Express & Sequelize

Rounding Out

Express & Sequelize

An assortment of tips, tricks, time savers and conversation starters...

What we will cover

- **Express**
  - Custom error handler
  - 404 Not Found
  - Template engines
- **Sequelize**
  - Eager Loading
  - Class methods / instance methods
  - Many-Many Relationships

Express: Custom Error Handler

Express: Custom Error Handler

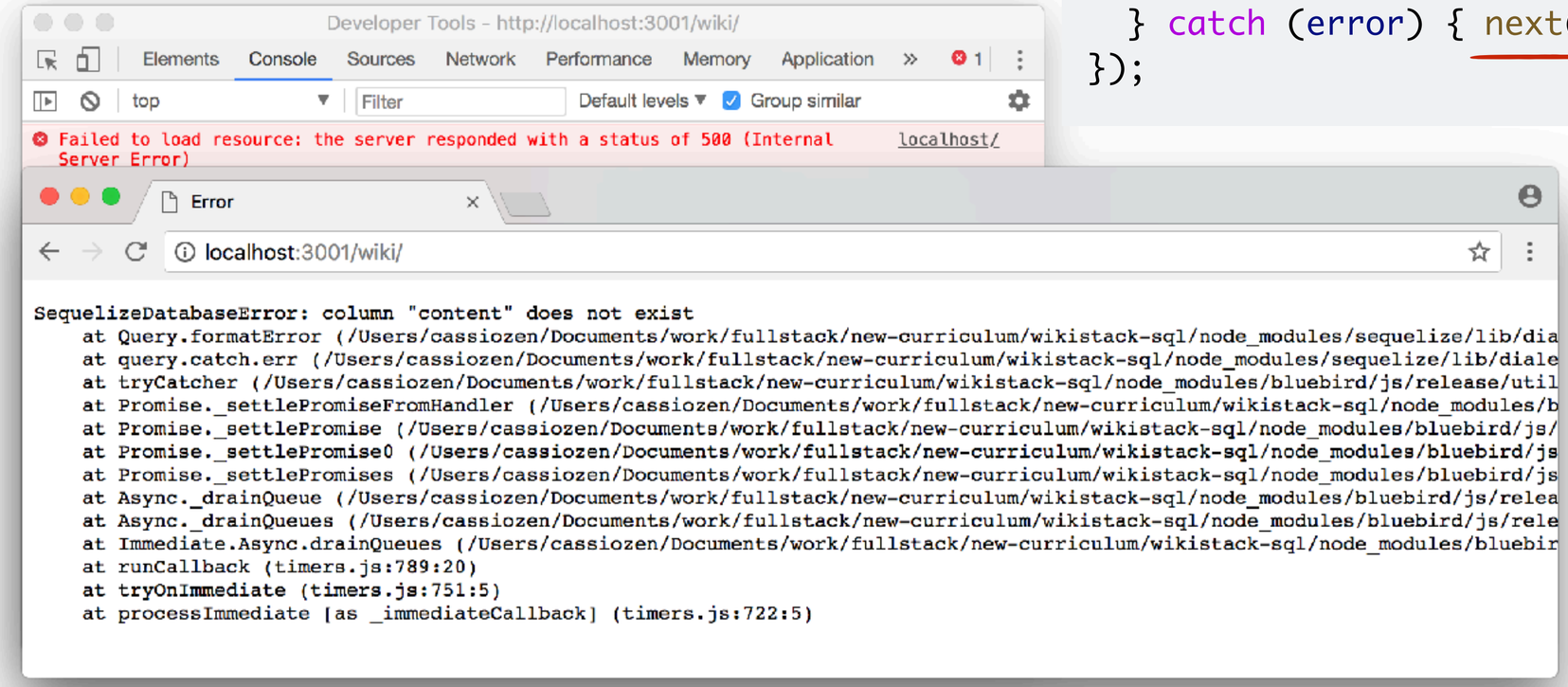
Express comes with a built-in error handler

```
router.get("/", async (req, res, next) => {
  try {
    const users = await User.findAll();
    res.send(userList(users));
  } catch (error) { next(error) }
});
```

