Almacenamiento

Cookies y WebStorage

Cookies

- Son pares “clave=valor” que se almacenan en ficheros en el navegador.
- Las cookies aparecieron como consecuencia de la falta de estado de HTTP.
- Su principal función es guardar variables en el cliente para:
  - Mantener preferencias del usuario.
  - Identificar al usuario.
  - Monitorizar la actividad del usuario (carrito de la compra, etc.).
- Algunas de sus principales características:
  - Tienen caducidad: pueden durar segundos, minutos, días, años…
  - Están asociadas a un único dominio: cada dominio tiene sus cookies y no se mezclan con otros dominios.
  - Pueden ser creadas por el navegador a petición del servidor cuando se descarga una página o por el cliente directamente desde javascript.
  - Una vez creadas y mientras no caduquen se enviarán al servidor en cada petición: todas las cookies del dominio no caducadas se envían al servidor en cada petición request.

Ejemplo cookies en PHP. cookie_crear.php

1. Abrir con VSCode el directorio **EjemplosPHP**
2. Abrir un terminal y ejecutar **.\php\php –S localhost:8080**

3.- Ejecutar: http://localhost:8080/cookie_crear.php

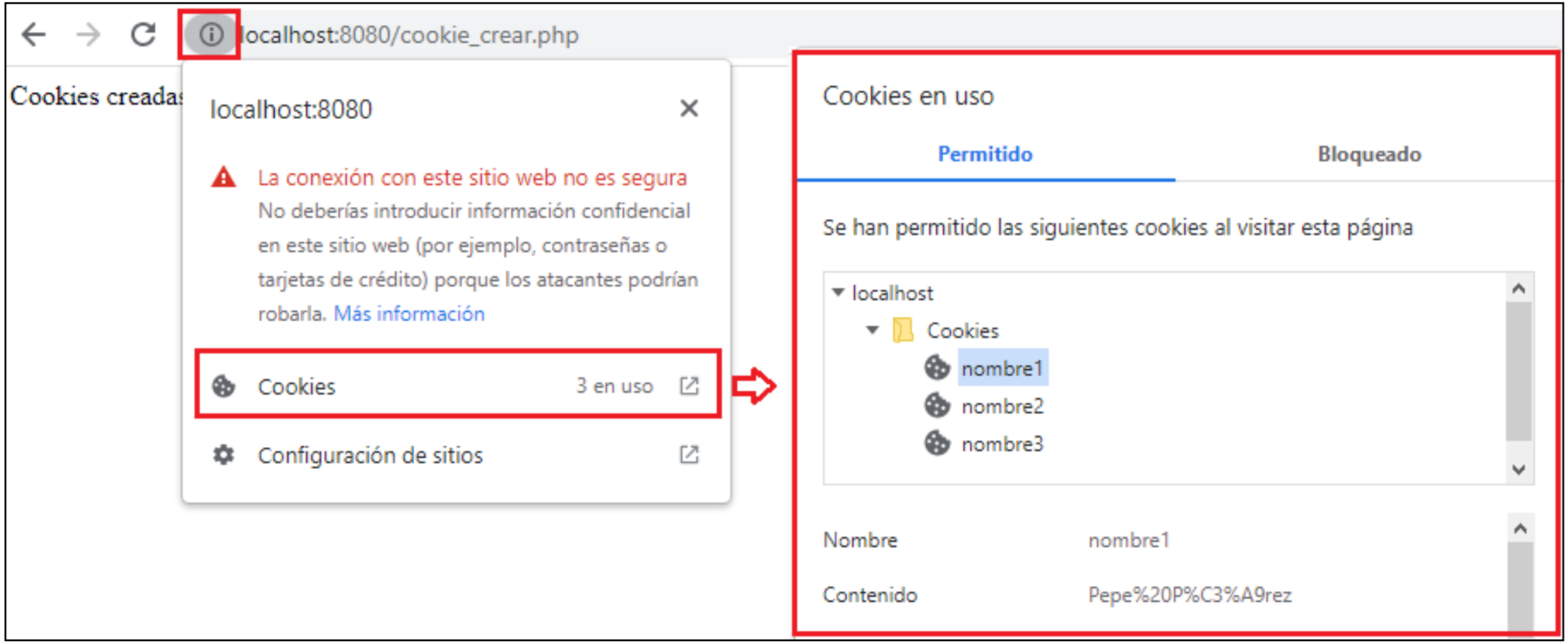
```php
<?
setcookie("nombre1", "Pepe Pérez",strtotime( '+30 days' ));
setcookie("nombre2", "Luis Gil",strtotime( '+2 years' ));
setcookie("nombre3", "Pedro Ruíz", time()+3600); // 60 seg x 60 min x 24 horas = 86400
echo "Cookies creadas";
?>
```

