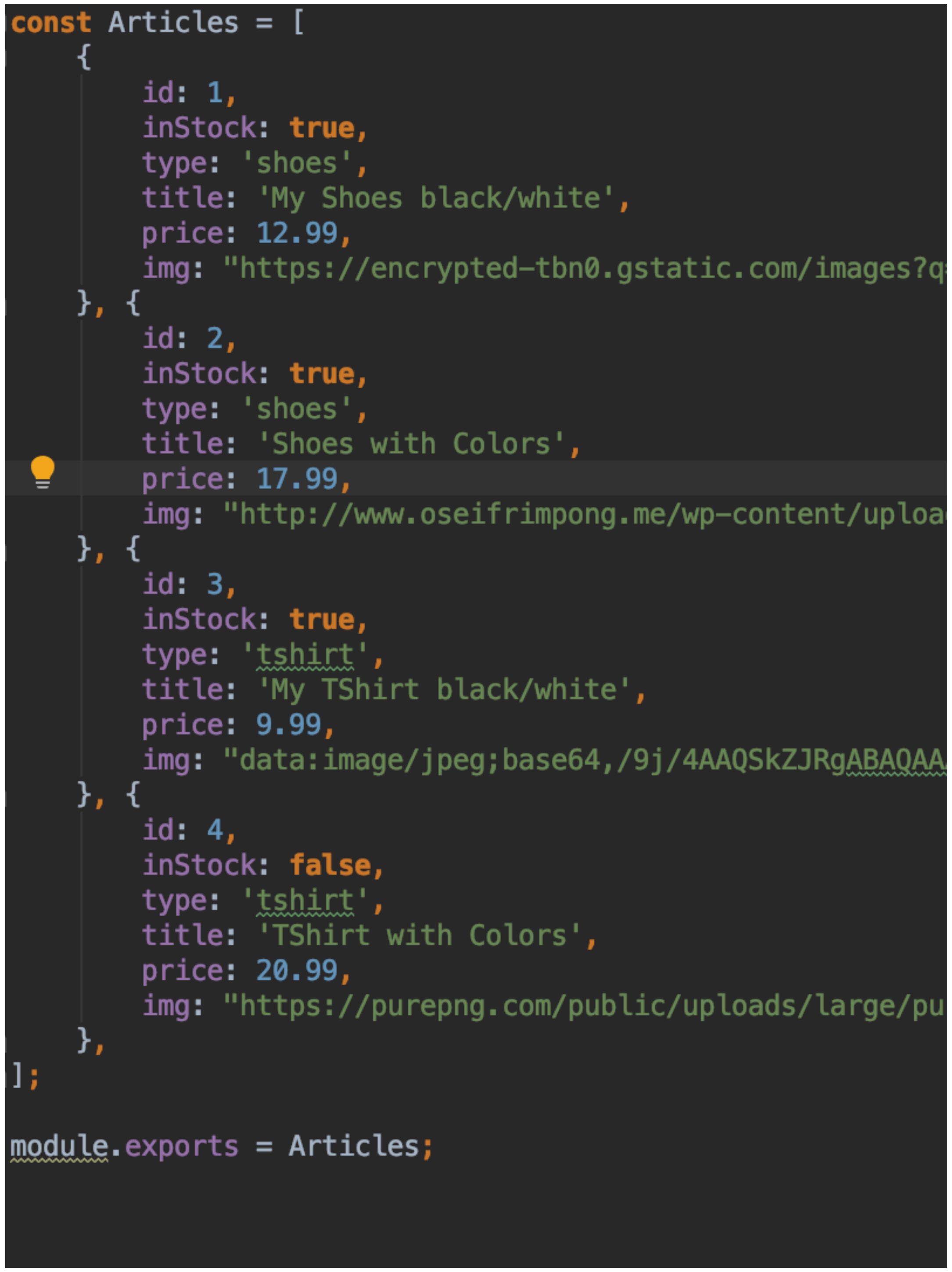
- Promises
- Article object refactoring
- GET params

