REASON

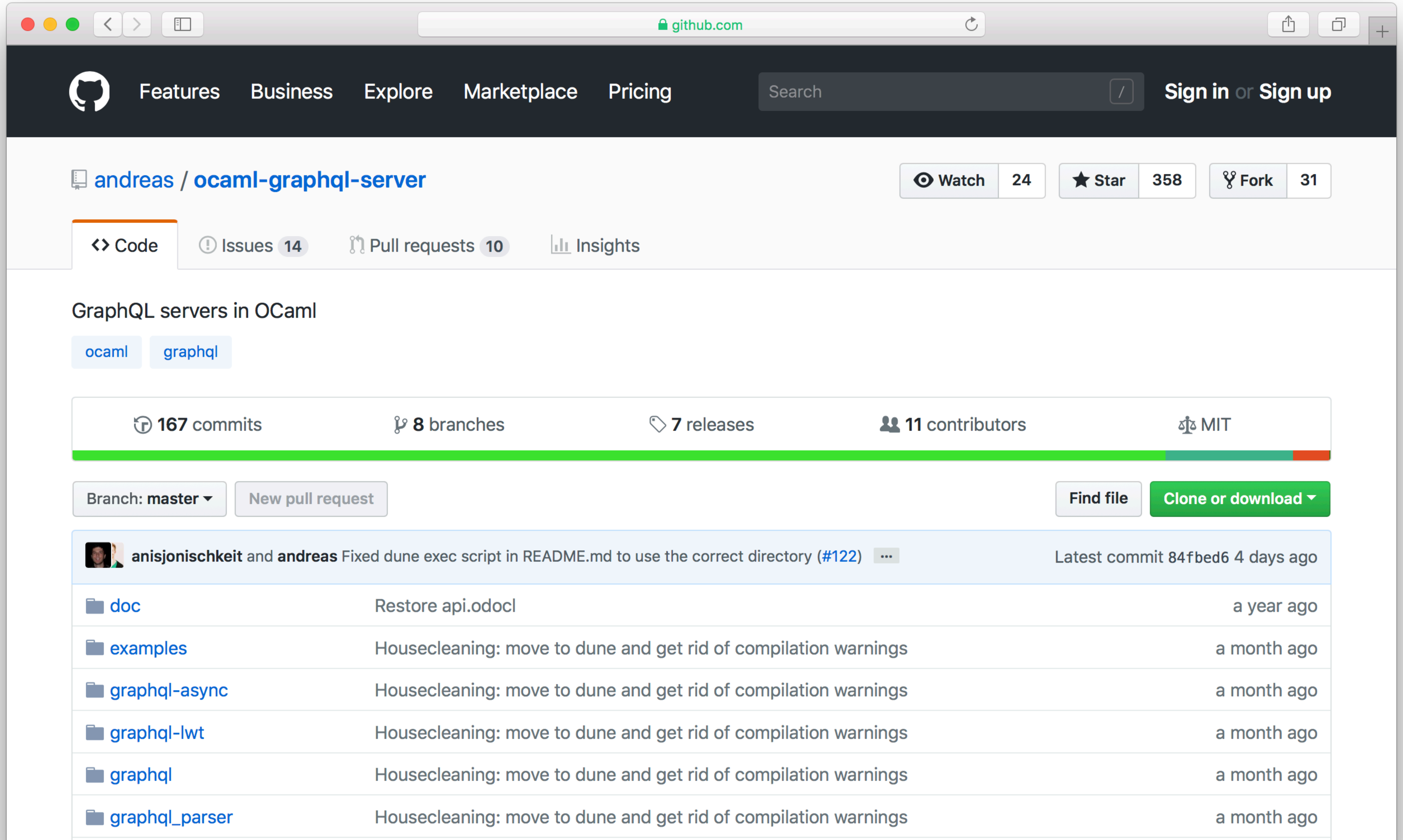
```
let fruits = ["Apple", "Orange"];
```