Con ello aparece el mensaje “Cookies creadas”

En Chrome realmente las cookies se almacenan codificadas en el fichero:
C:\Users\<usuario>\AppData\Local\Google\Chrome\User Data\Default\Cookies

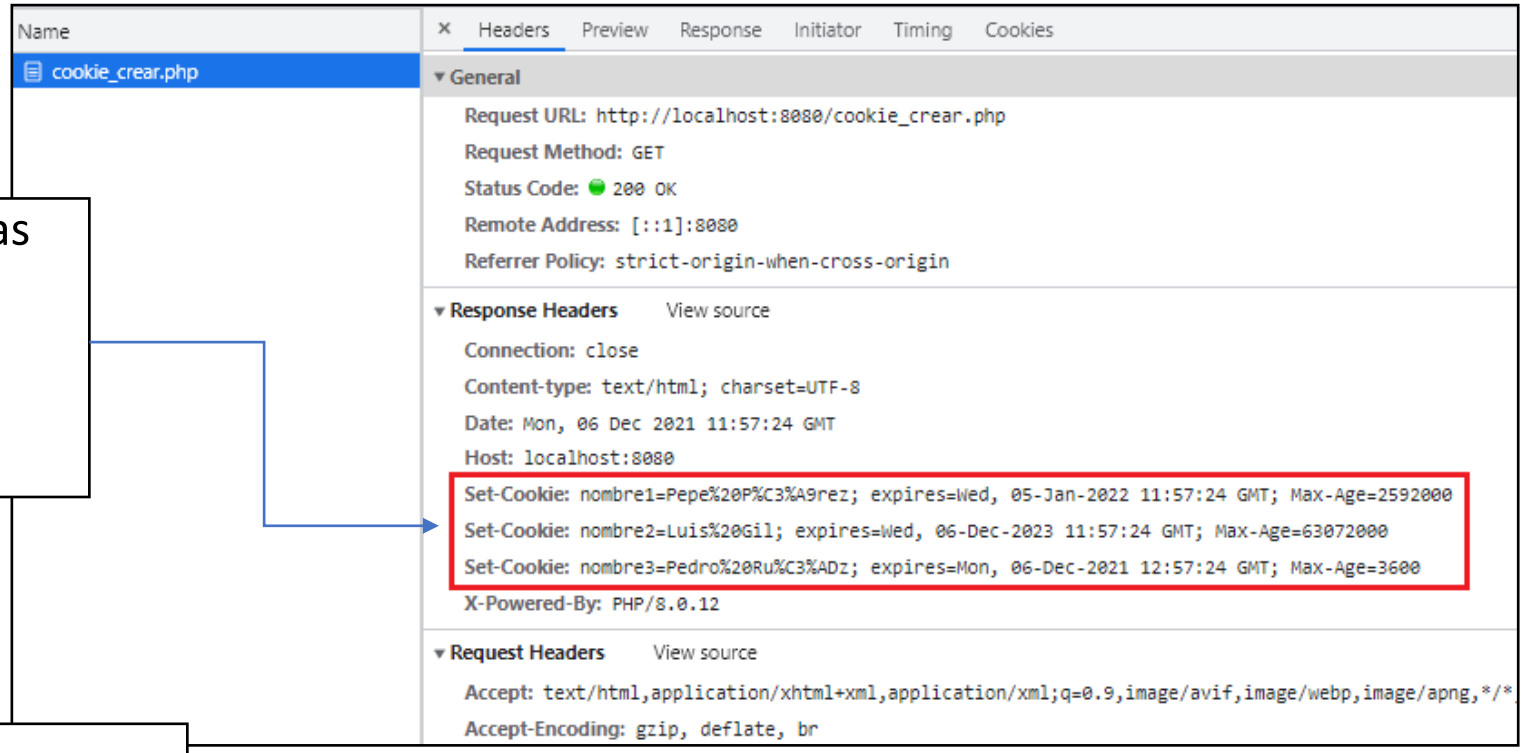
Ver las cookies creadas en Chrome:

- Pulsar en el icono “Información” justo antes de la URL y luego seleccionar “Cookies” para ver su contenido.



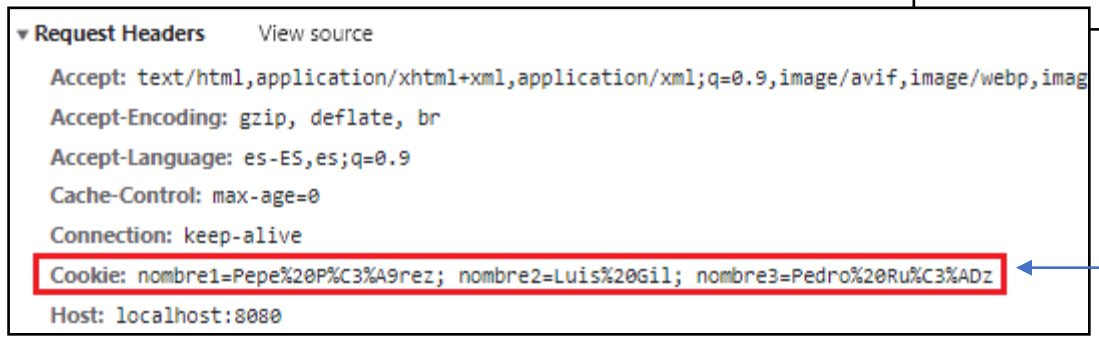
Ver las cookies creadas en Chrome:

- Abrir el inspector y comprobar (en el apartado “Network”) que en las cabeceras de respuesta se indica “Set_Cookie: <cookie>” para crear las nuevas cookies:



El servidor le pide al navegador que cree las cookies.

En la siguiente petición se enviarán al servidor todas las cookies que no hayan caducado.

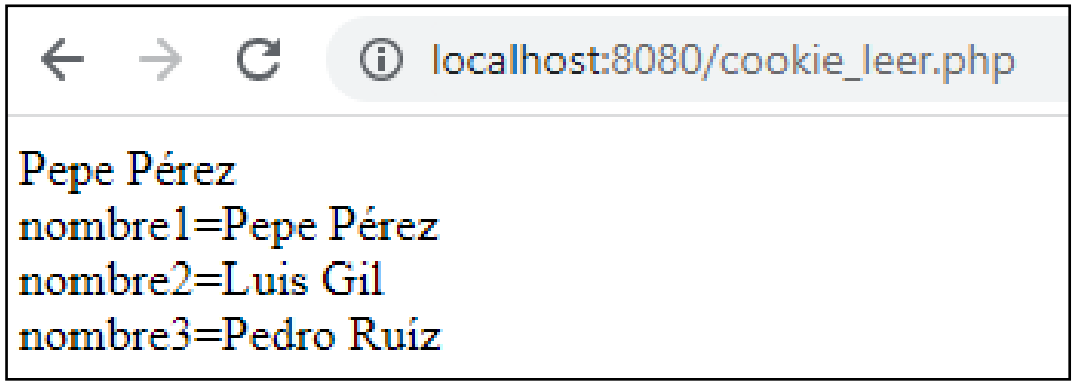


Recargar la página y comprobar que ahora las cabeceras de respuesta contienen las cookies generadas en el cliente.

