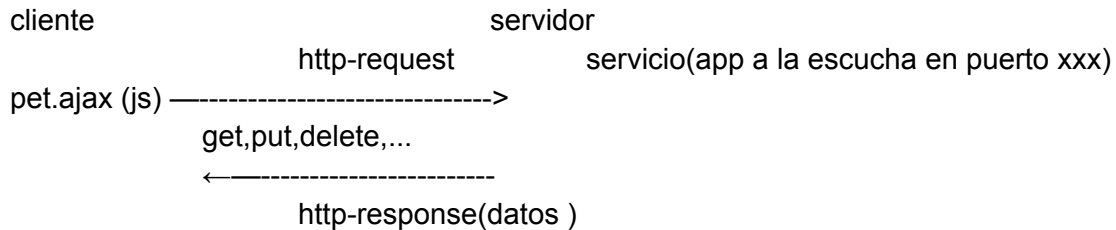


PET. AJAX con js

consiste en hacer una pet. a un servicio web instalado en un servidor remoto para SUBIR/RECIBIR datos (un servicio web es un app. corriendo en un servidor web q devuelve o admite datos en formato xml, json, texto,...)



1º forma: usando prototipo XMLHttpRequest
lo primero es crearnos un objeto del prototipo

```
var petAjax=new XMLHttpRequest();
```

este objeto tiene props y metodos.

Props:

- timeout ← tiempo de espera en seg para hacer pet.ajax al servicio web
- readystate ← entero q marca el desarrollo de la pet.ajax al servidor
- status ← el codigo de respuesta del servidor HTTP-RESPONSE

metodos:

- responseText, responseXML ← recuperas datos del servicio
- open('metodo http'. 'url') ← inicializas el servicio contra esa url y usando ese metodo http
- setRequestHeader('nombre cabecera', valor) ← inicializas cabecera en HTTP-REQUEST
- send(), send(datos) ← lanzar la peticion ajax, para detectar como cambia el readystate hay q añadir un manejador de eventos al evento "onreadystatechange"

```
var petAjax= new XMLHttpRequest();
```

```
petAjax.open('GET','https://pokeapi.co/api/v2/pokemon/pikachu/');
```

```
petAjax.addEventListener('readystatechange',
```

```
    function(ev){
        if(petAjax.readyState==4 && petAjax.status==200){
            //hemos recibido la respuesta del servicio
            readyState=4 y esta ok,con codigo status=200
            var pokemon = petAjax.responseText;
            console.log(pokemon);
        }
    });
```

```
petAjax.send();
```

2º forma usando jQuery, metodo .ajax('url', { opciones}) ←- devuelve objeto prototipo Promise

<https://api.jquery.com/jQuery.ajax/>

con metodo .then(funcion_recibe_datos) se recogen los datos si la pet. ha ido ok
con metodo .catch(funcion_recibe_errores) se recogen los errores si la pet. ha ido mal
con metodo .finally(funcion_sin_parametros) se ejecuta al finalizar la pet. ajax, tanto si ha ido ok como si ha ido mal

```
jQuery.ajax('GET','https://pokeapi.co/api/v2/pokemon/pikachu/')  
  .then( (datos)=> console.log(datos) )  
  .catch( (errores)=> console.log(errores))  
  .finally( ()=> console.log('peticion finalizada'))
```

3º forma usando api FETCH-API, para evitar el uso de librerias externas como jquery, esta dentro del motor javascript de los navegadores mas modernos, y funciona igual q jQuery.ajax() pero usando funcion:

fetch('url', { opciones}) ←- devuelve una promesa q se intercepta con .then() y .catch()
<https://developer.mozilla.org/en-US/docs/Web/API/fetch>