GraphQL

Goal:

full stack GraphQL in Reason

Let’s go native



andreas / ocaml-graphql-server

GraphQL servers in OCaml

anisjonischkeit and andreas Fixed dune exec script in README.md to use the correct directory (#122) Latest commit 84fbed6 4 days ago

doc	Restore api.odocl	a year ago
examples	Housecleaning: move to dune and get rid of compilation warnings	a month ago
graphql-async	Housecleaning: move to dune and get rid of compilation warnings	a month ago
graphql-lwt	Housecleaning: move to dune and get rid of compilation warnings	a month ago
graphql	Housecleaning: move to dune and get rid of compilation warnings	a month ago
graphql_parser	Housecleaning: move to dune and get rid of compilation warnings	a month ago

Benefits

- The type of a field agrees with the return type of the resolve function.
- The arguments of a field agrees with the accepted arguments of the resolve function.
- The source of a field agrees with the type of the object to which it belongs.
- The context argument for all resolver functions in a schema agree.

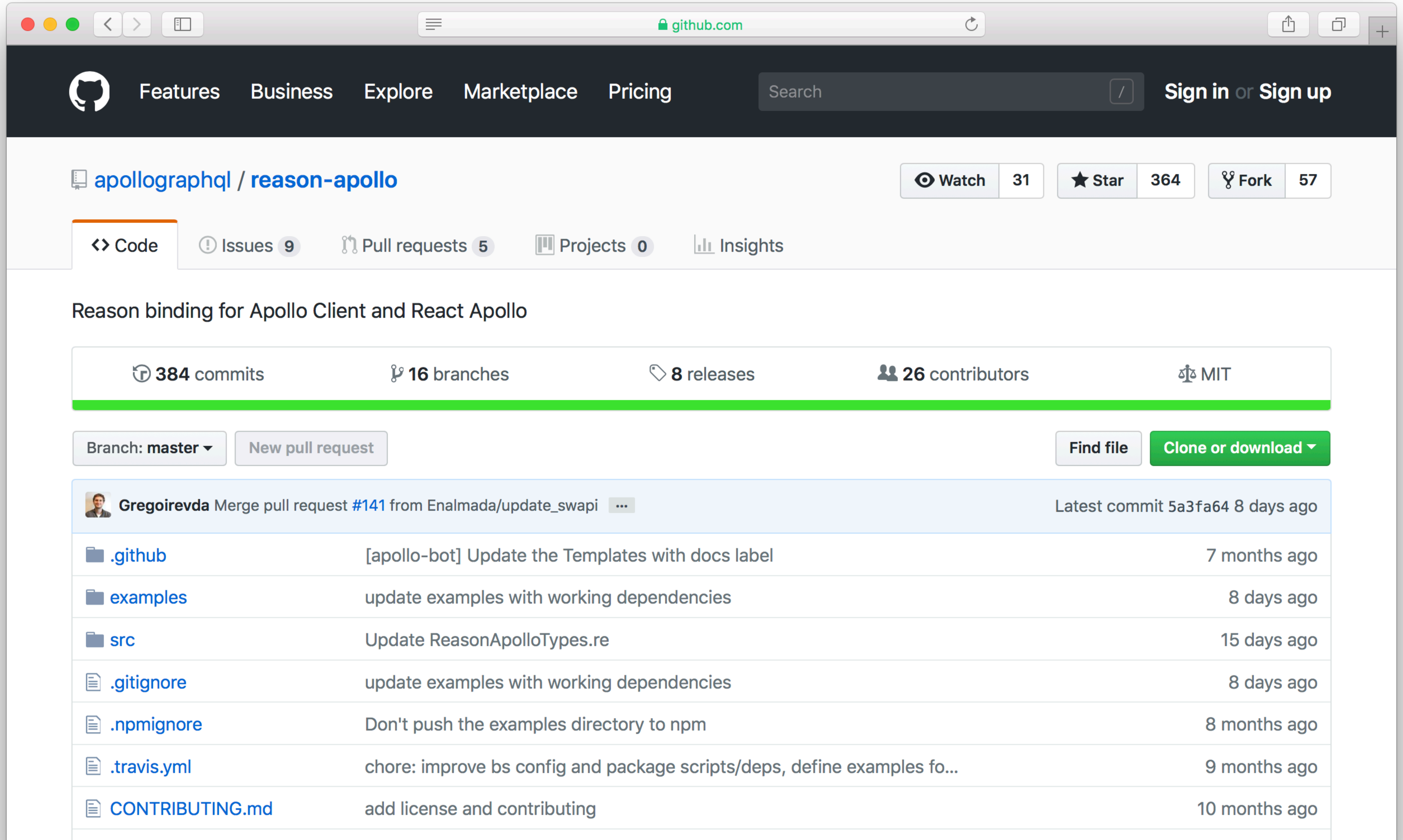
```
opam install graphql-lwt dune

git clone git@github.com:andreas/ocaml-graphql-server.git
cd ocaml-graphql-server/examples

dune exec ./server.exe
```

```
type role = User | Admin

type user = {
  id   : int;
  name : string;
  role : role;
  friends : user list;
}

let rec alice = { id = 1; name = "Alice"; role = Admin; friends = [bob] }
and bob = { id = 2; name = "Bob"; role = User; friends = [alice]}

let users = [alice; bob]

let role = Schema.(enum "role"
  ~values:[
    enum_value "USER" ~value:User ~doc:"A regular user";
    enum_value "ADMIN" ~value:Admin ~doc:"An admin user";
  ]
)

let user = Schema.(obj "user"
  ~fields:(fun user -> [
    field "id"
      ~args:Arg.[]
```

