Promises

Exercise 1(Todos)

- Open the url and https://jsonplaceholder.typicode.com/todos store it as an array of objects in your code.
- Create a function that returns a promise. This promise will resolve after a random time interval between 500 and 1000 milliseconds and will contain the array of objects.
- Handle the promise in a way that it initially logs "Loading" to the console. Once the promise resolves, log all data objects where the "completed" property is equal to true.

Exercise 2

Create a function named `squarePromise`, that takes a promise that resolves with a **number** or a **string**. Function should return a promise such that:

- If the input promise resolves with a number, the output promise resolves with the square of that number.
- If the input promise resolves with a string that we can turn into a number (like `"1234"`), the output promise resolves with the square of that number (`1522756` in this example)
- If the input promise resolves with a string that we cannot turn into a number (like `"asdf"`), then we reject with a message like `"Cannot convert 'asdf' to a number!"`
- If the input promise rejects with an error, the output promise rejects with the same error

Exercise 3

Create a function that takes in a single parameter and returns a new promise. Using `setTimeout`, after **500** milliseconds, the promise will either resolve or reject some value. If the parameter of the function is a **string**, the promise resolves with that same string uppercased. If the parameter of the function is anything but a string, it rejects with that same value.

Exercise 4

Create a function named `mapPromise` that can take in a promise and a transformer function and return a new transformed promise that follows these rules:

- If the first promise rejects with an error, the new promise rejects with that error.
- If the first promise resolves with a result, it calls the transformer with the value as an argument.
    - If the transformer returns with a value, the new promise resolves with that value.
    - If the transformer throws an error, the new promise rejects with that error.

Example:

```
const myPromise = new Promise((resolve, reject) => { resolve(2) })
const timesTwo = (val) => val * 2

mapPromise(myPromise, timesTwo).then(result => console.log(result))
```

Exercise 5

What will be the outputs of these codes?

```
Promise.resolve()
   .then(() => console.log(1))
   .catch(() => console.log(3))
   .then(() => { console.log(2); throw new Error(); })
   .then(() => console.log(4))
   .then(() => console.log(4))
Promise.resolve()
   .then(() => console.log(11))
   .then(() => { console.log(12); throw new Error(); })
   .catch(() => console.log(13))
   .catch(() => console.log(14))
   .then(() => console.log(15))
```

```
Promise.resolve()
   .then(data => {
       throw new Error('Api Error');
       return 1;
   })
   .then(data => console.log('ok'))
   .catch(error => {
       console.log(error.message);
       return "2";
   })
   .then(data => {
       console.log(data);
   })
```

```
console.log(1)
setTimeout(() => {
   console.log(2)
}, 20)
Promise.resolve()
   .then(() => {
       console.log(3)
       return 1
   })
   .catch(e => console.log(e, 4))
   .finally(res => console.log(res, 5))
   .then((res) => console.log(res))
console.log(6)
```

```
let a = 5;
setTimeout(() => {
   console.log(a);
   a = 10;
}, 0);
let p = new Promise((resolve, reject) => {
   console.log(a);
   a = 25;
   resolve();
});
p.then(res => {
   console.log('final result ', res)
});
console.log(a);
```