**Cookies**

Una cookie HTTP es una pequeña parte de datos que el servidor envía al navegador web del usuario, que puede guardarlo y mandarlo de nuevo junto con la siguiente petición al mismo servidor. Típicamente, se utiliza para saber si dos peticiones vienen del mismo navegador , permitiendo mantener la sesión del usuario "loggeada", por ejemplo.

Las cookies se utilizan principalmente para tres propósitos:

* Gestión de sesiones.
* Personalización.
* Seguimiento.

Para el manejo de cookies a través de Express.js es necesario utilizar el middleware cookie-parser, que puede ser instalado como cualquier otro módulo de node.js con

npm install cookie-parser

## Usando Cookie-Parser

Para utilizar cookie-parser es necesario incluirlo en la aplicación como se hace con cualquier middleware de Express.js

var express = require('express');

var cookieParser = require('cookie-parser');

var app = express();

app.use(cookieParser());

cookie-parser ubica en el objeto req, bajo req.cookies con un objeto que identifica las cookies por un nombre, un string. Para definir una cookie nueva se define una nueva ruta en la aplicación Express.js de la siguiente manera:

app.get('/cookie',function(req, res){

res.cookie(cookie\_name , 'cookie\_value').send('Cookie is set');

});

Para revisar si una cookie ha sido asignada correctamente, se puede ir a la consola del navegador que se esté usando e introducir el comando document.cookie.

El navegador manda de vuelta esa cookie al servidor cada vez que hace una petición al sitio. Para conseguir una cookie que el navegador está enviando al servidor mediante el objeto request podemos utilizar req.cookies. De esta manera el código siguiente registrará en la consola las cookies cuando el cliente envíe una:

app.get('/', function(req, res) {

console.log("Cookies : ", req.cookies);

});

Fijar el tiempo de expiración de una cookie se hace fijando una fecha mediante el campo expire de la siguiente manera:

res.cookie(name , 'value', {expire : new Date() + 9999});

Como con el tiempo de expiración, podemos fijar más opciones para la cookie de igual manera, afectando otros atributos. expire, por ejemplo, aguanta el tiempo de expiración en milisegundos. También se puede usar el atributo maxAge.

Para eliminar una cookie se utiliza el método clearCookie, que eliminará una cookie referenciada por su nombre. Por ejemplo, para eliminar la cookie pepito hacemos:

clearCookie("pepito");

**Localstorage y sessionStorage**

localStorage y sessionStorage son propiedades que acceden al objeto Storage y tienen la función de almacenar datos de manera local, la diferencia entre éstas dos es que localStorage almacena la información de forma indefinida o hasta que se decida limpiar los datos del navegador y sessionStorage almacena información mientras la pestaña donde se esté utilizando siga abierta, una vez cerrada, la información se elimina.

## Validar objeto Storage en el navegador

Aunque gran parte de los navegadores hoy en día son compatibles con el objeto Storage, no está de más hacer una pequeña validación para rectificar que realmente podemos utilizar dicho objeto, para ello utilizaremos el siguiente código:

if (typeof(Storage) !== 'undefined') {

// Código cuando Storage es compatible

} else {

// Código cuando Storage NO es compatible

}

## Guardar datos en Storage

Existen dos formas de guardar datos en Storage, que son las siguientes:

**localStorage**

// Opción 1 -> localStorage.setItem(name, content)

// Opción 2 ->localStorage.name = content

// name = nombre del elemento

// content = Contenido del elemento

localStorage.setItem('Nombre', 'Miguel Antonio')

localStorage.Apellido = 'Márquez Montoya'

**sessionStorage**

// Opción 1 -> sessionStorage.setItem(name, content)

// Opción 2 ->sessionStorage.name = content

// name = nombre del elemento

// content = Contenido del elemento

sessionStorage.setItem('Nombre', 'Miguel Antonio')

sessionStorage.Apellido = 'Márquez Montoya'

## Recuperar datos de Storage

Al igual que para agregar información, para recuperarla tenemos dos maneras de hacerlo

**localStorage**

// Opción 1 -> localStorage.getItem(name, content)

// Opción 2 -> localStorage.name

// Obtenemos los datos y los almacenamos en variables

let firstName = localStorage.getItem('Nombre'), lastName = localStorage.Apellido

console.log(`Hola, mi nombre es ${firstName} ${lastName}`)

// Imprime: Hola, mi nombre es Miguel Antonio Márquez Montoya

**sessionStorage**

// Opción 1 -> sessionStorage.getItem(name, content)

// Opción 2 -> sessionStorage.name

// Obtenemos los datos y los almacenamos en variables

let firstName = sessionStorage.getItem('Nombre'), lastName = sessionStorage.Apellido

console.log(`Hola, mi nombre es ${firstName} ${lastName}`)

// Imprime: Hola, mi nombre es Miguel Antonio Márquez Montoya

## Eliminar datos de Storage

Para eliminar un elemento dentro de Storage haremos lo siguiente:

**localStorage**

// localStorage.removeItem(name)

localStorage.removeItem(Apellido)

**sessionStorage**

// sessionStorage.removeItem(name)

sessionStorage.removeItem(Apellido)

## Limpiar todo el Storage

Ya para finalizar veremos la forma para eliminar todos los datos del Storage y dejarlo completamente limpio

**localStorage**

localStorage.clear()

**sessionStorage**

sessionStorage.clear()