AJAX & Promises

Some of examples and definitions are from very good JS docs - https://developer.mozilla.org/ .

infoShare <academy/>

Hello!

I am Mateusz Choma

I am a scientific mind, passionate of technology, an engineer "squared" - a graduate of two universities in Lublin :)
As well, I am a JS developer, entrepreneur and owner of small software house - Amazing Design.

1.
AJAX

AJAX

What async is at all?

```
window.setTimeout(function() {
   console.log("second");
}, 100);

console.log("first");
```

AJAX
What async is at all?

LIVE CODING EXAMPLE

AJAX
What is AJAX?

AJAX - Asynchronous JavaScript and XML

AJAX is not a technology in itself, is a term from 2005, that describes an approach to using a number of existing technologies together, including: HTML or XHTML, Cascading Style Sheets, JavaScript, The Document Object Model, XML, and most importantly the XMLHttpRequest object.

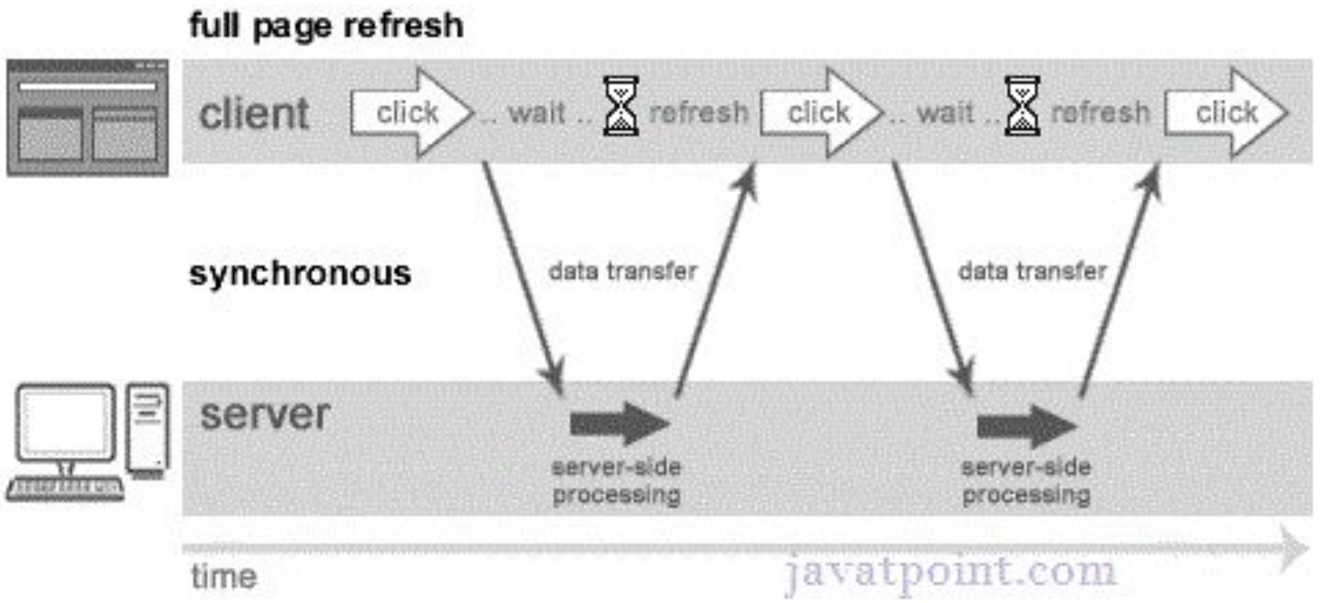
AJAX
What is AJAX?

