

Express & Sequelize

Rounding Out

Express & Sequelize

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

What we will cover

- **Express**

- Custom error handler (500)
- Handling 404 Not Found

- **Sequelize**

- Eager Loading
- Class methods / instance methods
- Many-Many Relationships

Express: Custom Error Handler

Express: Custom Error Handler

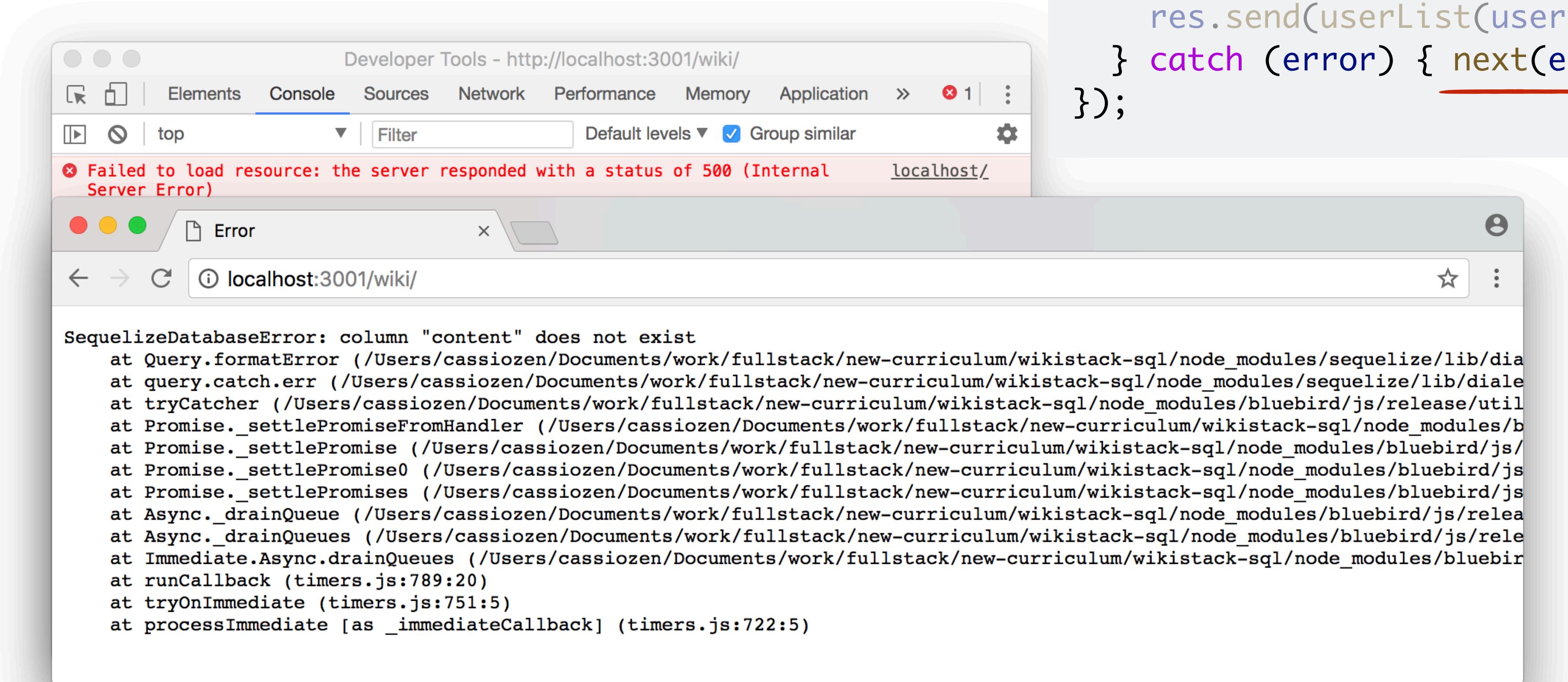
- Express comes with a built-in error handler

- This is what kicks in when you `next` an error
- The errors here are usually 500 errors
 - 500: “Internal Server Error”
 - Something broke in your server-side code

