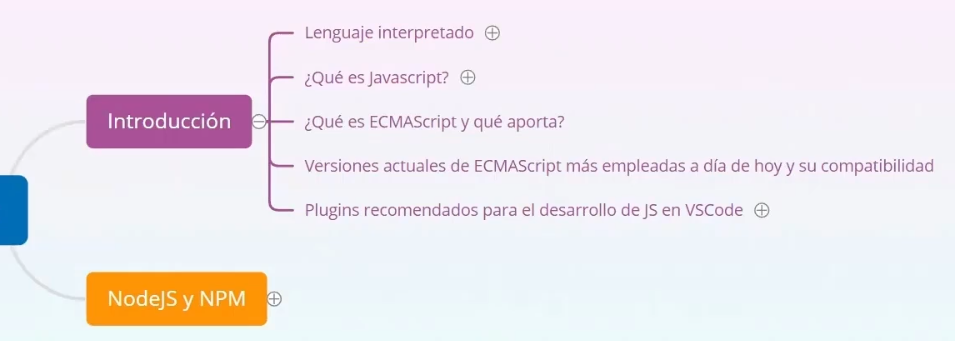
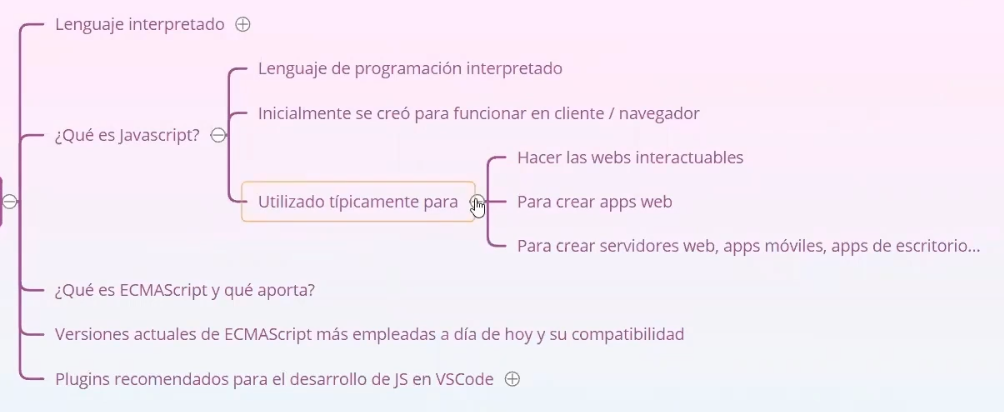
**JavaScript Basico**