Promises

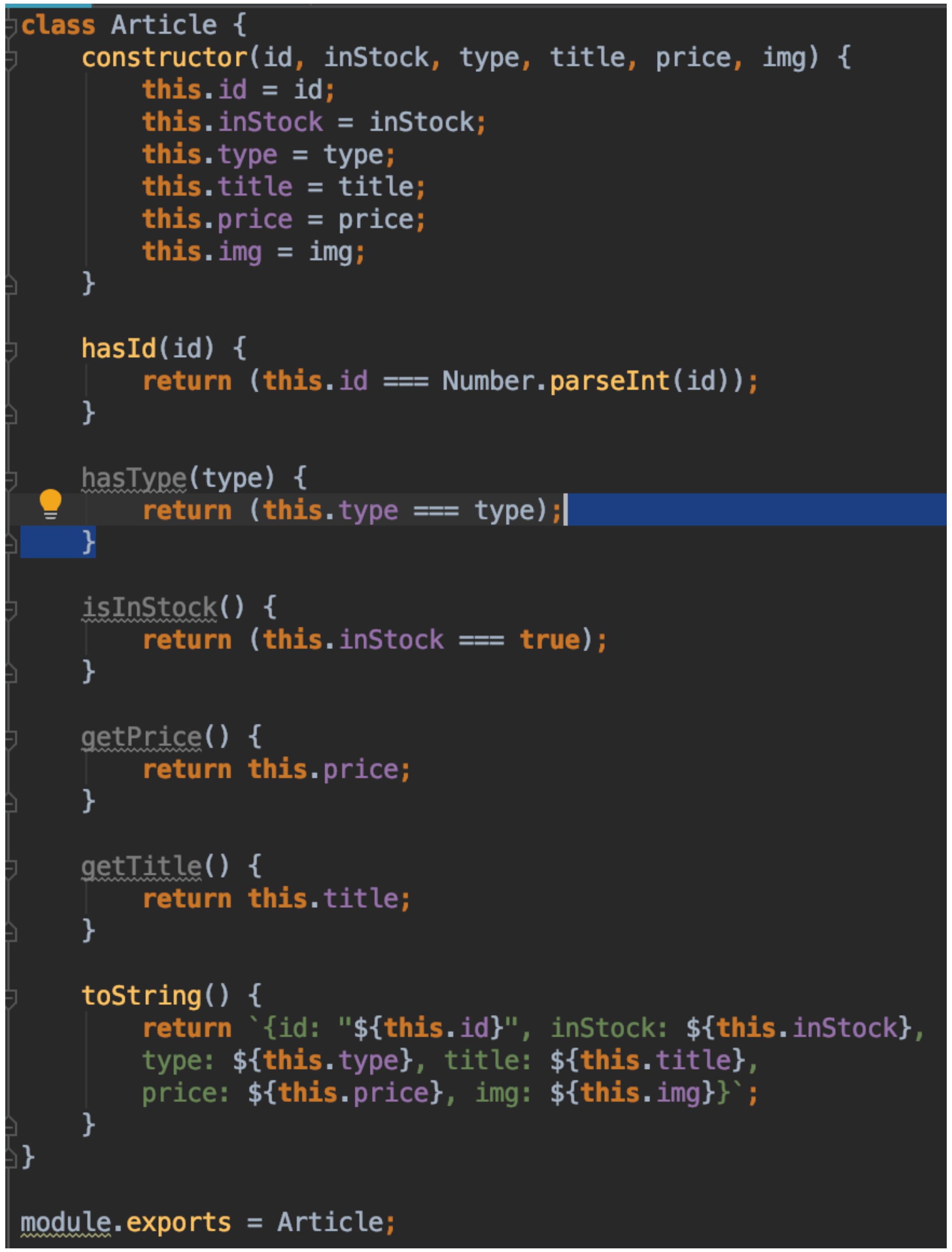
“A Promise is an object representing the eventual completion or failure of an asynchronous operation.”

