15. Promises, Async/Await

Monday, June 12, 2023 5:08 PM

Problem with Callbacks = Callback Hell

Dog API to get a random Dog with certain breed:
https://dog.ceo/dog-api/

https://dog.ceo/api/breed/retriever/images/random

```
const urls = ['https://dog.ceo/api/breed/retriever/images/random'];

Promise.all(urls.map(url => fetch(url).then(res =>
res.json() ))).then(results => {
   if (results) console.log(results);
   else throw Error;
})
.catch( err => console.log(err) );
```

// Need package.json & superagent module
npm init -y && npm i superagent

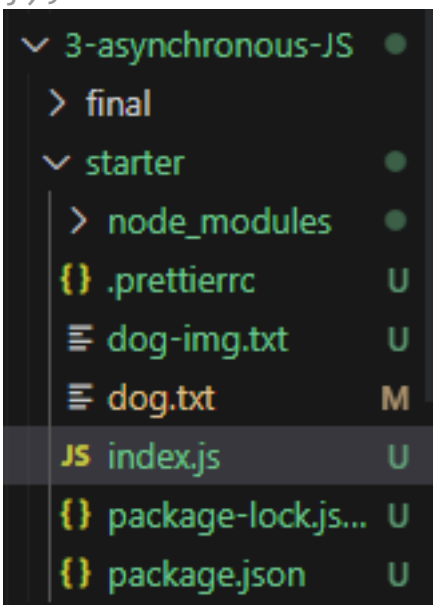
---
dog.txt:
retriever
---

index.js

```
const fs = require('fs');
const superagent = require('superagent');
fs.readFile(`${__dirname}/dog.txt`, (err, data) => {
```

```
    if (err) {
        return console.error(`Error reading File Content: ${err}`);
    }
    console.log(`Breed: ${data}`);
    console.log(`\n`);

    superagent
    .get(`https://dog.ceo/api/breed/${data}/images/random`)
    .end((err, res) => {
        if (err) {
            return console.log(err.message);
        };
        console.log('res.body', res.body);
        fs.writeFile('dog-img.txt', res.body.message, err => {
            if (err) {
                return console.log(err.message);
            }
            console.log('Random dog image saved to file')
        });
    });
});
```




```
Breed: labrador
[nodemon] starting `node index.js`
Breed: labrador
res.body {
  message: 'https://images.dog.ceo/breeds/labrador/img_6236.jpg',
  status: 'success'
}
Random dog image saved to file
[nodemon] clean exit - waiting for changes before restart
```

dog-img.txt

JS App.js M | dog.txt M | JS index.js U | dog-img.txt U

complete-node-bootcamp > 3-asynchronous-JS > starter > dog-img.txt

```
https://images.dog.ceo/breeds/labrador/n02099712_2228.jpg
```

==== from Callback Hell => Promises

index.js

```
const fs = require('fs');
const superagent = require('superagent');
// Create Arrow func to return new Promise that fs.readFile
const readFilePro = (file) => {
    return new Promise((resolve, reject) => {
         // executer func
        fs.readFile(file, (err, data) => {
            // if there's an error --> reject() --> piped err
into .catch()
            if (err) reject('Could NOT find file');
            // Promise returns data to us will be piped into .then()
            resolve(data);
        })
    });
}
const writeFilePro = (file, data) => {
    return new Promise((resolve, reject) => {
        fs.writeFile(file, data, (err) => {
            if (err) reject('Could NOT write file');
            // fs.writeFile doesn't need to return data
            resolve('write succeeded');
        })
    })
}

// readFilePro(fileName)
// returns a Promise before calling each of them
readFilePro(`${__dirname}/dog.txt`)
.then(data => {
    console.log(`Breed: ${data}`);
    console.log(`\n`);
    // To keep chaining .then(), must return a Promise
```

```
    // return a Promise before calling each of them
    return superagent.get(`https://dog.ceo/api/breed/
${data}/images/random`);
})
.then(res => {
    console.log('res.body.message', res.body.message);
    console.log(`\n`);
    // To keep chaining .then(), must return a Promise
    // return a Promise before calling each of them
    return writeFilePro('dog-img.txt', res.body.message);
    // fs.writeFile('dog-img.txt', res.body.message, err => {
    //     if (err) return console.log(err.message);
    //     console.log('Random dog image saved to file')
    // });
})
.then(() => {
    console.log('Random dog image saved to file!');
})
.catch(err => {
    console.log('err', err);
});

// Callback Hell
// Callbacks inside of Callbacks inside of Callbacks
// fs.readFile(`${__dirname}/dog.txt`, (err, data) => {
//     if (err) {
//         return console.log(err.message);
//     }
//     console.log(`Breed: ${data}`);
//     superagent
//     .get(`https://dog.ceo/api/breed/${data}/images/random`)
//     .then(res => {
//         console.log('res.body', res.body);
//         fs.writeFile('dog-img.txt', res.body.message, err => {
//             if (err) {
//                 return console.log(err.message);
//             }
//             console.log('Random dog image saved to file')
//         });
//     })
//     .catch(err => {
//         console.log('errors...', err);
//     });
// });
```