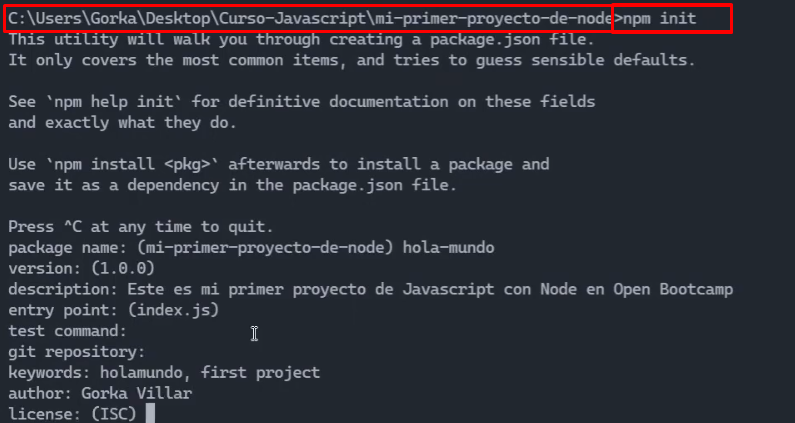






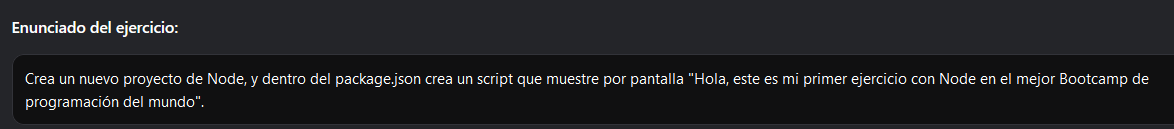


NodeJS

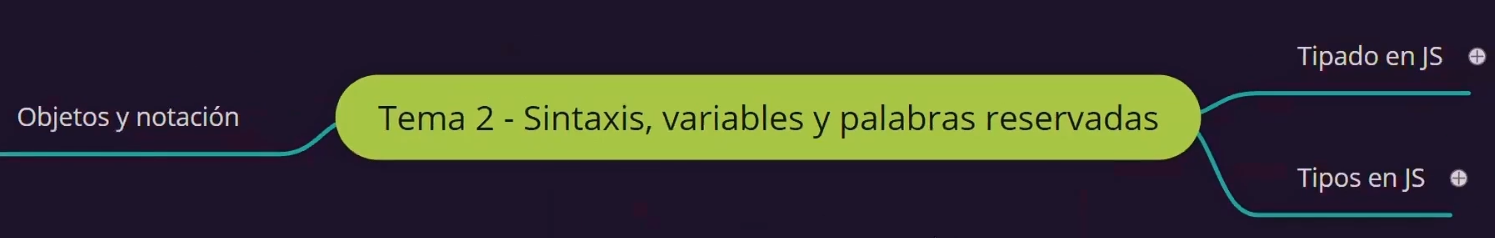


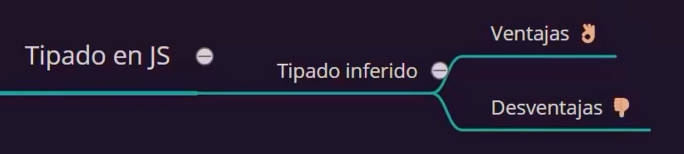
<https://campus.open-bootcamp.com/cursos/15/leccion/1745>

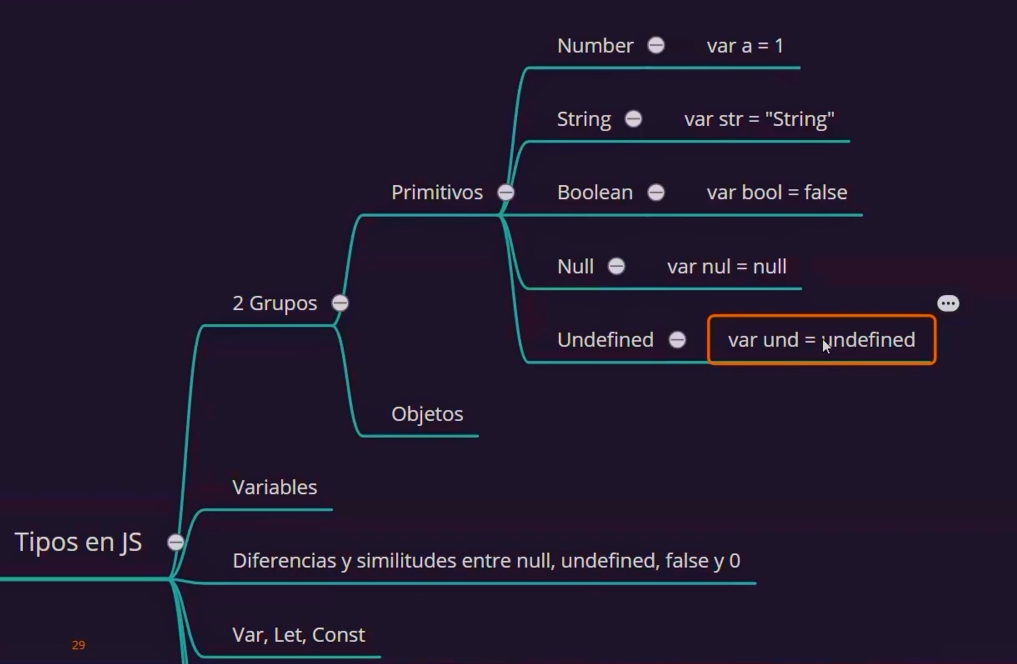
Ejercicio:



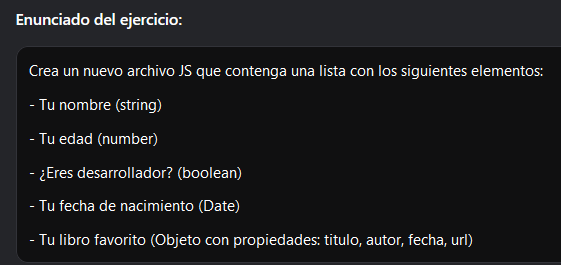
2.1



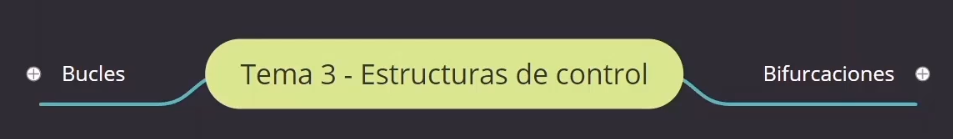


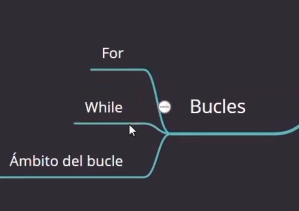


Ejercicio:

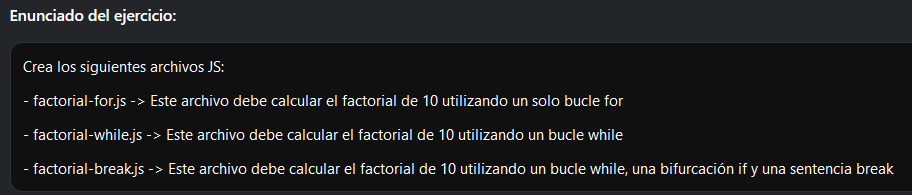


3.1





Ejercicio:

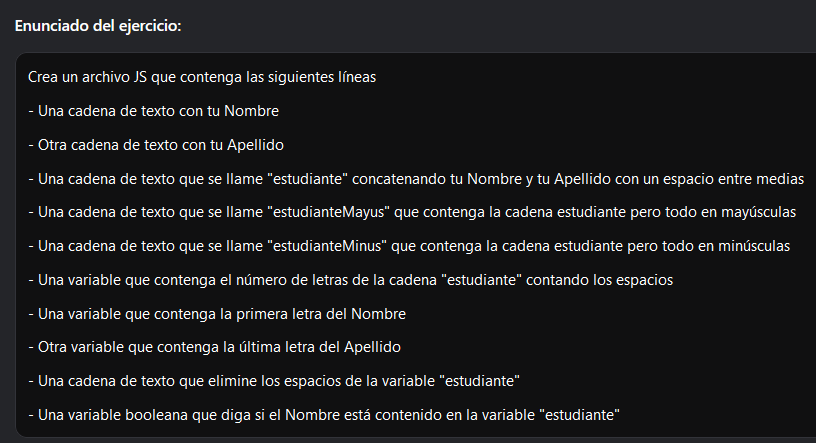


4.1

Tipos de declaraciones de Strings: <https://campus.open-bootcamp.com/cursos/15/leccion/259>

Manipulacion de cadenas de texto: <https://campus.open-bootcamp.com/cursos/15/leccion/1755>

Ejercicio:

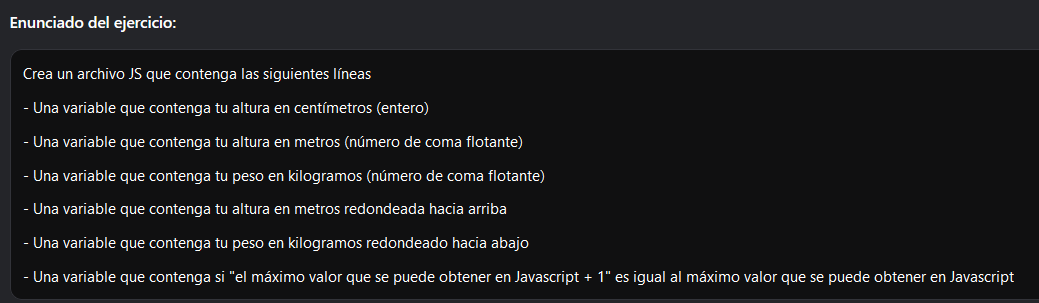


5.1

Operaciones y redondeo: <https://campus.open-bootcamp.com/cursos/15/leccion/1775>