GraphiQL
Prettify
Docs
QUERY VARIABLES

```
{

}
```

Client?



apollographql / reason-apollo

Reason binding for Apollo Client and React Apollo

384 commits | 16 branches | 8 releases | 26 contributors | MIT

Gregoirevda Merge pull request #141 from Enalmada/update_swapi ... Latest commit 5a3fa64 8 days ago

.github	[apollo-bot] Update the Templates with docs label	7 months ago
examples	update examples with working dependencies	8 days ago
src	Update ReasonApolloTypes.re	15 days ago
.gitignore	update examples with working dependencies	8 days ago
.npmignore	Don't push the examples directory to npm	8 months ago
.travis.yml	chore: improve bs config and package scripts/deps, define examples fo...	9 months ago
CONTRIBUTING.md	add license and contributing	10 months ago

```
yarn add reason-apollo

# Add graphql_ppx
yarn add --dev graphql_ppx

# Add JS dependencies
yarn add react-apollo apollo-client graphql …
```

```
"bs-dependencies": [
  "reason-react",
  "reason-apollo"
],
"ppx-flags": [
    "graphql_ppx/ppx"
]
```

```
yarn send-introspection-query http://localhost:8080/graphql
```

graphql_schema.json

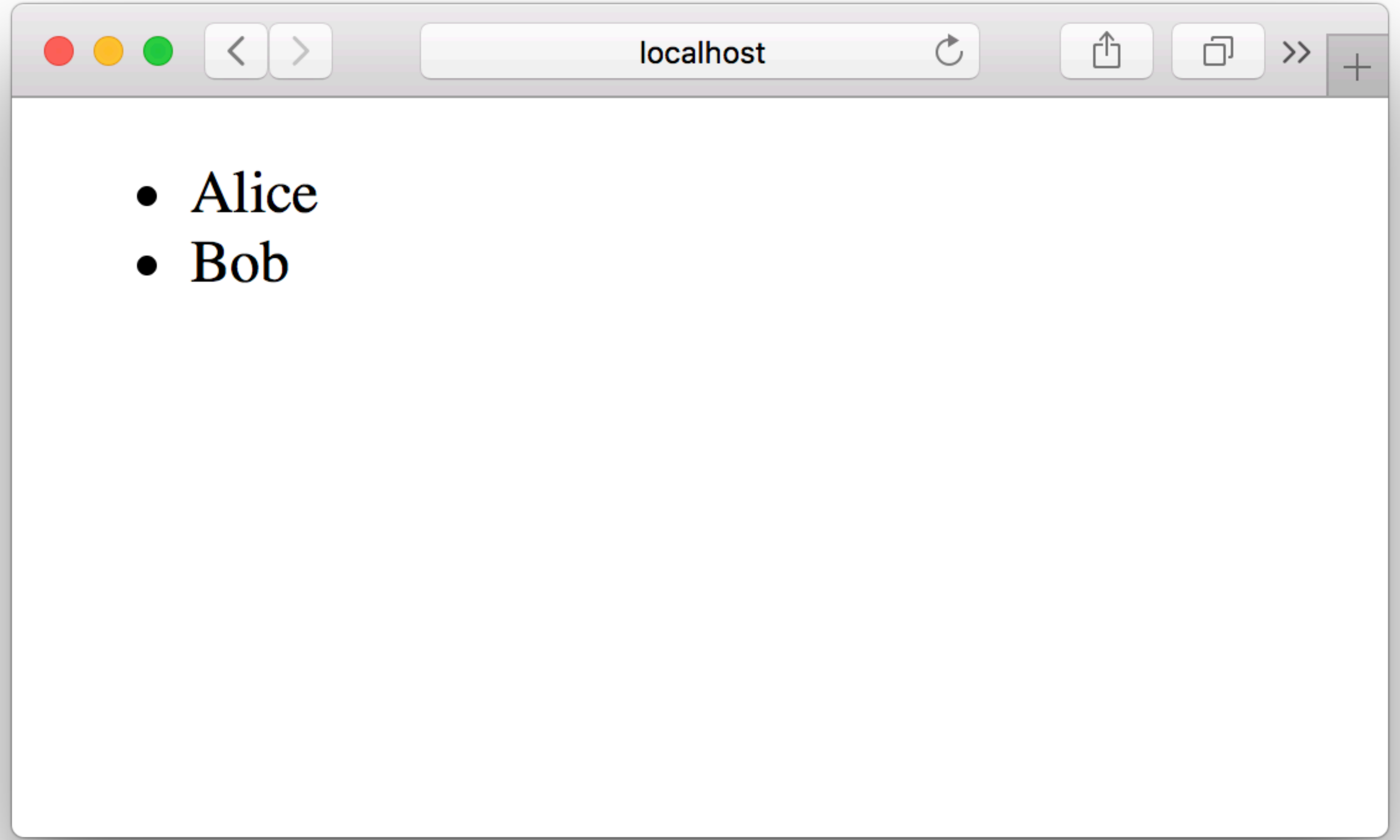
```json
{
  "data": {
    "__schema": {
      "queryType": {
        "name": "query"
      },
      "mutationType": null,
      "subscriptionType": {
        "name": "subscription"
      },
      "types": [
        {
          "kind": "OBJECT",
          "name": "subscription",
          "description": null,
          "fields": [
            {
              "name": "subscribe_to_user"
```

Instantiate a Apollo Client

```
let inMemoryCache = ApolloInMemoryCache.createInMemoryCache();

let httpLink = ApolloLinks.createHttpLink(~uri="/api/graphql", ());

let instance = ReasonApollo.createApolloClient(
  ~link=httpLink, ~cache=inMemoryCache, ()
);
```

