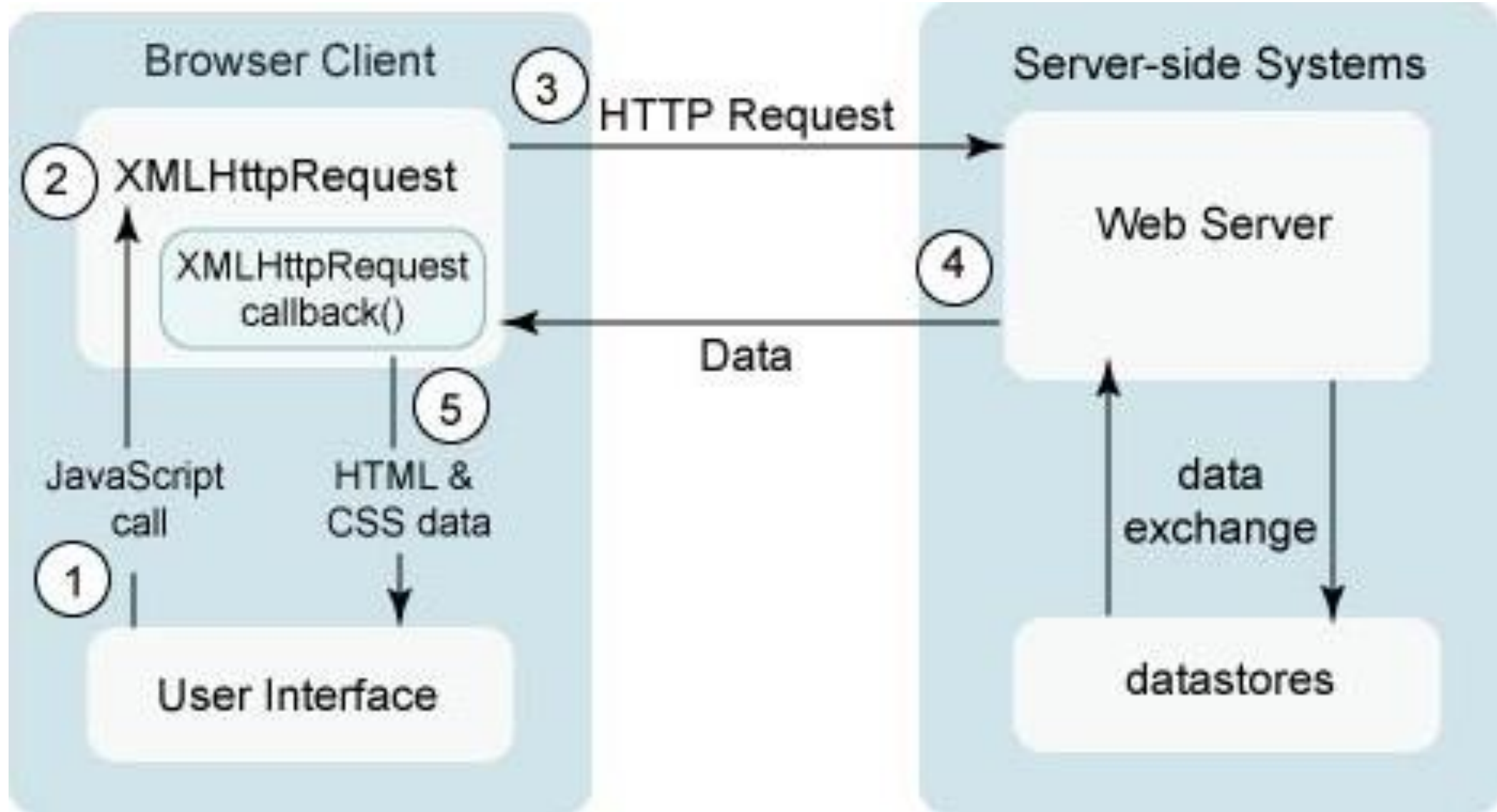


# AJAX

- Ajax es acrónimo de **A**synchronous **J**avaScript **A**nd **X**ML
- No se trata de un lenguaje ni de un protocolo concreto sino más bien de una tecnología utilizada en el desarrollo web.
- El objeto central utilizado por AJAX es **XMLHttpRequest (XHR)**.
- La petición y respuesta al servidor se realiza mediante HTTP.
- Gracias a AJAX es posible enviar y recibir información al servidor sin recargar la página.
- El formato de los datos enviados y recibidos puede ser: XML, HTML, JSON, texto plano, etc...
- AJAX es por naturaleza asíncrono: el cliente no tiene porqué esperar de manera síncrona la respuesta del servidor.

