# Create component with custom data API query incl. filters (GraphQL)

How to run a custom GraphQL query in your react component.

## Vendor bundle

Your vendor bundle must contain Apollo client see also https://www.npmjs.com/package/apollo-client.
To install run:

```
npm install apollo-boost graphql-tag graphql --save
```

Your index.js file then need the following import:

```
import gql from 'graphql-tag';
window.gql = gql;
```

After that run webpack and upload the bundle to your app.
unfamiliar with bundles? see: https://github.com/bettyblocks/cli/wiki/Tutorial:-Use-libraries

## Code

Replace "queryKey" with the model you like to get, oneArtist for one record, allArtist for multiple records.
Edit "select" with the properties you like to receive.
Edit "where" with the conditions you like to filter the data, "matches" can be replaced with for instance: "eq", "in", "starts_with" etc… See all filter types below.

```
const {gql} = window;
const {GetOneProvider, Query, env, getModel} = B;

const runQuery = () => {
      const queryKey = "allEntity"; //all<modelnaam> or one<modelnaam>
      const select = "results{id,name}"; //properties
      const q = "query ($skip: Int, $take: Int) {\
        " + queryKey + "(skip: $skip, where: $where) {\
          " + select + " \
        }\
      }";

      const g = gql(q);
      const where = {"where":{
          "<PROPERTYNAME>":{"matches":<VALUE>}},
          "skip":0,
          "take":1
      }
      return(
        <div>
          <Query query={g} variables={where}>
            {({data, loading, error}) => {
              if(loading || error) {
                return <div>LOADING</div>
              }
              console.log(data); //See how your data looks like.
              return <div>HERE YOU WILL OUTPUT YOUR HTML, .MAP() OR
    SIMILAR</div>;
            }}
          </Query>
        </div>
      );
    }
```

## Filter types

| in | if a value is in an array |
|---|---|
| nin | if a value is not in an array |
| eq | if value equals to |
| neq | if value not equals to |
| regex | if value matches a regex |
| starts_with | if value starts with |
| ends_with | if value ends with |
| matches | if value contains a value or string |
| does_not_match | if value does not contain a value or string |