Why not Relay?

A GraphQL and React story

require('lx') 20.02.2020

onfido

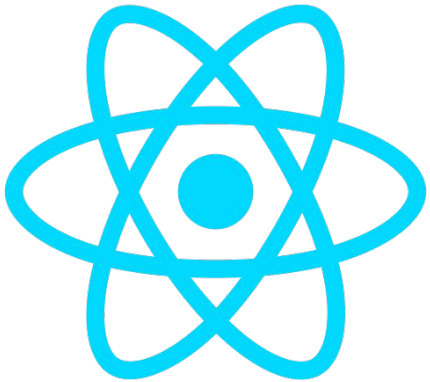
onfido

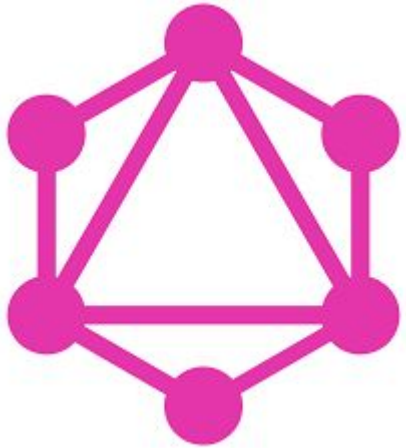
Giacomo Magini

Software Engineer
(lots of Front-End)

onfido

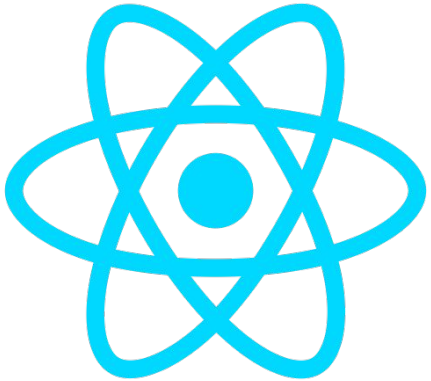
Who developed these two technologies you really love?

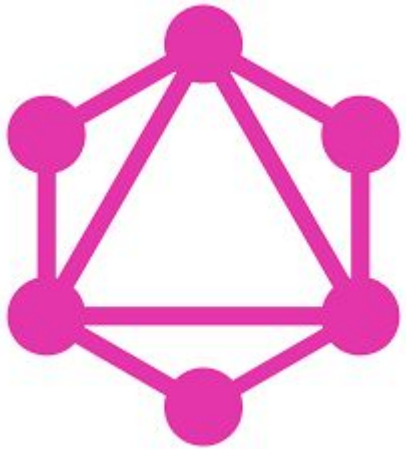




onfido

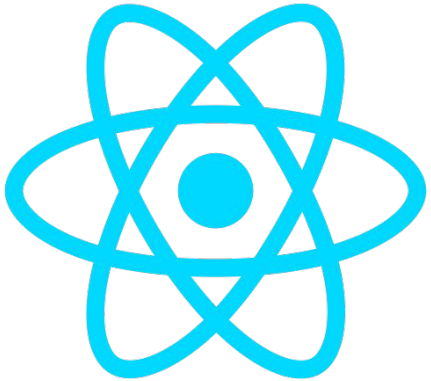
Relay?

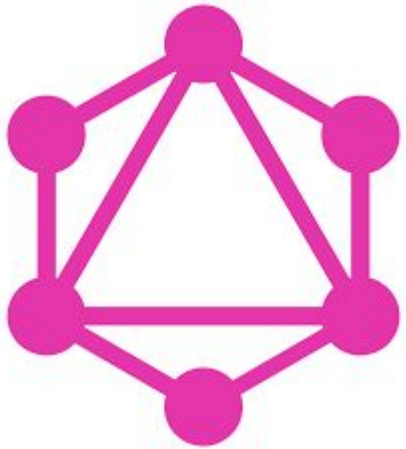




onfido

Apollo!





onfido

onfido

Is there any Apollo’s feature that Relay is missing?

