HTTP

Cordero Hernández,
Marco R.

En sesiones pasadas

(Hace mucho, mucho tiempo…)

- Introducción a JS
- Funciones
  - Función flecha
- Objetos de JS
  - JSON
  - Arreglos
- Programación asíncrona
  - Conceptos
  - Stack, loop de eventos
  - Callbacks
  - Promesas
  - Async/Await

CONTENIDOS

01	02
Conceptos	Solicitudes
03	**04**
Respuestas	AJAX / XHR

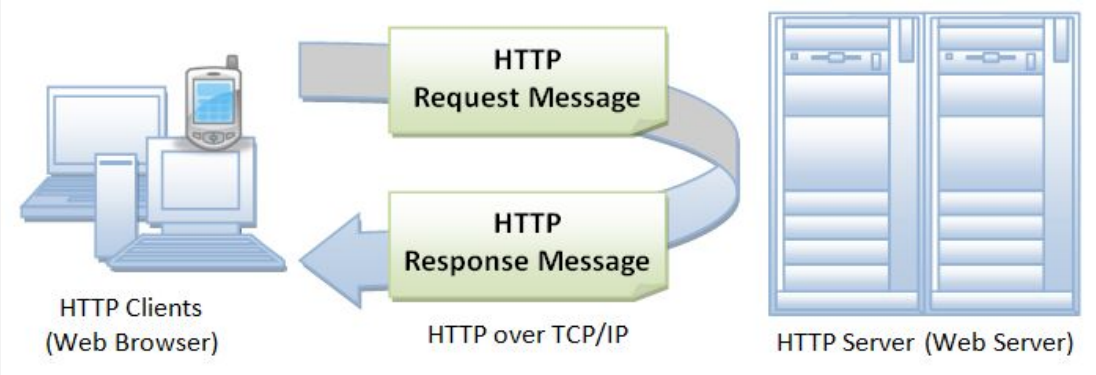
01

CONCEPTOS

HTTP

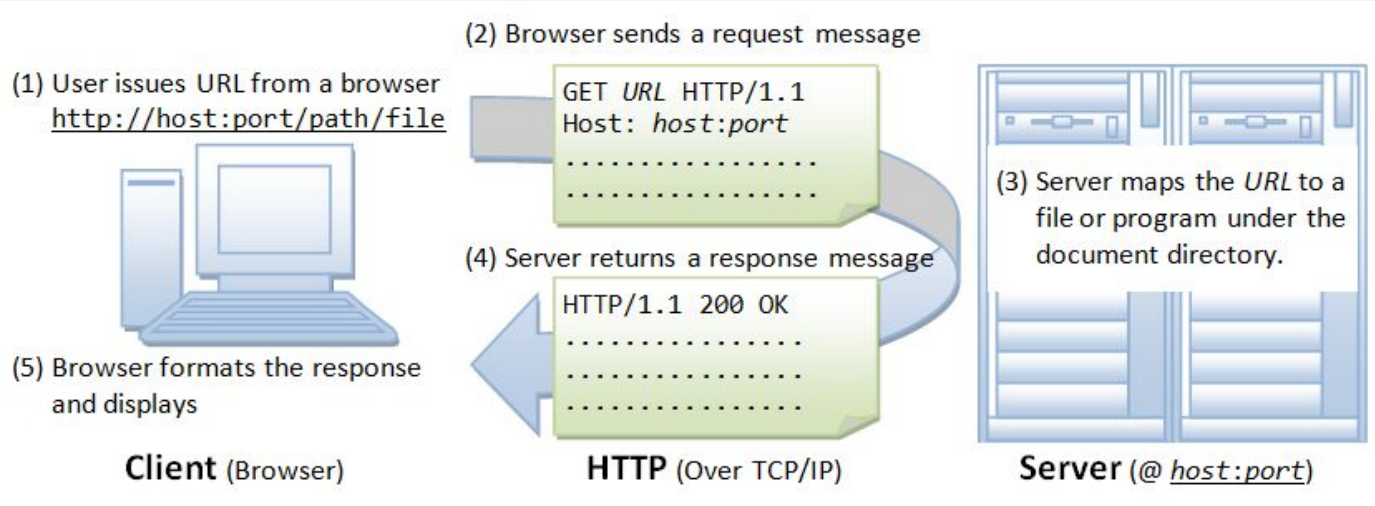
Hypertext Transfer Protocol (Protocolo de Transferencia de Hipetextos)
Se refiere al protocolo de la capa de aplicación para la transmisión de documentos, principalmente **HTML** pero podría ser cualquier tipo de documento.

