El concepto de promise chaining o encadenamiento de promesas es un clásico cuando hablamos de programación asíncrona. Vamos a construir un ejemplo que nos ayude a entenderlo de una forma sencilla . Para ello vamos a usar una aplicación de Node.js que nos devuelve una lista de datos via REST y que nos permite añadir mas elementos a la lista. Usamos yarn e instalados express y body parser (libreria para gestionar JSON)

yarn install express body-parser

Una vez hecho esto construimos la aplicación de servidor del lado de JavaScript.

```
var express = require('express');
var bodyParser = require('body-parser')
var app = express();



app.use(express.static('node_modules/jquery/dist/'));
app.use(express.static('cliente'));
app.use(bodyParser.json())

var lista=["hola","que","tal","estas"];

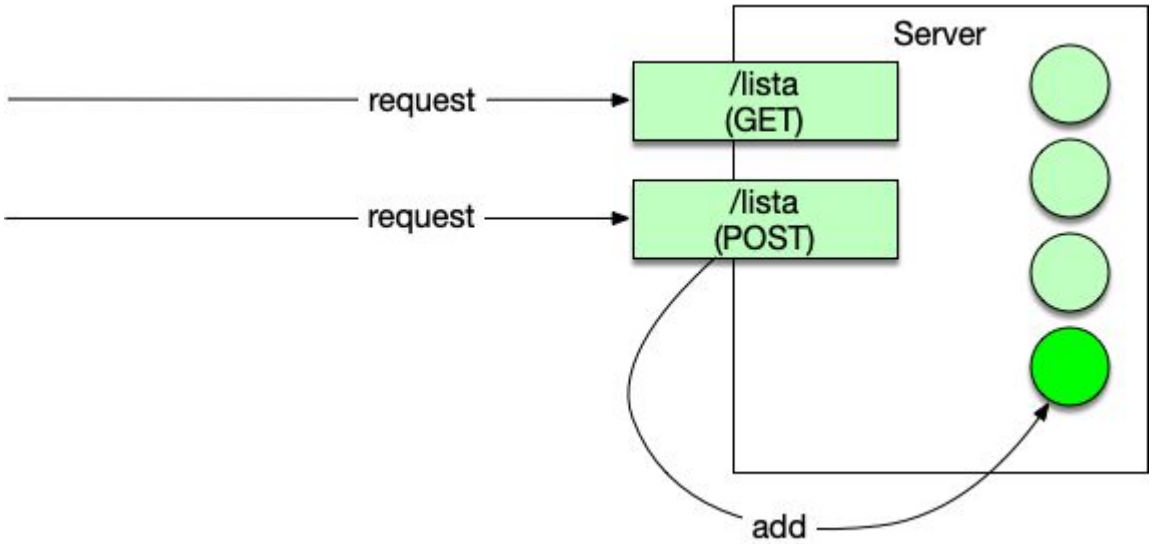
app.get('/lista', function (req, res) {
  res.send(lista);
});

app.post('/lista',function(request,response) {
lista.push(request.body.dato);

  response.send({mensaje:"ok"});
})
```

```
app.listen(3000, function () {
  console.log('servidor en el puerto 3000');
});
```

Este aplicación es muy sencilla y solo dispone de dos urls de acceso lista (get) y lista(post)



Cada vez que hacemos una petición GET nos devuelve la lista de elementos y cada vez que hacemos una petición POST nos añade nuevos elementos a la lista. Vamos a construir un interface de usuario que nos permita manejar ambas urls.

```
<html>
<script type="text/javascript" src="jquery.js">
</script>
<script type="text/javascript" src="001.js">
</script>

<body>
<div id="lista">

</div>
```

```
<input type="button" value="cargarLista" id="cargarLista"/>

<input type="text" id="texto"/>
<input type="button" value="nuevo" id="nuevo"/>
</body>
</html>
```