onfido

Fetching data

Apollo

```
function Dogs({ breed }) {
  const { loading, error, data } = useQuery(GET_DOGS, {
    variables: { breed },
  });

  if (loading) return 'Loading...';
  if (error) return `Error! ${error}`;

  // Use data.dogs
  ...
}
```

onfido

Fetching data

Relay

```
function renderQuery({error, props}){
  if (error) {
    return `Error! ${error.message}`;
  } else if (props) {
    // Use data.dogs
    ...
  }
  return 'Loading...';
}

function Dogs({ breed }) {
  return (
    <QueryRenderer
      environment={environment}
      query={GET_DOGS}
      variables={{ breed }}
      render={renderQuery}
    />
  );
}
```

onfido

Updating data

Apollo

```
function AddDog() {
  const [addDog, { data, loading, error }] = useMutation(ADD_DOG);
  ...
  addDog({ variables: { breed: 1, ... }});
}
```

onfido

Updating data

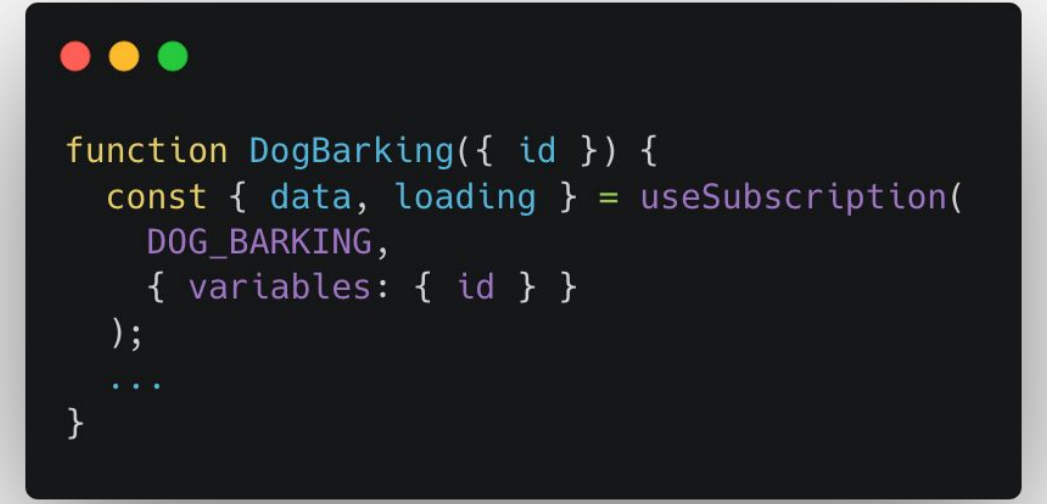
Relay

```
function addDog(environment, variables){
  commitMutation(
    environment,
    {
      ADD_DOG,
      variables,
      onCompleted: (response, errors) => {
        console.log('Response received from server.')
      },
      onError: err => console.error(err),
    },
  );
}

// React Component
function AddDog(environment) {
  ...
  addDog(environment, { breed: 1, ... });
}
```

onfido

Subscriptions

Apollo



```
function DogBarking({ id }) {
  const { data, loading } = useSubscription(
    DOG_BARKING,
    { variables: { id } }
  );
  ...
}
```

onfido

Subscriptions

Relay

```
function requestDogBarkingSubscription(environment, { id }){
  return requestSubscription(
    environment,
    {
      DOG_BARKING,
      { id },
      updater: store => { /* UPDATE THE QUERY CACHE */ }
      onCompleted: () => {/* server closed the subscription */},
      onError: error => console.error(error),
    }
  );
}
```

onfido

Others

Apollo & Relay

- Caching
- Optimistic update
- Local data
- Pagination

- Yes, Relay's API looks cluttered and verbose.
- Apollo is easier to integrate in the component lifecycle, thanks to hooks.
- Apollo has better docs.
- Apollo is just a library, while Relay has two packages, relay-compiler and relay-runtime.

onfido

Is there any Apollo's feature that Relay is missing?