Hexadecimales y similares: <https://campus.open-bootcamp.com/cursos/15/leccion/1776>

Ejercicio:



6.1

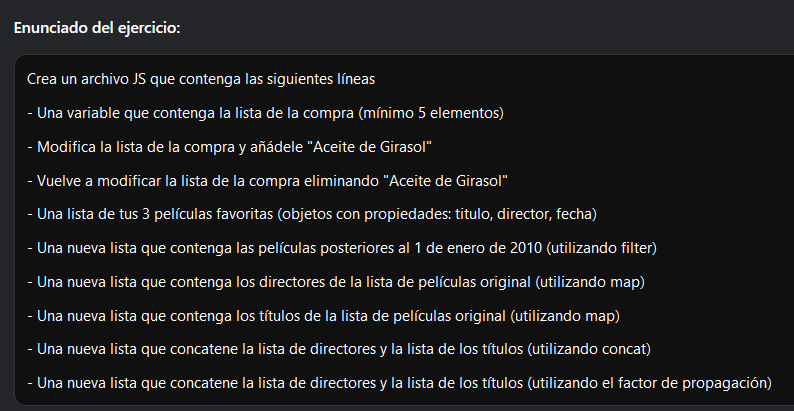
Concatenacion y obtencion de fragmentos de listas: <https://campus.open-bootcamp.com/cursos/15/leccion/1778>

Listas a traves de Listas: <https://campus.open-bootcamp.com/cursos/15/leccion/1780> -> Filter

Ordenar y Comparar listas: <https://campus.open-bootcamp.com/cursos/15/leccion/1781>

Identificar si existe un valor en un Array: <https://campus.open-bootcamp.com/cursos/15/leccion/1782>

Ejercicio:

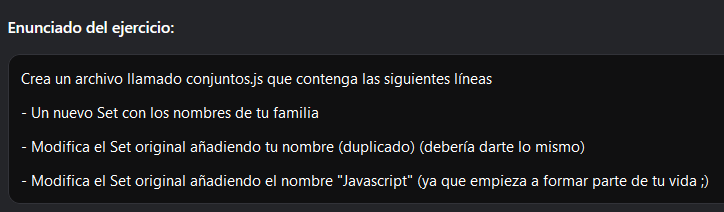


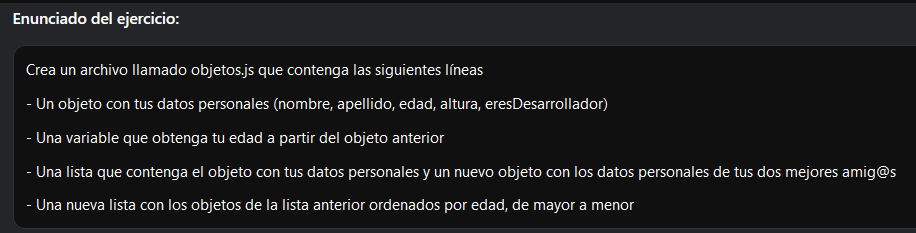
7.1

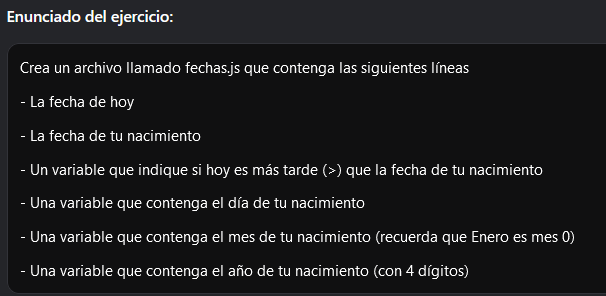
Set: <https://campus.open-bootcamp.com/cursos/15/leccion/1783>

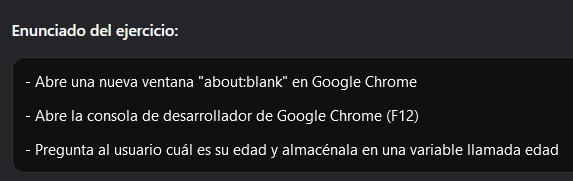
Objetos: <https://campus.open-bootcamp.com/cursos/15/leccion/1784>

Ejercicios:









8.1

Funciones tipo flecha y anonimas: <https://campus.open-bootcamp.com/cursos/15/leccion/1788>

const array = [1, 5, 6, 2, 7, 12, 8, 92]

const array2 = array.map((*valor*) => valor \* 2)

console.log(array2)

// const dobleDelValor = valor => {

//     return valor \* 2

// }

const dobleDelValor = *valor* => valor \* 2

console.log(doble(6))

console.log(dobleDelValor(6))

const array3 = array.map(dobleDelValor)

console.log(array3)

function doble(*valor*) {

    return valor \* 2

}

Async y promesas: <https://campus.open-bootcamp.com/cursos/15/leccion/1790>

function miAsinc() {

    // Hace una llamada a una base de datos externa

    // Puede darnos algún error

}

const miPromesa = **new** *Promise*((*resolve*, *reject*) => {

    const i = Math.floor(Math.random() \* 2)

    // Si está todo correcto

    if (i !== 0) {

        resolve()

    } else {

        reject()

    }

})

miPromesa

    .then(() => console.log("Se ha ejecutado de forma correcta"))

    .catch(() => console.log("ERROR"))

.finally(() => console.log("Yo me ejecuto siempre"))

// async await

Funciones Generadoras: <https://campus.open-bootcamp.com/cursos/15/leccion/1791>

function\* generaId() {

    let id = 0;

    while (true) {

        id++;

        if (id === 10) {

            return id;

        }

        yield id; // Esperando hasta que se vuelva a llamar

    }

}

const gen = generaId();

console.log(gen.next().value); // Retorna: 1

console.log(gen.next().value); // Retorna: 2

console.log(gen.next().value); // Retorna: 3

console.log(gen.next().value); // Retorna: 4

console.log(gen.next().value); // Retorna: 5

console.log(gen.next().value); // Retorna: 6

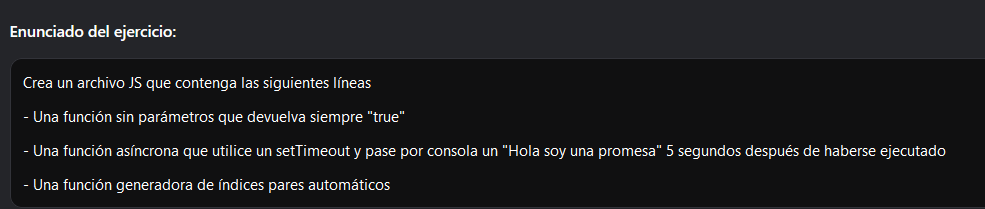
console.log(gen.next().value); // Retorna: 7

console.log(gen.next().value); // Retorna: 8

console.log(gen.next().value); // Retorna: 9

console.log(gen.next().value); // Retorna: 10

Ejercicio:



9.1

const miFuncion = *val* => {

    if (typeof *val* === "number") {

        return 2 \* *val*

    }

    throw **new** *Error*("El valor debe ser de tipo número")

}

// const elDoble = miFuncion("a1a")

const numero = "8";

try {

    // Código que puede fallar

    console.log("Está ejecutándose de manera correcta")

    const doble = miFuncion(numero)

    console.log(doble)

} catch (e) {

    // En caso de fallar, quiero que ejecutes

    console.error(`El valor de e es: ${e}`)

    console.error("ERROR!")

} finally {

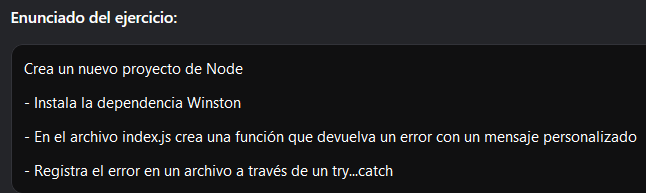
    console.log("Esto se va a ejecutar tanto si se produce algún error, como si no existe ninguno")

