CIP FP cecheste

Desarrollo Web en Entorno Cliente

Javascript

Asincronía [2]

API REST

API: Application Programming Interface (Interfaz de Programación de Aplicaciones) es una capa que permite que dos aplicaciones se comuniquen y compartan información.

REST: Representational State Transfer (Transferencia de Estado Representacional). Es cliente envía datos al servidor, este los procesa, y también devuelve datos. Define un conjunto de funciones, GET, POST, PUT, DELETE, … (CRUD)

Ajax

XMLHttpRequest: Objeto para realizar conexiones asíncronas con el servidor.

onreadystatechange: contiene el manejador del evento que es invocado cuando se dispara el evento `readystatechange`. Esto sucede cada vez que cambia el valor de la propiedad `readyState` de `XMLHttpRequest`

readyState: Guarda el estado de la petición, que puede ser:

- 0: Todavía no se ha llamado a `open()`.
- 1: Todavía no se ha llamado a `send()`.
- 2: send() ya ha sido invocado, cabeceras y estado preparados.
- 3: Descargando, `responseText/responseXML` tiene información parcial.
- 4: Completo.

status: El estado de la respuesta al pedido. Éste es el código `HTTPresult` que devuelve el servidor. `200`, `404`, `500`, …

statusText: Igual que el anterior, pero en vez de enviar el código, envía el texto.

responseText: La respuesta al pedido como texto.

responseXML: La respuesta al pedido como XML.

open(): Método donde elegimos el método de la petición, `GET`/`POST`, y la `URL` del archivo.

send(): Envía los parámetros de la petición de tipo `POST`, por defecto *`null`*.

Ajax

1. Creamos una instancia del objeto XMLHttpRequest.

```
peticion_http = new XMLHttpRequest();
```

2. Preparamos la función que se encargará de procesar la respuesta de nuestro servidor.

```
peticion_http.addEventListener("readystatechange", mostrar);
```

3. Realizamos la petición HTTP:

```
peticion_http.open('GET', 'http://dominio/movidas/texto.txt', true);
peticion_http.send(null);
```

4. Creamos la función mostrar() y valoramos la respuesta del servidor. Si todo está bien, mostramos un alert con el contenido del texto devuelto.

```
function mostrar() {
    if(peticion_http.readyState == 4 && peticion_http.status == 200) {
        alert(peticion_http.responseText);
    }
}
```

Ajax

Resumiendo...

```
let peticion = new XMLHttpRequest();

peticion.addEventListener("readystatechange", procesar);

peticion.open("GET", "URL de archivo", true);
peticion.send(null);

function procesar(){
    if (peticion.readyState == 4 && peticion.status == 200){
        console.log(peticion.responseText);
    }
}
```

¿Y si queremos enviar información de un formulario?

Ajax

GET

¿Cómo se envía la información cuando utilizamos `method="get"` en un formulario HTML?

```
<form method="get" action="destino.html">
    <input type="text" id="nombre" name="nombre">
    <input type="number" id="edad" name="edad">
</form>
```

```
https://destino.html?nombre=asdf&edad=45
```

¿Por lo tanto?¿Cómo enviaríamos esa información mediante AJAX?

```
let url = `https://destino.html?nombre=${nombre.value}&edad=${edad.value}`;
peticion.open("get", url);
```

Ajax

POST

¿Cómo se envía la información cuando utilizamos `method="post"` en un formulario HTML?

```
<form method="get" action="destino.html">
    <input type="text" id="nombre" name="nombre">
    <input type="number" id="edad" name="edad">
</form>
```

?

¿Por lo tanto?¿Cómo enviaríamos esa información mediante AJAX?

```
let url = `https://destino.html`;
peticion.open("post", url);
peticion.send(formulario);
```

FormData

La interfaz `FormData` proporciona una manera sencilla de construir un conjunto de parejas clave/valor que representan los campos de un formulario y sus valores

Los elementos (`entries`) de un `FormData` pueden recorrerse mediante un `for...of`.

```
let formData = new FormData();
```

Los métodos más interesantes de un `FormData` son:

`append(clave, valor);`

```
formData.append("nombre", "movidas");
```

`delete(clave);`

```
formData.delete("nombre");
```

FormData

```
entries();
```

```
for (const entrada of formData.entries()) {
    console.log(entrada[0], entrada[1]);
}
```

```
get();
```

```
formData.append("nombre", "movidas");
formData.get("nombre"); //movidas
```

```
 set();
```

```
formData.append("nombre", "movidas");
formData.set("nombre", "requetemovidas");
```

Ajax y FormData

```
<form method="get" action="destino.html" id="formulario">
    <input type="text" id="nombre" name="nombre">
    <input type="number" id="edad" name="edad">
</form>
```

```
let peticion = new XMLHttpRequest();

peticion.addEventListener("readystatechange", procesar);

let formData = new FormData(formulario);
let url = `https://destino.html`;

peticion.open("post", url);
peticion.send(formData);

function procesar(){
    if (peticion.readyState == 4 && peticion.status == 200){
        console.log(peticion.responseText);
    }
}
```

Json

XML es a HTML lo que **Json** (*JavaScript Object Notation*) a un array. Json es un formato sencillo para el intercambio de datos que se creó para Javascript que ha terminado por imponerse a XML, separándose del lenguaje y extendiéndose su uso al resto de lenguajes.

```
{
    'nombre': 'Gazpacho',
    'tipo': 'Sopa fría',
    'origen': {
    'pais': 'España',
        'region': 'Andalucía'
    },
    'ingredientes': [
        {
            'nombre': 'Tomate',
            'cantidad': '1 kg'
        },
        {
            'nombre': 'Pimiento verde',
            'cantidad': '100 g'
        }
    ]
}
```

Ajax y Json

Cuando recibimos la información del servidor, lo debemos hacer en modo texto (`responseText`), para luego convertirlo en Json, `JSON.parse()`.

```
let json;
let peticion = new XMLHttpRequest();

peticion.addEventListener("readystatechange", procesar);

peticion.open("GET", "URL de archivo", true);
peticion.send(null);

function procesar(){
    if (peticion.readyState == 4 && peticion.status == 200){
        json = JSON.parse(peticion.responseText);
    }
}
```

Ajax y Json

Con la información que tenéis en los datos abiertos del Ayuntamiento de Valencia, debéis hacer dos listados con la siguiente información del precio de la vivienda según Fotocasa e Idealista:

- Nombre del distrito.
- Precio del metro cuadrado en 2022.
- Precio del metro cuadrado en 2010.
- Indicar si el precio ha subido o bajado con el icono de una flecha verde hacia arriba o roja hacia abajo.
- Preció máximo histórico y año en el que ocurrió.