NO

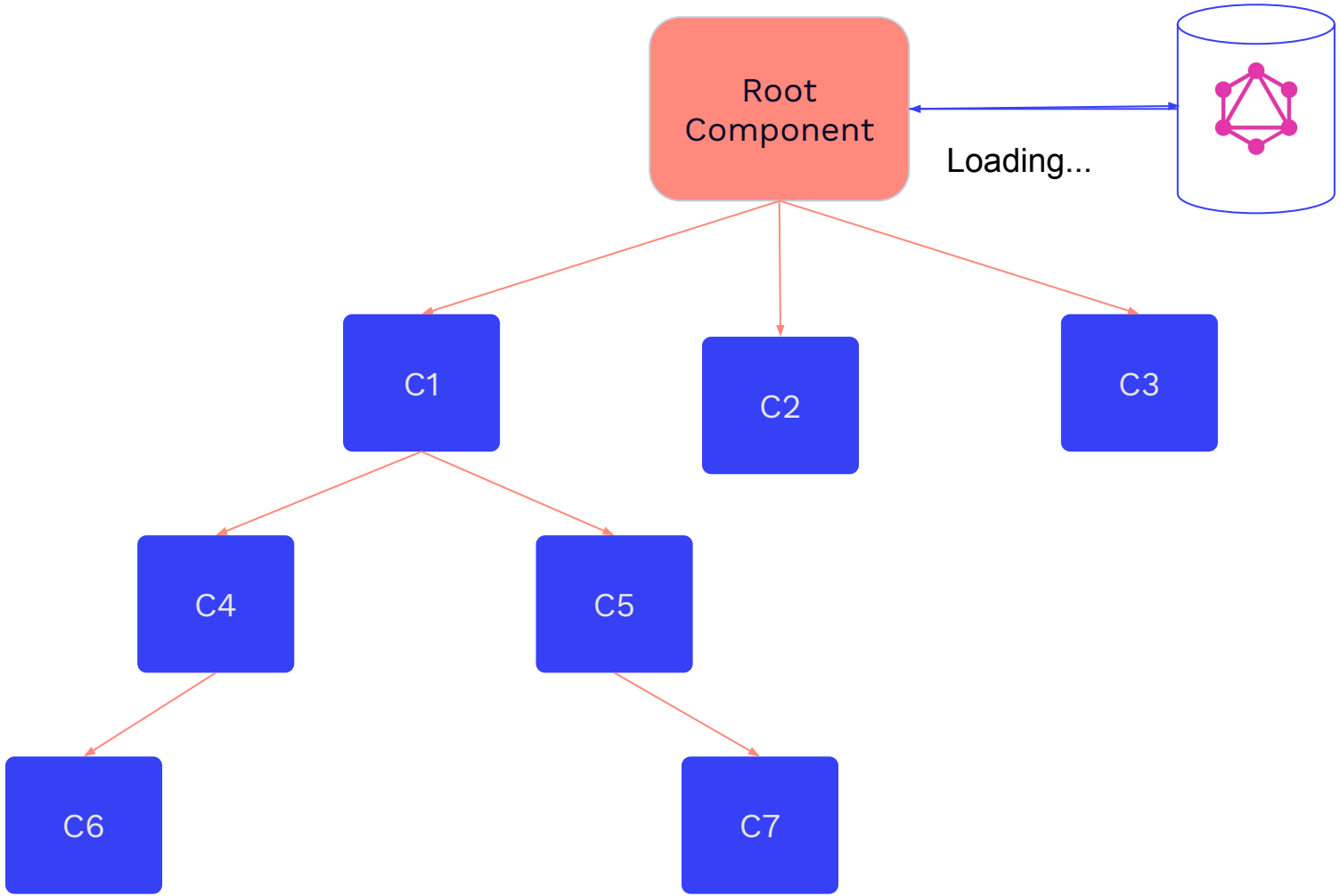
onfido

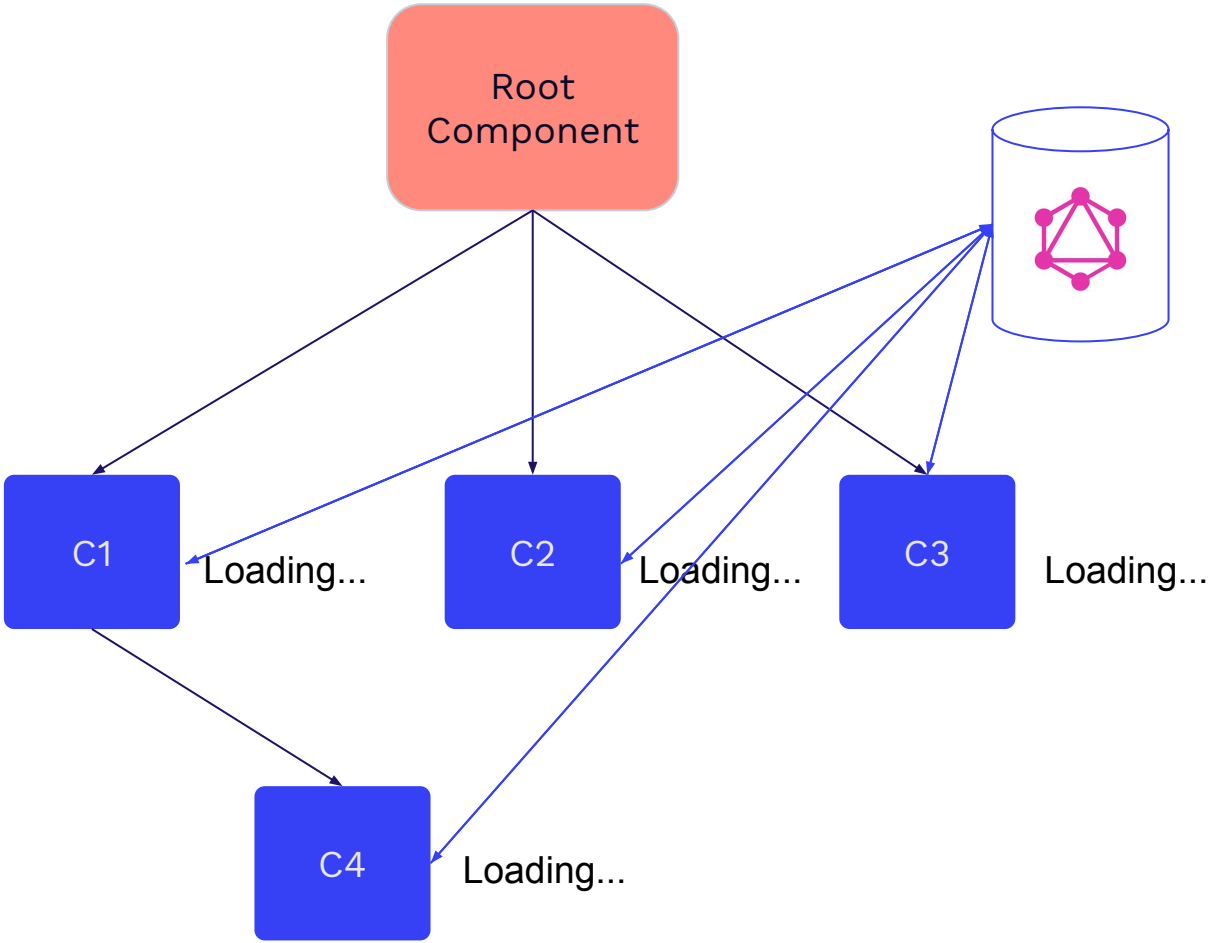
No seriously, why Relay?

onfido

Thinking in Realy

1. Fetching Data For a View
2. Data Components aka Containers
3. Render
4. Data Masking


Root Component
Loading...
C1
C2
C3
C4
C5
C6
C7


Root Component
C1
Loading...
C2
Loading...
C3
Loading...
C4
Loading...