====Consuming Promises with Async/Await

index.js

```
const fs = require('fs');
const superagent = require('superagent');
// Create Arrow func to return new Promise that fs.readFile
const readFilePro = (filePath) => {
    return new Promise((resolve, reject) => {
        // executer func
        fs.readFile(filePath, (err, fileContent) => {
        // if there's an error --> reject() --> piped err to .catch()
        if (err) reject('Could NOT find file');
        // Promise returns data to us will be piped into .then()
        resolve(fileContent);
        })
    });
}

const writeFilePro = (filePath, data) => {
    return new Promise((resolve, reject) => {
        fs.writeFile(file, data, err => {
            if (err) reject('Could NOT write file');
            // fs.writeFile doesn't need to return data
            resolve('write succeeded');
        })
    });
}

// Async func lets other tasks keep running in Event Loop
const getDogPic = async () => {
    try {
        // Stop operation of this line below, until it returns
        // & finally stores results to const data
        const data = await readFilePro(`${__dirname}/dog-1.txt`);
        console.log(`\n`);
        console.log(`Breed: ${data}`);
        console.log(`\n`);
        const res = await superagent.get(`https://dog.ceo/api/breed/
${data}/images/random`);
        console.log('res.body.message', res.body.message);
        console.log(`\n`);

        await writeFilePro('dog-img.txt', res.body.message);
        console.log('Random dog image saved to file');
```

```
    } catch (err) {
        console.log(err);
    }
}
getDogPic();
```

==== How Async Await works behind the scene

```
const fs = require('fs');
const superagent = require('superagent');
// Create Arrow func to return new Promise that fs.readFile
const readFilePro = (filePath) => {
    return new Promise((resolve, reject) => { // executer func
        fs.readFile(filePath, (err, fileContent) => {
            // if there's an error --> reject() --> piped
into .catch()
            if (err) reject('Could NOT find file');
            // Promise returns data to us will be piped into .then()
            resolve(fileContent);
        })
    });
}

const writeFilePro = (file, data) => {
    return new Promise((resolve, reject) => {
        fs.writeFile(file, data, err => {
            if (err) reject('Could NOT write file');
            // fs.writeFile doesn't need to return data
            resolve('write succeeded');
        })
    })
}

// Async func lets other tasks keep running in Event Loop
const getDogPic = async () => {
    try {
        // Stop operation of this line below, until it returns
        // & finally stores results to const data
        const data = await readFilePro(`${__dirname}/dog-1.txt`);
        console.log(`\n`);
        console.log(`Breed: ${data}`);
        console.log(`\n`);

        const res = await superagent.get(`https://dog.ceo/api/breed/
${data}/images/random`);
        console.log('res.body.message', res.body.message);
```

```
        console.log('res.body.message', res.body.message);
        await writeFilePro('dog-img.txt', res.body.message);
        console.log('Random dog image saved to file');
    } catch (err) {
        console.log(err);
    }
}

console.log('1: Will get dog pics!');
getDogPic();
console.log('2: Done getting dog pics!');
```

```
1: Will get dog pics!
2: Done getting dog pics!
Breed: labrador
res.body.message https://images.dog.ceo/breeds/labrador/n02099712_7411.jpg
Random dog image saved to file
```

```
// Async func lets other tasks keep running in Event Loop
const getDogPic = async () => {
    try {
        // Stop operation of this line below, until it returns
        // & finally stores results to const data
        const data = await readFilePro(`${__dirname}/dog-1.txt`);
        console.log(`\n`);
        console.log(`Breed: ${data}`);
        console.log(`\n`);

        const res = await superagent.get(`https://dog.ceo/api/breed/
${data}/images/random`);
        console.log('res.body.message', res.body.message);
        console.log(`\n`);

        await writeFilePro('dog-img.txt', res.body.message);
        console.log('Random dog image saved to file');
    } catch (err) {
        console.log(`Error writing file as a promise:\n${err}\n`);
    }
    console.log(`Returning something:`);
    return '2: READY';
};
console.log('1: Will get dog pics!');
const x = getDogPic();
console.log(x);
getDogPic();
console.log('3: Done getting dog pics!');
```

```
1: Will get dog pics!
Promise { <pending> }
3: Done getting dog pics!
Breed: labrador
Breed: labrador
res.body.message https://images.dog.ceo/breeds/labrador/n02099712_5853.jpg
Random dog image saved to file
res.body.message https://images.dog.ceo/breeds/labrador/n02099712_4550.jpg
Random dog image saved to file
```