The two major features of AJAX allow you to do the following:

- Make requests to the server without reloading the page
- Receive and work with data from the server

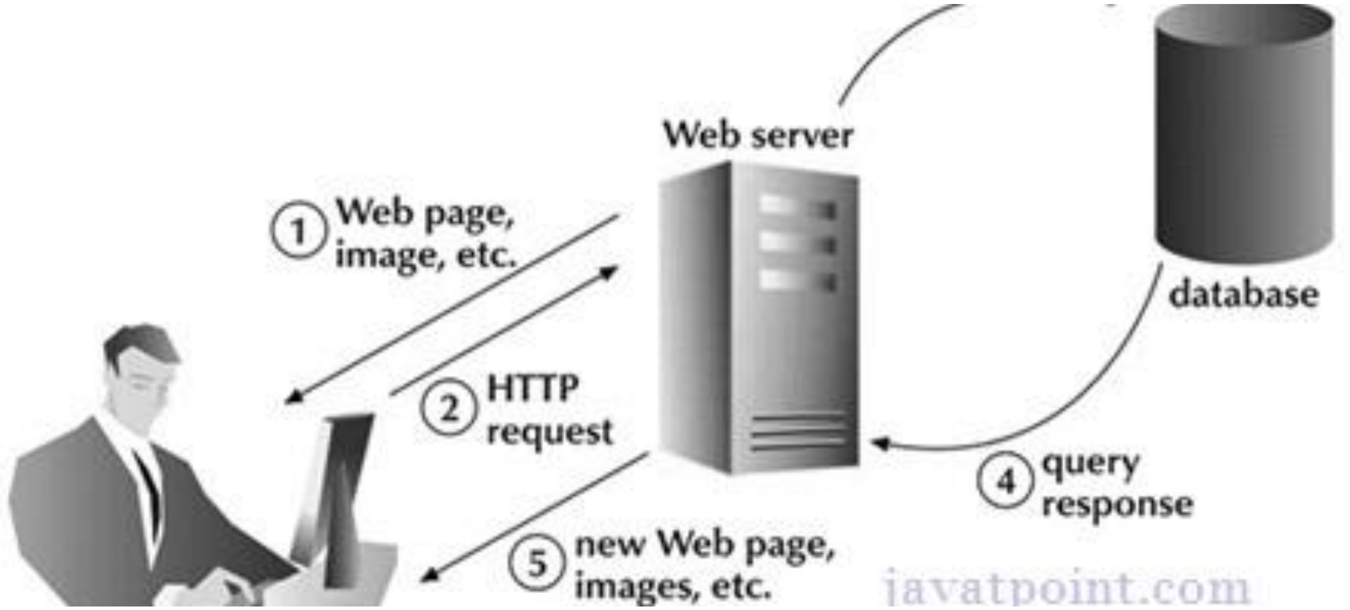
AJAX
Sync and async request

A synchronous request blocks the client until operation completes

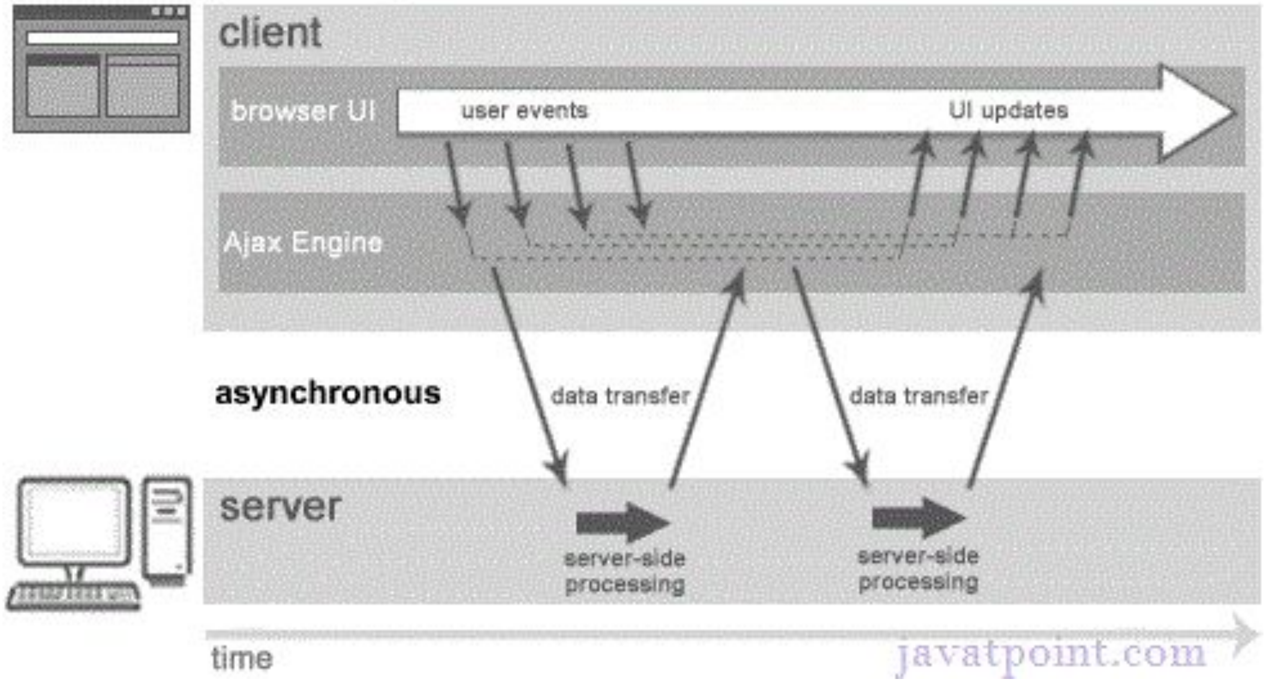


AJAX
Sync and async request

A synchronous request blocks the client until operation completes

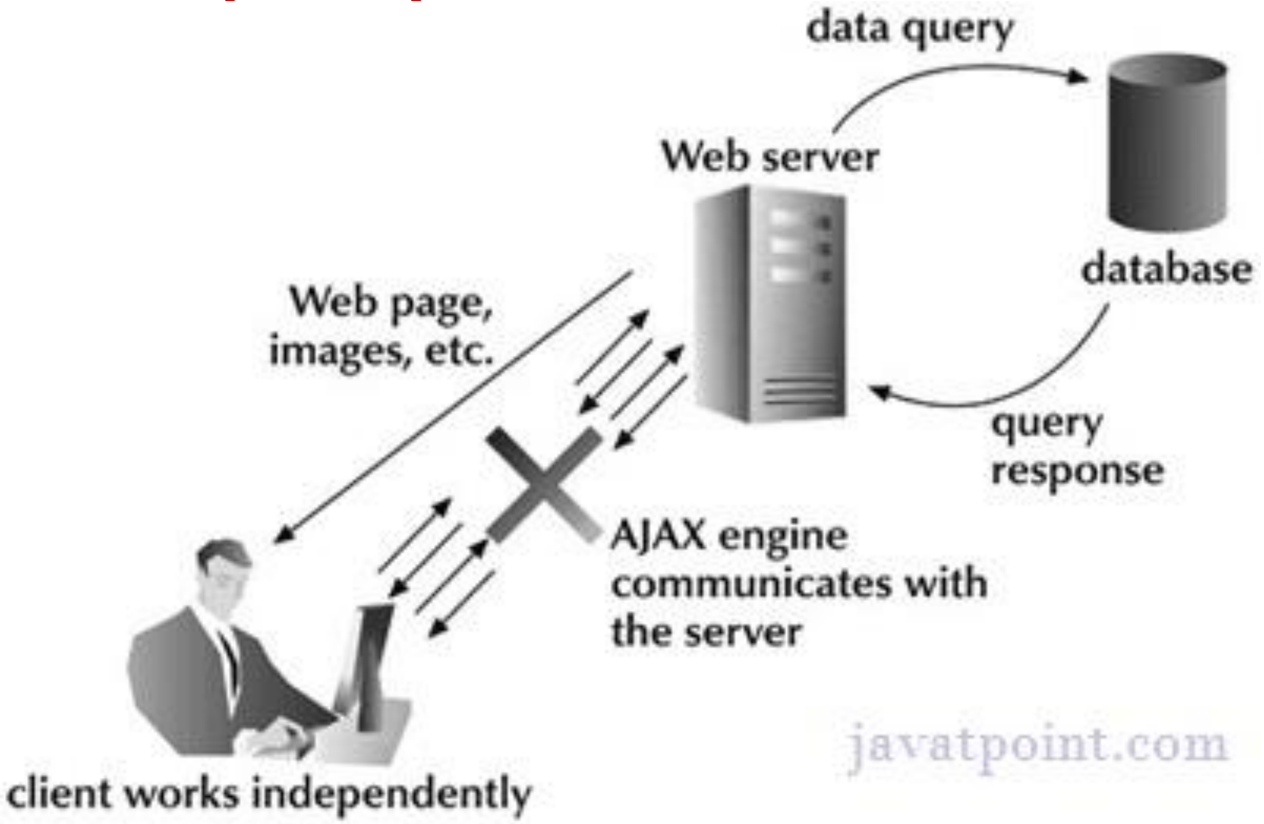


AJAX
Sync and async request



AJAX

Sync and async request



AJAX
XMLHttpRequest

XMLHttpRequest was originally designed by Microsoft.

Use XMLHttpRequest objects to interact with servers. You can retrieve data from a URL without having to do a full page refresh. This enables a Web page to update just part of a page without disrupting what the user is doing.

Despite its name, XMLHttpRequest can be used to retrieve any type of data, not just XML.

AJAX
XMLHttpRequest

You don’t have to operate on XMLHttpRequest object directly because it can be handled by jQuery methods or you can use Fetch API.

AJAX
XMLHttpRequest

```
// Old compatibility code, no longer needed

if (window.XMLHttpRequest) { // Mozilla, Safari, IE7+
    httpRequest = new XMLHttpRequest();
