/\* 'use strict'

console.log('//////////////////////////////////////');

//Objects

const persona = {

    nome: '',

    cognome: '',

    eta: '',

    famiglia: ''

}

console.log(persona);

persona.nome = 'nicola'

persona.cognome = 'pirotta'

persona.eta = 24

persona.famiglia = ['tommaso', 'francesco', ['ugo', 'maria']]

console.log(persona);

console.log(persona.nome);

console.log(persona['nome']);//can compute the field name

persona.numero = 3664306706

console.log(persona.numero);

console.log(Object.keys(persona));

console.log(Object.values(persona));

console.log('///////////////////////////////////////');

//array destructuring

const giorni = ['lunedi', 'martedi', 'mercoledi', 'giovedi', 'venerdi', 'sabato', 'domenica']

const [l, , m] = giorni//primo in l, terzo in m

console.log(l, m);

const [primo] = persona.famiglia//solo primo elemento

console.log(primo);

//REST (sinistra): al posto di VARIABILI divise da virgole

const [...family] = persona.famiglia

console.log(family);

const [tommy, ...restOfFamily] = persona.famiglia

console.log(tommy, restOfFamily);

//SPREAD (destra): al posto di VALORI divisi da virgole

const familia = [...persona.famiglia]

console.log(familia);

for (const giorno of giorni.entries()) {

    console.log(giorno);

}

const [...giorniConIndice] = giorni.entries()

console.log(giorniConIndice);

console.log('//////////////////////////////////////////');

//utilizzo di .entries(), .keys() e .values() con arrays e oggetti

const casa = ['tetto', 'cantina', 'comignolo']

const giardino = {

    casa: 'mia',

    città: 'vaprio',

    dove: function () {

        console.log(this.città);

    }

}

console.log(...casa.entries()); //(2) [0, 'tetto'] (2) [1, 'cantina'] (2) [2, 'comignolo']

console.log(...casa.keys()); //0 1 2

console.log(...casa.values()); //tetto cantina comignolo

console.log(...Object.entries(giardino)) //(2) ['casa', 'mia'] (2) ['città', 'vaprio']

console.log(...Object.keys(giardino));//casa città

console.log(...Object.values(giardino));//mia vaprio

giardino.dove() //vaprio

console.log('//////////////////////////////////////////');

//map, set, array

const newMap = new Map()

newMap.set(1, 'stringa')

newMap.set(2, 23234235)

newMap.set('terzo', [1, 2, 3])//key can be anything

console.log(newMap);

newMap.has('terzo') //true

// newMap.delete('terzo')

// console.log(newMap);

const arrayFromMap = [...newMap.entries()]//from MAP to ARRAY

console.log(arrayFromMap);//3 array contenenti le coppie key-value

const arrayOfValuesFromMap = [...newMap.values()]

console.log(arrayOfValuesFromMap);//1 array contenente le 3 values

const setFromMap = new Set(newMap.values()) //from MAP to SET

console.log(setFromMap);

console.log('//////////////////////////////////////////');

//string's methods

const s = 'arturito'

console.log(s.length);//8

console.log(s.indexOf('t'));//2

console.log(s.lastIndexOf('t'));//6

console.log(s.slice(1));//rturito

console.log(s.slice(1, 4));//rtu

console.log(s.trim());//removes white space from begin and end

console.log(s.replace('t', 's'));//arsurito (solo la prima)

console.log(s.replaceAll('t', 's'));//arsuriso (tutte)

console.log(s.startsWith('a'));//true

console.log(s.endsWith('a'));//false

console.log(s.charAt(0));//a

console.log(s.split('t'));//['ar','uri','o']

const sArray = ['ar', 'uri', 'o']

console.log(sArray.join('-')) //ar-uri-o

console.log(s.padStart(10, '!'));//!!arturito

console.log(s.padEnd(10, '!'));//arturito!!

console.log(s.repeat(2));//arturitoarturito

console.log('//////////////////////////////////////////');

//.call() e .bind()

const io = {

    nome: 'nicola',

    cognome: 'pirotta',

    getNome: function () {

        console.log(this.nome)

    }

}

//se assegno una funzione che usa 'this' a una costante, this li sarà undefined

const getNomeMaDiChi = io.getNome

//getNomeMaDiChi()  //da errore, this è undefined

getNomeMaDiChi.call(io)//uso .call(object) se voglio invocare immediatamente la funzione, passo come parametro il valore di 'this'

const getNomeMaDiChiBOUNDED = getNomeMaDiChi.bind(io)//uso .bind() per creare una nuova funzione che sia linkata al valore di 'this' voluto

getNomeMaDiChiBOUNDED()

//  MOLTO UTILE CON GLI EVENT LISTENERS!

console.log('//////////////////////////////////////////');

//array methods

let settimana = ['lunedi', 'martedi', 'mercoledi', 'giovedi', 'venerdi', 'sabato', 'domenica']