Express: Custom Error Handler

Express comes with a built-in error handler

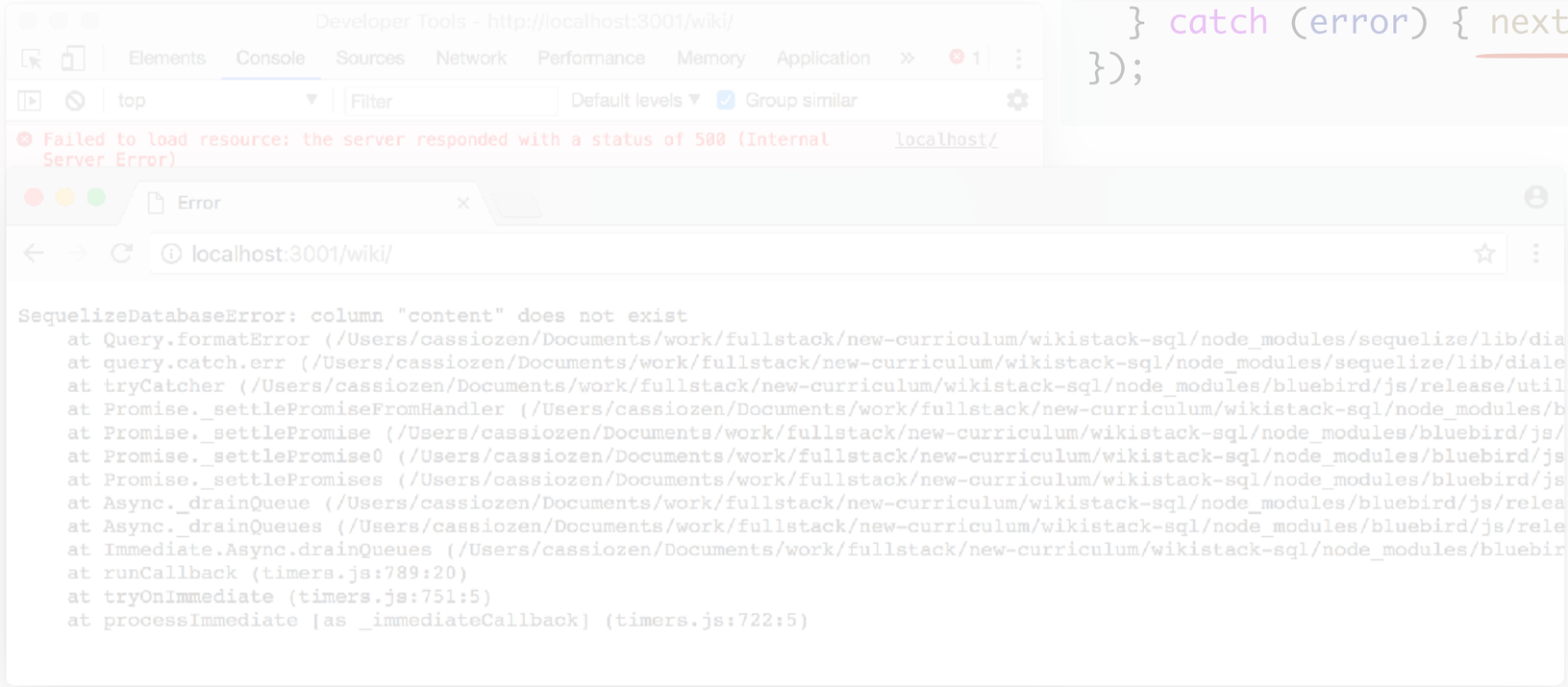
```
router.get("/", async (req, res, next) => {
  try {
    const users = await User.findAll();
    res.send(userList(users));
  } catch (error) { next(error) }
});
```



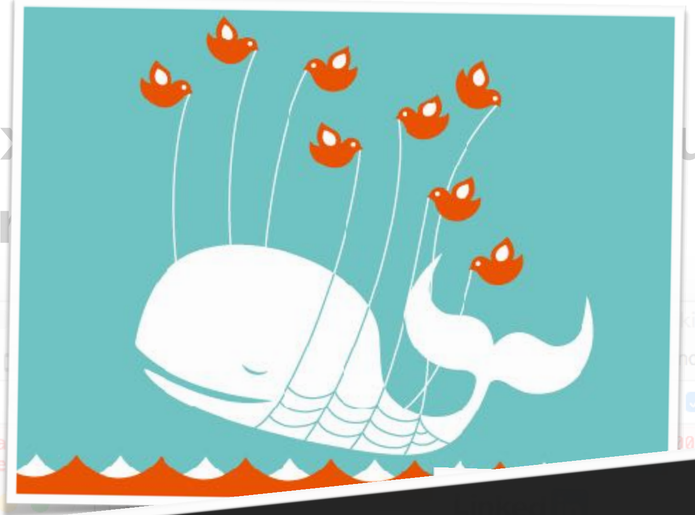
Express: Custom Error Handler

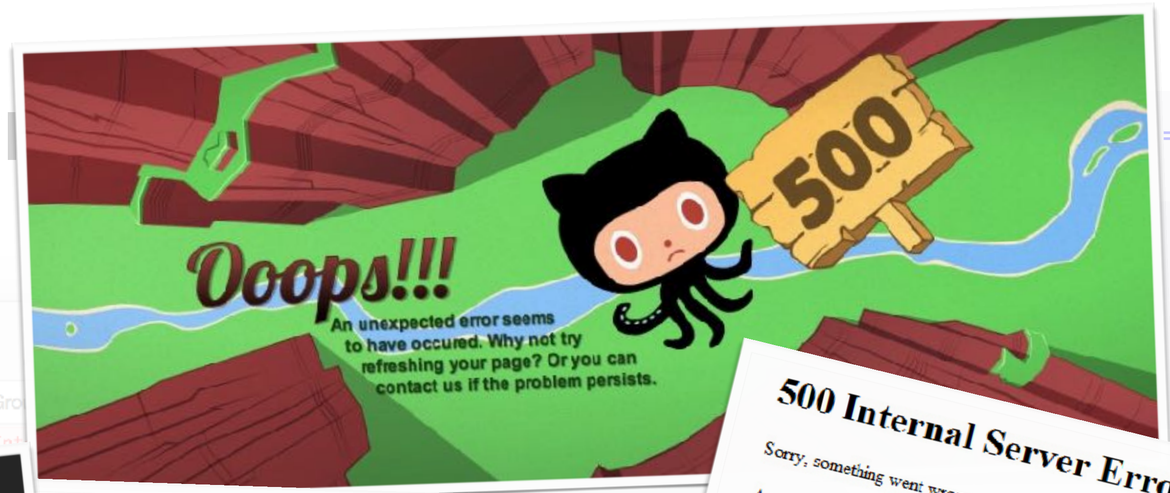
Express comes with a built-in error handler