Add a Apollo Provider

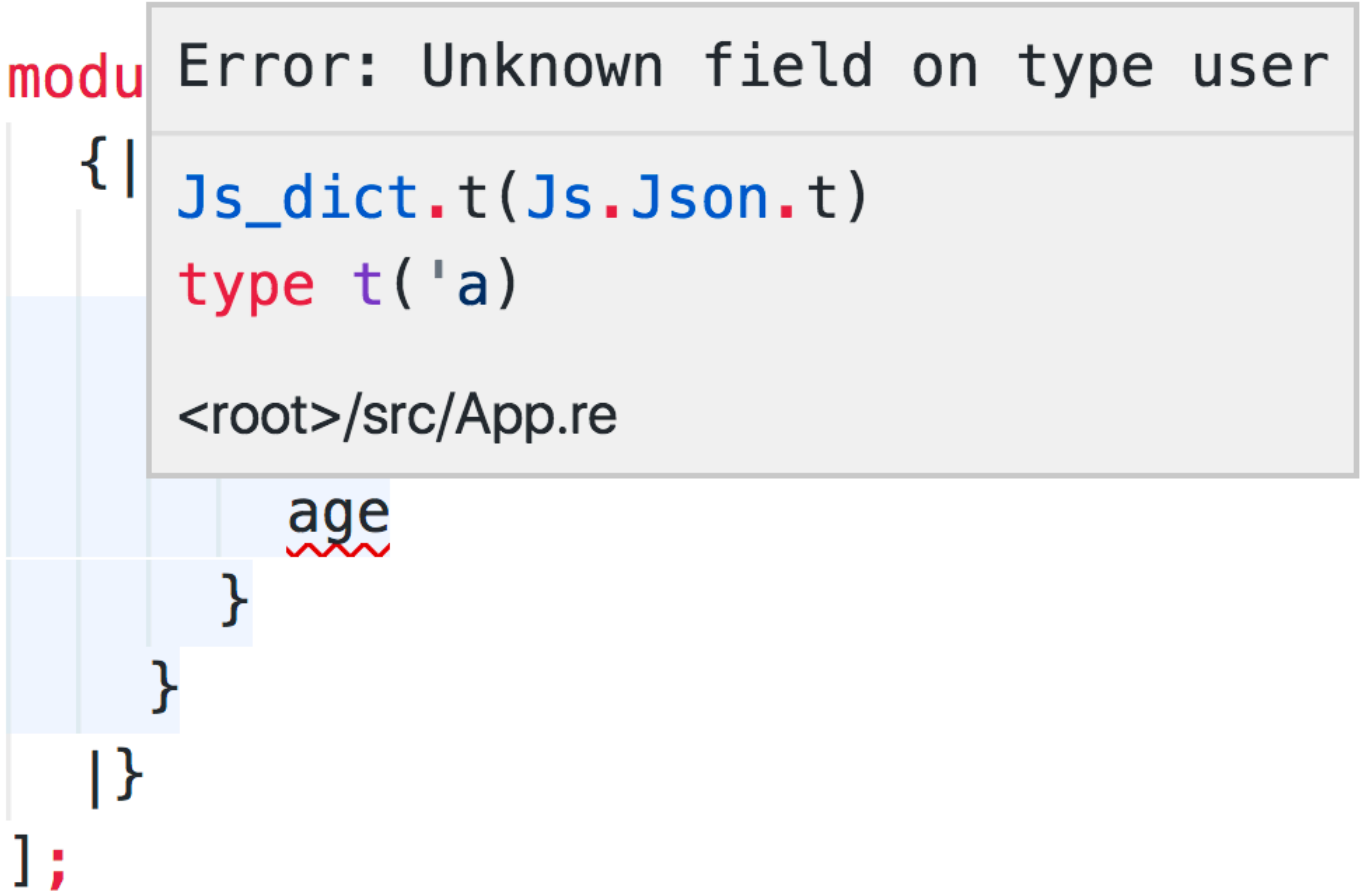
```
ReactDOMRe.renderToElementWithId(
  <ReasonApollo.Provider client=Client.instance>
    <App />
  </ReasonApollo.Provider>,
  "root",
);
```



- Alice
- Bob

```
module GetUsers = [%graphql
  {|
    query users {
      users {
        id
        name
      }
    }
  |}
];
```

```
module GetUsers = [%graphql
  {|
    query users {
      users {
        id
        age
      }
    }
  |}
];
```



```
modu
  {|
      age
    }
  }
  |}
];
```

```
Error: Unknown field on type user
```

```
Js_dict.t(Js.Json.t)
type t('a)
```