En este caso simplemente disponemos de un botón que carga la lista y una caja de texto con otro botón que nos añade nuevos elementos:



El código de JavaScript que tenemos asociado con jQuery es el siguiente:

```
$(document).ready(function() {

$("#cargarLista").click(function() {

  $.get("lista",function(datos) {

    var $lista=$("#lista").empty();
    datos.forEach(function(e){

      $(`<p>${e}</p>`).appendTo($lista);

    })

  })
})
```

```
$("#nuevo").click(function() {

  $.ajax({

    url:"lista",
    type:"post",
    dataType:"json",
    contentType: 'application/json',
    success: function (data) {
                console.log(data);
    },
    data: JSON.stringify({dato:$("#texto").val()})
  });

})


})
```

Hasta aquí todo es correcto un botón carga la lista y otro botón se encarga de insertar un nuevo elemento. No parece que tengamos ningún problema . Ahora bien el programa no funciona de la forma adecuada ya que cuando pulsamos el botón de cargar nos carga los elementos en la lista.



El problema le tenemos cuando insertamos un elemento , eso realizará una petición post al

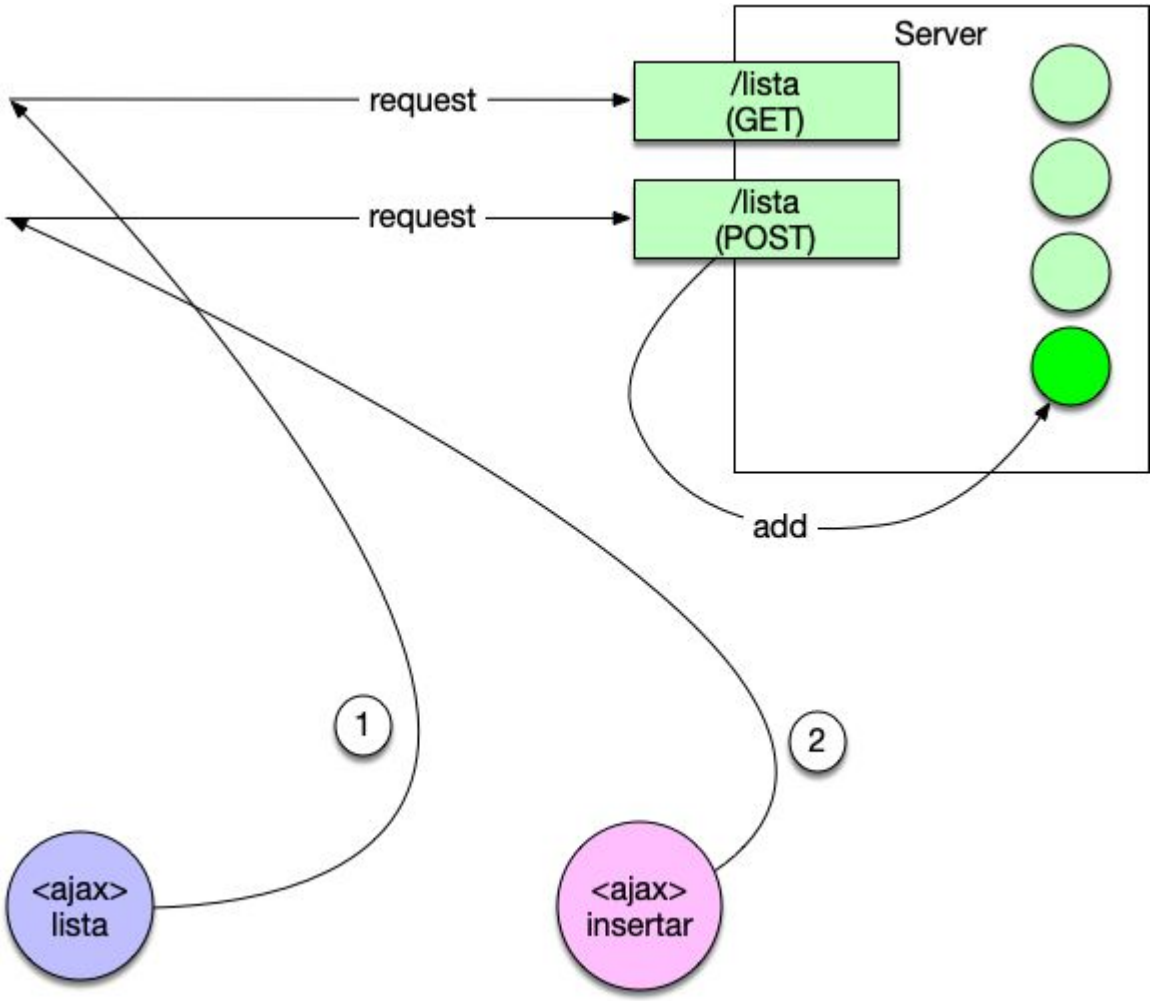
servidor e insertará el elemento en la lista. Sin embargo en el cliente no veremos ningún cambio.