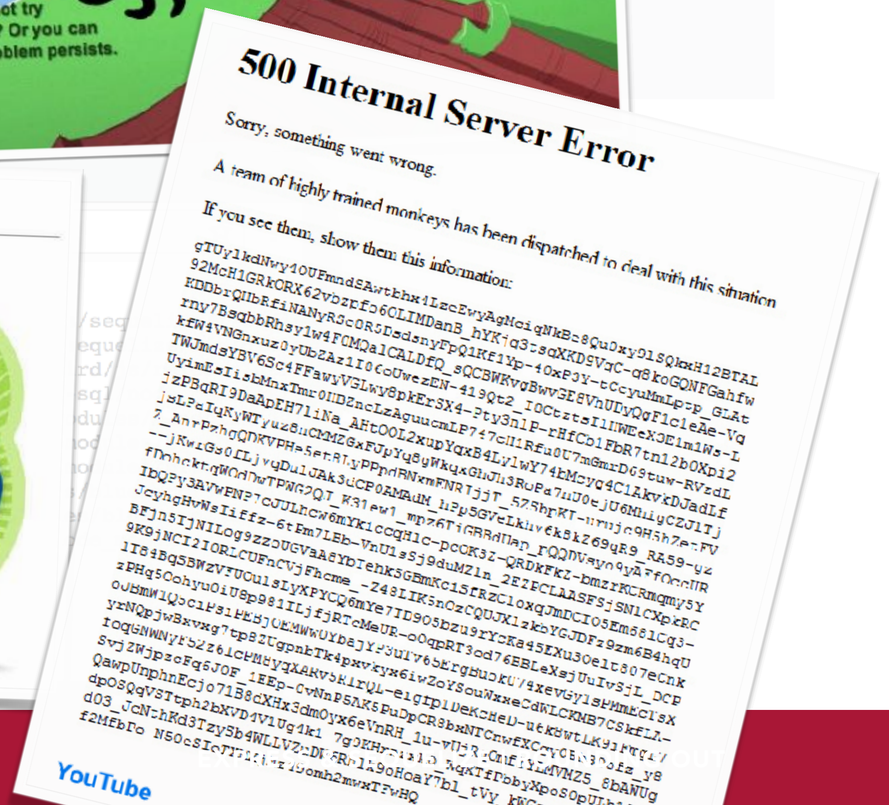
```
router.get("/", async (req, res, next) => {
  try {
    const users = await User.findAll();
    res.send(userList(users));
  } catch (error) { next(error) }
});
```



Express: Custom Error Handler







Express: Custom Error Handler

- Define error-handling middleware just like **other middleware**
- except **error-handling functions have four arguments instead of three:** (err, req, res, next)

```
app.use((err, req, res, next) => {
  console.error(err.stack)
  res.status(500)
    .send(/* Some friendly content */)
})
```

Express: 404 Not Found

Express: 404 Not Found

```
...
// All routes and middlewares above

// Last, Handle 404:
app.use((req, res) => {
  res.status(404).send(/*Not found*/);
});
```

Express: Template Engines

Express: Template Engines

- **JavaScript template literals ≠ template engine.**
- **Templates: static text files with special annotations.**
- **Template engines:**
  - Interprets the document;
  - Replaces variables with actual values
  - Send them as HTML in the HTTP response.

Express: Template Engines

userList.js

```
const html = require("html-template-tag");
const layout = require("./layout");

module.exports = (users) => layout(html`
  <h3>Users</h3>
  <hr>
  <ul class="list-unstyled">
    <ul>
      ${users.map(user => html`<li>
        <a href="/users/${user.id}">${user.name}</a>
      </li>`)}
    </ul>
  </ul>
`);
```

userList.html

```
{% extends "layout.html" %}

{% block content %}
<h3>Users</h3>
<hr>
<ul class="list-unstyled">
  <ul>
    {% for user in users %}
    <li>
       <a href="/users/{{ user.id }}">{{ user.name }}</a>
    </li>
    {% endfor %}
  </ul>
</ul>
{% endblock %}
```

Express: Template Engines

userList.js

```
const html = require("html-template-tag");
const layout = require("./layout");

module.exports = (users) => layout(html`
  <h3>Users</h3>
  <hr>
  <ul class="list-unstyled">
    <ul>
      ${users.map(user => html`<li>
        <a href="/users/${user.id}">${user.name}</a>
      </li>`)}
    </ul>
  </ul>
`);
```

userList.html

```
{% extends "layout.html" %}

{% block content %}
<h3>Users</h3>
<hr>
<ul class="list-unstyled">
  <ul>
    {% for user in users %}
    <li>
       <a href="/users/{{ user.id }}">{{ user.name }}</a>
    </li>
    {% endfor %}
  </ul>
</ul>
{% endblock %}
```