onfido

Thinking in Realy

1. Fetching Data For a View
2. Data Components aka Containers
3. Render
4. Data Masking

onfido

More than a library

1. relay-compiler
2. relay-runtime

onfido

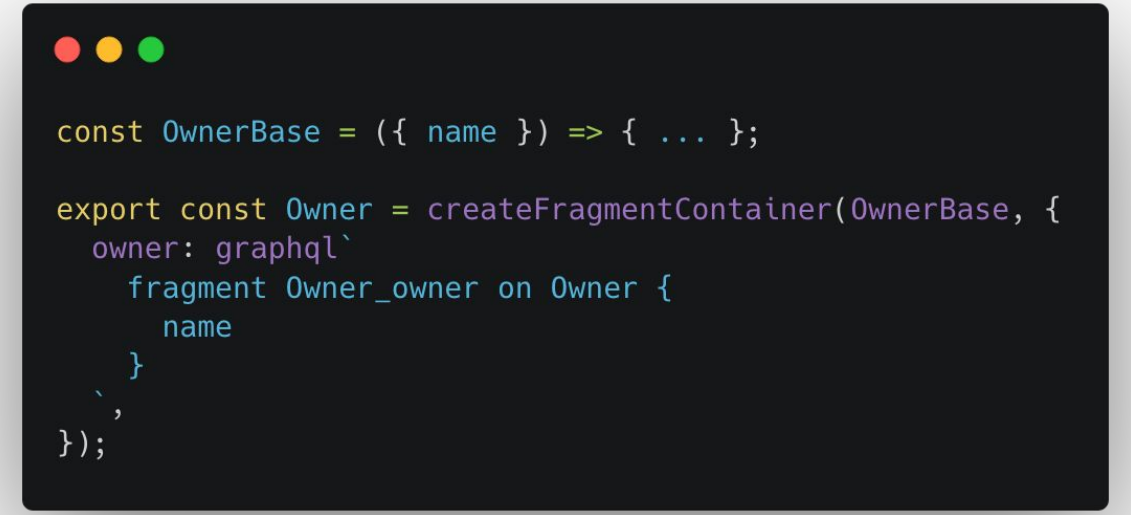
Why and when Relay

Finally

onfido

Colocation = Fragments

It’s all about fragments



```
const OwnerBase = ({ name }) => { ... };

export const Owner = createFragmentContainer(OwnerBase, {
  owner: graphql`
    fragment Owner_owner on Owner {
      name
    }
  `,
});
```

onfido

Include the fragment into a Query

It's all about fragments

```
const renderQuery = ({ error, props }) => {
  if (error) {
    return <div>{error.message}</div>;
  } else if (props) {
    return <Owner owner={props.dog.owner} />;
  }
  return <div>Loading</div>;
};

export const Dog = ({ key }) => {
  return (
    <QueryRenderer
      environment={environment}
      query={graphql`
        query DogQuery($key: ID!) {
          dog(key: $key) {
            key
            owner {
              ...Owner_owner
            }
          }
        }
      `}
      variables={{ key }}
      render={renderQuery}
    />
  );
};
```

onfido

Run compiler

```
yarn run v1.9.4
$ relay-compiler --src ./src --schema ./schema.graphql --extensions js jsx

Writing js
ERROR:
Unknown fragment 'Likes_dog'.
error Command failed with exit code 100.
```

onfido

Fragments composition

It’s all about fragments part 2

```
const OwnerBase = ({ name }) => { ... };

export const Owner = createFragmentContainer(OwnerBase, {
  owner: graphql`
    fragment Owner_owner on Owner {
      name
    }
  `,
});
```

onfido

Fragments composition

It’s all about fragments part 2

```
const OwnerAddressBase = ({ address }) => { ... };

export const OwnerAddress = createFragmentContainer(OwnerAddressBase, {
  address: graphql`
    fragment OwnerAddress_address on Address {
      street
      zip
      number
      city
    }
  `,
});
```

onfido

Fragments composition

It's all about fragments part 2

```
const OwnerBase = ({ name, addressList }) => {
  ...
  return addressList.map(address => (
    <OwnerAddress address={address}></OwnerAddress>
  ));
};

export const Owner = createFragmentContainer(OwnerBase, {
  owner: graphql`
    fragment Owner_owner on Owner {
      name
      addressList {
        ...OwnerAddress_address
      }
    }
  `
});
```

onfido

Type Emission

Typing is goooooood!

relay-compiler generates types based on your fragments and query.
By default it uses **Flow** but you can use **Typescript**.

It generates file alongside your **Fragment Container** components within the folder **__generated__**