🔗 https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Using_promises

Article object refactoring

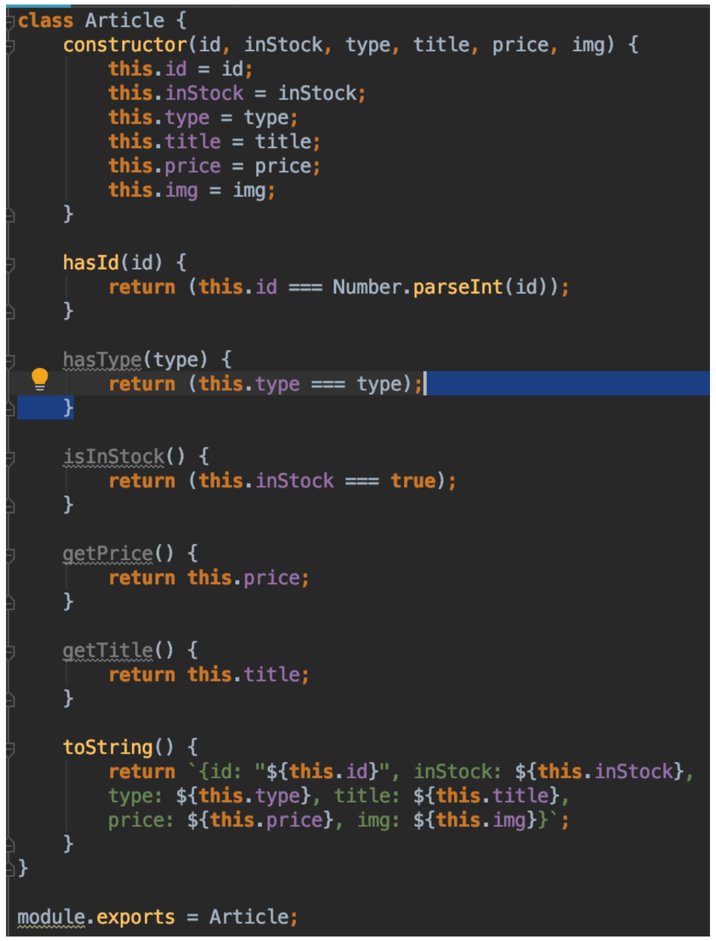


```
const Articles = [
    {
        id: 1,
        inStock: true,
        type: 'shoes',
        title: 'My Shoes black/white',
        price: 12.99,
        img: "https://encrypted-tbn0.gstatic.com/images?q
    }, {
        id: 2,
        inStock: true,
        type: 'shoes',
        title: 'Shoes with Colors',
        price: 17.99,
        img: "http://www.oseifrimpong.me/wp-content/uploa
    }, {
        id: 3,
        inStock: true,
        type: 'tshirt',
        title: 'My TShirt black/white',
        price: 9.99,
        img: "data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAA
    }, {
        id: 4,
        inStock: false,
        type: 'tshirt',
        title: 'TShirt with Colors',
        price: 20.99,
        img: "https://purepng.com/public/uploads/large/pu
    },
];

module.exports = Articles;
```