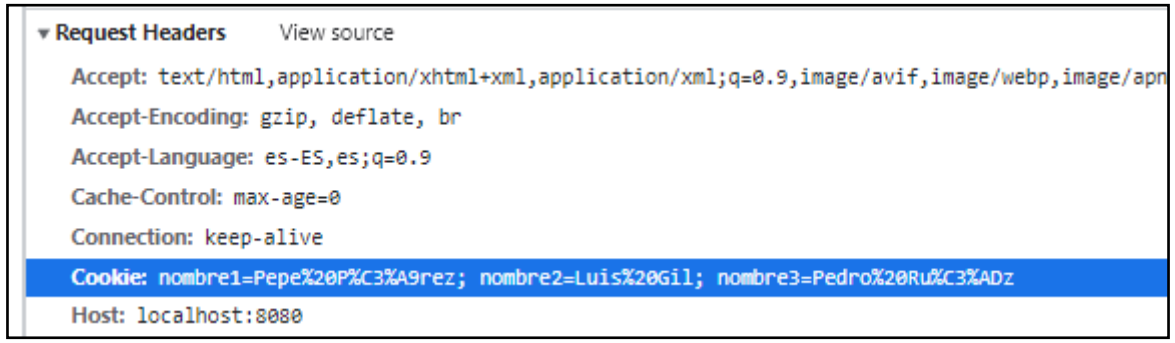
Ejemplo cookies en PHP. cookie_leer.php

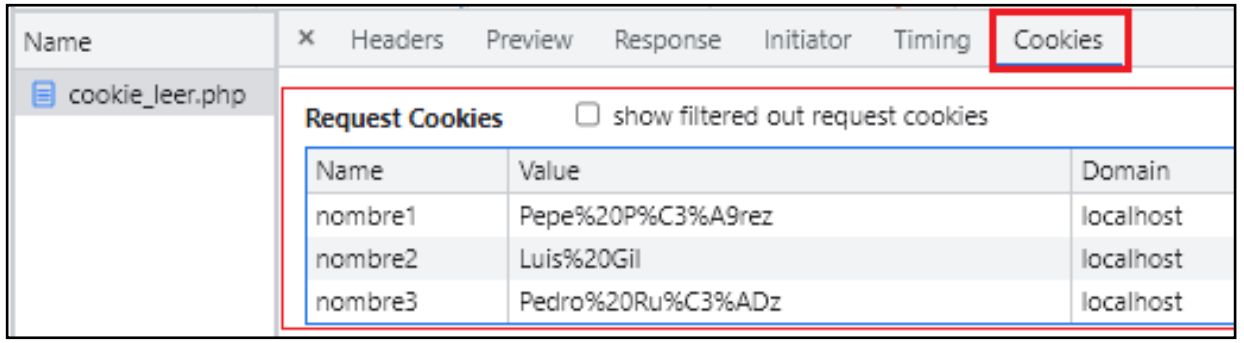
1. Ejecutar: http://localhost:8080/cookie_leer.php y comprobar la respuesta:

```php
<?
echo $_COOKIE["nombre1"];
echo "<br>";
foreach ($_COOKIE as $cookie_name => $cookie_value) {
    echo $cookie_name. "=" . $cookie_value. "<br>";
 }
?>
```



2. Comprobar en el inspector que se reciben las cookies en la cabecera de respuesta y también pulsando en “Cookies” en la sub-barra de “network”:





Leer cookies desde javascript. cookie_leer.html

- En todo momento se puede acceder desde javascript a las cookies disponibles (enviadas por el servidor o creadas localmente) mediante la propiedad “cookie” de document. Devuelve un string con las cookies separadas por “;”. Ejemplo:

```
console.log(document.cookie);
const cookieArray = document.cookie.split(';');
for (let i = 0; i < cookieArray.length; i++) {
  const cookiePar = cookieArray[i].split('=');
  console.log(`${cookiePar[0].trim()}->${decodeURIComponent(cookiePar[1])}`);
}
```

A tener en cuenta:

- PHP codifica los valores de las cookies y, por ello, es necesario utilizar “decodeURIComponent()”.

```
nombre1=Pepe%20P%C3%A9rez; nombre2=Luis%20Gil; nombre3=Pedro%20Ru%C3%ADz
nombre1->Pepe Pérez
nombre2->Luis Gil
nombre3->Pedro Ruíz
```

- Algunos navegadores añaden un espacio después del “;”. Por ello es necesario utilizar trim().