onfido

Type Emission

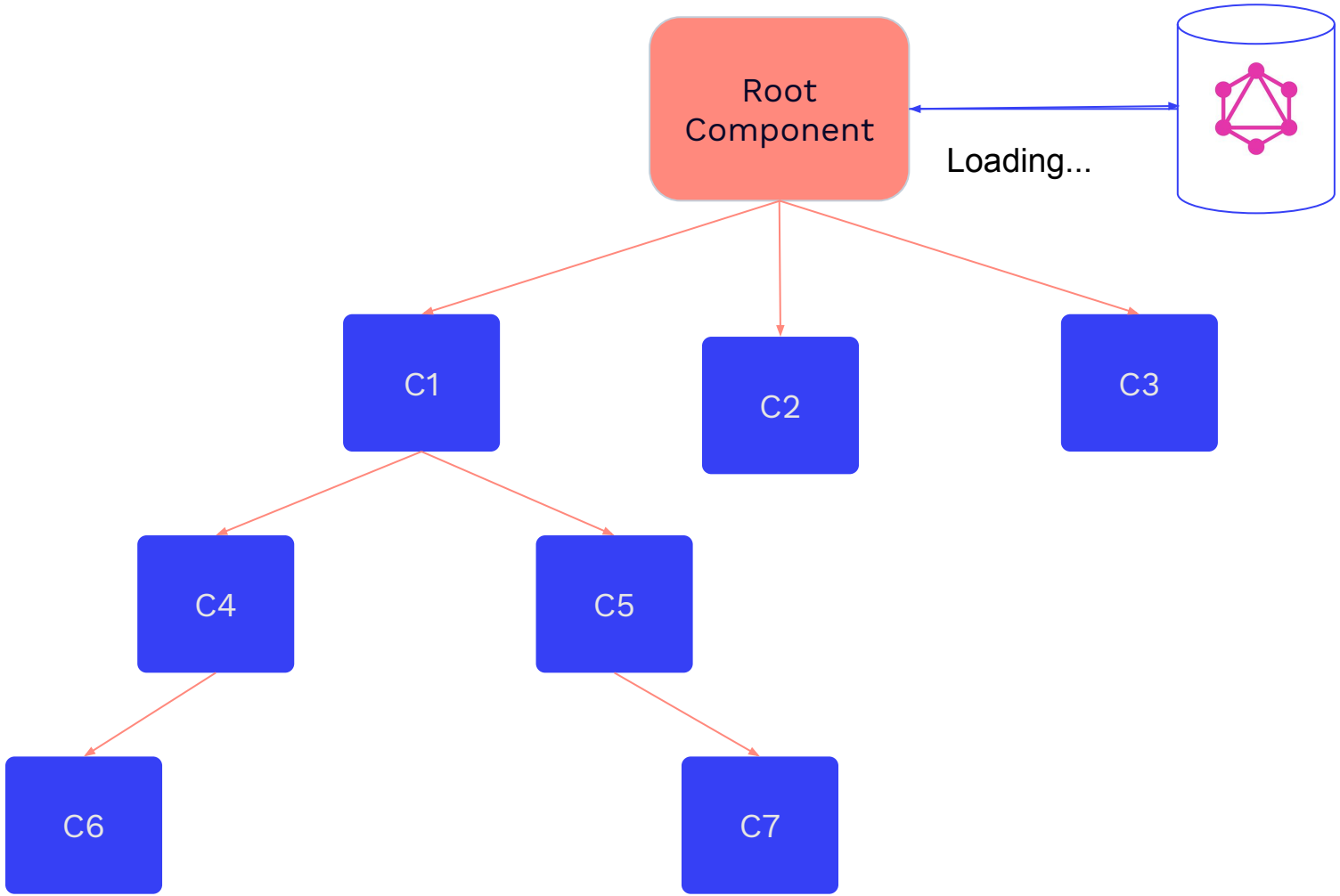
Typing is goooooood!