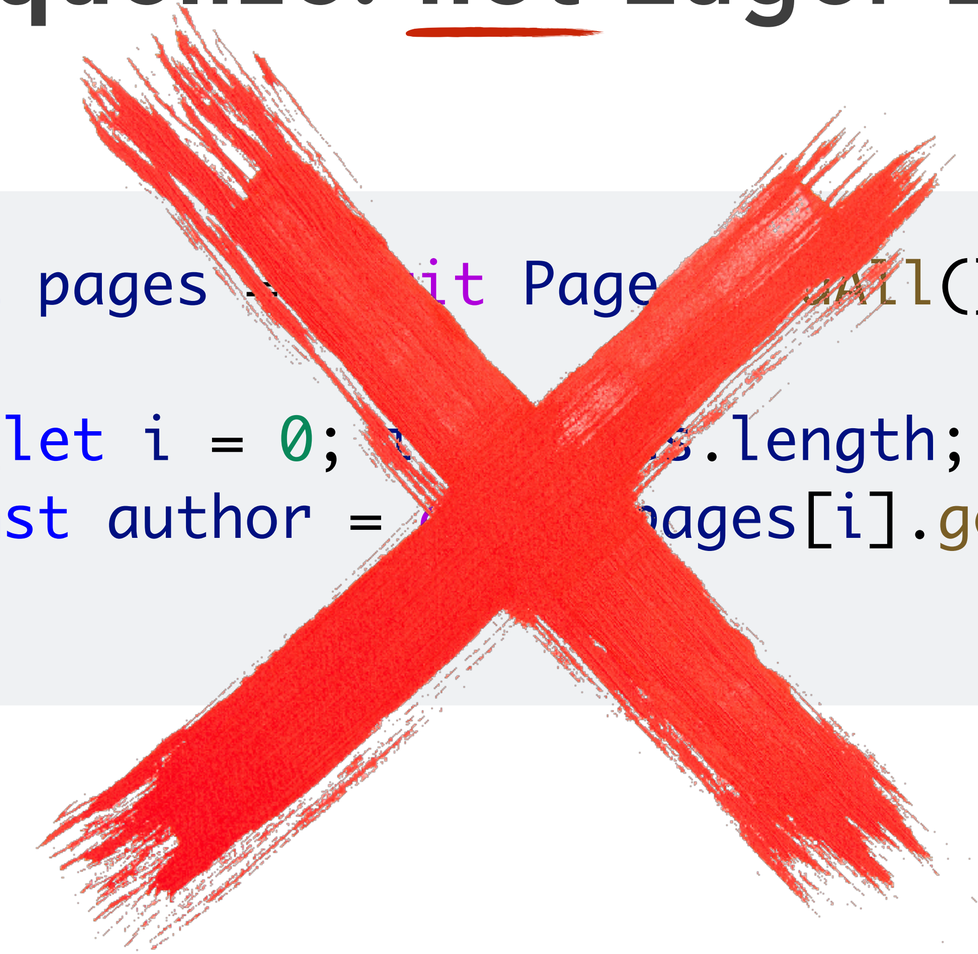
Sequelize: Eager Loading

Sequelize: not Eager Loading

```
const pages = await Page.findAll();

for (let i = 0; i < pages.length; i++) {
  const author = await pages[i].getAuthor();
}
```

Sequelize: not Eager Loading



```
const pages = await Page.findAll();

for (let i = 0; i < pages.length; i++) {
  const author = await pages[i].getAuthor();
}
```

Sequelize: Eager Loading

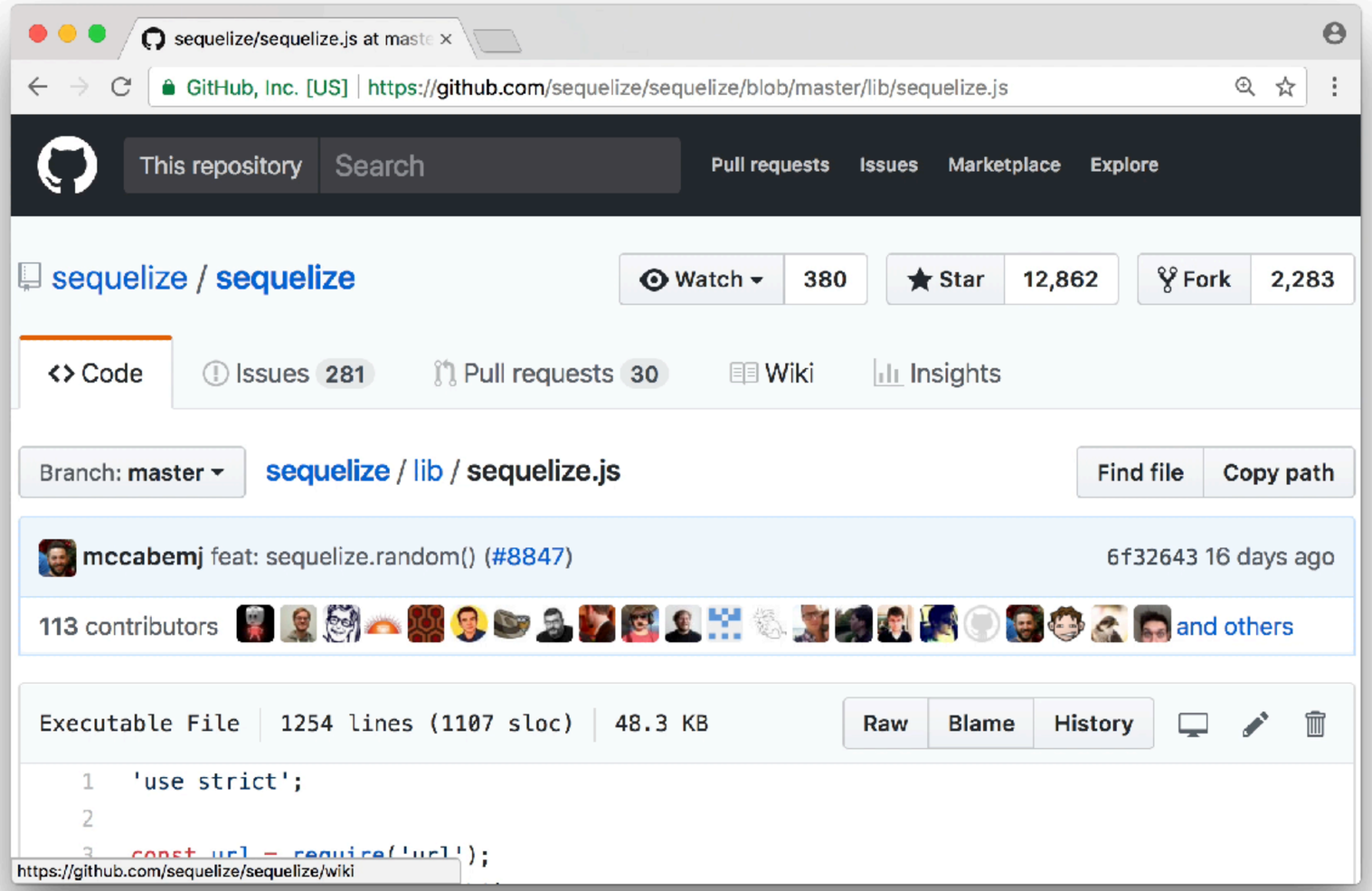
- In raw SQL queries, we have JOIN
- In Sequelize - it just goes by the name of “eager loading”
  - Usually a LEFT OUTER JOIN
- Don't get hung up on the terminology
  when you see “eager loading”, think “join two tables”.