Crear cookies desde javascript. cookie_crear.html

- Crear una cookies es sencillo basta con asignar su valor a “document.cookie”
- El siguiente ejemplo crea dos cookies:

```
document.cookie = 'usuario=Pepe';
document.cookie = 'nombre=' + encodeURIComponent('Pepe Pérez');
```

Es recomendable utilizar encodeURIComponent() para codificar los datos y evitar problemas.

- Cuando se crea una cookie se pueden añadir varios atributos, entre los más interesantes están los que permiten indicar la caducidad. Ejemplo:
  - **expires**: fecha de expiración (una fecha concreta en formato UTC)
  - **max-age**: duración máxima en segundos (ej: 60 * 60 * 24 * 15 para 15 días).

```
// La siguiente cookie caduca en 10 días desde la fecha actual:
const expires = new Date();
expires.setTime(expires.getTime() + 10 * 24 * 60 * 60 * 1000);
document.cookie = 'telefono1=983434343; expires=' + expires.toUTCString();
// La siguiente cookie caduca en 15 días desde la fecha actual:
document.cookie = 'telefono2=983202020; max-age=' + (15 * 24 * 60 * 60);
```

Si no se establece la caducidad de una cookie, se borrará cuando se cierre el navegador.

Funciones para cookies. cookie_funciones.html

- Las siguientes funciones permiten crear y leer cookies:

```
function setCookie (name, value, daysToLive) {
  let cookie = name + '=' + encodeURIComponent(value);
  if (typeof daysToLive === 'number') {
    cookie += '; max-age=' + (daysToLive * 24 * 60 * 60);
    document.cookie = cookie;
  }
}
```

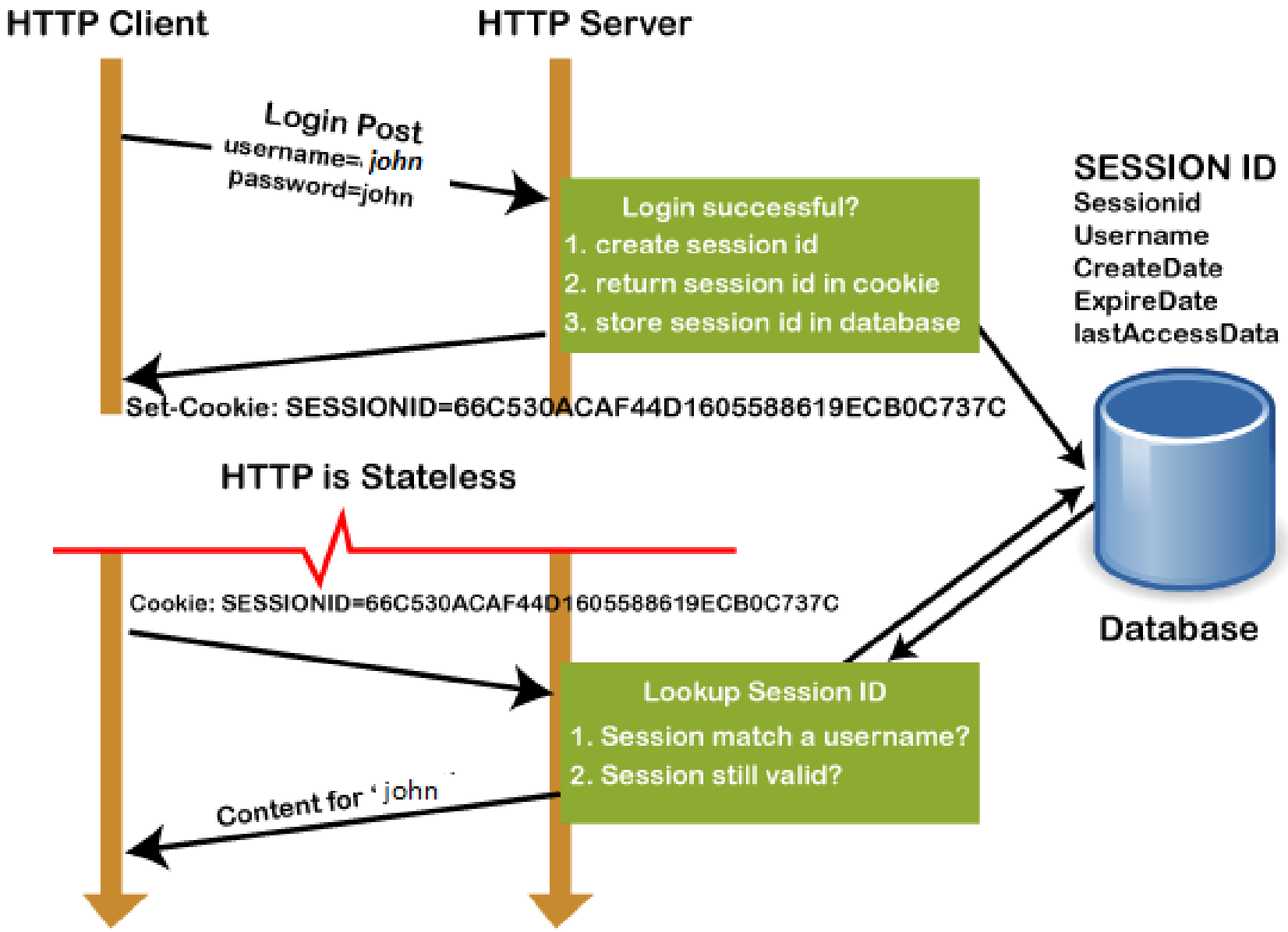
```
function getCookie (name) {
  const cookieArr = document.cookie.split(';');
  for (let i = 0; i < cookieArr.length; i++) {
    const cookiePair = cookieArr[i].split('=');
    if (name === cookiePair[0].trim()) {
      return decodeURIComponent(cookiePair[1]);
    }
  }
  return null;
}
```

