
  let message = {
    from,
    message,
    timestamp,
  }

  message
}

let encode = (message) => {
  [
    ("from", String(message.from)),
    ("message", String(message.message)),
    ("timestamp", Number(message.timestamp)),
  ]
  -> Dict.fromArray
  -> Object
}

let serializeOne = (payload) => {
  payload
  -> encode
  -> stringify
}

let serializeMany = (payload) => {
  payload
  -> Array.map(encode)
  -> Array
  -> stringify
}

let decode = (json) => {
  switch json {
    | Object(dict) => {
      let from = Dict.get(dict, "from")
      let message = Dict.get(dict, "message")
      let timestamp = Dict.get(dict, "timestamp")

      switch (from, message, timestamp) {
        | (
          Some(String(from)),
          Some(String(message)),
          Some(Number(timestamp)),
        ) if from !== "" && message !== "" => Ok({ from, message, timestamp })
        | _ => Error("Expected non empty string from, string message and number timestamp")
      }
    }
    | _ => Error("Expected an object")
  }
}

let deserializeOne = (payload) => {
  switch parse(payload) {
    | Error(errMsg) => Error("Error parsing JSON: " ++ errMsg)
    | Ok(json) => decode(json)
  }
}

let deserializeMany = (payload) => {
  switch parse(payload) {
    | Error(errMsg) => Error("Error parsing JSON: " ++ errMsg)
    | Ok(Array(jsons)) => {
      let decodedJsons = Array.map(jsons, decode)
      let successes = []
      let failures = []
```

```
      Array.forEach(decodedJsons, (json) => switch json {
        | Error(error) => Array.push(failures, error)
        | Ok(json) => Array.push(successes, json)
      })

      if Array.length(failures) > 0 {
        failures
        -> Array.joinWith(", ")
        -> Error
      } else {
        Ok(successes)
      }
    }
    | _ => Error("Expected an array")
  }
}
```

========== End file

========== Start file Message_ToClient.resi

```
type t = {
  from: string,
  message: string,
  timestamp: float,
}

let getServerUsername: () => string

let create: (~from: string, ~message: string) => t

let serializeOne: (t) => result<string, string>

let serializeMany: (array<t>) => result<string, string>

let deserializeOne: (string) => result<t, string>

let deserializeMany: (string) => result<array<t>, string>
```

========== End file

chat-script--backend--app

========== Start file App.res

```
open Fastify
open Node
open Message

module Seed = {
  let getFakeChats = () => {
    let now = Date.getTime(Date.make())

    let chats: array<ToClient.t> = [
      {
        from: "GitHub Copilot",
        timestamp: now,
        message: "Have you guys seen those AI robots? They can do some crazy things!"
      },
      {
        from: "Bard",
        timestamp: now -. 12000.0,
        message: "Yeah, I heard they can even beat humans in chess."
      },
      {
        from: "ChatGPT",
        timestamp: now -. 30000.0,
        message: "Well, that's nothing. My AI assistant once translated \"I love you\" to \"Error 404: Romance n
ot found.\"",
      }
    ]

    chats
  }
}

module Db = {
  let clients: Dict.t<WebSocket.t> = Dict.make()

  let chats: array<ToClient.t> = Seed.getFakeChats()
}

let getChatHistory = () => Array.copy(Db.chats)
```

```
let broadcast = (~payload, ~onError, ~exceptTo=?) => {
  switch ToClient.serializeOne(payload) {
    | Error(error) => onError(error)
    | Ok(message) => Db.clients
      -> Dict.toArray
      -> Array.forEach(((username, socket)) => {
        switch exceptTo {
          | None => socket->WebSocket.send(message)
          | Some(exceptTo) => {
            if username !== exceptTo {
              socket->WebSocket.send(message)
            }
          }
        }
      })
  }
}

let handleClient = (~username, ~socket, ~onError) => {
  Dict.set(Db.clients, username, socket)

  let payload = ToClient.create(
    ~from=ToClient.getServerUsername(),
    ~message=`${username} joined`,
  )

  broadcast(~payload, ~onError, ~exceptTo=username)

  socket->WebSocket.onMessage((buffer) => {
    let string = buffer->Buffer.toString
    let message = ToServer.deserialize(string)
    switch message {
      | Error(error) => onError(error)
      | Ok(message) => {
        let payload = ToClient.create(~from=username, ~message=message.message)

        Array.push(Db.chats, payload)

        broadcast(~payload, ~onError)
      }
    }
  })

  socket->WebSocket.onClose(() => {
    let payload = ToClient.create(
      ~from=ToClient.getServerUsername(),
      ~message=`${username} left`,
    )

    broadcast(~payload, ~onError, ~exceptTo=username)
  })
}
```

========== End file

========== Start file Chat.resi

```
let getChatHistory: () => array<Message.ToClient.t>

let handleClient: (
  ~username: string,
  ~socket: Fastify.WebSocket.t,
  ~onError: (string) => (),
) => ()
```

========== End file

res-x--h

Rescript v11

Repo: https://github.com/zth/res-x

========== Start file package.json (part or full code)

```
{
  "name": "rescript-x",
  "version": "0.1.0-alpha.7",
  "scripts": {
    "res:build": "rescript",
    "res:clean": "rescript clean",
    "res:dev": "rescript build -w"
  },
```