<root>/src/App.re

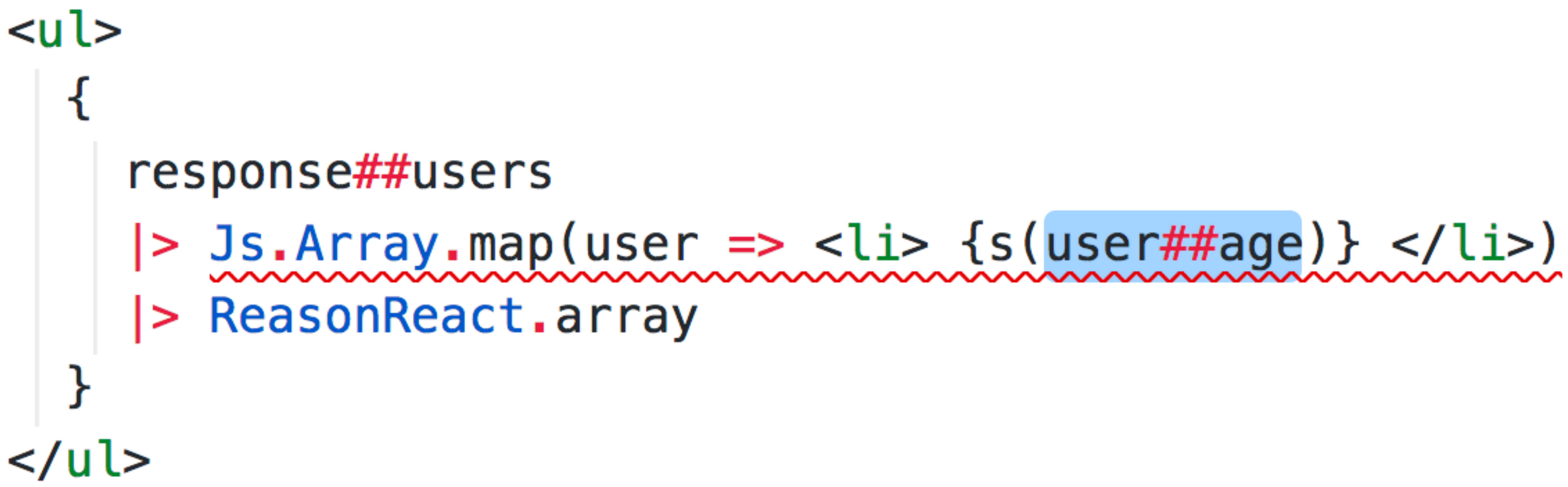
```
module GetUsers = [%graphql
  {|
    query users {
      users {
        id
        name
      }
    }
  |}
];

module GetUsersQuery = ReasonApollo.CreateQuery(GetUsers);
```

```
let make = _children => {
  ...component,
  render: _ =>
    <GetUsersQuery>
      ...{
           ({result}) =>
             switch (result) {
             | Loading => <div> {s("Loading")} </div>
             | Error(error) => <div> {s(error##message)} </div>
             | Data(response) =>
               <ul>
                 {
                   response##users
                   |> Js.Array.map(user => <li> {s(user##name)} </li>)
                   |> ReasonReact.array
                 }
               </ul>
             }
         }
    </GetUsersQuery>,
};
```