// We'd get Promise { <pending> }, rather the string '2: READY';
// instead of logging '2: READY', const x (Async func) is still running

```
// Async func lets other tasks keep running in Event Loop
const getDogPic = async () => {
    try {
        // Stop operation of this line below, until it returns
        // & finally stores results to const data
        const data = await readFilePro(`${__dirname}/dog-1.txt`);
        console.log(`Breed: ${data}`);
        console.log(`\n`);

        const res = await superagent.get(`https://dog.ceo/api/breed/
${data}/images/random`);
        console.log('res.body.message', res.body.message);
        console.log(`\n`);

        await writeFilePro('dog-img.txt', res.body.message);
        console.log('Random dog image saved to file');
    } catch (err) {
        console.log(`Error node fetching API:\n${err}\n`);
    }
    return '2: READY';
};

console.log('1: Will get dog pics!');
getDogPic()
.then(x => {
    console.log(x);
    console.log('3: Done getting dog pics!');
})
.catch(err => {
    console.log('Error!');
});
```

```
1: Will get dog pics!
Breed: labrador
res.body.message https://images.dog.ceo/breeds/labrador/n02099712_2223.jpg
Random dog image saved to file
2: READY
3: Done getting dog pics!
```

// We can use Immediately Invoke Function Execution
// instead of Flat Async.then().catch()

```
// Use IIFE
(async () => {
    try {
        console.log('1: Will get dog pics!');
        const x = await getDogPic();
        console.log(x);

    } catch(err) {
        console.log(`Error!\n${err}\n`);
    } finally {
        console.log('3: Done getting dog pics!');
    }
}) ();
```

```
1: Will get dog pics!
Breed: labrador
res.body.message https://images.dog.ceo/breeds/labrador/n02099712_6426.jpg
Random dog image saved to file
2: READY
3: Done getting dog pics!
```

Async funcs called from other Async funcs
**** Async func() => {...} automatically returns a Promise**
**** Value returned from an Async func = Resolved value of Promise**

==== **Waiting for Multiple Promises simultaneously**

```
const getDogPic = async () => {
    try {
        // Stop operation of this line below, until it returns
        // & finally stores results to const data
        const fileContent = await readFilePro(`
${__dirname}/dog-1.txt`);
```

```
${__dirname}/dog-1.txt');
        console.log(`Breed: ${fileContent}`);
        console.log(`\n`);

        // Suppose we wanna get 3 random Dog images at the same time
        // Storing a const resPro = superagent.get(`url`);
        // will NOT get us a resolved Promise
        const res1Pro = superagent.get(`https://dog.ceo/api/breed/
${data}/images/random`);
        const res2Pro = superagent.get(`https://dog.ceo/api/breed/
${data}/images/random`);
        const res3Pro = superagent.get(`https://dog.ceo/api/breed/
${data}/images/random`);

        const all = await Promise.all([res1Pro, res2Pro, res3Pro]);
        const imgs = all.map(element => {
         console.log('element.body:\n', element.body);
         console.log('element.body.message:\n', element.body.message);
         console.log(`\n`);

          return element.body.message;
        })

        console.log('imgs\n', imgs);

        // Writing the 3 images to './dog-img.txt'
        // join images each by a new line
        await writeFilePro('dog-img.txt', imgs.join('\n'));
        console.log('Random dog image saved to file');

    } catch (err) {
        console.log(`Error:\n${err}\n`);
        throw Error;
    }
    return '2: READY';
};
```

```
1: Will get dog pics!
Breed: labrador
element.body:
 {
  message: 'https://images.dog.ceo/breeds/labrador/JessieIncognito.jpg',
  status: 'success'
}
element.body.message:
 https://images.dog.ceo/breeds/labrador/JessieIncognito.jpg
```

```
element.body:
 {
  message: 'https://images.dog.ceo/breeds/labrador/n02099712_7406.jpg',
  status: 'success'
}
element.body.message:
 https://images.dog.ceo/breeds/labrador/n02099712_7406.jpg
element.body:
 {
  message: 'https://images.dog.ceo/breeds/labrador/n02099712_5853.jpg',
  status: 'success'
}
element.body.message:
 https://images.dog.ceo/breeds/labrador/n02099712_5853.jpg
imgs:
 [
  'https://images.dog.ceo/breeds/labrador/JessieIncognito.jpg',
  'https://images.dog.ceo/breeds/labrador/n02099712_7406.jpg',
  'https://images.dog.ceo/breeds/labrador/n02099712_5853.jpg'
]
Random dog image saved to file
2: READY
3: Done getting dog pics!
```