} else if (window.ActiveXObject) { // IE 6 and older
    httpRequest = new ActiveXObject("Microsoft.XMLHTTP");
}
// Now, Microsoft implemented XMLHttpRequest as well
```

AJAX
XMLHttpRequest

Example of direct XMLHttpRequest use:

https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started#Step_3_–_A_Simple_Example

2.
AJAX in jQuery

AJAX in jQuery
$.ajax

$.ajax - jQuery method to perform an asynchronous HTTP (Ajax) request.

```
$.ajax({
  url: 'https://randomuser.me/api/',
  dataType: 'json',
  success: function(data) {
    console.log(data);
  }
});
```

https://randomuser.me/
http://api.jquery.com/jquery.ajax/

AJAX in jQuery
$.ajax error handling

```
$.ajax({
  url: 'https://randomuSE.me/api/',
  dataType: 'json',
  success: function(data) {
    console.log(data);
  },
  error: function(err) {
    console.log('Error! ', err);
  }
});
```

Task 0

Get the user data from randomuser API and display his/her name, photo and contact data in HTML.

“ Task 0.1

Try to make an error :)
Handle it!

AJAX in jQuery
Simple HTTP methods

GET - is used to retrieve data - when we want to specify what data we want to retrieve i**t uses a query string in the URL.**

Example: https://randomuser.me/api/**?gender=female**