//looping

settimana.forEach((giorno, i) => console.log(giorno, i)) //cant break

for (const [i, giorno] of settimana.entries()) { console.log(giorno, i) }

//MUTATE ORIGINAL ARRAY

settimana.push('sabaterdì')//aggiunge alla fine

settimana.unshift('lunedomenica')//aggiunge all'inizio

settimana.pop()//rimuove dalla fine

settimana.shift()//rimuove dall'inizio

settimana.splice(2, 2, 'Mercoledì\_1', 'giovedì\_2')//togli 2 elementi a partire da index=2, possibilità di specificare elementi con cui sostituire quelli tolti

settimana.reverse()//inverte ordine elementi

settimana.sort((a, b) => a.length - b.length)//ordina array in base a funzione passata, di default ordine alfabetico

settimana.fill('riempio', 1, 3)//riempio con l'oggetto passato gli elementi compresi tra i due index specificati (begin compreso, end escluso)

//RETURN A NEW ARRAY

settimana = ['lunedi', 'martedi', 'mercoledi', 'giovedi', 'venerdi', 'sabato', 'domenica']//ripristino array iniziale

console.log(settimana.map((a) => a + 'M'));//elementi modificati dalla callback

console.log(settimana.filter((a) => a[0] === 'm'));//solo elementi filtrati

console.log(settimana.slice(0, 3));//restituisce sezione (start comp., end esc.)

console.log(settimana.concat(['gennario', 'febbrario']));//concatena nuovo array

console.log(settimana.flat());//flats the original, specify the depth

//console.log(settimana.flatMap());//equivalent to .map().flat(1)

//ALTRI

console.log(settimana.indexOf('martedi'));//indice (strict comparison)

console.log(settimana.find((a) => a[0] === 'g'));//cerca in base a condizione

console.log(settimana.join('-'));//stringa formata unendo elementi e divisore

console.log(settimana.reduce((a, b) => a + b));//accumulates to a single value

console.log(settimana.includes('venerdi'));//true se c'è l'elemento

console.log(settimana.some((a) => typeof a == 'string'));//true se almeno uno rispetta condizione

console.log(settimana.every((a) => typeof a == 'string'));//true se tutti rispettano condizione

console.log('//////////////////////////////////////////');

//Number

console.log(23 === 23.0);//true, numbers are always stored like floating

console.log(Number.parseInt('30cm'));//30, can ignore ending characters

console.log(Number.parseFloat('3.5px'));//3.5

console.log(Number.isFinite(30.5));//true if finite number

//Math.

console.log(Math.sqrt(4));//2

console.log(Math.max(4, 6, 4, 23546, 6255, 2, 45, 6, 66));//23546

console.log(Math.floor(4.8));//primo intero minore o uguale

let x = 45.1234

console.log(x.toFixed(2));//specify number of decimals (returns a string!!!)

//random number between min and max

const randomNumber = (min, max) => Math.floor(Math.random() \* (max - min) + 1) + min

console.log('//////////////////////////////////////////');

//DATES

console.log(new Date());

console.log(new Date(1997, 6, 29, 20, 22, 55));//(mesi da 0 a 11)

const data = new Date()

console.log(data.getDate());//giorno del mese

console.log(data.getDay());//giorno della settimana (0-6)

console.log(data.getMonth());//mese (0-11)

console.log(data.getFullYear());//2021

console.log(data.getTime());//tempo in millisecondi

console.log(data.toISOString());//formato ISO

//timers

setTimeout(() => {

    console.log('sono passati 3 secondi');

}, 3000);//delay in millisecondi

//!!! TIMERS ARE ASYNCHRONOUS !!! code execution does not stop!

console.log('//////////////////////////////////////////');

//DOM MANIPULATION

//Select

document.querySelector('.class oppure #id oppure nome tag')

document.querySelectorAll('.class oppure #id oppure nome tag')

document.getElementById('id')

document.getElementsByTagName('tag')

document.getElementsByClassName('class')

//create

const elemento = document.createElement('div')

elemento.style.width = '50px'

elemento.style.height = '20px'

elemento.style.backgroundColor = 'black'

//insert

const body = document.querySelector('body')

body.append(elemento)//insert after lastChild

const elemento1 = elemento.cloneNode()

elemento1.style.backgroundColor = 'red'

body.prepend(elemento1)//insert before firstChild

const elemento2 = elemento.cloneNode()

elemento2.style.backgroundColor = 'yellow'

body.before(elemento2)//insert before, as sibling

const elemento3 = elemento.cloneNode()

elemento3.style.backgroundColor = 'green'

body.after(elemento3)//insert after, as sibling

body.insertAdjacentHTML("afterbegin", 'html da inserire')

elemento.insertAdjacentHTML('afterbegin', '<div>sono dentro a elemento</div>')

//delete

//elemento.remove()

//some tips

//current scroll posion

console.log(window.pageXOffset, window.pageYOffset);

//viewport dimensions

console.log(document.documentElement.clientHeight, document.documentElement.clientWidth)