JS App.js M | JS index.js U | dog-img.txt U ✕ | dog-1.txt U

complete-node-bootcamp > 3-asynchronous-JS > starter > dog-img.txt

```
https://images.dog.ceo/breeds/labrador/JessieIncognito.jpg
https://images.dog.ceo/breeds/labrador/n02099712_7406.jpg
https://images.dog.ceo/breeds/labrador/n02099712_5853.jpg
```

```
// Solution 2

const getDogPic = async () => {
    try {
        // Stop operation of this line below, until it returns
        // & finally stores results to const data
        const data = await readFilePro(`${__dirname}/dog-1.txt`);
        console.log(`Breed:\n${data}\n`);

         // Suppose we wanna get 3 random Dog images at the same time
        // Storing a const resPro = superagent.get(`url`);
```

```
    // will NOT get us a resolved Promise

    // Store urls as an array

    const urls = [
      `https://dog.ceo/api/breed/${data}/images/random`,
      `https://dog.ceo/api/breed/${data}/images/random`,
      `https://dog.ceo/api/breed/${data}/images/random`
    ];

      // Destructuring urls to Promise.all => fetch each url
      const [res1Pro, res2Pro, res3Pro] = await
      Promise.all(urls.map(url =>
             fetch(url).then(res => res.json())))
             console.log('res1Pro.message: \n', res1Pro.message)
             console.log('res2Pro.message: \n', res2Pro.message)
             console.log('res3Pro.message: \n', res3Pro.message)

     // Using template strings to writeFilePromise for each url
     await writeFilePro('./dog-img.txt', `${res1Pro.message}\n
${res2Pro.message}\n${res3Pro.message}\n`);

    } catch (err) {
       console.log(err);
       throw Error;
    }
    return '2: READY';
};
```

```
1: Will get dog pics!
Breed: labrador
res1Pro.message:
 https://images.dog.ceo/breeds/labrador/n02099712_610.jpg
res2Pro.message:
 https://images.dog.ceo/breeds/labrador/n02099712_4913.jpg
res3Pro.message:
 https://images.dog.ceo/breeds/labrador/IMG_2752.jpg
2: READY
3: Done getting dog pics!
```

JS App.js M | JS index.js U | dog-img.txt U ✕ | dog-1.txt U

complete-node-bootcamp > 3-asynchronous-JS > starter > dog-img.txt

```
1   https://images.dog.ceo/breeds/labrador/n02099712_610.jpg
```

```
https://images.dog.ceo/breeds/labrador/n02099712_4913.jpg
https://images.dog.ceo/breeds/labrador/IMG_2752.jpg
```