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

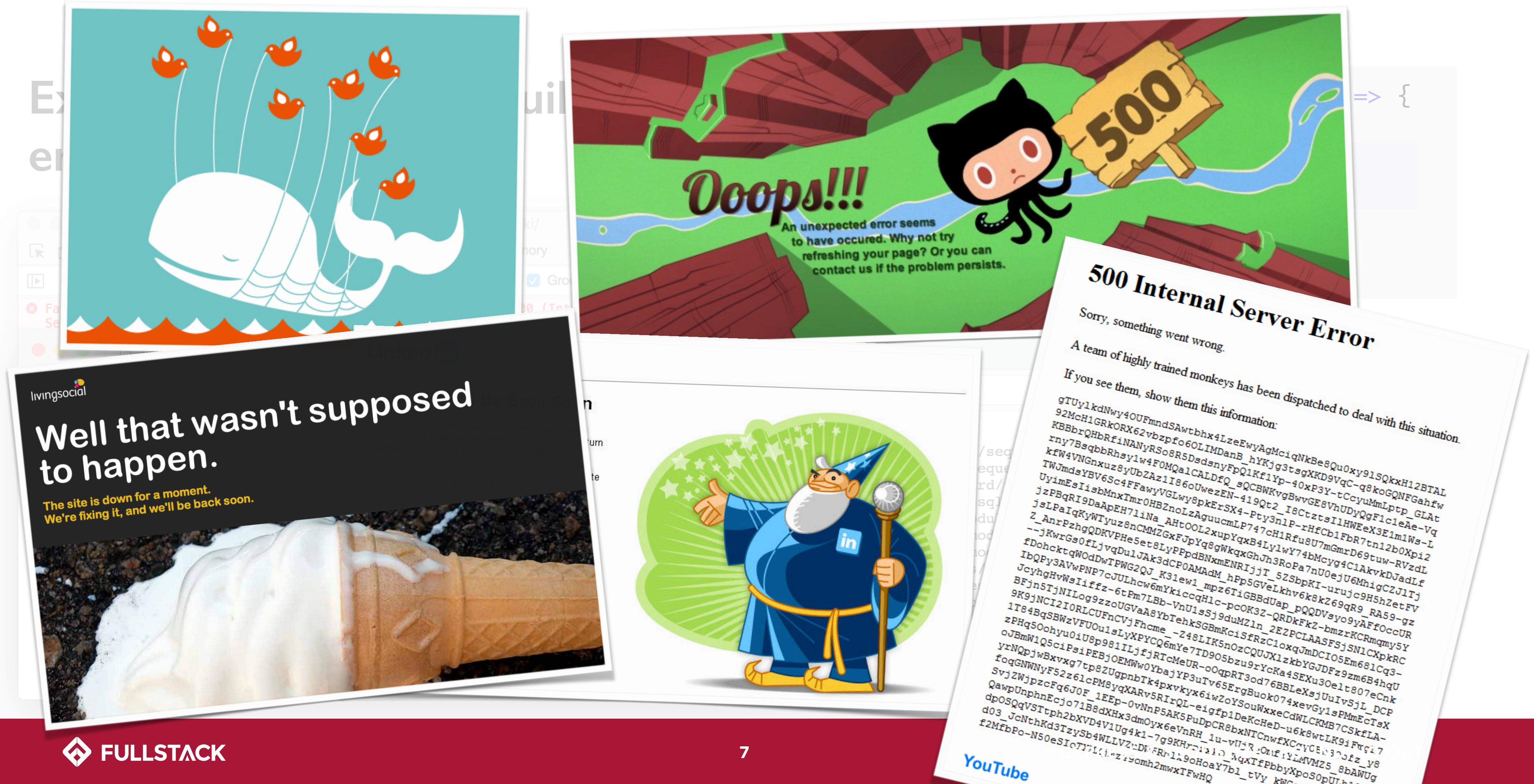
Express: Custom Error Handler

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)
- Now when you `next` an error, it will go here
- Error-handling middleware goes at the end of your routes

```
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

You may see a 404 error if the client:

1. Requested a route that doesn't exist
 - GET /api/notreal/route => 404: this route isn't real
2. Requested a resource that doesn't exist (but did hit a valid route)
 - GET /api/puppies/10001 => 404,:this route is real... but we don't have a puppy with that id

Case 1: Requested a ROUTE that does not exist

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

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

Case 2: Requested a RESOURCE that does not exist