Sequelize: Eager Loading

```
const pages = await Page.findAll({
  include: [
    {model: User, as: 'author'}
  ]
});
```

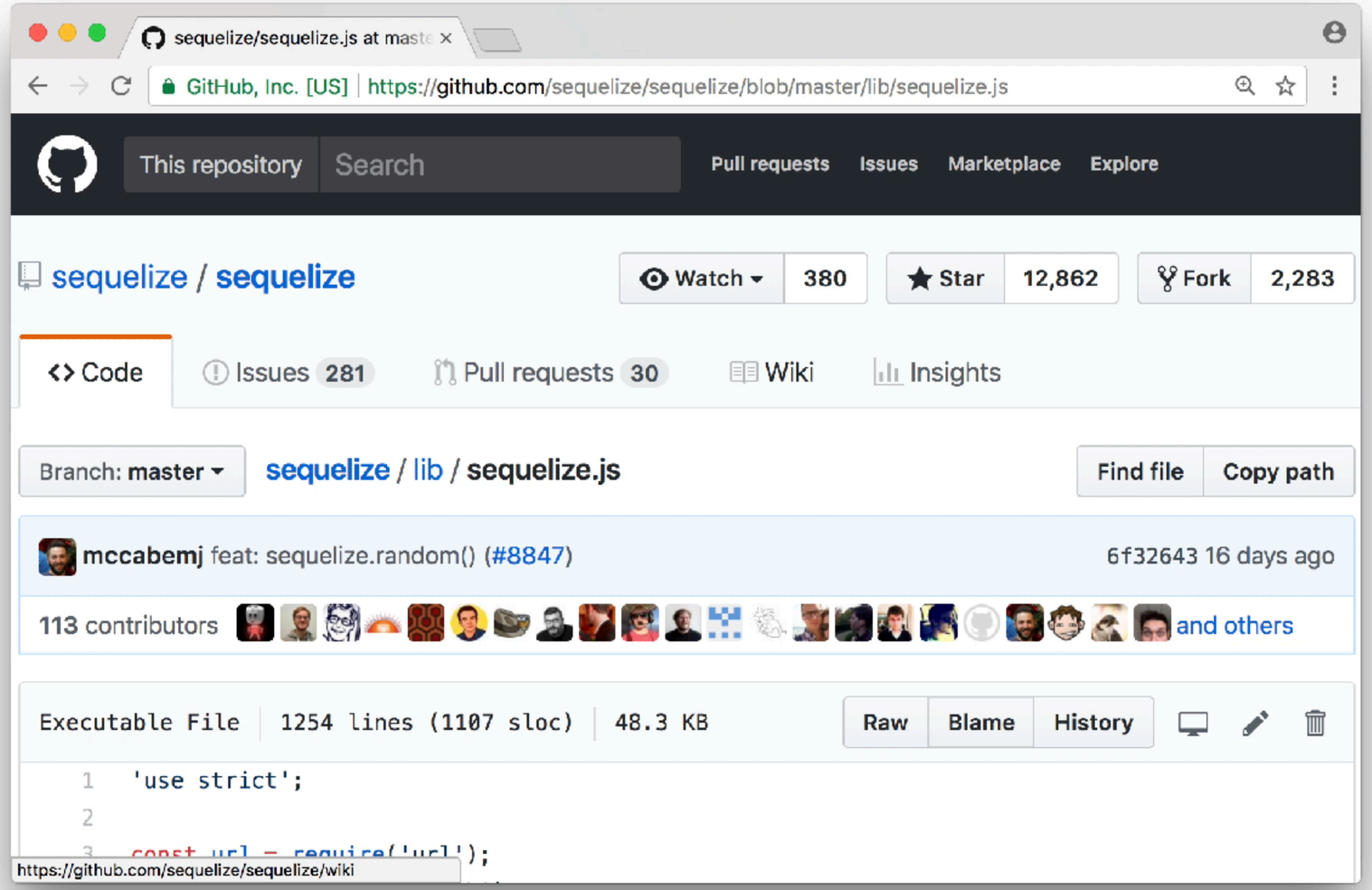
Sequelize: Class & instance methods

Sequelize: Class & instance methods



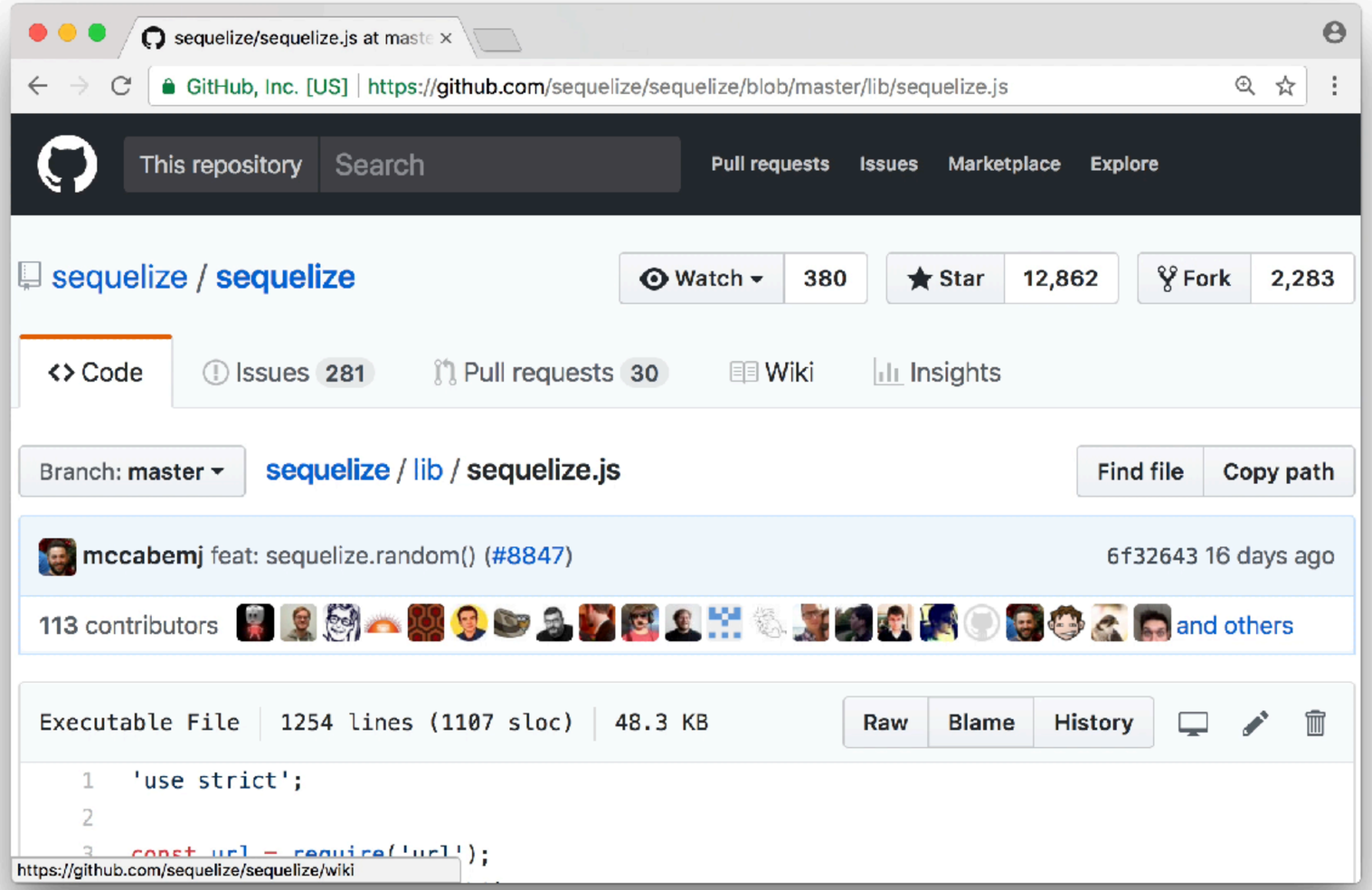
Sequelize: Class & instance methods

- Common/Convenient ways to add functionality to your Sequelize models



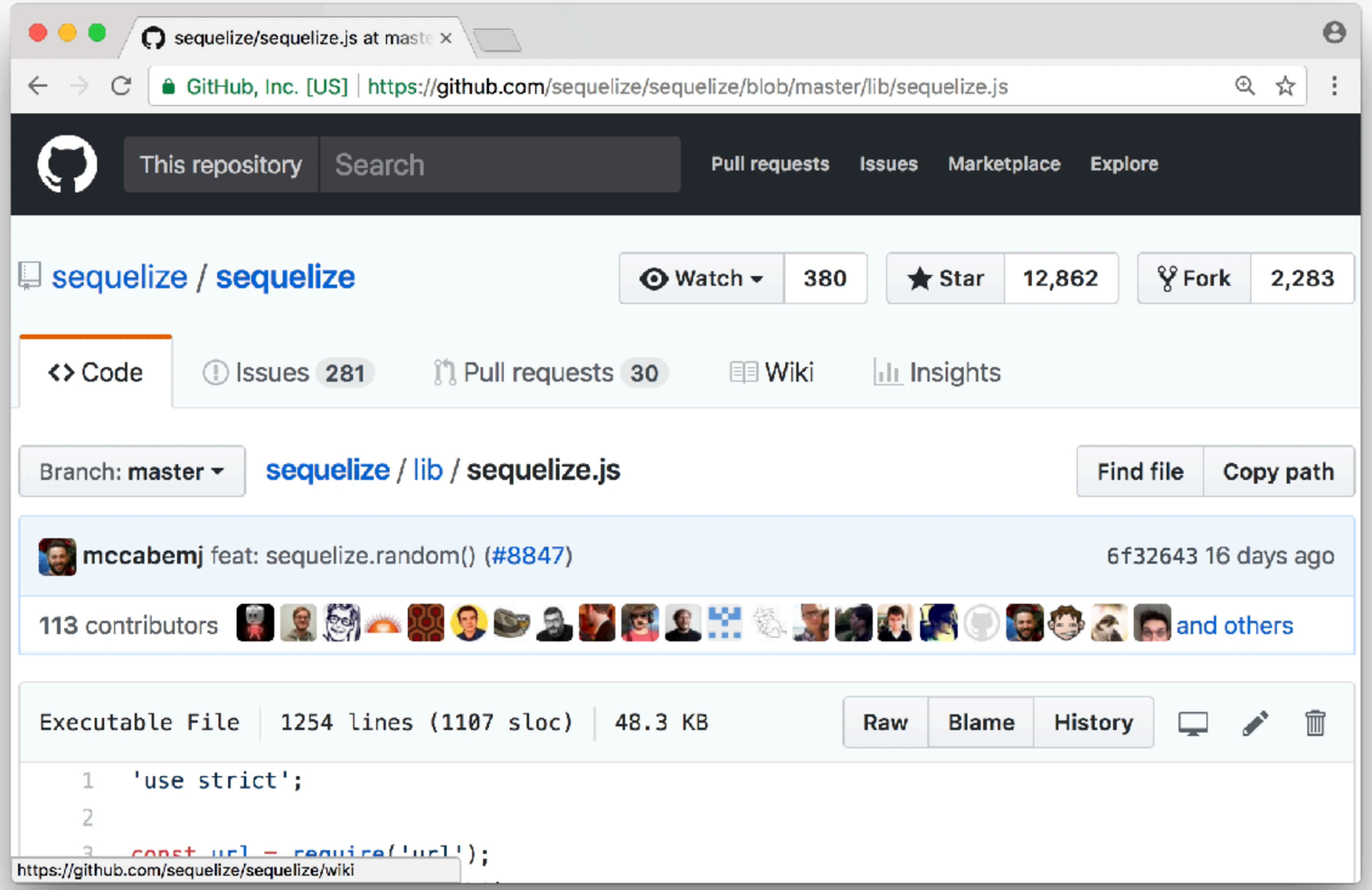
Sequelize: Class & instance methods

- Common/Convenient ways to add functionality to your Sequelize models



Sequelize: Class & instance methods

- **Common/Convenient ways to add functionality to your Sequelize models**
- It's just Javascript:
  We can add functions to the constructor function OR to its prototype.



Sequelize: Class & instance methods

```
const Dog = db.define('dog', {/* etc*/})
```

Sequelize: Class & instance methods

```
const Dog = db.define('dog', {/* etc*/})

Dog.findPuppies = function () {
  // 'this' refers directly back to the model
  return this.findAll({ // Dog.findAll
    where: {
      age: {$lte: 1}
    }
  })
}
```

Sequelize: Class & instance methods

```
const Dog = db.define('dog', {/* etc*/})

Dog.findPuppies = function () {
  // 'this' refers directly back to the model
  return this.findAll({ // Dog.findAll
