function hello(name){

return 'hola' + name

}

console.log(hello('Maria'))

console.log(hello('Alejandra'))

console.log(hello('Maleja'))

sumar

function add (x, y) {

return x + y

}

console.log(add(10, 20))

console.log(add(4, 5))

console.log(add(10, 7))

NaN no tiene los dos datos (10 + undefined)

function add (x, y) {

return x + y

}

console.log(add(10, ))

function add (x, y) {

if (y === undefined) {

y = 0

}

return x + y

}

console.log(add(20, 30 ))

puedo asiganra valores por defecto

function add (x = 23.3, y = 0) {

return x + y

}

console.log(add(20, 10))

objetos

const user = {

name: 'maria',

lastName: 'vallejo',

age: 30,

address: {

country:'colombia',

city: 'la ceja', PROPIETIES son clave valor

street: 'main street 18',

},

friends: ['Billy', 'Ann'],

active: true,

sendMail () { or

sendMail: function () {

return 'sending email...'

}

}

console.log(user)

acceso desde el punto .name

console.log(user.name)

console.log(user.friends)

console.log(user.active)

ejecutando la funcion con los ()

console.log(user.sendMail())

**Shorthand property names**

const name = 'laptop'

const price = 3000

const newProduct = {

name: *name*,

price: price

}

console.log(newProduct)

**se puede resumir en**

const newProduct = {

name,

price

}

**Manipulacion del DOM**

**Mostrar por consola**

const title = document.createElement('h1')

console.log(title)

**desde document puedo inspeccionarlo en elementos**

const title = document.createElement('h1')

document.body.append(title)

**inserter textos**

const title = document.createElement('h1')

title.innerText = 'hola mundo desde js'

document.body.append(title)

crear botones

const button = document.createElement

('button')

button.innerText = 'click'

document.body.append(button)

añadir escucha al boton para que haga: EVENT HANDLERS por consola

const button = document.createElement

('button')

button.innerText = 'click'

button.addEventListener('click', function () {

console.log('hola mundo')

actualizar un valor o ejecutar una logica. Apartir del navegador alerta

button.addEventListener('click', function () {

title.innerText = 'Texto actualizado con Js'

alert('se realizo un click')

})

Objetos como parametros desde la ejecucuion de la funcion

const user = {

name: 'Maria',

age: 30,

}

function printInfo(user){

return '<h1> Hola ' + user.name +'</h1>'

}

console.log(printInfo(user))

document.body.innerHTML = printInfo (user)

Destructuring de Objetos con [] en vez de .

function printInfo(user){

return '<h1> Hola ' + user['name'] +'</h1>'

o los Destructuring

function printInfo({name}){

return '<h1> Hola ' + name +'</h1>'

extaer de un objecto

const user = {

name: 'Maria',

age: 30,

}

function printInfo(user) {

const {name, age} = user;

console.log(name, age)

return '<h1> Hola ' + name +'</h1>'

}

console.log(printInfo(user))

document.body.innerHTML = printInfo (user)

funciones anonimas

console.log(function star (){

return 'starting...'

})

Crear y rjrcutar al mismo tirmpo ()

console.log(function star (){

return 'starting...'

}())

Igual si quito el nombre

console.log(function (){

return 'starting...'

}())

Sin nombrar.

const button = document.createElement

('button')

button.innerText = 'click me'

button.addEventListener('click', function () {

alert('clicked')

})

document.body.append(button)

nombrabnola antes

const button = document.createElement

('button')

button.innerText = 'click me'

function handleclick() {

alert('clicked')

}

button.addEventListener('click', handleclick)

document.body.append(button)

arrow function con flechas

function add (x, y) {

return x + y

}

Es igual a

const add = (x, y) => {

return x + y

}

const button = document.createElement

('button')

button.innerText = 'click me'

button.addEventListener('click', function () {

alert('clicked')

})

document.body.append(button)

es igual a

const button = document.createElement

('button')

button.innerText = 'click me'

button.addEventListener('click', () => {

alert('clicked')

})

document.body.append(button)

inline arrow functions

const showText = () =>{

return 'hola mundo'

}

console.log(showText())

const button = document.createElement

('button')

button.innerText = 'click me'

button.addEventListener('click', () => {

alert('clicked')

})

document.body.append(button)

Tambien puedo escribir teniendo el return implicito

const showText = () => 'hola mundo'

console.log(showText())

return

const showText = () => 'hola mundo'

const showNumber = () => 22;

const showBolean = () => true;

const showArray = () => [1, 2, 3]

const showObject = () => ({name: 'Maria

'})console.log(showText())

console.log(showNumber())

console.log(showBolean())

console.log(showArray())

console.log(showObject())

return

const button = document.createElement

('button')

button.innerText = 'click me'

const isAuthorized = true

button.addEventListener('click', () => {

if (isAuthorized) {

alert('autorizado')