```
// Get puppy by id route:  
app.get('/puppies/:id', (req, res, next) => {  
  const puppy = await Puppy.findByPk(req.params.id); // id: 10001  
  if(!puppy) return res.sendStatus(404);  
  res.send(puppy);  
});
```

Sequelize: Eager Loading



Sequelize: not Eager Loading

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

Sequelize: Eager Loading

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



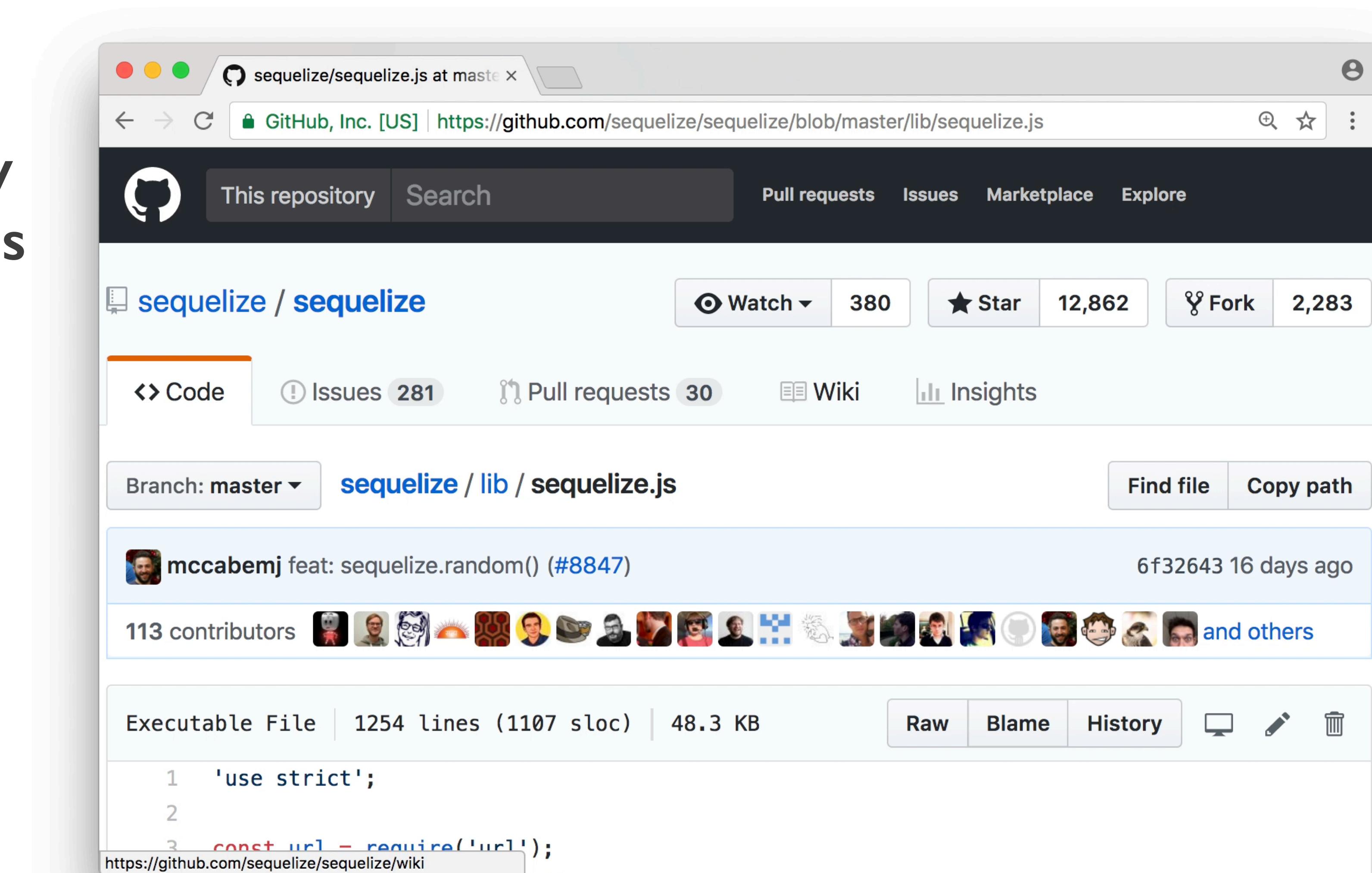
Sequelize: Eager Loading

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

Sequelize: Class & instance methods

Sequelize: Class & instance methods

- 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 */)

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

In your routes, for example...
const foundPuppies = await Dog.findPuppies()
```