POST - used mostly for sending data, it **sends an object with our data not the data itself in the URL.**

AJAX in jQuery
Parsing an URL

encodeURIComponent(str);

decodeURIComponent(str);

```
encodeURIComponent(set1); // %3B%2C%2F%3F%3A%40%26%3D%2B%24
encodeURIComponent(set2); // -_.!~*'()
encodeURIComponent(set3); // %23
encodeURIComponent(set4); // ABC%20abc%20123 (the space
                             gets encoded as %20)
```

AJAX in jQuery
$.ajax POST method

Sending a form to formspree.io

```
$.ajax({
  method: "POST",
  url: "http://formspree.io/chomamateusz@gmail.com",
  data: { name: "John", location: "Boston" }
})
```

Task 1

Try to make a form and send it contents by $.ajax POST method to formspree.io

AJAX in jQuery
$.get & $.post shorten methods

The **$.get()** method requests data from the server with an HTTP GET request.

```
$.get("demo test.asp", function(data){
        console.log(data);
    });
```

The **$.post()** method requests data from the server using an HTTP POST request.

```
$.post("http://formspree.io/chomamateusz@gmail.com ",
    { name: "John", location: "Boston" },
    function(data){
        console.log(data);
    });
```

“ *Task 2*

Get the content of a .txt file, by $.get method on our server and put it in a div in the body.