}

// InternalError, SyntaxError, TypeError, RangeError y ReferenceError

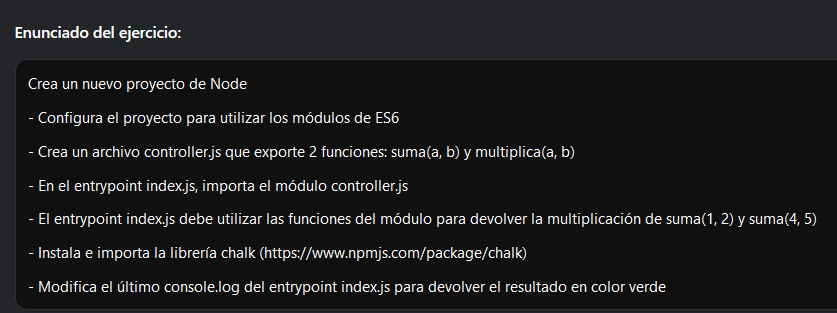
// https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global\_Objects/Error

Ejercicio:



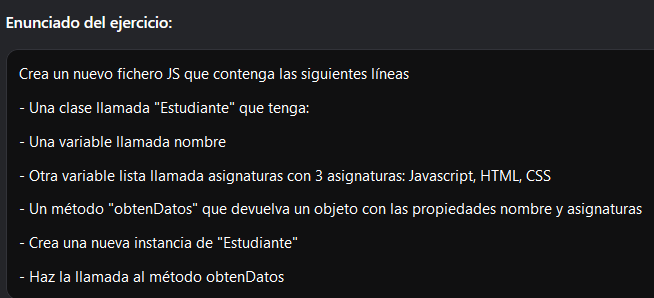
10.1

Ejercicio:



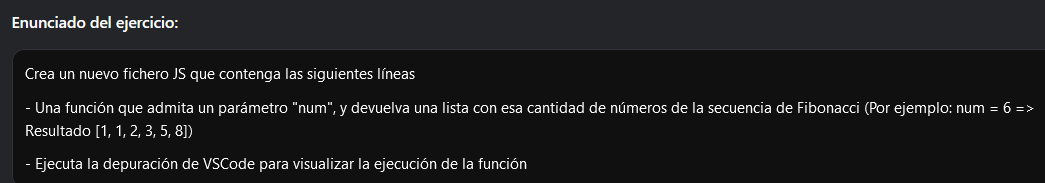
11.1

Ejercicio:



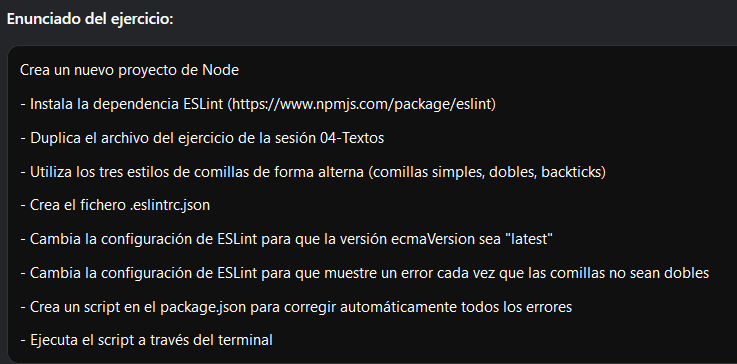
12.1

Ejercicio:



13.1

Ejercicio:



14.1

const hTexto = document.getElementById("h-texto")

console.log(hTexto)

hTexto.addEventListener("cambioTexto", *evento* => {

    console.log(evento.detail)

    hTexto.innerText = evento.detail.texto

    hTexto.style.color = evento.detail.color

})

function cambiarTexto(*nuevoTexto*, *color*) {

    const evento = **new** CustomEvent("cambioTexto", {

        detail: {

            texto: nuevoTexto,

            color

        }

    })

    hTexto.dispatchEvent(evento)

}

Importando Jquery:

<script *src*="https://code.jquery.com/jquery-3.6.0.min.js"></script>

Jquery.js

// $(selector).acción()

// $("h1").hide()

// $(document).ready(() => {

$(() => {

    // Selectores:

    // id="el-1" => "#el-1"

    // class="el-1" => ".el-1"

// $("#el-1").hide()

    $(".hide-btn").click(() => {

        // $("h1").hide()

        $("h1").fadeOut()

    })

    $(".show-btn").click(() => {

        // $("h1").show()

        $("h1").fadeIn()

})

    $("li").dblclick(() => {

        $("h1").css({ color: "red" })

})

    $(".new-element").click(() => {

        // $("ul").append("<li>New Element</li>")