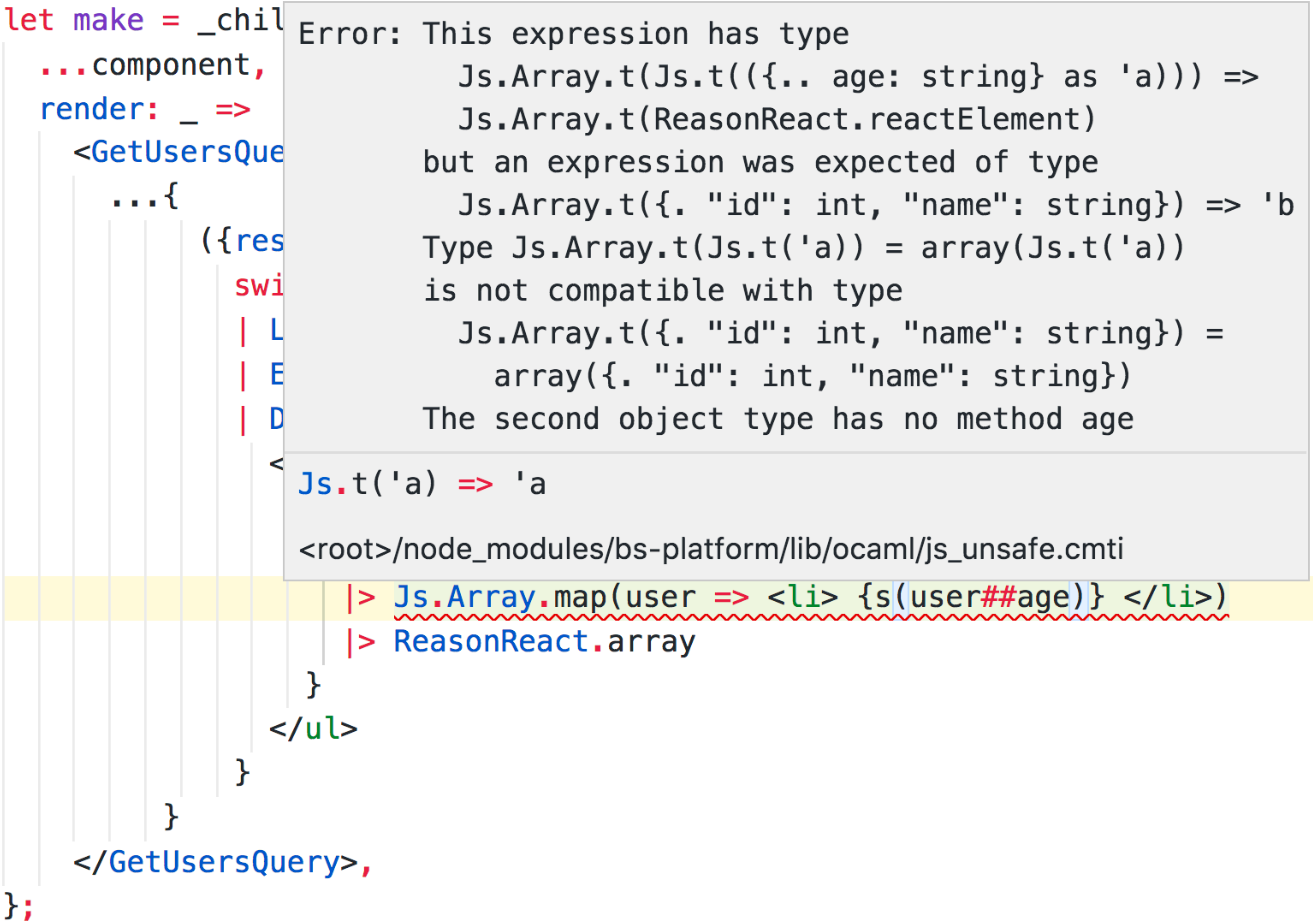
```
let make = _children => {
  ...component,
  render: _ =>
    <GetUsersQuery>
      ...{
           ({result}) =>
             switch (result) {
             | Loading => <div> {s("Loading")} </div>
             | Error(error) => <div> {s(error##message)} </div>
             | Data(response) =>
               <ul>
                 {
                   response##users
                   |> Js.Array.map(user => <li> {s(user##name)} </li>)
                   |> ReasonReact.array
                 }
               </ul>
             }
         }
    </GetUsersQuery>,
};
```

```
let make = _children => {
  ...component,
  render: _ =>
    <GetUsersQuery>
      ...{
          ({result}) =>
            switch (result) {
            | Loading => <div> {s("Loading")} </div>
            | Error(error) => <div> {s(error##message)} </div>
            | Data(response) =>
              <ul>
                {
                  response##users
                  |> Js.Array.map(user => <li> {s(user##name)} </li>)
                  |> ReasonReact.array
                }
              </ul>
            }
        }
    </GetUsersQuery>,
};
```

```
<ul>
  {
    response##users
    |> Js.Array.map(user => <li> {s(user##name)} </li>)
    |> ReasonReact.array
  }
</ul>
```



```
<ul>
  {
    response##users
    |> Js.Array.map(user => <li> {s(user##age)} </li>)
    |> ReasonReact.array
  }
</ul>
```



```
let make = _chil
  ...component,
  render: _ =>
    <GetUsersQue
      ...{
          ({res
            swi
            | L
            | E
            | D
              <


                  |> Js.Array.map(user => <li> {s(user##age)} </li>)
                  |> ReasonReact.array
                }
              </ul>
            }
          }
    </GetUsersQuery>,
};
```

```
Error: This expression has type
         Js.Array.t(Js.t(({.. age: string} as 'a))) =>
         Js.Array.t(ReasonReact.reactElement)
       but an expression was expected of type
         Js.Array.t({. "id": int, "name": string}) => 'b
       Type Js.Array.t(Js.t('a)) = array(Js.t('a))
       is not compatible with type
         Js.Array.t({. "id": int, "name": string}) =
           array({. "id": int, "name": string})
       The second object type has no method age

Js.t('a) => 'a

<root>/node_modules/bs-platform/lib/ocaml/js_unsafe.cmti
```