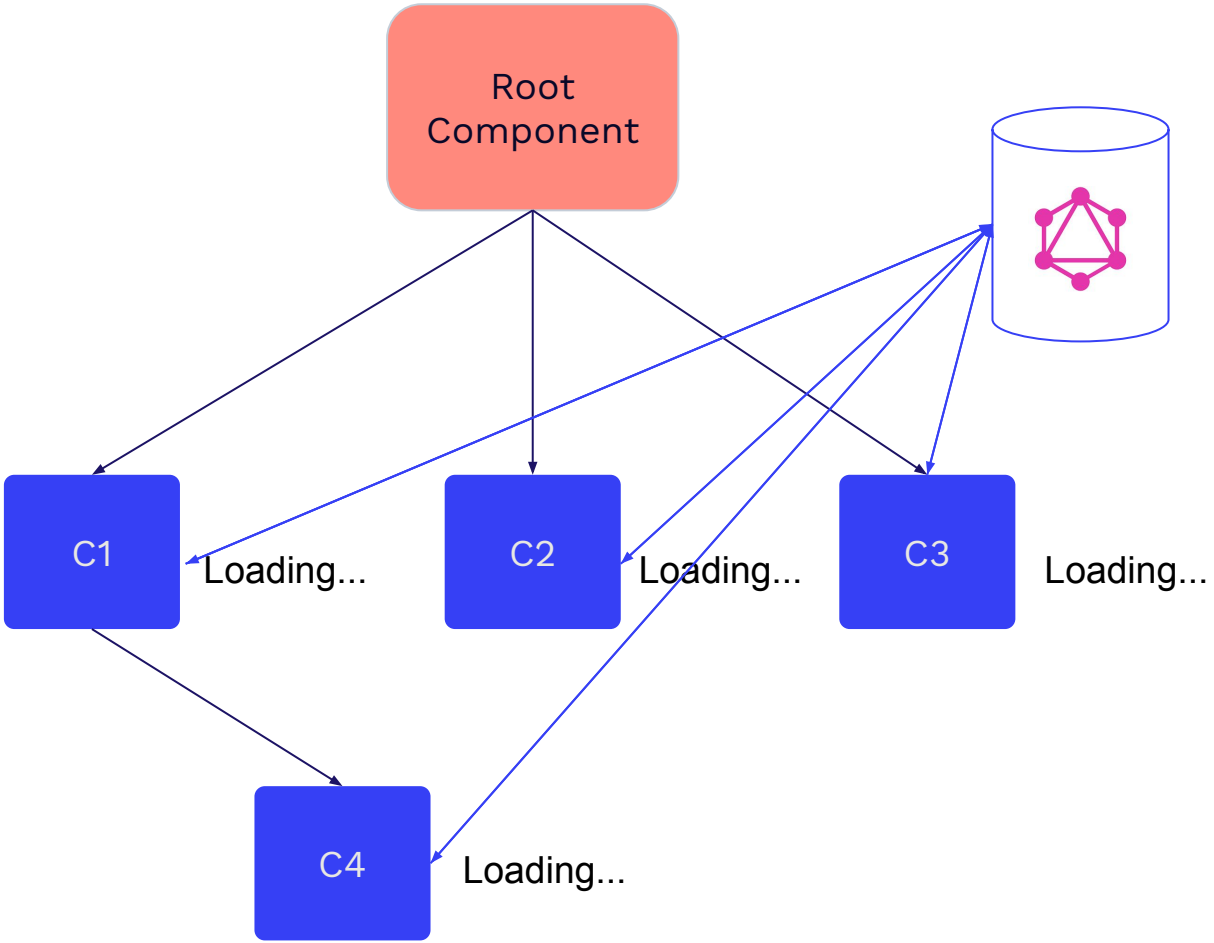
```
/**
 * export type OwnerAddress_address = {
 *   readonly street: string
 *   readonly zip: string
 *   readonly number: string
 *   readonly city: string
 * }
 */
import { OwnerAddress_address } from "__generated__/OwnerAddress_address.graphql"

const OwnerAddressBase = ({ address } : { address: OwnerAddress_address }) => { ... };

export const OwnerAddress = createFragmentContainer(OwnerAddressBase, {
  address: graphql`
    fragment OwnerAddress_address on Address {
      street
      zip
      number
      city
    }
  `,
});
```

onfido

Summary

1. Data colocation decouples from the root component running the query and avoids implicit dependencies.
2. Types help you to develop fast and safe
3. Relay-compiler optimise the query and enforce all the frameworks rules


Root Component
Loading...
C1
C2
C3
C4
C5
C6
C7


Root Component
C1
Loading...
C2
Loading...
C3
Loading...
C4
Loading...

onfido

Relay seems really powerful for **large applications**

onfido

Future of Relay

https://relay.dev/docs/en/experimental/step-by-step

Hooks

- RelayEnvironmentProvider
- useRelayEnvironment
- **usePreloadedQuery**
- useLazyLoadQuery
- useFragment
- useRefetchableFragment
- usePaginationFragment
- useBlockingPaginationFragment

Suspense

Loading state will be fully managed with **React Suspense**. Synchronise multiple loading states of the app will be super easy.

Thanks for listening!

👏👏👏👏

onfido