```
  "keywords": [
    "rescript"
  ],
  "files": [
    "README.md",
    "CHANGELOG.md",
    "rescript.json",
    "src/**/*",
    "res-x-vite-plugin.mjs"
  ],
  "author": "Gabriel Nordeborn",
  "license": "MIT",
  "peerDependencies": {
    "rescript": ">=11.0.0-rc.5",
    "@rescript/core": ">=0.5.0",
    "vite": ">=4.4.11",
    "rescript-bun": ">=0.1.0"
  },
  "devDependencies": {
    "@rescript/core": "^0.5.0",
    "fast-glob": "^3.3.1",
    "rescript": "11.0.0-rc.5",
    "rescript-bun": "0.1.0"
  },
  "dependencies": {
    "fast-glob": "^3.3.1"
  }
}
```

========== End file

========== Start file H.res

```
@val external null: Jsx.element = "null"

external float: float => Jsx.element = "%identity"
external int: int => Jsx.element = "%identity"
external string: string => Jsx.element = "%identity"
external array: array<Jsx.element> => Jsx.element = "%identity"

@module("./vendor/hyperons.js")
external renderToString: Jsx.element => promise<string> = "render"

@module("./vendor/hyperons.js")
external renderToStream: (Jsx.element, ~onChunk: string => unit=?) => promise<unit> = "render"

module Context = {
  type t<'context>

  type props<'context> = {
    value: 'context,
    children: Jsx.element,
  }

  @module("./vendor/hyperons.js")
  external createContext: 'context => t<'context> = "createContext"

  @module("./vendor/hyperons.js")
  external useContext: t<'context> => 'context = "useContext"

  @get external provider: t<'context> => Jsx.component<props<'context>> = "Provider"
}

module Fragment = {
  type fragmentProps = {children: Jsx.element}
  @module("./vendor/hyperons.js")
  external make: fragmentProps => Jsx.element = "Fragment"
}
```

========== End file

res-x--reactdom

Rescript v11

Repo: https://github.com/zth/res-x

========== Start file package.json (part or full code)

```
{
  "name": "rescript-x",
  "version": "0.1.0-alpha.7",
  "scripts": {
    "res:build": "rescript",
```

```
    "res:clean": "rescript clean",
    "res:dev": "rescript build -w"
  },
  "keywords": [
    "rescript"
  ],
  "files": [
    "README.md",
    "CHANGELOG.md",
    "rescript.json",
    "src/**/*",
    "res-x-vite-plugin.mjs"
  ],
  "author": "Gabriel Nordeborn",
  "license": "MIT",
  "peerDependencies": {
    "rescript": ">=11.0.0-rc.5",
    "@rescript/core": ">=0.5.0",
    "vite": ">=4.4.11",
    "rescript-bun": ">=0.1.0"
  },
  "devDependencies": {
    "@rescript/core": "^0.5.0",
    "fast-glob": "^3.3.1",
    "rescript": "11.0.0-rc.5",
    "rescript-bun": "0.1.0"
  },
  "dependencies": {
    "fast-glob": "^3.3.1"
  }
}
```

========== End file

========== Start file ResX__ReactDOM.res

```
@module("./vendor/hyperons.js")
external jsx: (string, H__domProps.domProps) => Jsx.element = "h"

@module("./vendor/hyperons.js")
external jsxs: (string, H__domProps.domProps) => Jsx.element = "h"

external someElement: Jsx.element => option<Jsx.element> = "%identity"
```

========== End file

fullstack--app

Rescript v10

Repo: https://github.com/skonky/fullstack

========== Start file package.json (part or full code)

```
{
  "name": "rescript-web",
  "version": "0.0.0",
  "author": "skonky",
  "private": true,
  "license": "Apache-2.0",
  "dependencies": {
    "next": "10.2.3",
    "react": "17.0.1",
    "react-dom": "17.0.1"
  },
  "scripts": {
    "dev": "concurrently \"next dev -p 5000\" \"rescript build -w\"",
    "debug": "NODE_OPTIONS='--inspect' next",
    "build": "rescript && next build",
    "now-build": "rescript && next build",
    "export": "next export",
    "start": "next start -p $PORT",
    "res:build": "rescript",
    "res:clean": "rescript clean",
    "res:start": "rescript build -w"
  },
  "devDependencies": {
    "@rescript/react": "0.10.3",
    "autoprefixer": "10.1.0",
    "concurrently": "^7.6.0",
    "cssnano": "5.0.5",
    "daisyui": "^2.51.3",
    "gentype": "4.1",
    "next-transpile-modules": "7.1.2",
```

```
    "postcss": "8.2.15",
    "postcss-cli": "8.3.1",
    "rescript": "9.1",
    "tailwindcss": "^3.0.23"
  }
}
```

========== End file

========== Start file App.res

```
// This type is based on the getInitialProps return value.
// If you are using getServerSideProps or getStaticProps, you probably
// will never need this
// See https://nextjs.org/docs/advanced-features/custom-app
type pageProps

module PageComponent = {
  type t = React.component<pageProps>
}

type props = {
  @as("Component")
  component: PageComponent.t,
  pageProps: pageProps,
}

// We are not using `@react.component` since we will never
// use <App/> within our ReScript code.
// It's only used within `pages/_app.js`
let default = (props: props): React.element => {
  let {component, pageProps} = props

  let router = Next.Router.useRouter()

  let content = React.createElement(component, pageProps)

  switch router.route {
  | "/examples" => <MainLayout> content </MainLayout>
  | _ => <MainLayout> content </MainLayout>
  }
}
```

========== End file

========== Start file App.resi

```
type props
let default: props => React.element
```

========== End file