    where: {
      age: {$lte: 1}
    }
  })
}
```

Sequelize: Class & instance methods

```
const Dog = db.define('dog', {/* etc*/})

Dog.findPuppies = function () {
  // 'this' refers directly back to the model
  return this.findAll({ // Dog.findAll
    where: {
      age: {$lte: 1}
    }
  })
}
```

In your routes, for example…

```
const foundPuppies = await Dog.findPuppies()
```

Sequelize: Class & instance methods

```
const Dog = db.define('dog', {/* etc*/})
```

Sequelize: Class & instance methods

```
const Dog = db.define('dog', {/* etc*/})

Dog.prototype.isBirthday = function () {
  const today = new Date()
  // 'this' refers to the instance itself
  if (this.birthDay === today.getDate()) {
    return true;
  } else {
    return false;
  }
}
```

Sequelize: Class & instance methods

```
const Dog = db.define('dog', {/* etc*/})

Dog.prototype.isBirthday = function () {
  const today = new Date()
  // 'this' refers to the instance itself
  if (this.birthDay === today.getDate()) {
    return true;
  } else {
    return false;
  }
}
```

Sequelize: Class & instance methods

```
const Dog = db.define('dog', {/* etc*/})

Dog.prototype.isBirthday = function () {
  const today = new Date()
  // 'this' refers to the instance itself