El cliente envía un mensaje de solicitud (*request*) y el servidor regresa un mensaje de respuesta (*response*).



HTTP

HTTP no mantiene el estado (es *stateless*), es decir, las nuevas solicitudes realizadas al servidor no conocen lo que se ha realizado con las anteriores.



HTTP

Por lo regular, los mensajes de *HTTP* son entedibles por los humanos ***a menos que se desee lo contrario*** (directamente en el diseño).

Los encabezados (*headers*) de *HTTP* se pueden extender y crear nuevas funcionalidades embebidas directamente en las peticiones.

Haciendo alusión a la diapositiva pasada, el hecho de no tener estado en una petición *no significa que no puedan existir sesiones*, para ello, se pueden usar las cookies para crear sesiones y compartir el mismo contexto o estado.

02

SOLICITUDES

Estructura

Una solicitud (*request*) consiste de los siguientes elementos:

- **Método**: usualmente algún verbo como *GET, POST, PUT* o sustantivos como *OPTIONS, HEAD* que definen la ***operación*** que el cliente quiere realizar.
  - Obtener un recurso := GET
  - Enviar los valores de un formulario := POST
  - Actualizar los valores de algo := PUT
  - Eliminar una entidad := DELETE
  - …

Estructura

- **Ruta del recurso:** *hostname:puerto/ruta*
- **Versión:** del protocolo de HTTP
- **Headers** (opcionales): información adicional
- **Body**: Datos de la petición (solo en algunos métodos [como el POST])

URLs

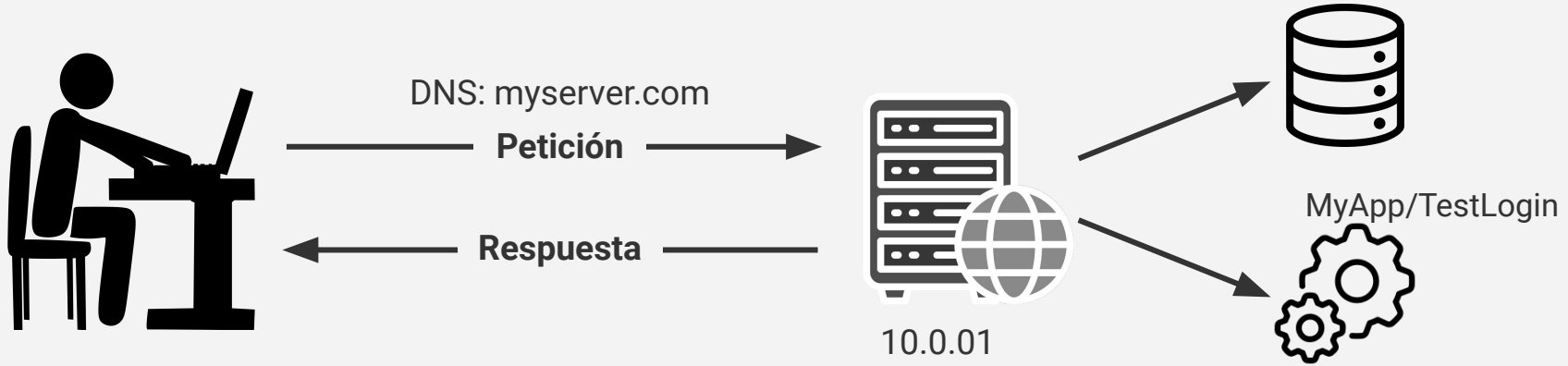
http_url: [http[s]://] host [:port] [/ abs_path [? query]]

- **http -** Hypertext Transfer Protocol [Secure]
- **host -** Servidor al que se le realiza la petición
- **port -** Puerto por el cual se encuentra escuchando el servidor
- **abs_path -** Ruta dentro del servidor donde se encuentra el recurso deseado
- **query -** Modificadores de la petición (parámetros de búsqueda)

URLs

http_url: [http[s]://] host [:port] [/ abs_path [? query]]

http://www.myserver.com/MyApp/TestLogin?user=jorge&password=123456



Métodos

GET - Solicita un recurso específico; Obtiene datos

HEAD - Similar a **GET** pero la respuesta no tendrá contenido (sin body)

POST - Envia un conjunto de valores a un recurso específico; Por lo regular, este método cambia el estado en el servidor, ejemplo: nuevas tablas, nuevos registros, nuevos archivos, etc.

PUT - Reemplaza el recurso anterior por uno nuevo

Métodos

PATCH - Aplica modificaciones parciales a un recurso

DELETE - Borra un recurso específico

CONNECT - Establece un túnel al servidor identificado por el recurso objetivo

OPTIONS - Describe las opciones de comunicación para un objetivo destino

TRACE - Permite monitorear el camino por donde viaja el mensaje hacia el recurso destino

Ejemplo

```
GET /en-US/docs/Web/HTTP/Methods HTTP/2
Host: developer.mozilla.org
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:123.0) Gecko/20100101 Firefox/123.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: _ga_B9CY1C9VBC=GS1.1.1703087146.3.1.1703087355.0.0.0; _ga=GA1.2.1813491906.1684358034;
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

Método

Ruta del recurso

Versión del protocolo

Encabezados

Clasificación de métodos

Métodos seguros (safe): Si la acción es de solo lectura; No se espera que ocurra algún cambio en el servidor (obtención de datos, búsquedas, filtros, etc.)

Idempotentes: Si al ejecutarse solicitudes idénticas varias veces, el efecto en el servidor es el mismo que el efecto de la primera solicitud; Después de ejecutar la primera solicitud ya no hay cambios en el servidor

Clasificación de métodos

MÉTODO	SEGURO	IDEMPOTENTE
GET	SÍ	SÍ
HEAD	SÍ	SÍ
POST	NO	NO
CONNECT	NO	NO
PUT	NO	SI
PATCH	NO	NO
DELETE	NO	SI
OPTIONS	SI	SI
TRACE	SI	SI

Encabezados

Los encabezados (*headers*) permiten pasar información adicional entre el cliente y el servidor.

Se definen como pares **propiedades** : **valor**

Existen muchos tipos de headers (consultar la referencia)

Encabezados

Algunos de los más comúnes son:

- **Content-Type:** Media type del recurso (html, jpg, pdf)
- **Accept:** Listado de content-types separado por comas; indica lo que el cliente es capaz de entender
- **Content-Length:** Tamaño del cuerpo de la petición dado en bytes
- **Content-Language:** Lenguaje de preferencia del agente que realiza la petición (≠ lang de html)
- **Cache-Control:** Directivas para el mecanismo de caching

03

RESPUESTAS

Estructura

Las respuestas tienene los siguientes elementos:

- **Versión** del protocolo HTTP
- **Código de estatus:** Indicando si fue exitosa, o no y por qué
- **Mensaje de estatus:** Descripción breve del código de estatus
- **Encabezados de HTTP:** Similar a la solicitud
- **Cuerpo (body) (*opcional*):** Descripción más detallada del contenido

Ejemplo