AJAX in jQuery
$.load method

$.load loads data from the server and place the returned HTML into the matched element.

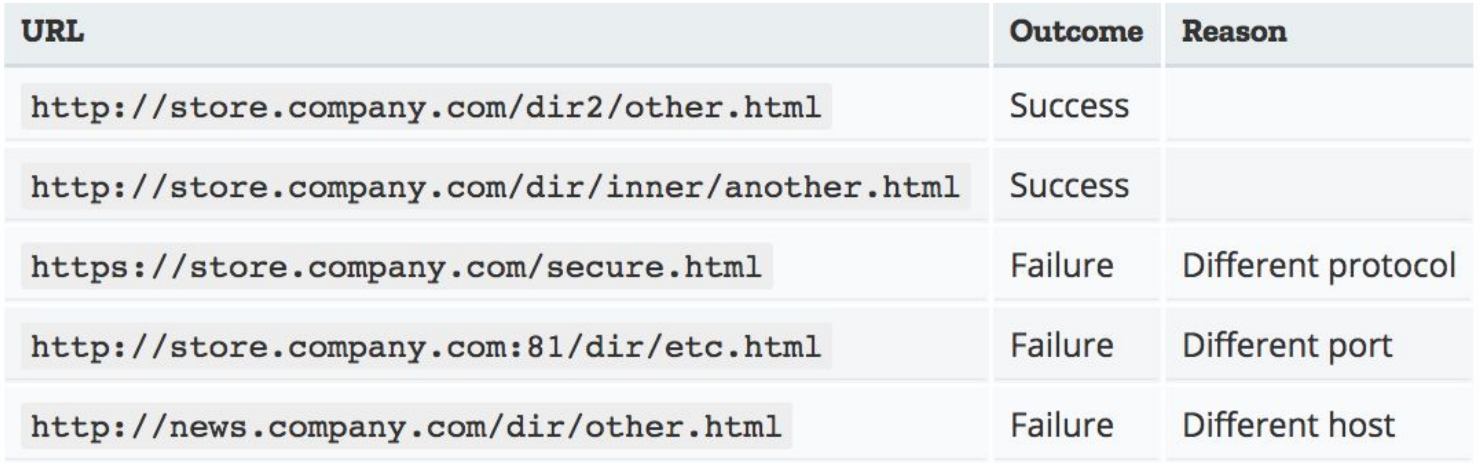
```
$( "#result" )
    .load( "/text.txt", function() {
      alert( "Load was performed." );
    });
```

Task 2.1

Get the content of a .txt file, by $.load method on our server and put it in a div in the body.

AJAX in jQuery
Single origin policy

Due to security reasons a web browser permits scripts contained in a web page to access data in a second web page, but only if both web pages have the same origin! An origin is defined as a combination of protocol, hostname, and port number.



URL	Outcome	Reason
`http://store.company.com/dir2/other.html`	Success	
`http://store.company.com/dir/inner/another.html`	Success	
`https://store.company.com/secure.html`	Failure	Different protocol
`http://store.company.com:81/dir/etc.html`	Failure	Different port
`http://news.company.com/dir/other.html`	Failure	Different host

AJAX in jQuery
CORS

To improve web applications, developers asked browser vendors to allow cross-domain requests.

The **Cross-Origin Resource Sharing** standard works by adding new HTTP headers that allow servers to describe the set of origins that are permitted to read that information using a web browser.

CORS proxy is a server that allows access from any domain and downloads data from other domains and passes it to your site without errors :)

http://cors-proxy.htmldriven.com/

Task 3

Let’s console.log lorem ipsum text from https://loripsum.net/api which doesn’t have CORS configured by proxy.

3.
Fetch API & promises

Fetch API & promises
What is Fetch API

“The **Fetch API** provides a JavaScript interface for accessing and manipulating parts of the HTTP pipeline, such as requests and responses. It also provides a global fetch() method that provides an easy, logical way to fetch resources asynchronously across the network.”