  if (this.birthDay === today.getDate()) {
    return true;
  } else {
    return false;
  }
}
```

In your routes, for example…

```
const createdDog = new Dog({name: 'Pork Chop'})

// the instance method is invoked *on the instance*
if(createdDog.isBirthday) console.log("Happy birthday!")
```

Sequelize: Class & instance methods

Sequelize: Class & instance methods

Sequelize: Many-Many Relationships

ONE-ONE

```
Pug.belongsTo(Owner)
Owner.hasOne(Pug)
```

PUGS

id	name	ownerId
1	Cody	1
2	Doug	NULL
3	Maisy	4
4	Frank	NULL

OWNERS

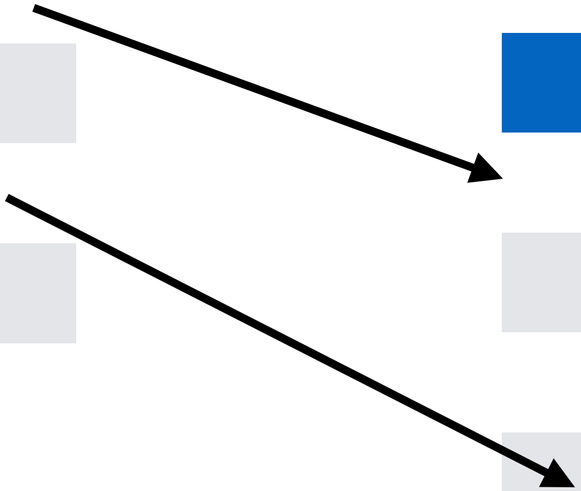
id	name
1	Tom
2	Kate
3	Cassio
4	Karen

ONE-ONE

```
Pug.belongsTo(Owner)
Owner.hasOne(Pug)
```

PUGS

id	name	ownerId
1	Cody	1
2	Doug	NULL
3	Maisy	4
4	Frank	NULL



OWNERS

id	name
1	Tom
2	Kate
3	Cassio
4	Karen

ONE-MANY

```
Pug.belongsTo(Owner)
Owner.hasMany(Pug)
```

PUGS

id	name	ownerId
1	Cody	1
2	Doug	1
3	Milo	2
4	Frank	NULL

OWNERS

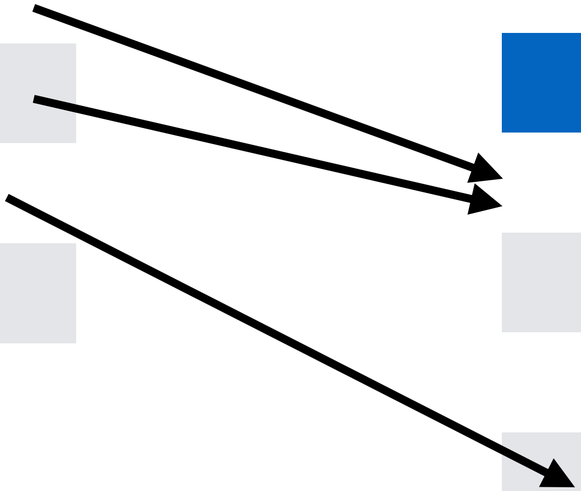
id	name
1	Tom
2	Kate
3	Cassio
4	Karen

ONE-MANY

```
Pug.belongsTo(Owner)
Owner.hasMany(Pug)
```

PUGS

id	name	ownerId
1	Cody	1
2	Doug	1
3	Milo	2
4	Frank	NULL



OWNERS

id	name
1	Tom
2	Kate
3	Cassio
4	Karen

MANY-MANY

```
Pug.belongsToMany(Friend, {through: “pug_friend”})
Friend.belongsToMany(Pug, {through: “pug_friend”})
```