# AJAX

- Fases en la comunicación cliente-servidor con AJAX:



# AJAX. Ejemplo1.html

- Ajax permite enviar una petición y obtener una respuesta sin recargar la página.
- El ejemplo más sencillo para cargar un fichero de texto plano:

```
<div class="container"></div>
<input type="button" id="btn" value="Leer">
<script>
  const xhr = new XMLHttpRequest();
  xhr.addEventListener('load', function () {
    if (xhr.status === 200) {
      document.querySelector('.container').innerHTML = xhr.response;
    }
  });
  document.querySelector('#btn').addEventListener('click', () => {
    xhr.open('get', 'http://localhost:5500/assets/datos.txt');
    xhr.send();
  });
</script>
```

En la propiedad "response" obtenemos la respuesta del servidor (puede ser un texto, un blob, un document, etc...).

Pasos a seguir:

- 1.- Crear el objeto **XMLHttpRequest**.
- 2.- Capturar el evento **load** que se produce cuando se devuelven los datos.
- 3.- Configurar la petición con el método **open()**.
- 4.- Enviar la petición con el método **send()**.

El Status HTTP 200 significa que todo es correcto.

# AJAX. Ejemplo2.html

- También es posible leer mediante AJAX el contenido de un fichero y cargarlo y renderizarlo en nuestra propia página web. Ejemplo:

```
<div class="container"></div>
<input type="button" id="btn" value="Leer">
<script>
  const xhr = new XMLHttpRequest();
  xhr.addEventListener('load', function () {
    if (xhr.status === 200) {
      document.querySelector('.container').innerHTML = xhr.response;
    }
  });
  document.querySelector('#btn').addEventListener('click', () => {
    xhr.open('get', 'http://localhost:5500/assets/datos.html');
    xhr.send();
  });
</script>
```

Datos.html:

```
<ul>
  <li>Azul</li>
  <li>Rojo</li>
  <li>Negro</li>
</ul>
```

Fichero renderizado:

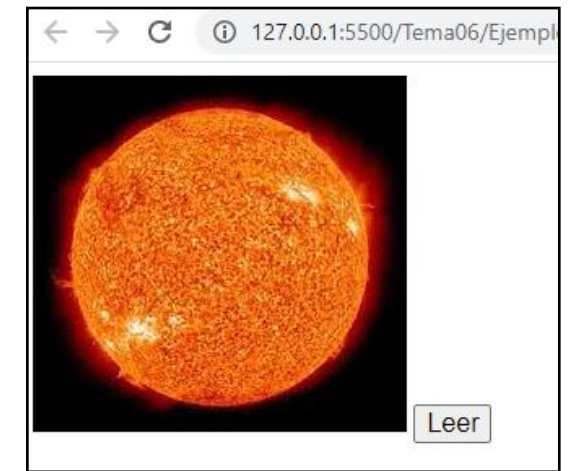
- Azul
- Rojo
- Negro

Leer

# AJAX. Ejemplo3.html

- Descargar un fichero del servidor.

```
<img id="img">
<input type="button" id="btn" value="Leer">
<script>
  const xhr = new XMLHttpRequest();
  xhr.addEventListener('load', function () {
    if (xhr.status === 200) {
      const img=document.querySelector('#img');
      img.src=URL.createObjectURL(xhr.response);
    }
  });
  document.querySelector('#btn').addEventListener('click', () => {
    xhr.open('get', 'http://localhost:5500/assets/sol.jpg');
    xhr.responseType="blob";
    xhr.send();
  });
</script>
```



A partir de la respuesta se construye una imagen con el método "URL.createObjectURL()"

No se puede utilizar "responseText", usamos "response" porque los datos son binarios.

En este caso especificamos mediante la propiedad "responseType" el tipo de datos que deseamos como respuesta del servidor.