Fetch API & promises
What is Fetch API

https://caniuse.com/#search=fetch

https://github.com/github/fetch

Fetch API & promises
Simple data fetch using Fetch API

Basic use of fetch:

```
fetch('https://randomuser.me/api')
  .then(function(response) {
      return response.json()
  }).then(function(data) {
      console.log(data);
  });
```

Fetch API & promises
Wait!!! What is .then ???

This is fetch:

```
fetch('https://randomuser.me/api')
```

Fetch returns a Promise object.
Lines above, are promise state handlers.

```
   .then(function(response) {
         return response.json()      <- it also returns a promise
   }).then(function(data) {
         console.log(data);
   });
```

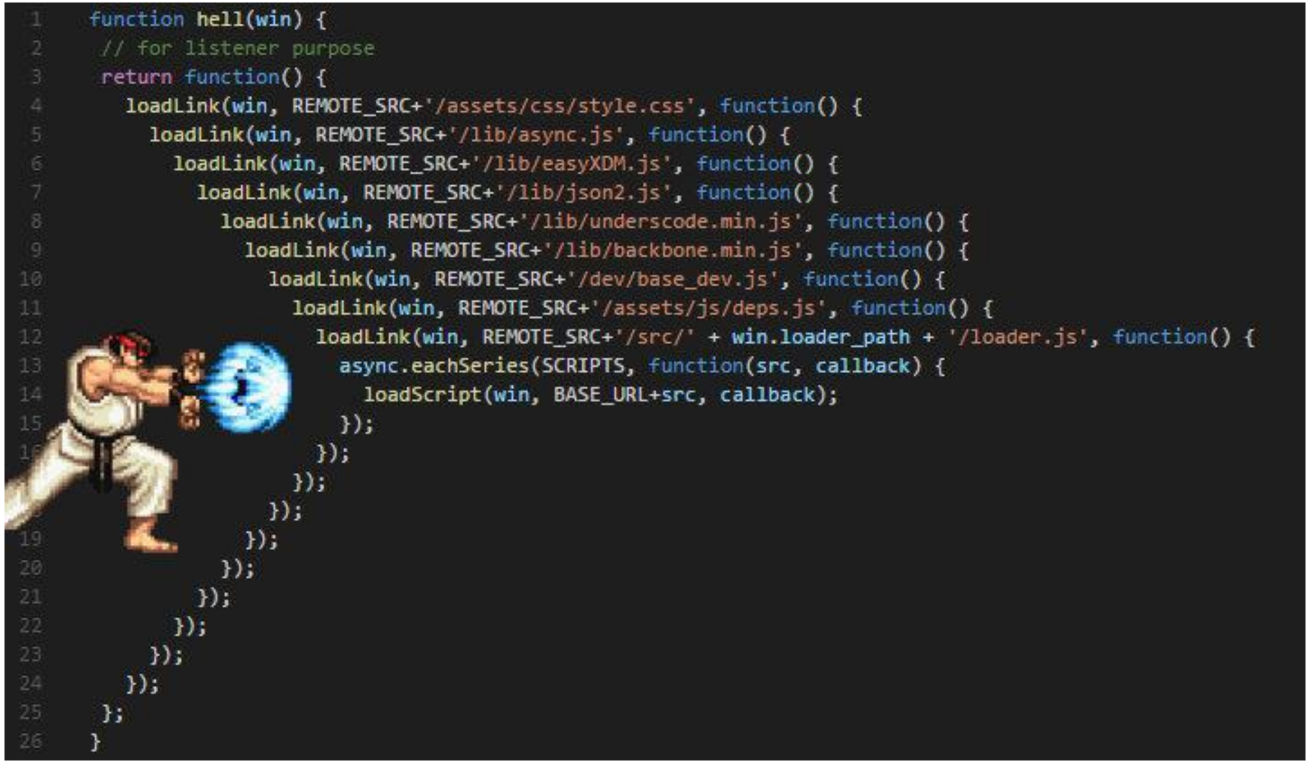
Fetch API & promises
Promise object

Let’s checkout in the browser what a Promise **object** is!

Notice that Promise is an object and it can be passed around, and assigned to variables!