PUG_FRIEND

pug_id	friend_id
1	1
1	2
2	1

PUGS

id	name	ownerId
1	Cody	1
2	Doug	1
3	Milo	2
4	Frank	NULL

FRIENDS

id	name
1	Ashi
2	Gabe
3	Omri
4	Ceren

MANY-MANY

```
Pug.belongsToMany(Friend, {through: “pug_friend”})
Friend.belongsToMany(Pug, {through: “pug_friend”})
```

PUG_FRIEND

pug_id	friend_id
1	1
1	2
2	1

PUGS

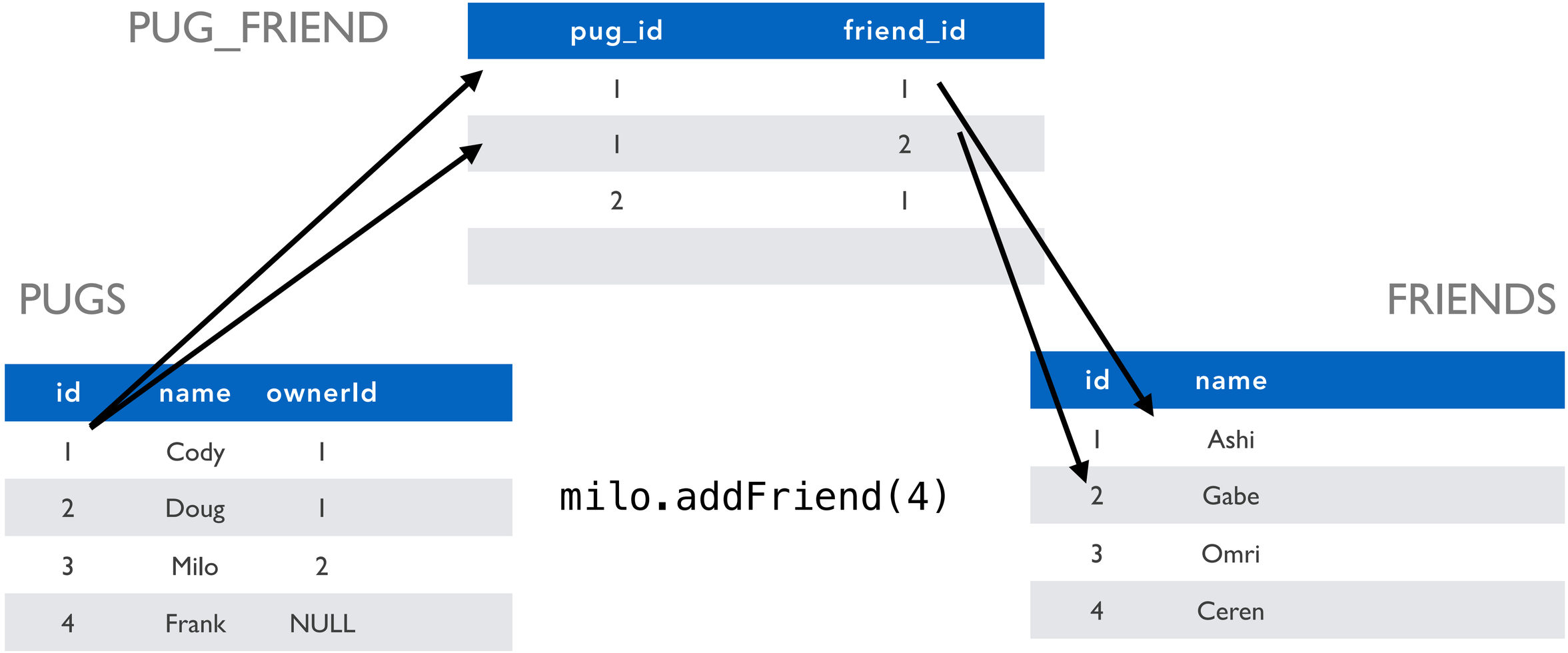
id	name	ownerId
1	Cody	1
2	Doug	1
3	Milo	2
4	Frank	NULL

FRIENDS

id	name
1	Ashi
2	Gabe
3	Omri
4	Ceren

MANY-MANY

```
Pug.belongsToMany(Friend, {through: “pug_friend”})
Friend.belongsToMany(Pug, {through: “pug_friend”})
```



MANY-MANY

```
Pug.belongsToMany(Friend, {through: “pug_friend”})
Friend.belongsToMany(Pug, {through: “pug_friend”})
```

PUG_FRIEND

pug_id	friend_id
1	1
1	2
2	1
3	4

PUGS

id	name	ownerId
1	Cody	1
2	Doug	1
3	Milo	2
4	Frank	NULL

```
milo.addFriend(4)
```

FRIENDS

id	name
1	Ashi
2	Gabe
3	Omri
4	Ceren

Workshop

- **Express**
  - Custom error handler
  - 404 Not Found
  - Template engines
- **Sequelize**
  - Eager Loading
  - Class methods / instance methods
  - Many-Many relationships