Deberemos volver a cargar la pagina para ver las novedades.

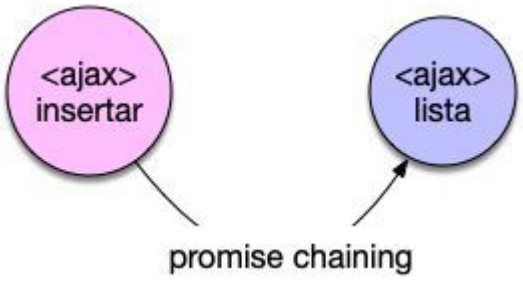


Este no es el comportamiento que queremos , sin embargo es lo que esta ocurriendo ¿A qué se debe ? . Se debe a que cada una de las peticiones Ajax es totalmente independiente.

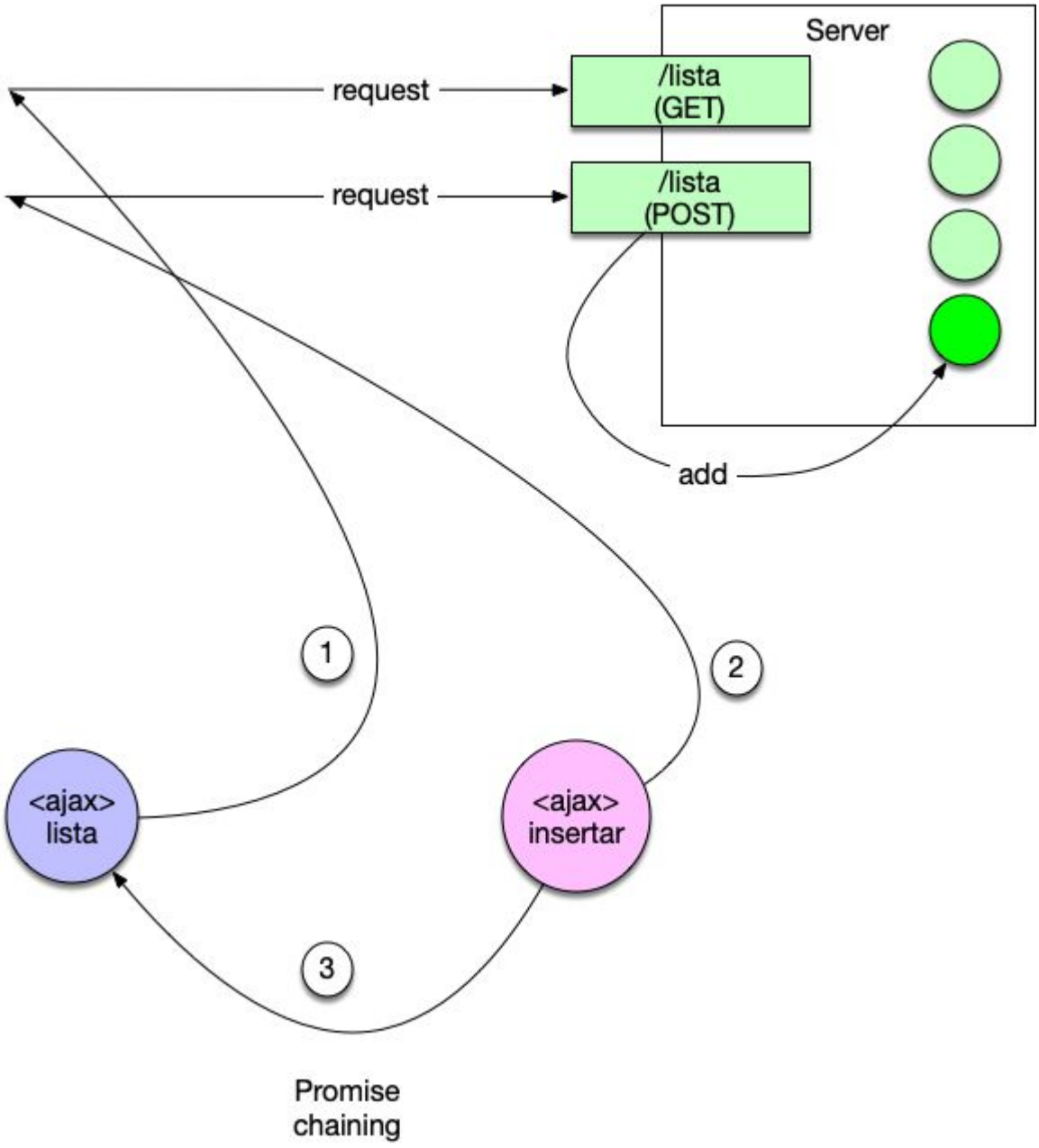


Promise Chaining y JavasScript

Nosotros lo que necesitamos es que estas peticiones Ajax estén relacionadas que funcionen como promesas encadenadas (promise chaining) de JavaScript . Es decir cuando yo realice una petición de inserción que automáticamente se realice otra de listado al terminar la primera. A nivel del diagram sería algo así:



En nuestro caso nuestras peticiones serían de este estilo :



Vamos a verlo en código:

```
$(document).ready(function() {

$("#cargarLista").click(function() {

    cargarLista().then(function(datos) {
```

```
      var $lista=$("#lista").empty();
      datos.forEach(function(e){


        $(`<p>${e}</p>`).appendTo($lista);

      })


    });


  })

function cargarLista() {

    return $.get("lista");

}

function insertar() {

return  $.ajax({

    url:"lista",
    type:"post",
    dataType:"json",
    contentType: 'application/json',
    data: JSON.stringify({dato:$("#texto").val()})
  });
}
```

```
$("#nuevo").click(function() {

  insertar().then(cargarLista).then(function(datos) {


    var $lista=$("#lista").empty();
    datos.forEach(function(e){
      $(`<p>${e}</p>`).appendTo($lista);

    })


  })

})


})
```

Lo que hemos hecho es convertir las peticiones de ajax en funciones . Estas funciones devuelven JQuery Promises y se pueden enlazar unas con otras a través del método then

```
insertar().then(cargarLista).then(function(datos) {


    var $lista=$("#lista").empty();
    datos.forEach(function(e){
      $(`<p>${e}</p>`).appendTo($lista);

    })
```

Ahora cada vez que insertemos un elemento la lista se recargará de forma automática ya

que se produce un promise chaining en Javascript



Con esto las cosas se ven de una forma más natural

Otros artículos relacionados:

CURSO Introducción Patrones Diseño

GRATIS

APUNTATE!!

1. jQuery getScript y carga dinámica
2. Utilizando jQuery on off
3. jQuery $.ajax ,$.get, $.post
4. Api de promesas