==== Entire index.js:

```
const fs = require('fs');
const superagent = require('superagent');

// Create Arrow func to return new Promise that fs.readFile
const readFilePro = (filePath) => {
    return new Promise((resolve, reject) => { // executer func
        fs.readFile(filePath, (err, fileContent) => {
        // if there's an error --> reject() --> piped into .catch()
        if (err) reject('Could NOT find file');
        // Promise returns data to us will be piped into .then()
        resolve(fileContent);
        })
    });
}

// Create Arrow func to return new Promise that fs.writeFile
const writeFilePro = (filePath, data) => {
    return new Promise((resolve, reject) => {
        fs.writeFile(filePath, data, err => {
            if (err) reject('Could NOT write file');
            // fs.writeFile doesn't need to return data
            resolve('write succeeded');
        })
    })
}

// Async func lets other tasks keep running in Event Loop
const getDogPic = async () => {
    try {
        // Stop operation of this line below, until it returns
        // & finally stores results to const data
        const data = await readFilePro(`${__dirname}/dog-1.txt`);
        console.log(`Breed: ${data}`);

        // Suppose we wanna get 3 random Dog images at the same time
        // Storing a const resPro = superagent.get(`url`);
        // will NOT get us a resolved Promise
        // Solution 1
        const res1Pro = superagent.get(
```

```
            `https://dog.ceo/api/breed/${data}/images/random`
        );
        const res2Pro = superagent.get(
            `https://dog.ceo/api/breed/${data}/images/random`
        );
        const res3Pro = superagent.get(
            `https://dog.ceo/api/breed/${data}/images/random`
        );

        const all = await Promise.all([res1Pro, res2Pro, res3Pro]);
        const imgs = all.map(element => {
            // console.log('element: \n', element);
            console.log('element.body: \n', element.body);
            console.log('element.body.message: \n',
element.body.message);
            return element.body.message;
        });
        console.log('imgs: \n', imgs);
        await writeFilePro('./dog-img.txt', imgs.join('\n'));
        console.log('Random dog image saved to file');

        // Solution 2
        // Store urls as an array
        // const urls = [
        //     `https://dog.ceo/api/breed/${data}/images/random`,
        //     `https://dog.ceo/api/breed/${data}/images/random`,
        //     `https://dog.ceo/api/breed/${data}/images/random`
        // ];
        // // Destructuring urls to Promise.all => fetch each url
        // const [res1Pro, res2Pro, res3Pro] = await
Promise.all(urls.map(url =>
        //     fetch(url).then(res => res.json())))
        //     console.log('res1Pro.message: \n', res1Pro.message)
        //     console.log('res2Pro.message: \n', res2Pro.message)
        //     console.log('res3Pro.message: \n', res3Pro.message)

        // // Using template strings to writeFilePromise for each url
        // await writeFilePro('./dog-img.txt', `${res1Pro.message}\n
${res2Pro.message}\n${res3Pro.message}\n`);

        // For 1 url only
        // console.log('all Promise.all\n', all);
        // console.log('res.body.message', res.body.message);
        // await writeFilePro('dog-img.txt', res.body.message);
        // Writing the 3 images to './dog-img.txt'
```

```
        // join images each by a new line

    } catch (err) {
        console.log(err);
        throw Error;
    }
    return '2: READY';
};

/*
console.log('1: Will get dog pics!');
// const x = getDogPic();
// console.log(x);
// getDogPic();
getDogPic()
.then(x => {
    console.log(x);
    console.log('3: Done getting dog pics!');
})
.catch(err => {
    console.log('Error!');
});
*/

// Use IIFE
(async () => {
    try {
        console.log('1: Will get dog pics!');
        const x = await getDogPic();
        console.log(x);

    } catch(err) {
        console.log('Error!\n', err);
    } finally {
        console.log('3: Done getting dog pics!');
    }
}) ();

/*
// readFilePro(fileName)
// return a Promise before calling each of them
readFilePro(`${__dirname}/dog-1.txt`)
.then(data => {
    console.log(`Breed: ${data}`);
    // To keep chaining .then(), must return a Promise
```

```
    // return a Promise before calling each of them
    return superagent.get(`https://dog.ceo/api/breed/
${data}/images/random`);
})
.then(res => {
    console.log('res.body.message', res.body.message);
    // To keep chaining .then(), must return a Promise
    // return a Promise before calling each of them
    return writeFilePro('dog-img.txt', res.body.message)
    // fs.writeFile('dog-img.txt', res.body.message, err => {
    //     if (err) return console.log(err.message);
    //     console.log('Random dog image saved to file')
    // });
})
.then(() => {
    console.log('Random dog image saved to file!');
})
.catch(err => {
    console.log('err', err);
});
*/
```