Fetch API & promises

Why to use promises? To avoid hell :)



```
function hell(win) {
 // for listener purpose
 return function() {
   loadLink(win, REMOTE_SRC+'/assets/css/style.css', function() {
     loadLink(win, REMOTE_SRC+'/lib/async.js', function() {
       loadLink(win, REMOTE_SRC+'/lib/easyXDM.js', function() {
         loadLink(win, REMOTE_SRC+'/lib/json2.js', function() {
           loadLink(win, REMOTE_SRC+'/lib/underscode.min.js', function() {
             loadLink(win, REMOTE_SRC+'/lib/backbone.min.js', function() {
               loadLink(win, REMOTE_SRC+'/dev/base_dev.js', function() {
                 loadLink(win, REMOTE_SRC+'/assets/js/deps.js', function() {
                   loadLink(win, REMOTE_SRC+'/src/' + win.loader_path + '/loader.js', function() {
                     async.eachSeries(SCRIPTS, function(src, callback) {
                       loadScript(win, BASE_URL+src, callback);
                     });
                   });
                 });
               });
             });
           });
         });
       });
     });
   });
 };
}
```

Fetch API & promises
Promise states

A **Promise** can be in one of these states:

- **pending**: initial state, neither fulfilled nor rejected
- **fulfilled**: meaning that the operation completed successfully
- **rejected**: meaning that the operation failed

Fetch API & promises
Promise methods

`.catch(onRejected)`
Appends a rejection handler callback to the promise, and returns a new promise resolving to the return value of the callback if it is called.

`.then(onFulfilled, onRejected)`
Appends fulfillment and rejection handlers to the promise, and returns a new promise resolving to the return value of the called handler.

Fetch API & promises
Fetch methods to get the response body

response.json() - we use it to parse JSON from response and obtain a JavaScript object

response.text() - we use it to obtain a plain text from response

Both methods returns a Promise so we can call then to continue work when the JSON or text will be extracted from response body.

Task 4

Get the 10 users data from randomuser API and display his/her name & lastname, photo and telephone data in HTML using Fetch.

“ *Task 4.1*

Make and catch an error!

Fetch API & promises
Making Promises

```
let myFirstPromise = new Promise(function(resolve, reject) {
  // We call resolve(...) when what we were doing asynchronously was successful, and
reject(...) when it failed.
  // In this example, we use setTimeout(...) to simulate async code.
  setTimeout(function(){
    resolve("Success!"); // Yay! Everything went well!
  }, 250);
});

myFirstPromise.then(function(successMessage) {
  // successMessage is whatever we passed in the resolve(...) function above.
  // It doesn't have to be a string, but if it is only a succeed message, it probably
will be.
  console.log("Yay! " + successMessage);
});
```

Task 5

Make new promise that, wraps Fetch call to randomuser API which parse JSON data, and resolve the promise when data is parsed.

Fetch API & promises

Combining Promises

The **Promise.all()** method returns a single Promise that resolves when all of the promises in the iterable argument have resolved or when the iterable argument contains no promises. It rejects with the reason of the first promise that rejects.

```
Promise.all(iterable);
```

Iterable is an object that can be iterated - an array for example.

Fetch API & promises
Combining Promises

```
var p1 = new Promise(function(resolve, reject) {
  setTimeout(resolve, 1000, 'one');
});
var p2 = new Promise(function(resolve, reject) {
  setTimeout(resolve, 2000, 'two');
});
var p3 = new Promise(function(resolve, reject) {
  setTimeout(resolve, 3000, 'three');
});
var p4 = new Promise(function(resolve, reject) {
  setTimeout(resolve, 4000, 'four');
});
var p5 = new Promise(function(resolve, reject) {
  reject('reject');
});
```

Fetch API & promises
Combining Promises

```
//You can also use .catch
Promise.all([p1, p2, p3, p4, p5]).then(values => {
  console.log(values);
}).catch(reason => {
  console.log(reason)
});

//From console:
//"reject"
```

Task 6

Combine 2 fetch and data parse randomuser API, and console.log the parsed data.