Let’s do it in TypeScript

1. Unique names for all your queries and mutation … (per directory?)

2. Download the schema

```
apollo schema:download –endpoint=http://example.com graphql-schema.json
```

3. Generate the types

```
apollo codegen:generate genTypes --schema=graphql-schema.json –
queries=‘packages/**/src/**/*.ts*’ --passthroughCustomScalars --
customScalarsPrefix=GraphQl --addTypename --globalTypesFile=./packages/
types/src/global-graphql.ts
```

4. Import the Type and extend the Component

```
import { UsersQuery } from "./genTypes/UsersQuery";

const USERS_QUERY = gql`
  query UsersQuery {
    users {
      id
      name
    }
  }
`;

export default () => (
  <Query<UsersQuery> query={USERS_QUERY}>
    {({ loading, error, data }) => {
      if (loading) return <div>Loading…</div>;
      if (error) return <div>Error</div>;
      if (!data) return null; // NOTE guarding that data is not null
      <ul>
        {data.users.map(user => (
          <li key={user.id}>{user.name}</li>
        ))}
      </ul>;
    }}
  </Query>
);
```

```
import { UsersQuery } from "./genTypes/UsersQuery";

const USERS_QUERY = gql`
  query UsersQuery {
    users {
      id
      name
    }
  }
`;

export default () => (
  <Query<UsersQuery> query={USERS_QUERY}>
    {({ loading, error, data }) => {
      if (loading) return <div>Loading…</div>;
      if (error) return <div>Error</div>;
      if (!data) return null; // NOTE guarding that data is not null
      <ul>
        {data.users.map(user => (
          <li key={user.id}>{user.name}</li>
        ))}
      </ul>;
    }}
  </Query>
);
```

```
import { UsersQuery } from "./genTypes/UsersQuery";

const USERS_QUERY = gql`
  query UsersQuery {
    users {
      id
      name
    }
  }
`;

export default () => (
  <Query<UsersQuery> query={USERS_QUERY}>
    {({ loading, error, data }) => {
      if (loading) return <div>Loading…</div>;
      if (error) return <div>Error</div>;
      if (!data) return null; // NOTE guarding that data is not null
      <ul>
        {data.users.map(user => (
          <li key={user.id}>{user.name}</li>
        ))}
      </ul>;
    }}
  </Query>
);
```

Is it perfect?

Things I like to see

- Records instead of objects
- Lists instead of Js.Array
- Correct auto-completion inside the GraphQL PPX
- Formatting of PPX

The End