//scroll to element

elemento3.scrollIntoView()

//EVENTS

elemento.addEventListener('click', function (e) { })

elemento.removeEventListener('click', function (e) { })

//e.target = dove ho triggerato    e.currentTarget = dove si è propagato

//e.stopPropagation() per fermare la propagazione dell'eventListener

//DOM traversing

elemento.querySelector('div')//cerca tra i childNodes

console.log(elemento.childNodes)//lista childNodes

console.log(elemento.parentNode)//parentNode

console.log(elemento.closest('div'))//closest parent with specified selector

console.log(elemento.previousSibling)//sibling precedente

console.log(elemento.nextSibling)//sibling seguente

console.log('//////////////////////////////////////////');

//OOP

//contructor functions: create object from a function

const Person1 = function (name, year) {

    this.firstName = name

    this.birthday = year

}

const nicola = new Person1('Nicola', 1997)

//add methods (never create them inside the constructor!!!)

Person1.prototype.calcAge = function (currentYear) {

    console.log(currentYear - this.birthday)

}

//add properties

Person1.prototype.species = 'homo sapiens'

//inheritance

//far ereditare l'intero prototype

const Student = function (name, year, course) {

    this.name = name

    this.birthday = year

    this.course = course

}

Student.prototype = Object.create(Person1.prototype)

const marco = new Student('ciao', 1989, 'awrwe')

console.log(marco)

marco.calcAge(2022)

//far ereditare solo campi/costruttori/metodi indicati

const Student1 = function (name, year, course) {

    Person1.call(this, name, year)//eredito costruttore

    this.course = course //campo aggiuntivo

}

const luca = new Student1('Luca', 2002, 'matematica')

console.log(luca)

//ES6 Classes

class Person2 {

    constructor(name, year) {

        this.name = name

        this.year = year

    }

    calcAge(currentYear) {//viene aggiunto nel prototype, non nell'object --> buone performance

        console.log(currentYear - this.year)

    }

}

//class inheritance

class Student2 extends Person2 {

    constructor(name, year, course) {

        super(name, year)//eredito costruttore

        this.course = course

    }

}

const abele = new Student2('abele', 2012, 'storia')

console.log(abele)

abele.calcAge(2023)

//Object.create(prototype)

const tommaso = Object.create(Person2.prototype)

tommaso.name = 'tommaso'

tommaso.year = 2000

console.log(tommaso)

tommaso.calcAge(2020)

//inheritance

const PersonProto = {

    calcAge() {

        console.log(2037 - this.birthYear);

    },

    init(firstName, birthYear) {

        this.firstName = firstName

        this.birthYear = birthYear

    }

}

const StudentProto = Object.create(PersonProto)

StudentProto.init = function (name, year, course) {

    PersonProto.init.call(this, name, year)

    this.course = course

}

const jay = Object.create(StudentProto)

jay.init('jay', 2001, 'physics')

console.log(jay)

console.log('//////////////////////////////////////');

//ASYNCHRONOUS JAVASCRIPT

//Ajax call with the fetch API

let receivedData = []

fetch("https://api.publicapis.org/random")

    .then(response => response.json())

    .then(data => {

        console.log(data.entries)

        receivedData = data.entries[0]

        console.log(`

           API: ${receivedData.API}

           Category: ${receivedData.Category}

           Description: ${receivedData.Description}

        `)

        return fetch("https://api.publicapis.org/random")

    }

    ).catch(err => console.log(err.message))

//another one

const url = 'https://restcountries.com/v2/'

const getCountryData = function (country) {

    fetch(url + 'name/' + country)//fetches something

        .then(response => response.json())//tranform the response to json

        .then(data => console.log(data[0]))//use the data

}

//getCountryData('italy')

//oppure, con async await

const getCountryData2 = async function (country) {

    const res = await (fetch(url + 'name/' + country))

    const data = await res.json()

    console.log(data[0])

}

getCountryData2('germany')

//PROMISES

//build a Promise

const promise = new Promise(function (resolve, reject) {

    //the asynchronous behavior we want to handle

    setTimeout(() => {

        if (Math.random() > 0.5) resolve('you won')

        else reject('you lost')

    }, 2000);

}).catch((rejection) => { console.log(rejection) })

promise

    .then(resolve => console.log(resolve))

    .catch(reject => console.log(reject))

const wait = function (seconds) {

    return new Promise(function (resolve) {

        setTimeout(resolve, seconds \* 1000)

    })

}

wait(2)

    .then(() => {

        console.log('1 second passed');

        return wait(1)

    })

    .then(() => {

        console.log('2 second passed');

        return wait(1)

    })

    .then(() => {

        console.log('3 second passed');

        return wait(1)

    })

 \*/

consents = {}

userConsents = [

  { id: 1234, accepted: true },

  { id: 1235, accepted: false },

  { id: 1236, accepted: true },

]

userConsents.forEach((u) => (consents[u.id] = u.accepted))

console.log(consents)