- Las cookies se modifican simplemente reasignado su valor.
- El tamaño máximo de una cookies es 4Kb.
- Puede haber hasta 50 cookies para un dominio.
- Las cookies se borran estableciendo la fecha de caducidad a una fecha anterior a la actual.
- En este ejemplo, al establecer “max-age=0”, la cookie se borra cuando se cierra el navegador.

```
setCookie('nombre', 'Pepe', 30); // Crea la cookie
let nombre = getCookie('nombre');
console.log(nombre);
setCookie('nombre', 'Paco', 10); // Modifica
nombre = getCookie('nombre');
console.log(nombre);
setCookie('nombre', '', 0); // Borra
nombre = getCookie('nombre');
console.log(nombre);
```

Sesiones

- Las “sesiones” son objetos del servidor.
- Almacenan información en el servidor para ser utilizada a través de las diferentes páginas.
- La información se puede persistir en ficheros o bases de datos.
- Cuando se inicia sesión en el servidor se crea un **SessionID** que luego será almacenado en una cookie en el cliente.



SessionStorage y LocalStorage

- En HTML 5 se introducen estos dos objetos para aumentar la capacidad de almacenamiento de datos locales en el navegador.
- El límite de datos almacenados es mayor: 5Mb (depende del navegador).
- Los datos se almacenan como pares clave/valor y no tienen fecha de caducidad.
- Los información se almacena en el navegador y "no" se envía al servidor.
- Los datos están vinculados al dominio: diferentes dominios no pueden compartir datos.
- SessionStorage: permite mantener datos dentro de una pestaña del navegador. Cuando la pestaña se cierra los datos se borran. Mantiene los datos cuando se refresca la página.
- LocalStorage: mantiene los datos incluso cuando se cierra el navegador. Los datos no desaparecen hasta que no sean borrados explícitamente.