```
HTTP/2 200 OK
x-guploader-uploadid: ABPtcPqbuHLN7fSJ0nOcWBEXaBhzCHK-Q3sSn9rmnnvisfTA0cf3E3AwdnRli6jtl9ks2HcD8f0
x-goog-generation: 1710808608641301
x-goog-metageneration: 1
x-goog-stored-content-encoding: identity
x-goog-stored-content-length: 144181
x-goog-meta-goog-reserved-file-mtime: 1710807831
x-goog-hash: crc32c=88Si8g==, md5=NZ7qC1tySzkR3C+yDRFEpQ==
x-goog-storage-class: STANDARD
accept-ranges: none
alt-svc: h3=":443"; ma=2592000,h3-29=":443"; ma=2592000
alt-svc: clear
x-content-type-options: nosniff
strict-transport-security: max-age=63072000
content-security-policy: default-src 'self'; script-src 'report-sample' 'self' www.google-analytics.com/analytics.js
x-frame-options: DENY
```

Versión del protocolo

Código de estatus

Mensaje

Encabezados

Códigos de estatus (o de respuesta)

- **1xx := Informativo -** Solicitud recibida, continuar proceso
- **2xx := Éxito -** La acción fue recibida correctamente, entendida y aceptada
- **3xx := Redirección -** Otra acción debe ser tomada para satisfacer la solicitud
- **4xx := Error en el cliente -** La petición contiene error(es) de sintaxis o no puede ser completada
- **5xx := Error en el servidor -** El servidor falló al atender una petición aparentemente correcta

100 Continue	**206** Partial Content	**400 Bad Request**	**409** Conflict	**417** Expectation Failed	**451** Unavailable For Legal Reasons
101 Switching protocols	**300** Multiple Choices	**401 Unathorized**	**410** Gone	**418** I'm a teapot	**500 Internal Server Error**
200 OK	**301** Moved Permanently	**403 Forbidden**	**411** Length Required	**422** Unprocessable Entity	**501** Not Implemented
201 Created	**302** Found	**404 Not Found**	**412** Precondition Failed	**425** Too Early	**502** Bad Gateway
202 Accepted	**303** See Other	**405** Method Not Allowed	**413** Payload Too Large	**426** Upgrade Required	**503** Service Unavailable
203 Non-Authoritative Information	**304** Not Modified	**406** Not Acceptable	**414** URI Too Long	**428** Precondition Required	**504** Gateway Timeout
204 No Content	**307** Temporary Redirect	**407** Proxy Authentication Required	**415** Unsupported Media Type	**429** Too Many Requests	**505** HTTP Version Not Supported
205 Reset Content	**308** Permanent Redirect	**408** Request Timeout	**416** Range Not Satisfiable	**431** Request Header Fields Too Large	**511** Network Authentication Required

Ejercicio

Dentro de VS Code, instalar la extensión ***REST Client*** (Huachao Mao).

Dentro de un archivo test.http realizar lo siguiente:

- Hacer una petición a jsonplaceholder.typicode.com a *users*
- También a *users/10*
- Hacer una petición a un id de usuario inexistente
- Hacer un *POST* a *users*

04

AJAX / XHR

Definición y uso

AJAX = Asynchronous **J**avaScript **A**nd **X**ML

- Permite la lectura de datos de un ***servidor web***
- Actualizar datos de la **página web** (donde se implementa) *sin recargarla*
- Enviar datos a un ***servidor web*** en *segundo plano*
- No está limitado a **XML**, podría ser texto plano o **JSON**
- **AJAX** es una combinación del uso del objeto **XMLHttpRequest** (o la función ***fetch***) y **JS** para actualizar el **DOM**

```
function guardarEnJSON(datos) {
    // 1. Crear objeto de tipo XMLHttpRequest
    let xhr = new XMLHttpRequest();

    // 2. Configurar método (PUT = Actualizar)
    xhr.open('PUT', urlJSON);

    // 3. Indicar tipo de datos (JSON)
    xhr.setRequestHeader('Content-Type', 'application/json');

    // 4. Enviar solicitud a la red
    xhr.send([JSON.stringify(datos)]);

    // 5. Definir comportamiento de recepción de respuesta
    xhr.onload = () => {
        if (xhr.status != 200) { // Fallo de solicitud
            console.log('Ocurrió un error...');
            console.log(`${xhr.status}: ${xhr.statusText}`);
        } else { // Solicitud exitosa
            console.log(xhr.responseText);
        }
    };
}
```

Ejercicio

Usar el objeto **XMLHttpRequest** para obtener la información de JSONplaceholder (users). Mostrar los siguientes datos de cada usuario:

- Id
- Name
- Email
- Suite, Street, City
- Phone

Solicitar un usuario por id:

- Si existe, se muestran los mismos datos de antes
- Si no existe, mostrar un mensaje indicándolo

Función *fetch*

```
async function leerDatosDeJSON() {
    let response = await fetch(urlJSON);

    if (response.status != 200) return [];

    let arreglo = await response.json();
    console.log(arreglo);
    return arreglo;
}
```

Función *fetch*

```
fetch(url, {
    method: 'POST',
    mode: 'cors',
    body: {"id": 9, "name": 'Ricardo'},
    // headers: {'Content-Type': 'application/json; charset=UTF-8'}
}).then(res => res.json())
.then(json => console.log(json));
```

Soporte de *fetch*

Chrome	Edge *	Safari	Firefox	Opera	IE
4-39			2-33	10-26	
2 40			1 4 34-38	2 27	
2 3 41	12-13	3.1-10	4 39	2 3 28	
42-121	14-121	10.1-17.3	40-122	29-107	6-10
122	122	17.4	123	108	11
123-125		TP	124-126		

https://caniuse.com/fetch

Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	KaiOS Browser
	3.2-10.2									
	10.3-17.3	4-22		12-12.1		2.1-4.4.4				2.5
122	17.4	23	all	80	15.5	122	123	14.9	13.52	3.1