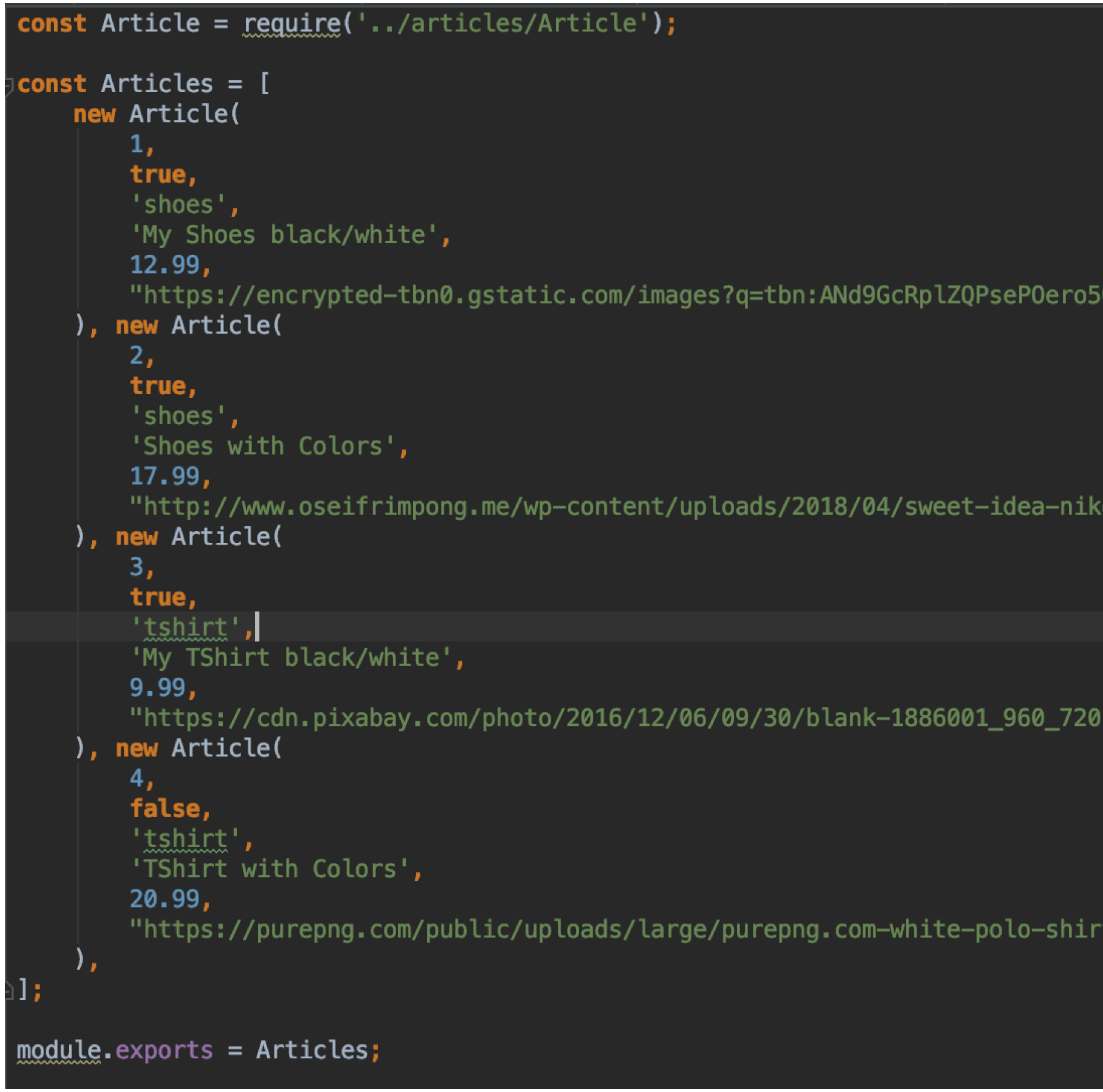


```
class Article {
    constructor(id, inStock, type, title, price, img) {
        this.id = id;
        this.inStock = inStock;
        this.type = type;
        this.title = title;
        this.price = price;
        this.img = img;
    }

    hasId(id) {
        return (this.id === Number.parseInt(id));
    }

    hasType(type) {
        return (this.type === type);
    }

    isInStock() {
        return (this.inStock === true);
    }

    getPrice() {
        return this.price;
    }

    getTitle() {
        return this.title;
    }

    toString() {
        return `{id: "${this.id}", inStock: ${this.inStock},
        type: ${this.type}, title: ${this.title},
        price: ${this.price}, img: ${this.img}}`;
    }
}

module.exports = Article;
```

Article object refactoring



```
class Article {
    constructor(id, inStock, type, title, price, img) {
        this.id = id;
        this.inStock = inStock;
        this.type = type;
        this.title = title;
        this.price = price;
        this.img = img;
    }

    hasId(id) {
        return (this.id === Number.parseInt(id));
    }

    hasType(type) {
        return (this.type === type);
    }

    isInStock() {
        return (this.inStock === true);
    }

    getPrice() {
        return this.price;
    }

    getTitle() {
        return this.title;
    }

    toString() {
        return `{id: "${this.id}", inStock: ${this.inStock},
        type: ${this.type}, title: ${this.title},
        price: ${this.price}, img: ${this.img}}`;
    }
}

module.exports = Article;
```