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.birt In your routes, for example...  
    return true const createdDog = new Dog({name: 'Pork Chop'})  
  } else {  
    return false // the instance method is invoked *on the instance*  
  }  
  if(createdDog.isBirthday()) console.log("Happy birthday!")  
}
```

Sequelize: Class & instance methods



It's just JavaScript

Sequelize: Many-Many Relationships

ONE-ONE

Pug.belongsTo(Owner)
OwnerhasOne(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)
OwnerhasMany(Pug)

PUGS

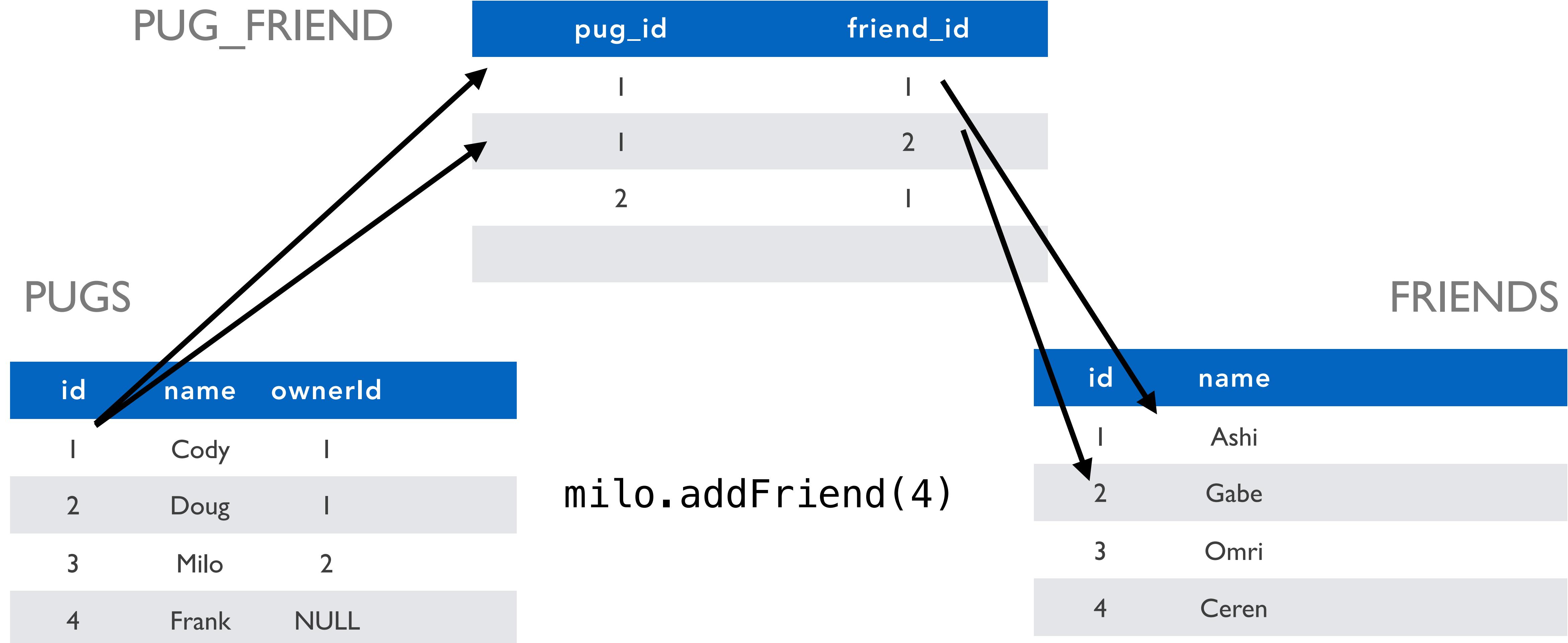
id	name	ownerId
1	Cody	1
2	Doug	1
3	Milo	2
4	Frank	NULL

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"})



Workshop

- **Express**

- Custom error handler
- 404 Not Found

- **Sequelize**

- Eager Loading
- Class methods / instance methods
- Many-Many relationships