        $("ul").prepend("<li>New Element</li>")

})

    $("li").mouseenter((*elem*) => {

*elem*.target.style.color = "blue"

})

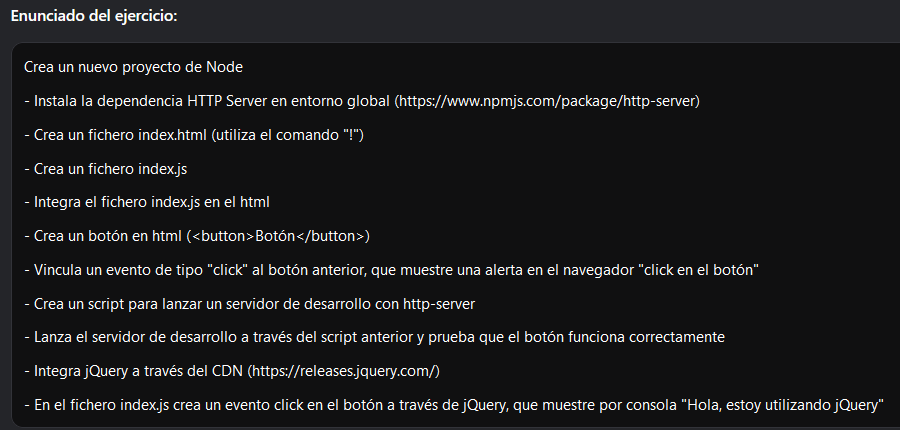
    $("li").mouseleave(*elem* => {

*elem*.target.style.color = "black"

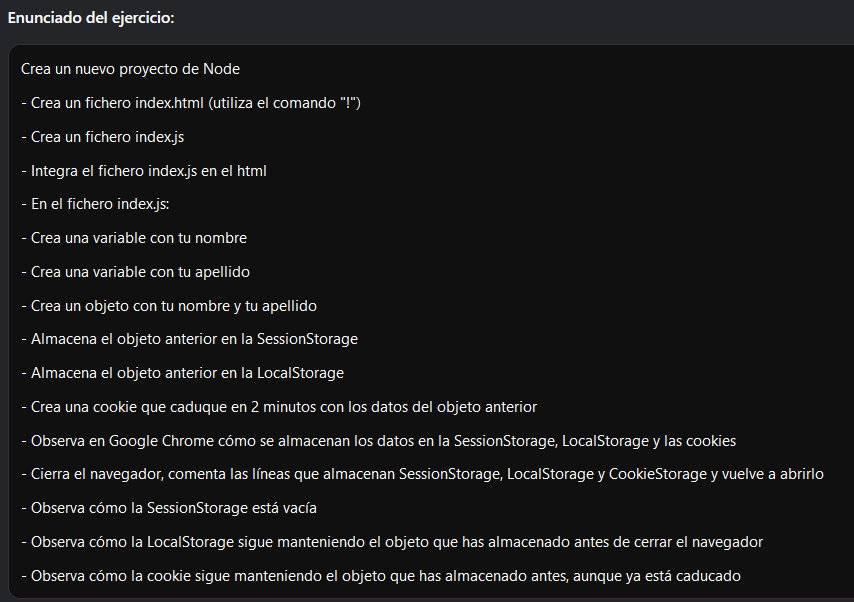
    })

})

Ejercicio:

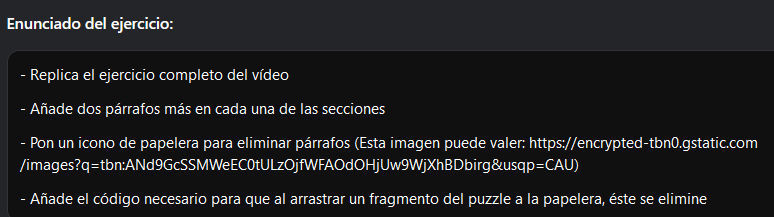


15.1



16.1

<https://campus.open-bootcamp.com/cursos/15/leccion/1868>



17.1

Mapas en HTML: <https://campus.open-bootcamp.com/cursos/15/leccion/1871>

Geolocalizacion: <https://campus.open-bootcamp.com/cursos/15/leccion/1870>