} else {

alert('no autorizado')

}

})

document.body.append(button)

ahorrar else por return

const button = document.createElement

('button')

button.innerText = 'click me'

const isAuthorized = false

button.addEventListener('click', () => {

if (isAuthorized) {

return alert('autorizado')

} alert('no autorizado');

});

document.body.append(button)

string literals

onst button = document.createElement

('button')

button.innerText = 'click me'

button.*style* = 'background:red; color:white;'

const background = 'red'

const color = 'white'

const result = background + color

console.log(result)

const button = document.createElement

('button')

button.innerText = 'click me'

button.*style* = 'background:red; color:white;'

mejor para concatenar

const result = `estos son estilos: ${background} ${color}`

console.log(result)

= = = true es igual a el ? operador ternario si es true

const background = 'yellow'

const color = 'black'

const isAuthorized = true;

const button = document.createElement

('button')

button.innerText = 'click me'

button.*style* = 'background:red; color:white;'

button.*style* = `background: ${isAuthorized ? background : 'red'}; color: ${color}`

button.addEventListener('click', () => {

if (isAuthorized) {

return alert('autorizado');

}

alert('no autorizado');

});

document.body.append(button)

metodos de arrays (map, filter, foreach, concat)

const names = ['mari', 'billy', 'ann']

for (let i = 0; i < names.length; i++) {

const element = names[i];

console.log(element)

}

For Each

names.forEach (function (name){

console.log(name)

})

Map

const newNames = names.map(function (name) {

return `hola ${name}`

})

console.log(newNames)

console.log(names)

find PARA ENCONTRAR

const nameFound = names.find(function (name) {

if (name === 'billy') {

return name

}

})

console.log(nameFound)

filter si es igual

const newNames = names.filter(function (name) {

if (name === 'billy') {

return name

}

})

console.log(newNames)

filter si es diferente

const newNames = names.filter(function (name) {

if (name !== 'billy') {

return name

}

})

console.log(newNames)

concat fusionar dos arreglos

const names = ['maria', 'billy', 'ann']

const newNames = ['adele', 'amy', 'cory']

console.log(names.concat(newNames)

javascript arrays / DEVELOPER.MOZILLA.org

FAZTweb arrays

spread operator mas sencillo

const names = ['maria', 'billy', 'ann']

const newNames = ['adele', 'amy', 'cory']

console.log([...names,...newNames])

con objetos

const user = {

name: "maria",

lastName: "vallejo",

}

const address = {

country:'colombia',

city: 'la ceja',

street: 'main street 18'

}

const userInfo = {

...user,

...address

}

console.log(user)

console.log(address)

console.log(userInfo)

Ecmascript modules

En index js

mport {add, multiply} from './add.js';

console.log(add(10, 20))

console.log(multiply(10, 20))

en add js

export function add (x, y) {

return x + y

}

export function multiply (x, y) {

return x \* y

}

En index. Html

<script type= "module" src="./index.js"></script>

Tambien se pueden exporter string, boolean array

En index.js

import {add, multiply,title, active, points} from './add.js';

console.log(active, points, title)

en add.js

export const title = 'calculator'

export const active = true

export const points = [20, 30, 40]

export default

en add.js

export default title // o cualquiera que desee exporter

con funciones debo poner x=0 y=0

en index.js

import addModule from './add.js';

console.log(addModule)

optional Chaining

datos que traemos de un backend con ?.

name:'Maria',

address:{

city:'Medellin'

}

}

console.log(person.address.city)

const person = {

name:'Maria',

address:{

city:'Medellin'

}

}

console.log(person.location?.city)

Async/await sincronia

Fetch

Biblioteca de js

let data = fetch ('https://jsonplaceholder.typicode.com/posts')

console.log(data)

devuelve una promesa asincrona mientras pide datos hace otras cosas con la funcion

.then ()

fetch('https://jsonplaceholder.typicode.com/posts')

.then(function (){

console.log('finalizo la carga')

})

console.log('linea 2')

para verlo

fetch('https://jsonplaceholder.typicode.com/posts')

.then(function (response){

return response.json()

}) .then(function (data) {

console.log(data)

})

console.log('linea 2')

const ul =document.createElement("ul")

*//fetch('https://jsonplaceholder.typicode.com/posts')*

*//.then(function (response){*

*// return response.json()*

*//}) .then(function (data) {*

*//console.log(data)*

*//data.forEach(function (post) {*

*//const li = document.createElement("li");*

*//li.innerText = post.title;*

*//ul.append(li)*

*//});*

*//document.body.append(ul)*

*//})*

async function loadData(){

const response = await fetch('https://jsonplaceholder.typicode.com/posts')

const data = await response.json()

console.log(data)

data.forEach(function (post) {

const li = document.createElement("li");

li.innerText = post.title;

ul.append(li)

});

document.body.append(ul);

}

loadData()

console.log('linea 2')