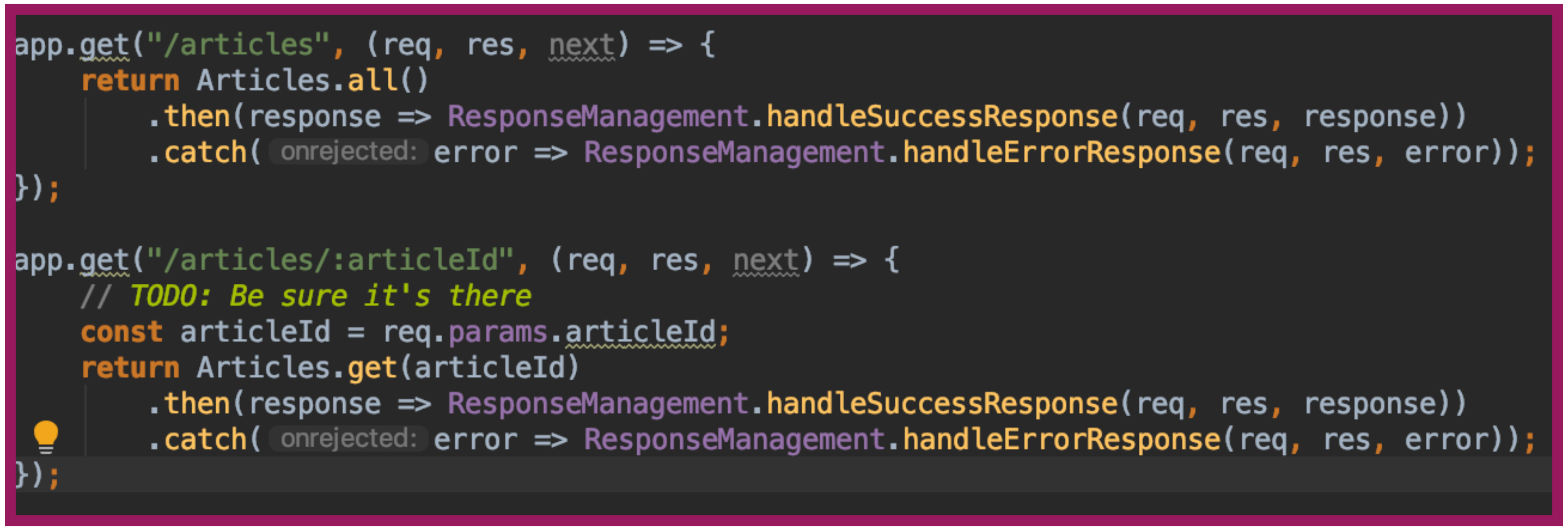
```
const Article = require('../articles/Article');

const Articles = [
    new Article(
        1,
        true,
        'shoes',
        'My Shoes black/white',
        12.99,
        "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRplZQPsePOero5
    ), new Article(
        2,
        true,
        'shoes',
        'Shoes with Colors',
        17.99,
        "http://www.oseifrimpong.me/wp-content/uploads/2018/04/sweet-idea-nik
    ), new Article(
        3,
        true,
        'tshirt',
        'My TShirt black/white',
        9.99,
        "https://cdn.pixabay.com/photo/2016/12/06/09/30/blank-1886001_960_720
    ), new Article(
        4,
        false,
        'tshirt',
        'TShirt with Colors',
        20.99,
        "https://purepng.com/public/uploads/large/purepng.com-white-polo-shir
    ),
];

module.exports = Articles;
```

©Lorenzo Sfarra

GET params

Route parameters are **named URL segments** that are used to capture the values specified at their position in the URL.

The captured values are populated in the **req.params** object, with the **name of the route parameter specified in the path as their respective keys**.



```
app.get("/articles", (req, res, next) => {
    return Articles.all()
        .then(response => ResponseManagement.handleSuccessResponse(req, res, response))
        .catch( onrejected: error => ResponseManagement.handleErrorResponse(req, res, error));
});

app.get("/articles/:articleId", (req, res, next) => {
    // TODO: Be sure it's there
    const articleId = req.params.articleId;
    return Articles.get(articleId)
        .then(response => ResponseManagement.handleSuccessResponse(req, res, response))
        .catch( onrejected: error => ResponseManagement.handleErrorResponse(req, res, error));
});
```