# AJAX. Ejemplo4.html

- Leer datos JSON:

```
const xhr1 = new XMLHttpRequest();
xhr1.addEventListener('load', function () {
  if (xhr1.status === 200) {
    console.log(JSON.parse(xhr1.response));
  }
});
xhr1.open('get', 'http://localhost:5500/assets/datos.json');
xhr1.send();
```

```
const xhr2 = new XMLHttpRequest();
xhr2.addEventListener('load', function () {
  if (xhr1.status === 200) {
    console.log(xhr2.response);
  }
});
xhr2.open('get', 'http://localhost:5500/assets/datos.json');
xhr2.responseType = 'json';
xhr2.send();
```

```
[
  {
    "titulo": "Sol",
    "nombreImagen": "sol.jpg"
  },
  {
    "titulo": "Luna",
    "nombreImagen": "luna.jpg"
  }
]
```

Los dos ejemplos de código son iguales pero en el primer caso se obtienen los datos en modo texto y se parsean con `JSON.parse()`. En el segundo se indica mediante la propiedad `responseType` que los datos descargados son de tipo JSON.

```
▼ (2) [{...}, {...}] ⓘ
  ▶ 0: {titulo: 'Sol', nombreImagen: 'sol.jpg'}
  ▶ 1: {titulo: 'Luna', nombreImagen: 'luna.jpg'}
  length: 2
  ▶ [[Prototype]]: Array(0)

▼ (2) [{...}, {...}] ⓘ
  ▶ 0: {titulo: 'Sol', nombreImagen: 'sol.jpg'}
  ▶ 1: {titulo: 'Luna', nombreImagen: 'luna.jpg'}
  length: 2
  ▶ [[Prototype]]: Array(0)
```

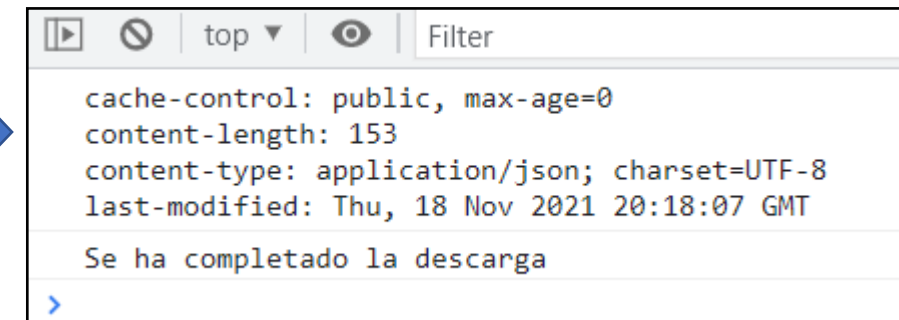
# Eventos load y readystatechange. Ejemplo5.html

- El evento **load** se produce cuando la petición se ha completado correctamente.
- Si se desea realizar un control más fino (controlar estados intermedios o posibles errores) es posible utilizar el evento **readystatechange** y la propiedad **readyState**.

```
const xhr = new XMLHttpRequest();
xhr.addEventListener('readystatechange', function () {
  switch (xhr.readyState) {
    case XMLHttpRequest.HEADERS_RECEIVED: // 2
      console.log(this.getAllResponseHeaders());
      break;
    case XMLHttpRequest.DONE: // 4
      console.log('Se ha completado la descarga');
      break;
    default:
      break;
  }
});
xhr.open('get', 'http://localhost:5500/assets/datos.json');
xhr.send();
```

Posibles estados por los que pasa XMLHttpRequest:

- UNSENT = 0; // initial state
- OPENED = 1; // open called
- HEADERS\_RECEIVED = 2; // response headers received
- LOADING = 3; // response is loading (a data packet is received)
- DONE = 4; // request complete



The screenshot shows a browser's developer console with the following content:

```
cache-control: public, max-age=0
content-length: 153
content-type: application/json; charset=UTF-8
last-modified: Thu, 18 Nov 2021 20:18:07 GMT

Se ha completado la descarga
```

# status y statusText. Ejemplo6.html

- **status** y **statusText** proporcionan información sobre el estado **HTTP**

```
const xhr1 = new XMLHttpRequest();
xhr1.addEventListener('readystatechange', function () {
  if (xhr1.readyState === 4 && xhr1.status === 200) {
    console.log(xhr1.statusText); // OK
  }
});
xhr1.open('get', 'http://localhost:5500/assets/datos.json');
xhr1.send();

const xhr2 = new XMLHttpRequest();
xhr2.addEventListener('load', function () {
  if (xhr2.status === 200) {
    console.log(xhr2.response);
  } else {
    console.log(xhr2.status, xhr2.statusText); // 404 'Not Found'
  }
});
xhr2.open('get', 'http://localhost:5500/assets/datos1.json');
xhr2.send();
```

Algunos de los códigos de respuesta HTTP son:

200 ok  
201 created  
204 no response  
400 bad request  
404 not found

Con xhr1 se utiliza readyState y status para determinar la lectura correcta (es el método antiguo).

Con xhr2 se utiliza el evento 'load' que se produce cuando se ha completado la lectura (método moderno). En este caso la lectura no es correcta (404) porque el fichero no existe.