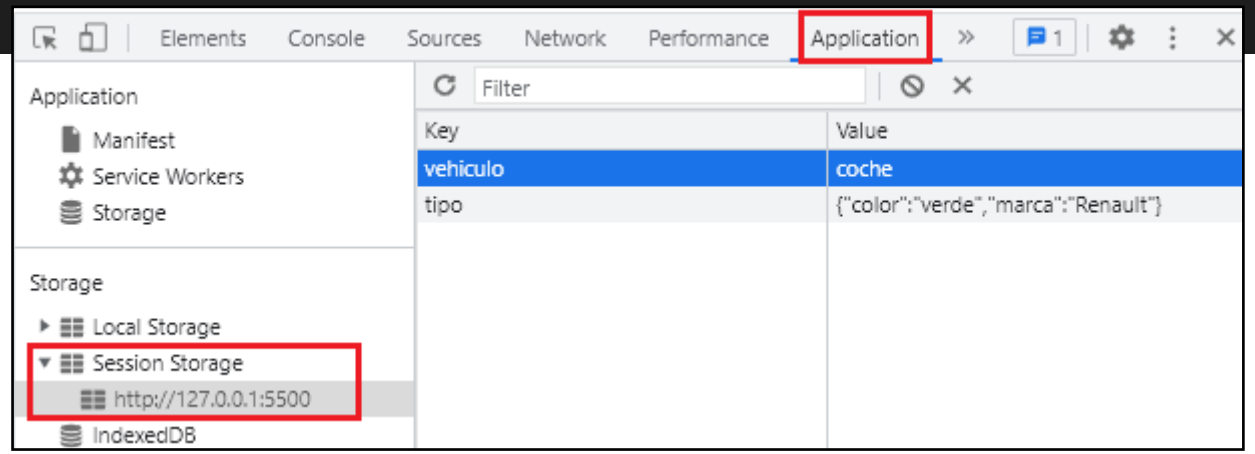
SessionStorage y LocalStorage

- Ambos objetos permiten establecer una clave/valor como en las propiedades de los objetos convencionales. Por ejemplo:
  - Crear o asignar una clave/valor: **localStorage.test=20;**
  - Borrar una clave: **delete.localStorage.test;**
- Además ambos objetos tienen la siguiente API:
  - **setItem(key, value)** : almacena el par key/value.
  - **getItem(key) :** obtiene un valor a partir de su clave.
  - **removeItem(key)** : borra una clave.
  - **clear() :** borra todo.
  - **key(index) :** obtiene una clave a partir de su posición (0,1,2…)
  - **length :** devuelve el número de elementos.
- El evento “storage” se dispara cuando se modifica sessionStorage o localStorage y devuelve un objeto con la clave, el antiguo valor, el nuevo valor, etc…

ver: https://developer.mozilla.org/en-US/docs/Web/API/StorageEvent

Ejemplo. sessionStorage.html

```
sessionStorage.setItem('vehiculo', 'coche');
const oTipo = { color: 'verde', marca: 'Renault' };
sessionStorage.setItem('tipo', JSON.stringify(oTipo));
sessionStorage.setItem('edad', 23);
console.log(sessionStorage.getItem('vehiculo'));
console.log(sessionStorage.getItem('tipo'));
console.log(sessionStorage.getItem('edad'));
sessionStorage.removeItem('edad');
console.log(sessionStorage.length);
for (let i = 0; i < sessionStorage.length; i++) {
  const indiceSesionStorage = sessionStorage.key(i);
  console.log(sessionStorage.getItem(indiceSesionStorage));
}
```



- La información se mantiene hasta que se cierra la pestaña actual del navegador.

```
coche
{"color":"verde","marca":"Renault"}
23
3
true
coche
{"color":"verde","marca":"Renault"}
```

Es posible ver las variables almacenadas en el apartado “Application” “Session Storage” en el inspector.

Ejercicio1

- Se desea crear dos páginas web:
- 1.- GraficoFormulario.html: permite introducir tres datos (nombre, valor y color) mediante los controles apropiados de un formulario. Cuando el usuario pulsa en “Añadir valor” se crea un objeto con la información y se guarda como una nueva variable en formato JSON en localStorage.
- 2.- GraficoCanvas.html: dibuja en un canvas de 800x800 pixeles los datos que hay en el localStorage como un gráfico de barras con el formato indicado. El gráfico se actualizará cuando se carga la página o cuando cambia un valor de localStorage: