//Ekrana girerken soru soran pencere

<div id="welcome"></div>

<script>

let name = prompt("What\'s your name?");

document.getElementById('welcome').innerHTML = `welcome ${name}`;

</script>

 ADDING JAVASCRIPT EXTERNALLY

// Before the closing body tag create and link app.js

<script src="../app.js"></script>

// add our previous JS Code into that file

let name = prompt("What\'s your name?");

document.getElementById('welcome').innerHTML = `welcome ${name}`;

// Select any element

document.querySelector('h1');

// Send an array as a table

console.table([1, 2, 3, 4]);

// Print an error

console.error("Ooops!, something went wrong!");

// Clean the console

console.clear();

// Print a warning

console.warn("A warning");

// Check how much time a code takes to execute

console.time(“zaman”);

console.warn("A warning");

console.warn("A warning");

console.warn("A warning");

console.warn("A warning");

console.warn("A warning");

console.warn("A warning");

console.warn("A warning");

console.timeEnd(“zaman”);

var ve let farkı:

var ile iki defa = değer verilir ama let ile sadece bir defa. Let \*\*\*, \*\*\* diye virgule iki değer verirsen sorun yok.

Const = değer vermeden hata verir, değiştirilemez

TÜRÜNÜ SORMA. Null object olarak geçer.

let değişken = "yunus"

console.log(değişken)

console.log(typeof değişken)

consta sayı ekleniyor, tek tek değişiyor ama komple değişmiyor

const numbers = [1,2,3,4,5];

numbers[3] = 3;

console.log(numbers);

numbers.push(6);

console.log(numbers);

// But you cannot re assign the whole array

numbers = [1,2,3]; olmuyor

 STRINGS IN JAVASCRIPT

“” İLE BAŞLARSAN SORUN YOK AMA ‘’ İLE YAPARSAN YUNUS’S YAZAMIYORSUN \'s YAPMAN GEREKİR.

let name;

// Quotes.

name = 'Juan Pablo';

name = 'Then i said: hey what's up? ';

name = 'Then i said: hey what\'s up? '; // Backslash

 STRING METHODS

const learning = 'Learning JavaScript is great!';

let output;

// concat. ekliyor

output = learning.concat(" ", " and fun");

// uppercase

output = learning.toUpperCase();

// lowercase

output = learning.toLowerCase();

// indexOf // indexof -1 doesn't exist

output = learning.indexOf('JavaScript');

output = learning.indexOf('PHP');

yoksa -1 verir

let email="yunus@hotmail"

output=email.indexOf("@")

if(output>0){

    console.log("geçerli posta")

}else{console.log("yanlış posta")}

console.log(output)

//indexof birşeyde bir karakter yoksa -1 veriyor, 0'dan küçükse şunu yaz dedik.

// substring() 0’dan başlayıp 10 karakteri seçer

output = learning.substring(0,10);

output = learning.substring(2,10);

// Substring from the end. son 4 karakteri seçer

output = learning.substring( learning.length - 4 );

// slice() substring gibi, ekside sondan başlar

// negative number will start from the end

output = learning.slice(0,4);

output = learning.slice(-3);

let learning = "yunus ha erai yunus ha tensai"

output = learning.slice(6)

console.log(output)

//output = learning.substring(6)  sonuç:ha erai yunus ha tensai

//output = learning.slice(6)  sonuç:ha erai yunus ha tensai

let learning = "yunus ha erai yunus ha tensai"

output = learning.slice(0,6)

console.log(output)

//output = learning.substring(0,6)  sonuç:yunus

//output = learning.slice(0,6)  sonuç:yunus

// Split “”arası boşlu koyunca tek tek öblüyor

output = learning.split(" ");

// Another wxample

const hobbies = 'read, walk, listen music, write, learn to program';

output = hobbies.split(", ");

// Replace. Ilki değişecek olan, ikinci yeni kelime

output = learning.replace('JavaScript', 'Modern JavaScript');

// includes şunu içeriyorsa true yazıyor

output = learning.includes('JavaScript');

output = learning.includes('PHP');

// repeat 3 kere hello yazdı.

let greet = "hello ";

output = greet.repeat(3);

console.log(output);

 NUMBERS IN JAVASCRIPT

const number1 = 30;

const number2 = 20;

const number3 = 20.20;

const number4 = .10213;

const number5 = -3;

console.log(number1);

let result;

// Modulos 100 99’a bölününce artan 1’I gösteriyor

result = number1 % number2;

// Pi

result = Math.PI;

// round ortalamasına yakını gösteriyor 2.3 ise 2

result = Math.round(2.5);

// round up or down (ceil or floor )

result = Math.ceil(2.2); ortalamanın yukarısı: 3

result = Math.floor(2.2); ortalama altı:2

// square root

result = Math.sqrt(144);

// absolute eksiyi artı yapıyor

result = Math.abs(-300);

// power 8\*8\*8:

result = Math.pow(8, 3);

// get the minimum number from a list

result = Math.min(3,5,1,2,9,4,2, -3);

// get the max number from a list

result = Math.max(4,1,21,4,15,5,11,5);

// generate a random number

result = Math.random();

console.log(result);

// 20% Discount from a Shopping Cart

const cartItems = (20 + 20 + 30 + 40);

const discount = (cartItems / 100) \* 20;

const totalPay = cartItems - discount;

console.log('Total: ' + cartItems)

console.log('Discount: ' + discount);

console.log('Pay: ' + totalPay);

// Increments or decrements.

let score = 5; sonra score yazarsan yine 5,sadece ikinci yazdığında +1 ekliyor. (Printten sonra) Hemen +1 eklemesi için ++score yazman lazım.

score++;

score--;

++score;

--score;

score += 3;

score -= 3;

console.log(score);

 DATA TYPES IN JAVASCRIPT

// In languages such as C, Java or C# you have to specify the data type

// In JavaScript you can add that functionality with TypeScript

// TypeOf operator is used to retrieve the data type

let name = 'Juan'

console.log(typeof name);

// Let's review the other data types

const name = "Juan";

console.log(typeof name );

// Numbers

// Boolean

// Null

// will return object

let bankSavings = null;

console.log(typeof bankSavings );

// Undefined

// Symbol (ES6)

let symbolES6 = Symbol('this is a symbol');

console.log(typeof symbolES6);

// Reference (Objetos)

// Arrays

let languages = ['HTML5', 'JS', 'PHP'];

console.log(typeof languages);

// Objects

let person = {

name: 'Juan',

city: 'mexico'

}

console.log(typeof person);

// Dates

Zaman sorarken object dedi

let today = new Date();

console.log(today);

let today = new Date();

console.log(typeof today);

 COMPARISON OPERATORS

// JavaScript can make comparisons between characters. A en küçük a küçük en büyük z

A en küçük a küçük en büyük z.

console.log('a' < 'b'); true veriyor

console.log('Z' < 'a'); false veriyor

// Equality Operator

console.log(2 == '2');

// will check the typeof also)

console.log(2 === '2');

// Not Equals

console.log( 2 != 2); false veriyor

console.log('hello' != ' hello'); tur Verdi. boşluk

// Comparison between null & undefined

console.log(null == undefined);

console.log(null === undefined);

 CONVERT STRINGS TO NUMBERS

let number1 = "50",

number2 = 10,

number3 = 'nine';

console.log(number1 + number2); sonuç 5060.

// Convert number1 to Number. Konsolda string beyaz sayılar mavi gösteriliyor. Number veya parseInt ikisi de sayıya çevirir.

console.log(Number(number1) + number2); sonuç 60

console.log(parseInt(number1) + number2);

// Convert number 3

console.log(number3);

// This will concatenate the value

console.log(number1 + number2); sonuç NAN

// But this will substract the value

console.log(number1 - number2); sonuç 40, eksilerde işe yarıyor

// Another Methods parse in böyle çalışmıyor

let number = "20";

number = Number("20");

number = Number("20.20102");

number = Number(true); true verenler 1

number = Number(false); false verenler 0 göründü

number = Number(null); 0 gösterdi

number = Number("Hello world"); 0 gösterdi

number = Number([1,2,3,4]); NAN gösterdi

console.log(number);

console.log(typeof number); number yazdı

// ParseInt & ParseFloat (küsüratlılarda)

number = parseInt("100");

number = parseInt("100.20"); sonuç 100

number = parseFloat("100.20"); sonuç 100.2

// ToFixed just for numbers

// ToFixed just for numbers

let number1 = "1209139";

let number2 = 1209139.101213;

console.log(Number(number1).toFixed(4) );

//stringi sayı yapıp sonuç çıkarır ama küsaratın sadece ilk 4 basamağı (parantezdeki sayı)

console.log(number2.toFixed() );

//toFixed()parantezi çi boş olunca küsüratı göstermez.

//console.log(number2.toFixed() );sayı olmadığından işe yaramıyor

 CONVERTING DATA TO STRING

// Numbers to string

let number = 90210,

output;

output = String(number);

// Anothers

dato = 4+4;

dato = "4" + "4";

console.log(output);

console.log(output.length);

console.log(typeof output);

// bool to string

output = true;

output = String(true);

// date to string

output = new Date();

output = String( new Date() ); bugünkü tarihi yzr

// array to string

output = String( [1,2,3,4] );

// toString()

output = 20.toString() ; tek sayıda hata verdi

output = true.toString() ;

output = [1,2,3,4].toString() ; 1234 göründü

// null cannot be converted since it doesn't exist

output = null.toString() ;

 TEMPLATE LITERALS

const product1 = 'Pizza';

const price1 = 30;

const product2 = 'Hamburger';

const price2 = 40;

// Old Method

let html;

html = '<ul>'+

'<li>Item: ' + product1 + '</li>' +

'<li>Price: $ ' + price1 + '</li>' +

'<li>Item: ' + product2 + '</li>' +

'<li>Price: $ ' + price2 + '</li>' +

'<li>Total: $' + (price1 + price2) + '</li>';

'</ul>';

// Template Strings

html = `

<ul>

<li>Item: ${product1}</li>

<li>Price: ${price1}</li>

<li>Item: ${product2}</li>

<li>Price: ${price2} </li>

<li>Total: ${total(price1, price2)}</li>

</ul>

`;

function total(param1, param2) {

return param1 + param2;

}

// HTML to inject the code

let app = document.querySelector('#app');

app.innerHTML = html ;

 ARRAYS

// Array is a variable that can hold more than one value at the time

// Usually arrays hold related data.

// Creating an array

const numbers = [10,20,30,40,50];

console.log(numbers);

// Array of Months

//BU İKİSİ AYNI

const months = ['January', 'February', 'March', 'April', 'May', 'June'];

console.log(months);

const months = new Array('January', 'February', 'March', 'April', 'May', 'June');

console.log(months);

//

const months = new Array('January', 'February', 'March', 'April', 'May', 'June');

console.log(months);

// Array with mixed values and data types!

const mixedArray = ["Hello", 10, true, "Yes", null];

console.log(mixedArray);

// Array methods

// Check the length in an array

console.log(months.length);

// Check if is an array Array.isArray komple forml

console.log(Array.isArray(months));

let name = 'Juan';

console.log(Array.isArray(name));

// Access any element in the array

console.log(months[0]);

console.log(months[3]);

// Change Values in the array

months[3] = 'December';

console.log(months);

// Find any element . Endeksini bul.

console.log(months.indexOf('December'));

months.push('Noviembre'); // Add in the end

months.unshift('Mes 0'); Add beginning

months.pop(); Remove from the end

months.shift();Remove beggining

// Remove from specific position

// Splice takes 2 parameters, first one is the position,

// second one, how many elements you want to remove

months.splice(0,2);

const months = new Array('January', 'February', 'March', 'April', 'May', 'June');

months.splice(2,null, "üçüncüye ekledik")

console.log(months);

//null yazarsan silmeden ekler

//[ 'January', 'February', 'üçüncüye ekledik', 'May', 'June' ]

months.splice(2,2,)

//ikinciden itibaren 2 tanesini sildi [ 'January', 'February', 'May', 'June' ]

months.splice(2,months.length-1,)

//[ 'January', 'February' ] sondan başlayarak 4 tanesini sildi

months.splice(months.length-1,2)

//sondan birini sildi [ 'January', 'February', 'March', 'April', 'May' ]

// Reverse sırayı tersine çevirir.

months.reverse();

console.log(months);

// Concatenate two arrays in JavaScript

const array1 = [1,2,3];

const array2 = [9,8,7];

console.log(array1.concat(array2));

// order an array

let fruits = ['Banana', 'Apple', 'Strawberry', 'Orange', 'WaterMelon'];

fruits.sort();

console.log(fruits);

//önce sıralayıp sonra tersine çevirmek için:

console.log(fruits.sort().reverse());

// Order numbers. Sort sadece ilk rakamı sıralıyor

const arrayNumbers = [1,3,100,4,6,7,3,2,14,67];

const numbers =[1,2,3,324,6,548,567,123,5,]

numbers.sort()

console.log(numbers)

//sadece ilk sayı sıralanıyor

// Order from lower to greater gerçek sıralama

arrayNumbers.sort(function(x, y) {

return x - y;

});

// Order from greater to lower

arrayNumbers.sort(function(x, y) {

return y - x;

});

console.log(arrayNumbers);

 JAVASCRIPT OBJECTS

// THis properties are defined by you and you access them with a dot

// Create an Object

const person = {

name: 'Juan',

lastName: 'De la torre',

job: 'Web Developer',

email: 'mail@mail.com'

}

console.log(person);

console.log(person.name);

console.log(person.job);

// Another way but not really common

console.log( person[name] );

// An Object can hold any data type

const person = {

name: 'Juan',

lastName: 'De la torre',

job: 'Web Developer',

email: 'mail@mail.com'

age : 20,

favoriteMusic: ['Trance', 'Rock', 'Grunge'],

living: {

city: 'Guadalajara',

country: 'Mexico'

},

birthYear: function() {

return new Date().getFullYear() - this.age;

}

}

// Access each element

console.log(person);

console.log(person.name);

console.log(person.favoriteMusic);

console.log(person.living);

console.log(person['living']['city']);

// An Object can contain also functions

birthYear: function() {

return new Date().getFullYear() - this.age;

}

// Access the function

console.log( person.birthYear() );

// Array of Objects

let cars = [

{model: 'Mustang', engine: 6.0},

{model: 'Camaro', engine: 6.1},

{model: 'Challenger', engine: 6.1},

];

// Iterate in the array of objects

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

console.log(cars[i].model);

}

 FUNCTIONS

// Function Declaration. // A function must be called

function helloVisitor() {

    console.log('Hello & Welcome ');}

helloVisitor();

//Hello & Welcome

//girilen ismin yazılması için

function helloVisitor(firstName, lastName) {

    return `Hello ${firstName}  ${lastName} `;}

console.log( helloVisitor('yunus', 'smile') );

//Hello yunus smile

console.log( helloVisitor('yunus', ) );

//bir tane girersen sonuç "Hello yunus  undefined "

console.log( helloVisitor('yunus', "") );

//"" yazarsan Hello yunus

// ESKİ YÖNTEM

function helloVisitor(firstName, lastName) {

    if(typeof firstName === 'undefined') {firstName = ''}

    if(typeof lastName === 'undefined') {lastName = ''}

    return `hello ${firstName}  ${lastName} `;

}

console.log( helloVisitor('yunus', 'smile') );

//sonuç:hello yunus  smile

console.log( helloVisitor('yunus', ) );

//sonu: hello yunus

console.log( helloVisitor() );

//sonuç:hello

//YENİ YÖNTEM

function helloVisitor(firstName = 'ziyaretçi' , lastName = 'ikinci rastgele') {

    return `Hello ${firstName}  ${lastName} `;}

console.log( helloVisitor('yunus', 'smile') );

//sonu:Hello yunus  smile

console.log( helloVisitor() )

//hello ziyaretçi. BOŞ BIRAKINCA FİRSTNAME İÇİNDEKİ YAZILIYOR.

//function helloVisitor(firstName = '' , lastName = '')BOŞ OLSA DA FORMÜL ÇALIŞIYOR

// A function must be called

helloVisitor();

// Function expressions a ve b herhangi birşey olabilir. function(a,b) da aynı.

const sum = function(a = 5, b = 2) {

return a + b;

};

console.log(sum(4, 8));

console.log(sum(14, 18));

console.log(sum(24, 28));

console.log(sum());

// FUnctions that are invoked inmediately (IIFEs)

// immediately-invoked function expression

(function() {

console.log('IIFES!!');

})(); //hemen 'IIFES!!' yazdı

// Passing arguments to functions

(function(technology) {

console.log('Learning ' + technology);

})('JavaScript');

// learning javascript yazdı

let formül=function(a,b){

    return a+b

    };

console.log(formül(1,2))

//diğer yazılış şekli

let formül = function(a,b){

    return console.log(a+b)

}

formül(1,2)

(function() {

    console.log('IIFES!!');

})();   //hemen 'IIFES!!' yazdı

// Passing arguments to functions

(function(formül){

    console.log(formül+"ikinci yazı")

})("ilk yazı ");

//ilk yazı ikinci yazı

// Property Methods (a function inside an object is a method)

const musicPlayer = {

play: function(id) {

console.log(`Playing song with the ID: ${id}`);

},

pause: function() {

console.log('paused....');

}

}

musicPlayer.play(30);

musicPlayer.pause();

// Methods can be outside (but variable name should match)

musicPlayer.remove = function(id) {

console.log(`Remove from my playlist: ${id}`)

}

musica.remove(20);

// Basically you can create your own functions, but remember

// JavaScript has set of functions also.

console.log();

alert();

prompt();

confirm();

 DATES IN JAVASCRIPT

// Dates in JavaScript are objects so you have to create a new date.

const today = new Date(); bu yeterli tarih çıkıyor

let output;

console.log(today);

// Date MM-DAY-YEAR

let birthday = new Date('1-5-1987');

// Another way boşluk nokta – hepsi oluyor

birthday = new Date('January 5 1987');

const birthday = new Date('August 19, 1975 23:15:30');

const day1 = birthday.getFullYear();

console.log(day1)

tarih değiştirmek için

output = today.setMonth(değişecek tarihi yaz);

output = today.getMonth();

output = today.getDate();

output = today.getDay();

output = today.getFullYear();

output = today.getMinutes();

output = today.getHours();

output = today.getTime();

output = today.getFullYear();

output = today.setFullYear(2018);

console.log(output);

weekday[0] = "Sunday";

weekday[1] = "Monday";

weekday[2] = "Tuesday";

weekday[3] = "Wednesday";

weekday[4] = "Thursday";

weekday[5] = "Friday";

weekday[6] = "Saturday";

getFullYear()   Get the year as a four digit number (yyyy)

getMonth()  Get the month as a number (0-11)

getDate()   Get the day as a number (1-31)

getHours()  Get the hour (0-23)

getMinutes()    Get the minute (0-59)

getSeconds()    Get the second (0-59)

getMilliseconds()   Get the millisecond (0-999)

getTime()   Get the time (milliseconds since January 1, 1970)

getDay()    Get the weekday as a number (0-6)

Date.now()  Get the time. ECMAScript 5.

 CONTROL STRUCTURES IN JAVASCRIPT

// A Control Structure will tell javaScript the flow

// where the program should be executed.

// If Operator

const score = 100;

// EQUAL

if(score == 100) {

console.log("Yes, is the same");

} else {

console.log("No is not the same");

}

// Not equal

if(score != 100) {

console.log("Yes, different!");

} else {

console.log("Not, not different, values are the same");

}

// Strict comparison Operator

if(score === '1000') {

console.log("Yes is the same");

} else {

console.log("Values are not the same");

}

// Strict comparison operator (not equal)

if(score !== 1000) {

console.log("Yes, is different !");

} else {

console.log("No, is not differet");

}

// Check if variable has a value

const score = 1000;

if(score) {

console.log(`Yes, and the score is: ${score}`);

} else {

console.log('No, there\'s no score');

}

let score;

if(score) {

    console.log(`Yes, and the score is: ${score}`);} else {

    console.log('No, there\'s no score');}

//const LET’e dönüştürülmek zorunda . score'a birşey yazmayınca : No, there\'s no score'

// Check variable exists. Cevap:skor yok

if(typeof score != 'undefined' ) {

console.log(`yes, and the score is ${score}`);

} else {

console.log('No, there\'s no score');

}

// Other comparison are < > and >= <=

cash >= cartTotals yaparsan eşit olduğunda yaplr

let cash = 500;

let cartTotals = 300;

if( cash > cartTotals ) {

console.log('successful payment');

} else {

console.log('insufficient funds');

}

// When you have 1 line you can skip curly braces

Tek şerit varken {} koymaya gerek yok.

let cash = 500;

let cartTotals = 300;

if( cash > cartTotals )

console.log('successful payment');

else

console.log('insufficient funds');

let cash = 500;

let cartTotals = 300;

if( cash > cartTotals )

{console.log('successful payment');}

    else

    {console.log('insufficient funds');}

if( cash > cartTotals )

    console.log('successful payment');

else

    console.log('insufficient funds');

// else if

let currentTime = 20;

if(currentTime <= 10) {

console.log('Good Mourning');

} else if(currentTime <= 18) {

console.log('Good Afternoon');

} else {

console.log('Good Night');

}

let currentTime = 25;

// Operador && will check both conditions

if(currentTime > 0 && currentTime <= 12 ) {

console.log('Good Mourning');

} else if( currentTime > 12 && currentTime <= 18) {

console.log('Good Afternoon');

} else if( currentTime > 18 && currentTime <= 24) {

console.log('Good Night');

} else {

console.log('Invalid...');

}

// Operator || OR

let cash = 300;

let credit = 300;

let cartTotals = 700;

if(cartTotals < cash || cartTotals < credit ) {

console.log('You can pay with cash or credit');

} else {

console.log('Insufficient Funds');

}

// More advanced Example.

let cash = 300;

let credit = 300;

let available = cash + credit;

let cartTotals = 700;

if(cash > cartTotals || credit > cartTotals ) {

console.log('You can pay with cash or credit');

} else if(available >= cartTotals) {

console.log('Pay with both!');

} else {

console.log('Insufficient Funds');

}

// Ternary

let loggedIn = false;

console.log( loggedIn === true ? 'The user is logged in' : 'Not logged, please log in' );

//kısa IF ekse formülü

let loggedIn = false;

console.log( loggedIn === true ? 'The user is logged in' : 'Not logged, please log in' );

let loggedIn = false;

//uzun yazılışı

if(loggedIn === true) {console.log('The user is logged in')} else {console.log('Not logged, please log in')}

 SWITCHES

// The switch statement evaluates an expression, it checks a case, and executes statements associated with that case.

const paymentMethod = 'cash';

switch(paymentMethod) {

case 'cash':

console.log(`Your Payment Method is: ${paymentMethod}`);

break;

case 'check':

console.log(`Your Payment Method is: ${paymentMethod} we will verify the funds...`);

break;

case 'card':

console.log(`Your Payment method is: ${metodoPago} processing...`);

break;

default:

console.log('Please select a payment Method');

break;

}

// Asign a variable from a switch case.

const cars = ['Camaro', 'Mustang', 'Challenger'];

const selected = 2;

let car;

switch(selected) {

case 0:

car = cars[0];

break;

case 1:

car = cars[1];

break;

case 2:

car = cars[2];

break;

}

console.log(`Your selected car is ${car}`);

const faction = ["dwarf", "elf", "human"];

let selected = 1;

let race;

switch(selected){

    case 0:

    race = faction[0];

    break;

    case 1:

    race = faction[1];

    break;

        case 2:

    race = faction[2];

    break;

}

console.log(`Your selected race is ${race}`);

 FOR LOOPS

// a for loop is used to run a code or statement until a

// condition is met

// For loop consist on 3 parts.

// Initial value, condition, and the increment

for(let i = 0; i < 10; i++) {

console.log(`Number: ${i} `);

}

// READ A VALUE

for(let i = 0; i < 10; i++) {

if(i == 2) {

console.log('Yes! 2!');

// test with and without continue;

continue;

}

console.log(`Number: ${i} `);

}

// Break the for Loop

for(let i = 0; i < 10; i++) {

if(i == 2) {

console.log('Yes! 2!');

break;

}

console.log(`Number: ${i} `);

}

// Odd or even number

for(let i = 0; i <= 10; i++) {

if(i % 2 == 0) {

console.log(`${i} is even `);

} else {

console.log(` ${i} is ODD `);

}

}

// For loop for a Shopping cart

const shoppingCart = ['Product 1', 'Product 2', 'Product 3'];

// Access each value manually

shoppingCart[0];

shoppingCart[1];

shoppingCart[2];

for(let i = 0; i < 3 ; i++ ){

console.log(`Product: ${shoppingCart[i]}`);

}

// Use shoppingCart.length

 WHILE & DO WHILE

let i = 0;

while(i < 10) {

console.log(`Number: ${i}`);

i++;

}

// Looping an array with While.

const shoppingCart = ['Product 1', 'Product 2', 'Product 3'];

//loop olmadan böyle tek tek yazılır

shoppingCart[0];

shoppingCart[1];

shoppingCart[2];

//for in ile

for(let i = 0; i < 3 ; i++ ){

    console.log(`Product: ${shoppingCart[i]}`);}

//while ile

    let i = 0;

    while(i < shoppingCart.length ) {

        console.log(`Product: ${shoppingCart[i]}`);

        i++;}

//do while

do {

    console.log(`Number: ${i}`)

    i++;

} while( i < 10);

// Do While will run at least 1 time, doesn't really matter if the cndition is meth or not

let i = 0; // Try with 1000

do {

console.log(`Number: ${i}`)

i++;

} while( i < 10);

 LOOP AN ARRAY WITH FOR, FOREACH & MAP

// Loop an array with for

let todoList = ['Homework', 'Food', 'Project', 'Learn JS'];

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

console.log(todoList[i] );

}

// loop an array with foreach

todoList.forEach(function(assignment, index) {

console.log(`${index} : ${assignment}`);

});

// Loop with MAP

const shoppingCart = [

{id: 1, product: 'Book' },

{id: 2, product: 'Shirt'},

{id: 3, product: 'Album'}

];

const productName = shoppingCart.map(function(shoppingCart) {

return shoppingCart.product;

const sepet = [

    {id: 1, product: 'Book' },

    {id: 2, product: 'Shirt'},

    {id: 3, product: 'Album'}

];

const isim = sepet.map(function(isim){

    return isim.product;

});

console.log(isim)

});

// This will extract just the product name in a new array.

console.log(productName);

// Iterators in ES6

let myCar = {

model: 'Camaro',

engine: '6.0',

yeah: 1969,

make: 'Chevrolet'

}

for(let key in myCar) {

console.log(`${key}: ${myCar[key]}`);

}

let myCar = {

    model: 'Camaro',

    engine: '6.0',

    yeah: 1969,

    make: 'Chevrolet'

}

for(let key in myCar) {

    console.log(`${key}: ${myCar[key]}`);

}

/\*

model: Camaro

engine: 6.0

yeah: 1969

make: Chevrolet

SADECE İÇERİĞİ AÇACAKSAN\*/

for(let key in myCar) {

    console.log(`${myCar[key]}`);

}

/\*

Camaro

6.0

1969

Chevrolet

 WINDOW OBJECT

// Window Object, type this in the window.

window

// You don't have to type console.log or alert with window since they're part of the global in javascript

window.console.log('hello');

window.alert("alert!");

// Prompt

const name = prompt('Your Name?');

// Confirm OK’e tıklanırsa şunu yap

if(confirm('Are you sure ?')) {

console.log('Deleted...')

} else {

console.log('Nothing happens...');

}

// Retrieve width and height of the window

let height, width;

height = window.outerHeight;

width = window.outerWidth;

// without interface

height = window.innerHeight;

width = window.innerWidth;

console.log(height);

console.log(width);

width=window.innerwidth

if(width>5000){

    document.body.style.backgroundColor='red';

}else if (width>100){

    document.body.style.backgroundColor='blue';}

    else {

        document.body.style.backgroundColor='green';

    }

    console.log(width);

    //yeşili çalıştırabildim. css gibi konsol rengini değiştiriyor.

// Location

let urlLocation = window.location;

console.log(urlLocation);

console.log(urlLocation.hostname);

console.log(urlLocation.port);

// append this in the url ?id=20&name=juan

console.log(ubicacion.search);

// redirect via JS

window.location.href = 'http://google.com';

 SCOPE

// Scope is the accessibility of variables, functions, and

// objects in your code.

// scope will determine the visibility of variables

// and their values in your code

Formülün içindeki silinirse formül dışındaki değer alınıyor ama formül dışındaki silinirse içindekiler görünmüyor.

let a = "a";

let b = "b";

const c = "c";

// fUNCTION Scope

function function\_scope() {

let a = 'A';

let b = 'B';

const c = 'C';

console.log('Function: '+ a,b,c);

}

function\_scope();

console.log('Global:'+ a,b,c);

// Block Scope (if, for, and others delimited by {} )

if(true) {

let a = 'AA';

let b = 'BB';

const c = 'CC';

console.log('BLOCK LEVEL: '+ a,b,c);

}

// FOR

for(let a = 0; a < 10; a++) {

console.log(a);

}

 DOM

let element;

element = document;

element = document.all;

element = document.all[0];

element = document.head;

element = document.body;

element = document.domain;

element = document.URL;

element = document.characterSet;

element = document.contentType;

element = document.links;

element = document.links[0].id;

element = document.links[0].className;

element = document.forms;

element = document.forms[0];

element = document.forms[0].id;

element = document.forms[0].method;

element = document.forms[0].action;

element = document.forms[0].classList;

element = document.forms[0].classList[0];

element = document.images;

element = document.scripts;

element = document.scripts[2].getAttribute('src');

// looping all images . ilki arrey’e dönüştürüyor. Ikincisi resimleri sıralı gösteriyor.

let images = document.images;

let imagesArray = Array.from(images);

imagesArray.forEach(function(image) {

console.log(image);

});

console.log(element);

let heading=document.getElementById('heading');

console.log(heading.textContent);

//görünen : online courses

 SELECTING ELEMENTS IN JS

// Selecting DOM Elements

console.log(document.getElementById('heading'));

// Retrieve id or class from heading

console.log( document.getElementById('heading').id );

console.log( document.getElementById('heading').className );

// Change the CSS

let heading = document.getElementById('heading');

heading.style.background = '#333';

heading.style.color = '#FFF';

heading.style.padding = '20px';

// Change the Text içeriği değiştirme

heading.textContent = 'The best courses';

// Alternative way

heading.innerText = 'Learn from the Experts';

//farklı seçme yöntemleri.

let card = document.querySelector('.card'); //class

const heading = document.querySelector('h1');//tag

const learningHeading = document.querySelector('#learn');//id

// Query SELECTOR with ID

const learnHeading = document.querySelector('#learn');

// Query Selector with Class

const tagline = document.querySelector('.tagline');

const tagline = document.querySelector('.tagline');

const newText = tagline.textContent.replace('$15', '$120');

tagline.textContent = newText;

//fiyatı değiştirdik. replace 15'ten 120 yaptık.

document.querySelector('#clear-cart').addEventListener('click', function(e) {

    e.preventDefault();

    console.log("it works");

});

// Another Method

const clearCartBtn = document.querySelector('#clear-cart');

clearCartBtn.addEventListener('click', function(e) {

    e.preventDefault();

    console.log("it works");

});

// Another Method

document.querySelector('#clear-cart').addEventListener('click', executeFunction );

function executeFunction(e) {

    e.preventDefault();

    console.log("It's working");

//......................

// Query selector with Tag

const heading = document.querySelector('h1');

// If there're different elements query selector will return the first

let card = document.querySelector('.card');

// Nesting like CSS

let image = document.querySelector('.card img');

// li:nth-child(3) or li:last-child or li:first-child

let link = document.querySelector('#primary a:last-child');

let link = document.querySelector('#primary a:nth-child(2)');

let link = document.querySelector('#primary a:first-child');

 SELECT MULTIPLE ELEMENTS IN JS

//eski yöntem

const links = document.getElementById('primary').getElementsByClassName('link');

//yeni yöntem

const links = document.querySelector('#primary').getElementsByClassName('link');

console.log(links);

// document.getElementsByClassName

const links = document.getElementsByClassName('link');

// console.log(links);

links[0].style.color = 'red';

links[2].textContent = 'New Text';

console.log(links[0]);

// You can also use queryselector and getElementsByClassName

const links = document.querySelector('nav').getElementsByClassName('link');

console.log(links);

// document.getElementsByTagName

let images = document.getElementsByTagName('img');

console.log(images[0]);

// Convert HTMLCollection to array

images = Array.from(images);

console.log(images);

// Loop through SRC of images.

images.forEach(function(image) {

console.log(image.src);

});

let images = document.getElementsByTagName('img');

images = Array.from(images);

images.forEach(function(image) {

    console.log(image.src);

});

// document.querySelectorAll

// returns a node list

const cards = document.querySelectorAll('.card');

console.log(cards);

const courses = document.querySelectorAll('.card h4')

courses.forEach(function(course) {

console.log(course.textContent);

});

// odd links

const oddLinks = document.querySelectorAll('#primary a:nth-child(odd)');

console.log(oddLinks);

oddLinks.forEach(function(odd) {

odd.style.backgroundColor = 'red';

odd.style.color = 'white';

});

// Even Links

const evenLinks = document.querySelectorAll('#primary a:nth-child(even)');

console.log(evenLinks);

evenLinks.forEach(function(even) {

even.style.backgroundColor = 'blue';

even.style.color = 'white';

});

// Change all add-to-cart texts.

const addCartBtns = document.querySelectorAll('.add-to-cart');

addCartBtns.forEach(function(button) {

button.textContent = 'New Text';

});

 TRAVERSING THE DOM

// traversing is how you move in your html code based on how

// this elements are related to the other

// In traversing you define the element to select

// and then you move until you reach the desired element

// Traversing

let element;

const navigation = document.querySelector('nav');

const links = document.querySelector('.link');

// Get ChildNodes // Nodelist de todo

element = navigation.childNodes;

// get Children // elements (doesn't add the text)

element = navigation.children;

// element = navigation.children[0].nodeName;

// element = navigation.children[0].nodeType;

// 1 = Element

// 2 - Attribute

// 3 - Text node

// 8 - Comment

// 9 - document

// 10 doctype

navigation.children[2].textContent = 'Hello!!';

// Children of the children

element = navigation.children[3].children[0].textContent;

// Last Child

element = navigation.lastChild;

element = navigation.lastElementChild;

// First Child

element = navigation.firstChild;

element = navigation.firstElementChild;

// Count the elements

element = navigation.childElementCount;

console.log( element ) ;

// Parent

let element;

let cartBtn = document.querySelector('.add-to-cart');

// Parent Node

element = cartBtn.parentNode;

element = cartBtn.parentElement;

element = cartBtn.parentElement.parentElement;

// sibling (next)

element = cartBtn.nextSibling;

element = cartBtn.nextElementSibling;

element = cartBtn.nextElementSibling.nextElementSibling;

// Siblings (previous)

element = cartBtn.previousSibling;

element = cartBtn.previousElementSibling;

element = cartBtn.previousElementSibling.previousElementSibling;

console.log(element);

 CREATE ELEMENTS WITH JAVASCRIPT

// add a new link

const newLink = document.createElement('a');

// add a class

newLink.className = 'link';

// add the href

newlink.href = '#';

newLink.setAttribute('href', '#');

// Add the Text

newLink.appendChild(document.createTextNode('New Link'));

// add the new link to the #primary or #secondary

document.querySelector('#primary').appendChild(newLink);

 MODIFYING HTML ELEMENTS

// Replace an element

const newHeading = document.createElement('h2');

// add an id

newHeading.id = 'heading';

// add a new text

newHeading.appendChild(document.createTextNode('The Best Courses'));

// Select the old element

const oldHeading = document.querySelector('#heading');

// Parent

const body = document.querySelector('body');

// Then, Replace (first the new element, then the old)

body.replaceChild(newHeading, oldHeading);

 REMOVE ELEMENTS

// Remove any element

// Remove by it's own

const links = document.querySelectorAll('a');

links[0].remove();

// Remove by the children

const navigation = document.querySelector('#primary');

navigation.removeChild(links[2]);

 CLASSES, ID'S AND ATTRIBUTES

// classes & attributes

const link = document.querySelector('.link');

let element;

element = link;

// Read the Class

element = link.className;

// Read the class as DOM Token List

element = link.classList;

// access to specific class

element = link.classList[0];

// Add a new class

link.classList.add('new-class');

// Remove a class

link.classList.remove('new-class');

// read attributes

element = link.getAttribute('href');

element = link.setAttribute('href', 'facebook.com');

element = link.setAttribute('data-link', '1');

element = link.hasAttribute('data-link');

element = link.removeAttribute('data-link');

element = enlace;

console.log(element);

 EVENT LISTENERS (CLICK)

document.querySelector('#clear-cart').addEventListener('click', function(e) {

e.preventDefault();

console.log("it works");

});

// Another Method

const clearCartBtn = document.querySelector('#clear-cart');

clearCartBtn.addEventListener('click', function(e) {

e.preventDefault();

console.log("it works");

});

document.querySelector('#clear-cart').addEventListener('click', executeFunction );

function executeFunction(e) {

e.preventDefault();

console.log("It's working");

// target

let element;

element = e;

// read the values.

element = e.target;

element = e.target.id;

element = e.target.className;

element = e.target.innerText;

element = e.target.innerText = 2 + 2;

console.log(element);

}

 MOUSE EVENTS

// Create the variables

const heading = document.querySelector('#heading');

const links = document.querySelector('nav');

const clearCartBtn = document.querySelector('#clear-cart');

// click

clearCartBtn.addEventListener('click', printEvent);

// Doble CLick

clearCartBtn.addEventListener('dblclick', printEvent);

// Mouse Enter hover

clearCartBtn.addEventListener('mouseenter', printEvent);

// mouse Leave buton mousetan uzaklaşınca

clearCartBtn.addEventListener('mouseleave', printEvent);

// Mouse over Mouse Enter hover ile aynı

clearCartBtn.addEventListener('mouseover', printEvent);

// mouse Out buton mousetan uzaklaşınca

clearCartBtn.addEventListener('mouseout', printEvent);

// MouseDown (click and hold)

clearCartBtn.addEventListener('mousedown', printEvent);

// Mouse Up (mouse click and on release)

clearCartBtn.addEventListener('mouseup', printEvent);

// MouseMove üzerinden geçince aktif

links.addEventListener('mousemove', printEvent);

function printEvent(e) {

console.log(`The Event is: ${e.type}` );

}

 INPUT EVENTS

// Create the variables

const searchForm = document.getElementById('search');

const searchInput = document.getElementById('search-course');

// esperar a submit

searchForm.addEventListener('submit', printEvent);

// Input Events

onkeydown is fired when the key is down (like in shortcuts; for example, in Ctrl+A, Ctrl is held 'down'.

onkeyup is fired when the key is released (including modifier/etc keys)

onkeypress is fired as a combination of onkeydown and onkeyup, or depending on keyboard repeat (when onkeyup isn't fired). (this repeat behaviour is something that I haven't tested. If you do test, add a comment!)

searchInput.addEventListener('keydown', printEvent);

searchInput.addEventListener('keyup', printEvent);

searchInput.addEventListener('keypress', printEvent);

searchInput.addEventListener('focus', printEvent);boşluğa tıklayınca

searchInput.addEventListener('blur', printEvent);

const searchForm = document.getElementById('search');

const searchInput = document.getElementById('search-course');

searchInput.addEventListener('blur', printEvent);

function printEvent(e){

    if(searchInput.value.length==0){

        alert("yazmayı unuttun yarram");

    }

}

searchInput.addEventListener('cut', printEvent);

searchInput.addEventListener('copy', printEvent);

searchInput.addEventListener('paste', printEvent);

searchInput.addEventListener('input', printEvent);

// Form

function printEvent(e) {

// read the values in the Input

console.log(searchInput.value);

console.log(`Type: ${e.type}`);

// searchForm.reset();

e.preventDefault();

}

 EVENT BUBBLING

// Event Bubling

const cards = document.querySelector('.card');

const infoCards = document.querySelector('.info-card');

const addCartbTN = document.querySelector('.add-to-cart');

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

console.log('You Clicked on the Card!');

});

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

console.log('You Clicked on the Info!');

});

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

console.log('You Clicked on the Add To Cart btn');

});

// You prevent this with stopPropagation() analı ocuklu parent child birbirine girmişse ayrı ayrı kod yürümesini sağlıyor

e.stopPropagation();

 DELEGATION

// Delegation

document.body.addEventListener('click', removeProductFromCart);

function removeProductFromCart(e) {

e.preventDefault();

console.log(e.target.classList);

//console.log(e.target.classList.contains( '.remove') );

if(e.target.classList.contains( '.remove') ){

e.target.parentElement.parentElement.remove();

}

}

 LOCAL STORAGE

// Add to local Storage

localStorage.setItem('name', 'Juan');

// add to session storage

sessionStorage.setItem('name', 'Pablo');

// remove from local storage

localStorage.removeItem('name');

// read the value

const name = localStorage.getItem('name');

// Limpiar todo

localStorage.clear();

// If you add something else this will override the Local Storage

localStorage.setItem('name', 'Juan');

localStorage.setItem('name', 'Walter White');

// There're 2 ways of fixing this...

localStorage.setItem('name1', 'Juan');

localStorage.setItem('name2', 'Walter White');

// The second method is better, since LocalStorage

// only saves data as a string, we are going to save

// this is an array

const localStorageContent = localStorage.getItem('name');

console.log(localStorageContent);

let name;

if(localStorageContent == null) {

name = [];

} else {

name = JSON.parse( localStorageContent) ;

}

name.push('Walter ');

localStorage.setItem('name', JSON.stringify( name ) );

 PROJECT: SAVE FROM FORM TO LOCALSTORAGE

// Variables

const tweetList = document.getElementById('tweet-list');

// Event Listeners

eventListeners();

function eventListeners() {

// Form Submission

document.querySelector('form').addEventListener('submit', newTweet);

// Remove Tweet from list

tweetList.addEventListener('click', removeTweet);

// Document Ready

document.addEventListener('DOMContentLoaded', localStorageLoad);

}

// New Tweet when form is submitted

function newTweet(e) {

// Read textarea value

const tweet = document.getElementById('tweet').value;

// Create the remove button

const removeBtn = document.createElement('a');

removeBtn.classList = 'remove';

removeBtn.textContent = 'X';

// Create an LI element

let li = document.createElement('li');

li.textContent = tweet;

// Add the remove button to each tweet

li.appendChild(removeBtn);

// Add into the list

tweetList.appendChild( li );

// Add to LocalStorage

addTweetLocalStorage(tweet);

// Prevent The Default when form is submitted

e.preventDefault();

}

// Remove Tweet from the DOM

function removeTweet(e) {

// Detect which element is clicked

e.preventDefault();

if( e.target.className === 'remove-tweet' ) {

e.target.parentElement.remove()

}

// Remove From Storage

removeTweetLocalStorage(e.target.parentElement.textContent);

}

// Add Tweet Intro Local Storage

function addTweetLocalStorage(tweet) {

// Read from Storage

let tweets;

tweets = getTweetsFromStorage();

// Add the new tweet

tweets.push(tweet);

// Convert tweet array into string

localStorage.setItem('tweets', JSON.stringify(tweets));

// Print an Alert

alert('Tweet Added');

}

// Removes the tweets from local Storage

function removeTweetLocalStorage(tweet) {

let tweets, tweetBorrar;

// Get tweets from storage

tweets = getTweetsFromStorage();

// Remove the tweet

tweetBorrar = tweet.substring( 0, tweet.length - 1 );

// Loop all the tweets and then remove it

tweets.forEach(function(tweet, index) {

if(tweetBorrar == tweet) {

tweets.splice(index, 1);

}

});

// Then save the data

localStorage.setItem('tweets', JSON.stringify(tweets));

}

// Read tweets from local storage

function getTweetsFromStorage() {

let tweets;

// Get the values, if null is returned then create an empty array

if(localStorage.getItem('tweets' ) === null) {

tweets = [];

} else {

tweets = JSON.parse(localStorage.getItem('tweets') );

}

return tweets;

}

// Read values from Local Storage when DOM is ready

function localStorageLoad() {

let tweets;

// Get from storage

tweets = getTweetsFromStorage();

// Loop trought storage and then print the values

tweets.forEach(function(tweet) {

// create the remove button

let removeBtn = document.createElement('a');

removeBtn.classList = 'remove-tweet';

removeBtn.textContent = 'X';

// Create the Li

let li = document.createElement('li');

li.textContent = tweet;

li.appendChild(removeBtn);

// Add into the DOM

tweetList.appendChild( li );

});

}

 PROJECT: ADD COURSES TO THE SHOPPING CART

// Variables

const shoppingCartContent = document.querySelector('#shopping-cart tbody'),

courses = document.querySelector('#courses-list'),

clearCartBtn = document.querySelector('#clear-cart');

// Listeners

loadEventListeners();

// Add event Listeners into a function

function loadEventListeners() {

// When a new course is added

courses.addEventListener('click', buyCourse);

// When the remove button is clicked

shoppingCartContent.addEventListener('click', removeCourse);

// Clear Cart Btn

clearCartBtn.addEventListener('click', clearCart);

// On Document Ready

document.addEventListener('DOMContentLoaded', getFromLocalStorage);

}

// Functions

function buyCourse(e) {

e.preventDefault();

// Use delegation to find the course that was added

if(e.target.classList.contains('add-to-cart') ){

// Read the actual course

const course = e.target.parentElement.parentElement;

// Read the values

getCourseInfo(course);

}

}

// Reads the HTML of the selected course

function getCourseInfo(course) {

// Create an Object with Course Data

const courseInfo = {

image: course.querySelector('img').src,

title: course.querySelector('h4').textContent,

price: course.querySelector('.price span').textContent,

id: course.querySelector('a').getAttribute('data-id')

}

// Insert into the Shopping cart

addIntoCart(courseInfo);

}

function addIntoCart(course) {

// Create a TR

const row = document.createElement('tr');

// Build the Template String

row.innerHTML = `

<tr>

<td>

<img src='${course.image}' width=100>

</td>

<td>${course.title}</td>

<td>${course.price}</td>

<td><a href="#" class="remove" data-id="${course.id}"> X</a></td>

</tr>

`;

// add into the shopping cart

shoppingCartContent.appendChild(row);

// add into the storage

saveIntoStorage(course);

}

// Remove Course from DOM

function removeCourse(e) {

let course, courseId;

// remove element from the DOM

if(e.target.classList.contains('remove')) {

e.target.parentElement.parentElement.remove();

course = e.target.parentElement.parentElement;

courseId = course.querySelector('a').getAttribute('data-id') ;

}

// Remove from storage when removed from DOM

removeCourseLocalStorage(courseId);

}

// Clear Cart

function clearCart() {

// first method

// shoppingCartContent.innerHTML = '';

// Ejemplo 2, más rápido.

while(shoppingCartContent.firstChild) {

shoppingCartContent.removeChild(shoppingCartContent.firstChild);

}

// Clear Local Storage

clearLocalStorage();

}

// Add the courses into Local Storage

function saveIntoStorage(course) {

let courses;

// If something exists on storage then we get value, otherwise, create empty array

if(localStorage.getItem('courses') === null) {

courses = [];

} else {

courses = JSON.parse(localStorage.getItem('courses'));

}

// Add the new course

courses.push(course);

// Since Storage only saves strings, we need to convert array into JSON

localStorage.setItem('courses', JSON.stringify(courses) );

}

// Remove from storage

function removeCourseLocalStorage(courseId) {

let coursesLS;

// Check if there's something on storage

if(localStorage.getItem('courses') === null) {

coursesLS = [];

} else {

coursesLS = JSON.parse(localStorage.getItem('courses'));

}

// Loop throught array and find the course

coursesLS.forEach(function( courseLS, index) {

if(courseId == courseLS.id) {

coursesLS.splice(index, 1);

}

});

// Add the rest of the array

localStorage.setItem('courses', JSON.stringify(coursesLS));

}

// Get courses from storage

function getFromLocalStorage() {

let coursesLS;

// If something on storage, then get the value

if(localStorage.getItem('courses') === null) {

coursesLS = [];

} else {

coursesLS = JSON.parse(localStorage.getItem('courses'));

}

// Loop throught the courses and print the values

coursesLS.forEach(function(course) {

// Creates a TR

const row = document.createElement('tr');

row.innerHTML = `

<tr>

<td>

<img src='${course.image}' width=100>

</td>

<td>${course.title}</td>

<td>${course.price}</td>

<td><a href="#" class="remove" data-id="${course.id}"> X</a></td>

</tr>

`;

shoppingCartContent.appendChild(row);

});

}

function clearLocalStorage() {

localStorage.clear();

}

 PROJECT: SIMULATE EMAIL SENT

// variables

const sendMailForm = document.getElementById('email-form'),

email = document.getElementById('email'),

subject = document.getElementById('subject'),

message = document.getElementById('message'),

sendBtn = document.getElementById('sendBtn'),

resetBtn = document.getElementById('resetBtn');

// Event Listeners

eventListeners();

// Functions

function eventListeners() {

// App Init

document.addEventListener('DOMContentLoaded', appInit);

// Validate the Form

email.addEventListener('blur', validateField);

subject.addEventListener('blur', validateField);

message.addEventListener('blur', validateField);

// Send Email & Reset Buttons

sendMailForm.addEventListener('submit', sendEmail);

resetBtn.addEventListener('click', resetForm);

}

// App Inicialization

function appInit() {

// Disable Button when loaded

sendBtn.disabled = true;

}

// Validates Fields

function validateField() {

let errors;

// Validates the length of the field value

validateLength(this);

// Validate email

if(this.type == 'email') {

validateEmail(this);

}

// Both will return errors, then check if any errors..

errors = document.querySelectorAll('.error');

// Loop Throught the fields

if( email.value !== '' && subject.value !== '' && message.value !== '' ) {

if(errors.length === 0) {

// Remove the disabled if everything is fine

sendBtn.disabled = false;

}

}

}

// Send the email

function sendEmail(e) {

// Show the spinner

let spinner = document.querySelector('#spinner');

spinner.style.display = 'block';

// Show the image

let sendEmailImg = document.createElement('img');

sendEmailImg.src = 'img/mail.gif';

sendEmailImg.style.display = 'block';

// Hidde spinner then show the sendEmailImg

setTimeout(function() {

// Hide the Spinner

spinner.style.display = 'none';

// Show the Image

document.querySelector('#loaders').appendChild( sendEmailImg );

// After 5 seconds hide the image and reset the form

setTimeout(function() {

sendMailForm.reset();

sendEmailImg.remove();

}, 5000);

}, 3000 );

e.preventDefault();

}

// Reset the form

function resetForm(e) {

sendMailForm.reset();

e.preventDefault();

}

// Validate length in the fields,

function validateLength(field) {

if(field.value.length > 0 ) {

field.style.borderBottomColor = 'green';

field.classList.remove('error');

} else {

field.style.borderBottomColor = 'red';

field.classList.add('error');

}

}

// Validate email

function validateEmail(field) {

let emailText = field.value;

// Check if the email contains the @ sign

if( emailText.indexOf('@') !== -1 ) {

field.style.borderBottomColor = 'green';

field.classList.remove('error');

} else {

field.style.borderBottomColor = 'red';

field.classList = 'error';

}

}

 CONSTRUCTOR & THIS

// You can create objects with 2 methods

// The first one is called the object literal

// Object Literal

const client = {

name: 'Juan',

balance: 2000,

membership : function() {

let name;

// Check different Balance

if(this.balance > 1000) {

name = 'Gold';

} else if(this.saldo > 500) {

name = 'Platinum';

} else {

name = 'Normal';

}

return name;

}

}

console.log(client);

console.log(client.name);

console.log(client.balance);

console.log(client.membership() );

// The second method is know as the constructor and this one is more

// powerful and will provide more dynamic behaviour

// Object Constructor

function Client(name, balance) {

this.name = name;

this.balance = balance;

this.membership = function() {

let name;

// check for the different balances

if(this.saldo > 1000) {

name = 'Gold';

} else if(this.saldo > 500) {

name = 'Platinum';

} else {

name = 'Normal';

}

return name;

}

}

const person = new Client('Juan', 2000);

const person2 = new Client('Karen', 600);

// You can access the method with this code

console.log(person.membership() );

 OTHER CONSTRUCTORS

// String

const name1 = 'Karen';

const name2 = new String('Karen');

console.log(typeof name1);

console.log(typeof name2);

// try with name1 & 2

if(name1 === name2) {

console.log('Yes');

} else {

console.log('No');

}

// Numbers

const number1 = 20;

const number2 = new Number(20);

// boolean

const boolean1 = true;

const boolean2 = new Boolean(true);

// Functions

const function1 = function(a, b) {

return a + b;

}

const function2 = new Function('a','b', 'return a + b');

console.log(function2(1, 2));

// Objects

const person1 = {name: 'Juan'};

const person2 = new Object({name: 'Juan'});

// Arrays

const array1 = [1,2,3,4];

const array2 = new Array(1,2,3,4);

 ES5 PROTOTYPES

// All The Objects in JavaScript will contain a Prototype

Client.prototype;

String.prototype;

// Instead of making your objects full of methods, you can create a prototype

function Client(name, balance) {

this.name = name;

this.balance = balance;

}

// then attach the prototype

Client.prototype.membership = function() {

let name;

if(this.saldo > 1000) {

name = 'Gold';

} else if(this.saldo > 500) {

name = 'Platinum';

} else {

name = 'Normal';

}

return name;

}

// Second Protoype with name and balance..

Client.prototype.clientInfo = function() {

return `Name: ${this.name}, Balance ${this.balance}, Category: ${this.membership() } `;

}

// Another method to withdraw money from the account

Client.prototype.withdraw = function(amount) {

this.balance -= amount;

}

// Deposit money

Client.prototype.deposit = function(amount) {

this.balance += amount;

}

// Check Balance

Client.prototype.getBalance = function() {

return this.balance;

}

// Instanciate the method

const client = new Client('Karen', 600);

// Then access the prototypes

console.log ( client.membership() );

// Print the client info

console.log ( client.clientInfo() );

// withdraw money

client.withdraw(2000);

// check balance

console.log ( client.getBalance() );

// Deposit

client.deposit(2000);

console.log ( client.getBalance() );

// Check for properties...

console.log(client.hasOwnProperty('getBalance'));

console.log(client.hasOwnProperty('clientInfo'));

 INHERITING PROTOTYPES

function Client(name, balance) {

this.name = name;

this.balance = balance;

}

// Create the prototype

Client.prototype.clientInfo = function() {

return `Name: ${this.name}, Balance: ${this.balance} } `;

}

// instanciate, then run the method

const client = new Client('Juan', 1000);

console.log( client.clientInfo() );

// Business

function Business(name, balance, phone, category) {

// in this case you don't use this, you should use .call

Client.call(this, name, balance);

this.phone = phone;

this.category = category;

}

// Inherit client info

Business.prototype = Object.create(Client.prototype);

// Return the prototype for Business

Business.prototype.constructor = Business;

// Create a Business

const business = new Business('Udemy', 1000000, 012345678, 'Education');

console.log(business);

// Attach a new Prototype with all the properties

Business.prototype.businessInfo = function() {

return `Hello from proto Business ${this.name}, balance is: ${this.balance} & category: ${this.category}`;

}

// Test the previous Prototype

console.log(business.businessInfo() );

 OBJECT CREATE

// Object Create

const Client = {

getBalance: function() {

return `hello ${this.name} ${this.balance}`;

},

withdraw: function(amount) {

return this.balance -= amount;

},

deposit: function(amount) {

return this.balance += amount;

}

}

// Create a new object called mary and give a balance of 1000

const mary = Object.create(Client);

// Attach mary and balance

mary.name = 'Mary';

mary.balance = 1000;

// Send to the console

console.log(mary);

console.log(mary.getBalance() );

// Withdraw some money

mary.withdraw(500);

console.log(mary.getBalance() );

// Deposit some money

mary.deposit(1200);

console.log(mary.getBalance() );

// Another way...

const juan = Object.create(Client, {

name : {value: 'Juan'},

lastName : {value: 'De la torre'},

job: {value: 'Web Developer'}

});

console.log(juan.job );

 ES6 CLASSES

// In ES6 you will have access to Classes instead of Prototypes

class Client {

// Create the constructor

constructor(name, balance) {

this.name = name;

this.balance = balance;

}

// Any method inside the class will be added to the prototype...

// Print client information

clientInfo() {

return `Hello ${this.name}, your balance: ${this.balance}`;

}

// Membership

membership() {

let name;

if(this.balance > 1000) {

name = 'Gold';

} else if(this.balance > 500) {

name = 'Platinum';

} else {

name = 'Normal';

}

return name;

}

withdraw(amount) {

this.balance -= amount;

}

// Static methods doesn't require instanciate

static welcome() {

return `Welcome to your bank.`;

}

}

const mary = new Client('Mary', 1000);

console.log(mary);

// Access the methods

console.log(mary.clientInfo() );

console.log(mary.membership() );

// Withdraw some money

mary.withdraw(600);

// Check Again

console.log(mary.clientInfo() );

console.log(mary.membership() );

// This will cause an error since is not parte of current instance...

console.log(mary.welcome() );

// But this will work!

console.log(Client.welcome());

 SUBCLASSES

// In ES6 you can inherit a class, constructor and properties really easy..

class Client {

// Create the constructor

constructor(name, balance) {

this.name = name;

this.balance = balance;

}

// Print client information

clientInfo() {

return `Hello ${this.name}, your balance: ${this.balance}`;

}

// Static methods doesn't require instanciate

static welcome() {

return `Welcome to your bank.`;

}

}

class Business extends Client {

constructor(name, balance, phone, category) {

// Access the parent constructor properties...

super(name, balance);

// New attributes require this.

this.phone = phone;

this.category = category;

}

// Print client information

clientInfo() {

return `Hello ${this.name}, your balance: ${this.balance}, Category: ${this.category}`;

}

// Print the balance...

balance() {

return ${this.balance};

}

static welcome(){

return `Welcome to Bank for Business`;

}

}

// Instanciate and call the methods...

const john = new Client('John', 3000);

console.log(john);

console.log(john.clientInfo() );

// Instanciate the subclass

const business = new Business('Business Name', 10000, 10290193, 'Real State');

// Since this is a subclass you have access to the methods...

// if you remove the method from the subclass, the parent method will be loaded...

console.log(business.clientInfo() );

console.log(business.balance() );

// You can still have access to both static methods...

console.log(Client.welcome() );

console.log(Business.welcome() );

 PROJECT: CAR INSURANCE QUOTE

// Variables

const form = document.getElementById('request-quote');

// Constructor

function Insurance(make, year, level) {

this.make = make;

this.year = year;

this.level = level;

}

// Prototypes

Insurance.prototype.calculateQuotation = function(insurance) {

//console.log(insurance);

let price;

const base = 2000;

// get make

make = insurance.make;

/\* Makes

1 = American \* 1.15

2 = Asian \* 1.05

3 = European \* 1.35

\*/

switch(make) {

case '1':

price = base \* 1.15

break;

case '2':

price = base \* 1.05

break;

case '3':

price = base \* 1.30

break;

}

// Get the year

year = insurance.year;

// Get the years difference.

const difference = this.getYearDifference(year);

// Each year the cost of the insurance is going to be 3% cheapear

price = price - ((difference \* 3) \* price) / 100;

// Check for level of protection

level = insurance.level;

price = this.calculateLevel(price, level);

return price;

}

Insurance.prototype.getYearDifference = function(year) {

return new Date().getFullYear() - year;

}

Insurance.prototype.calculateLevel = function(price, level) {

/\*

Basic insurance is going to increase the value by 30%

Complete insurance is going to increase the value by 50%

\*/

if(level === 'basic') {

price = price \* 1.30;

} else {

price = price \* 1.50;

}

return price;

}

// HTML Elements

function HTMLUI() {}

HTMLUI.prototype.showResults = function(insurance, total) {

const result = document.getElementById('result');

// Get make from insurance object

let make = insurance.make;

switch(make) {

case '1':

make = 'American';

break;

case '2':

make = 'Asian';

break;

case '3':

make = 'European';

break;

}

// Create div

const div = document.createElement('div');

// Insert the result

div.innerHTML = `

<p class="header">Summary:</p>

<p>Make: ${make}</p>

<p>Year: ${insurance.year}</p>

<p>Level: ${insurance.level}</p>

<p class="total">Total: $ ${total}</p>

`;

result.appendChild(div);

}

HTMLUI.prototype.printError = function(message) {

// Create the Div

const div = document.createElement('div');

div.classList = 'error';

// Insert

div.innerHTML = `

<p>${message}</p>

`;

form.insertBefore(div, document.querySelector('.form-group'));

setTimeout( function() {

document.querySelector('.error').remove();

}, 3000 ) ;

}

// This Insurance company has a policy

// where they offer insure for newer than 20 years.

// Create the differents options on the fly

HTMLUI.prototype.displayYears = function() {

// Print the <option> for years

const max = new Date().getFullYear(),

min = max - 20;

// Generate a List from 20 previous Years

const selectYears = document.getElementById('year');

// Print the values

for(let i = min; i <= max; i++ ) {

// Correct for...

for(let i = max; i > min; i-- ) {

let option = document.createElement('option');

option.value = i;

option.innerHTML = i;

selectYears.appendChild(option);

}

}

/\* When web is loaded\*/

document.addEventListener('DOMContentLoaded', function() {

const html = new HTMLUI();

html.displayYears();

});

// When form is submitted

form.addEventListener('submit', function(e) {

e.preventDefault();

// Get Values from the form

const make = document.getElementById('make').value;

const year = document.getElementById('year').value;

const level = document.querySelector('input[name="level"]:checked').value;

if(selectedMake === '' || selectedYear === '' || level === '') {

// print error from previous isntanciate HTMLUI();

html.printError('Please fill all the fields');

} else {

// Clear previous Quotes

const prevResult = document.querySelector('#result div');

if(prevResult != null) {

prevResult.remove();

}

// Make the Quotation

const insurance = new Insurance(selectedMake, selectedYear, level);

const price = insurance.calculateQuotation(insurance);

// Print result from previous HTMLUI();

html.showResults(insurance, price);

}

});

 PROJECT: CAR INSURANCE WITH CLASSES

// Insurance Class

class Insurance{

constructor(make, year, level) {

this.make = make;

this.year = year;

this.level = level;

}

calculateQuote(insurance) {

console.log(insurance);

let price;

const base = 2000;

// get make

make = insurance.make;

/\* Makes

1 = American \* 1.15

2 = Asian \* 1.05

3 = European \* 1.35

\*/

switch(make) {

case '1':

price = base \* 1.15

break;

case '2':

price = base \* 1.05

break;

case '3':

price = base \* 1.30

break;

}

// Get the year

year = insurance.year;

// Get the years difference.

const difference = this.getYearDifference(year);

// Each year the cost of the insurance is going to be 3% cheapear

price = price - ((difference \* 3) \* price) / 100;

// Check for level of protection

level = insurance.level;

price = this.calculateLevel(price, level);

return price;

}

getYearDifference(year) {

return new Date().getFullYear() - year;

}

calculateLevel(price, level) {

/\*

Basic insurance is going to increase the value by 30%

Complete insurance is going to increase the value by 50%

\*/

if(level === 'basic') {

price = price \* 1.30;

} else {

price = price \* 1.50;

}

return price;

}

}

// HTML Elements

class HTMLUI{

// Shows the result in the HTML

showResults(insurance, total) {

// gets the result div

const result = document.getElementById('result');

// Get make from insurance object

let make = insurance.make;

switch(make) {

case '1':

make = 'American';

break;

case '2':

make = 'Asian';

break;

case '3':

make = 'European';

break;

}

// Create div

const div = document.createElement('div');

// Insert the result

div.innerHTML = `

<p class="header">Summary:</p>

<p>Make: ${make}</p>

<p>Year: ${insurance.year}</p>

<p>Level: ${insurance.level}</p>

<p class="total">Total: $ ${total}</p>

`;

result.appendChild(div);

}

printError(message) {

// Create the Div

const div = document.createElement('div');

div.classList = 'error';

// Insert

div.innerHTML = `

<p>${message}</p>

`;

form.insertBefore(div, document.querySelector('.form-group'));

setTimeout( function() {

document.querySelector('.error').remove();

}, 3000 ) ;

}

}

// When form is submitted

form.addEventListener('submit', function(e) {

e.preventDefault();

// Get Values from the form

const make = document.getElementById('make').value;

const year = document.getElementById('year').value;

const level = document.querySelector('input[name="level"]:checked').value;

const html = new HTMLUI();

if(selectedMake === '' || selectedYear === '' || level === '') {

// print error from previous isntanciate HTMLUI();

html.printError('Please fill all the fields');

} else {

// Clear previous Quotes

const prevResult = document.querySelector('#result div');

if(prevResult != null) {

prevResult.remove();

}

// Make the Quotation

const insurance = new Insurance(selectedMake, selectedYear, level);

const price = insurance.calculateQuotation(insurance);

// Print result from previous HTMLUI();

html.showResults(insurance, price);

}

});

 PROJECT: WEEKLY BUDGET APP

// Variables

const addExpenseForm = document.querySelector('#add-expense'),

budgetTotal = document.querySelector('span#total'),

budgetLeft = document.querySelector('span#left');

let budget, userBudget;

// Event Listeners

eventListeners();

function eventListeners() {

// App Init

document.addEventListener('DOMContentLoaded', function() {

// Aask the visitor the weekly budget

userBudget = prompt('What\'s your budget for this week?');

// Check the value

if(userBudget === null || userBudget === '' || userBudget === '0') {

window.location.reload();

} else {

// Instanciate the Budget Class

budget = new Budget(userBudget);

// Instanciate HTML Class

const ui = new HTML();

ui.insertBudget(budget.budget);

}

});

}

// Listen for form submission

addExpenseForm.addEventListener('submit', function(e) {

// Get values from budget

const expenseName = document.querySelector('#expense').value;

const amount = document.querySelector('#amount').value;

// Instanciate a new class

const ui = new HTML();

// Check they're not empty

if(expenseName === '' && amount === '') {

// 2 parameters, message and type

ui.printMessage('There was an error, all the fields are mandatory', 'alert-danger');

} else {

ui.addToExpenseList(expenseName, amount);

ui.trackBudget(amount);

ui.printMessage('Added', 'alert-success');

}

} );

// Budget Class

class Budget {

// Pass the Weekly Budget

constructor(budget){

this.budget = Number( budget );

this.budgetLeft = this.budget;

}

// Substract from the budget

substractFromBudget(amount = 0) {

return this.budgetLeft -= amount;

}

}

// Everything related to HTML or UI

class HTML {

// Add Budget into Then on init

insertBudget(amount) {

// Insert into HTML

budgetTotal.innerHTML = `${amount}`;

budgetLeft.innerHTML = `${amount}`;

}

trackBudget(amount) {

// Substract from budget

const budgetLeftQuantity = budget.substractFromBudget(amount);

budgetLeft.innerHTML = `${budgetLeftQuantity}`;

// console.log(budget.budget);

// console.log(budgetLeftQuantity);

// Check 25%

if( (budget.budget / 4) > budgetLeftQuantity ) {

// Add the class: danger

budgetLeft.parentElement.parentElement.classList.remove('alert-success', 'alert-warning');

budgetLeft.parentElement.parentElement.classList.add('alert-danger');

} else if( (budget.budget / 2) > budgetLeftQuantity) {

// add the class: warning

budgetLeft.parentElement.parentElement.classList.remove('alert-success');

budgetLeft.parentElement.parentElement.classList.add('alert-warning');

}

}

addToExpenseList(name, amount) {

const expensesList = document.querySelector('#expenses ul');

// Create the li

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

li.className = 'list-group-item d-flex justify-content-between align-items-center';

// Insertar columns

li.innerHTML = `

${name}

<span class="badge badge-primary badge-pill">$ ${amount}</span>

`;

// Insert Into HTML

expensesList.appendChild(li);

}

printMessage(message, className) {

const messageWrapper = document.createElement('div');

messageWrapper.classList.add('text-center', 'alert', className);

messageWrapper.appendChild(document.createTextNode(message));

document.querySelector('.primary').insertBefore(messageWrapper, addExpenseForm);

setTimeout(function() {

document.querySelector('.primary .alert').remove();

addExpenseForm.reset();

}, 3000);

}

}

 AJAX

document.getElementById('button').addEventListener('click', loadData);

function loadData() {

// Create the xmlttprequest object

const xhr = new XMLHttpRequest();

// Open the connection

xhr.open('GET', 'data.txt', true);

// Print ready states if needed

//console.log('Ready States', xhr.readyState)

// Execute ajax call

xhr.onload = function() {

if(this.status === 200) {

document.getElementById('output').innerHTML = `<h1>${this.responseText}</h1>`;

}

}

/\* Second method

xhr.onreadystatechange = function() {

// console.log('Ready States', xhr.readyState)

if(this.status === 200 && this.readyState === 4 ) {

console.log(this.responseText);

}

// console.log('Ready States', xhr.readyState)

}

\*/

// Send the request

xhr.send();

// ReadyStates

// 0 : Unsent

// 1: Opened

// 2: received

// 3: loading

// 4: done

// Codes

// 200: Correct

// 403: Forbidden

// 404: not found

}

 AJAX & JSON

// Employee.json

{

"id" : 1,

"name" : "Juan",

"company" : "EasyWebDev",

"job" : "Web Developer"

}

// Employees.json

[

{

"id" : 1,

"name" : "Juan",

"company" : "EasyWebDev",

"job" : "Desarrollador Web"

},

{

"id" : 2,

"name" : "Mary",

"company" : "EasyWebDev",

"job" : "Designer"

},

{

"id" : 3,

"name" : "Alexa",

"company" : "EasyWebDev",

"job" : "App Developer"

}

]

document.getElementById('button1').addEventListener('click', loadEmployee);

document.getElementById('button2').addEventListener('click', loadEmployees);

function loadEmployee() {

// Create the object

const xhr = new XMLHttpRequest();

// Open the connection

xhr.open('GET', 'employee.json', true);

// Execute

xhr.onload = function() {

if(this.status === 200) {

const employee = JSON.parse(this.responseText);

const output = `

<ul>

<li>ID: ${employee.id}</li>

<li>Name: ${employee.name}</li>

<li>Company: ${employee.company}</li>

<li>Job: ${employee.job}</li>

</ul>

`;

// Print into html

document.getElementById('employee').innerHTML = output;

}

}

// Send the request

xhr.send();

}

// Print all the employees from json

function loadEmployees() {

const xhr = new XMLHttpRequest();

// Open the connection

xhr.open('GET', 'employees.json', true);

xhr.onload = function() {

if(this.status === 200) {

const employees = JSON.parse(this.responseText);

let output = '';

employees.forEach(function(employee) {

output += `

<ul>

<li>ID: ${employee.id}</li>

<li>Name: ${employee.name}</li>

<li>Company: ${employee.company}</li>

<li>Job: ${employee.job}</li>

</ul>

`;

});

document.getElementById('employees').innerHTML = output;

}

}

xhr.send();

}

// When Working with json i recommend to install jsonview

 CONSUMING A REST API WITH AJAX

document.querySelector('#load').addEventListener('click', loadPosts);

// Fetch posts from API

function loadPosts(e) {

const xhr = new XMLHttpRequest();

xhr.open('GET', 'https://jsonplaceholder.typicode.com/posts', true);

xhr.onload = function(){

if(this.status === 200) {

const response = JSON.parse(this.responseText);

let content = '';

response.forEach(function(post) {

content += `

<h3>${post.title}</h3>

<p>${post.body}</p>

`;

});

document.querySelector('#result').innerHTML = content;

}

}

xhr.send();

e.preventDefault();

}

 PROJECT: GENERATE NAMES FROM REST API WITH AJAX

document.querySelector('#generate-names').addEventListener('submit', loadNames);

function loadNames(e) {

e.preventDefault();

// variables

const origin = document.getElementById('country').value;

const genre = document.getElementById('genre').value;

const quantity = document.getElementById('quantity').value;

// URL Constructor

let url = '';

url += 'http://uinames.com/api/?';

// If we have a name, append it to the uRL

if(origin !== '') {

url += `region=${origin}&`;

}

if(genre !== '') {

url += `gender=${genre}&`;

}

if(quantity !== '') {

url += `amount=${quantity}&`;

}

console.log(url);

// Start AJAX CALL

const xhr = new XMLHttpRequest();

xhr.open('GET', url, true);

xhr.onload = function(){

if(this.status === 200) {

const names = JSON.parse(this.responseText);

let html = '<h2>Generated Names</h2>';

html += `<ul class="list">`;

names.forEach(function(name) {

html += `

<li>${name.name}</li>

`;

});

html += `</ul>`;

document.querySelector('#result').innerHTML = html;

}

}

xhr.send();

}

 CALLBACKS

/\*

callbacks are the the cornestone of asynchronous programming in javascript

We have already write a lot of callbacks!

A callback is just a function inside another function

\*/

// Callbacks!

const cities = ['London', 'New York', 'Madrid', 'Paris', 'Berlin'];

// Inline Callback

cities.forEach(function(city) {

console.log(city);

});

// Same callback with a function declaration

function callback(city) {

console.log(city);

}

cities.forEach(callback);

// Let's create an array of countries

const countries = ['France', 'Spain', 'Portugal', 'Australia', 'England', 'Ireland'];

// Then we add a new country 2 seconds later

function newCountry(country, callback) {

setTimeout(function() {

// Add into the array

countries.push(country);

// Execute the callback

callback();

}, 2000 );

}

// The countries are displayed after 1 second

function displayCountries() {

setTimeout(function() {

let html = '';

countries.forEach(function(country) {

html += `<li>${country}</li>`;

});

document.body.innerHTML = html;

}, 1000 );

}

// Add a new Country

newCountry('Germany', displayCountries);

// Print them all

displayCountries();

 PROMISES

//The Promise object represents when a function or task is completed (or has failed)

// of an asynchronous operation, and its resulting value.

// Example with promises

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

// resolve when the promise is succesful

// reject when the promise has failed

// Change to false to run the reject

const discount = true;

if(discount) {

resolve('Discount Applied');

} else {

reject('Discount failed...');

}

});

applyDiscount.then(function(result) {

console.log(result);

}).catch(function(result) {

console.log(result);

});

 FETCH API

document.getElementById('txtBtn').addEventListener('click', loadTxt);

document.getElementById('jsonBtn').addEventListener('click', loadJSON);

document.getElementById('apiBTN').addEventListener('click', loadREST);

// load TXT

function loadTxt() {

fetch('data.txt')

.then(function(response) {

return response.text();

})

.then(function(data) {

console.log(data);

document.getElementById('result').innerHTML = data;

})

.catch(function(error) {

console.log(error);

});

}

// load json

function loadJSON() {

fetch('employees.json')

.then(function(response) {

return response.json();

})

.then(function(data) {

console.log(data);

let html = ''

data.forEach(function(employee) {

html += `

<li>${employee.name} ${employee.job}</li>

`;

});

document.getElementById('result').innerHTML = html;

})

.catch(function(error) {

console.log(error);

});

}

function loadREST() {

fetch('https://picsum.photos/list')

.then(function(response) {

return response.json();

})

.then(function(images) {

console.log(images);

let html = ''

images.forEach(function(image) {

html += `

<li>

<a href="${image.post\_url}">View Image</a>

${image.author}

</li>

`;

});

document.getElementById('result').innerHTML = html;

})

.catch(function(error) {

console.log(error);

});

}

 PROJECT: NAME GENERATOR WITH FETCH API

document.querySelector('#generate-names').addEventListener('submit', loadNames);

function loadNames(e) {

e.preventDefault();

// variables

const origin = document.getElementById('country').value;

const genre = document.getElementById('genre').value;

const quantity = document.getElementById('quantity').value;

// URL Constructor

let url = '';

url += 'http://uinames.com/api/?';

// If we have a name, append it to the uRL

if(origin !== '') {

url += `region=${origin}&`;

}

if(genre !== '') {

url += `gender=${genre}&`;

}

if(quantity !== '') {

url += `amount=${quantity}&`;

}

console.log(url);

// Fetch API

fetch(url)

.then(function(response) {

return response.json();

})

.then(function(names) {

let html = '<h2>Generated Names</h2>';

html += `<ul class="list">`;

names.forEach(function(name) {

html += `

<li>${name.name}</li>

`;

});

html += `</ul>`;

document.querySelector('#result').innerHTML = html;

})

.catch(function(error) {

console.log(error);

});

}

 ARROW FUNCTIONS

// Arrow Functions

const learning = function() {

console.log('Learning Modern JS');

}

// Using Arrow FUnctions

const learning = () => {

console.log('Learning Modern jS');

}

// If your function is one line long you can skip the braces

const learning = () => console.log('Learning Modern jS');

// return a value

const learning = () => 'Learning Modern jS;

console.log(learning());

// Returning objects

const message = () => ({message: 'Hello'});

console.log(message());

// Parameters

const learning = (tech) => console.log(`learning ${tech}`);

learning('JavaScript');

// if you're passing one parameter you can skip parenthesis

const learning = tech => console.log(`learning ${tech}`);

learning('JavaScript');

// Multiple parameters will require de parenthesis

const learning = (tech1, tech2) => console.log(`Learning ${tech1} ${tech2}`);

learning('JS', 'ES6');

// Arrow functions with a callback

const shoppingCart = ['Album', 'Shirt', 'Guitar'];

const productQuantity = shoppingCart.map(function(product) {

return product.length;

});

// with arrow function arrow

const productQuantity = shoppingCart.map(product => {

return product.length;

});

// shorter way

const productQuantity = shoppingCart.map(product => product.length);

// Example with for each

const shoppingCart = ['Album', 'Shirt', 'Guitar'];

shoppingCart.forEach(function(product) {

console.log(product)

});

// Arrow function

shoppingCart.forEach(product => {

console.log(product);

});

 FETCH API WITH ARROW FUNCTIONS

document.getElementById('txtBtn').addEventListener('click', loadTxt);

document.getElementById('jsonBtn').addEventListener('click', loadJSON);

document.getElementById('apiBTN').addEventListener('click', loadREST);

// load TXT

function loadTxt() {

fetch('data.txt')

.then(res => res.text())

.then(data => document.getElementById('result').innerHTML = data )

.catch(error => console.log(error) );

}

// load json

function loadJSON() {

fetch('employees.json')

.then(res => res.json() )

.then(data => {

console.log(data);

let html = ''

data.forEach(function(employee) {

html += `

<li>${employee.name} ${employee.job}</li>`;

});

document.getElementById('result').innerHTML = html;

})

.catch(error => console.log(error) );

}

function loadREST() {

fetch('https://picsum.photos/list')

.then(res => res.json() )

.then(data => {

console.log(data);

let html = ''

data.forEach(image => {

html += `

<li>

<a href="${image.post\_url}">View Image</a>

${image.author}

</li>

`;

});

document.getElementById('result').innerHTML = html;

})

.catch(error => console.log(error) );

}

 PROJECT: NAME GENERATOR WITH FETCH API & ARROW FUNCTIONS

document.querySelector('#generate-names').addEventListener('submit', loadNames);

function loadNames(e) {

e.preventDefault();

// variables

const origin = document.getElementById('country').value;

const genre = document.getElementById('genre').value;

const quantity = document.getElementById('quantity').value;

// URL Constructor

let url = '';

url += 'http://uinames.com/api/?';

// If we have a name, append it to the uRL

if(origin !== '') {

url += `region=${origin}&`;

}

if(genre !== '') {

url += `gender=${genre}&`;

}

if(quantity !== '') {

url += `amount=${quantity}&`;

}

console.log(url);

// Fetch API

fetch(url)

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

.then(names => {

let html = '<h2>Generated Names</h2>';

html += `<ul class="list">`;

names.forEach(function(name) {

html += `

<li>${name.name}</li>

`;

});

html += `</ul>`;

document.querySelector('#result').innerHTML = html;

})

.catch(error => console.log(error) );

}

 ASYNC AWAIT

// The async function defines an asynchronous function, which returns an AsyncFunction object.

// Async await

async function getClients() {

// Create a new Promise

const clients = new Promise((resolve, reject) => {

setTimeout(() => {

resolve(`Client List Downloaded...`);

}, 1000);

});

// error or not...

const error = true;

if(!error) {

const response = await clients; // Will wait until client function is executed

return response;

} else {

// If error is presented then we reject with a global Promise

await Promise.reject(`There was an error...`);

}

}

// Execute the promise

// Try without .then

getClients()

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

.catch(error => console.log(error));

// Second Example with REST API

async function getPosts() {

// wait until the posts are downloaded

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

// Execute then

const data = await response.json();

// Until second await is done...

return data;

}

getPosts().then( posts => console.log(posts) );

 PROJECT: NAME GENERATOR WITH FETCH API, ARROW FUNCTIONS AND ASYNC AWAIT

document.querySelector('#generate-names').addEventListener('submit', loadNames);

function loadNames(e) {

e.preventDefault();

// variables

const origin = document.getElementById('country').value;

const genre = document.getElementById('genre').value;

const quantity = document.getElementById('quantity').value;

// URL Constructor

let url = '';

url += 'http://uinames.com/api/?';

// If we have a name, append it to the uRL

if(origin !== '') {

url += `region=${origin}&`;

}

if(genre !== '') {

url += `gender=${genre}&`;

}

if(quantity !== '') {

url += `amount=${quantity}&`;

}

const names = getNames(url)

.then(result => {

let html = '<h2>Generated Names</h2>';

html += `<ul class="list">`;

result.names.forEach(function(name) {

html += `

<li>${name.name}</li>

`;

});

html += `</ul>`;

document.querySelector('#result').innerHTML = html;

})

}

async function getNames(url) {

// Fetch API

const response = await fetch(url);

const names = await response.json()

return {

names

}

}

 PROJECT: CRYPTOCURRENCIES RATES WITH FETCH API & ASYNC AWAIT

// cryptoAPI.js

class CryptoAPI{

async queryAPI(currency, crypto ) {

// Query the url

const url = await fetch(`https://api.coinmarketcap.com/v1/ticker/${crypto}/?convert=${currency}`);

// Return as json

const result = await url.json();

// return the response

return {

result

}

}

// Get all the cryptocurrencies

async getCryptoCurrenciesList(){

const url = await fetch('https://api.coinmarketcap.com/v1/ticker/')

const cryptocurrencies = await url.json();

return {

cryptocurrencies

}

}

}

UI.js

class UI {

constructor() {

this.init();

}

init() {

this.printCryptoCurrencies();

}

// Print <option> from select

printCryptoCurrencies() {

cryptoAPI.getCryptoCurrenciesList()

.then(data => {

// read value from api

const cryptoCurrencies = data.cryptocurrencies;

const select = document.getElementById('cryptocurrency');

// Build the <select> from the REST API

cryptoCurrencies.forEach(currency => {

// Add the id and value

const option = document.createElement('option');

option.value = currency.id;

option.appendChild(document.createTextNode(currency.name));

select.appendChild(option);

})

})

}

// Displays a message

printMessage(message, className) {

const div = document.createElement('div');

div.className = className;

div.appendChild(document.createTextNode(message));

const messageDiv = document.querySelector('.messages');

messageDiv.appendChild(div);

setTimeout(() => {

document.querySelector('.messages div').remove();

}, 3000 );

}

// Show results

displayResult(result, currency) {

const prevResult = document.querySelector('#result > div');

if(prevResult) {

prevResult.remove();

}

// Display Spinner

this.showSpinner();

// Read the currency

const currencyName = `price\_${currency}`;

// Get the currency value

const value = result[currencyName];

// Construir el template

let templateHTML = '';

templateHTML += `

<div class="card cyan darken-3">

<div class="card-content white-text">

<span class="card-title">Result</span>

<p>The price of ${result.name} from ${currency} is: $ ${value}</p>

<p>Last hour: ${result.percent\_change\_1h} %</p>

<p>Last Day: ${result.percent\_change\_24h} %</p>

<p>Last 7 Days: ${result.percent\_change\_7d} %</p>

</div>

</div>

`;

// After 3 seconds print the result and hide spinner

setTimeout(() => {

// Insert HTML Template

const divResult = document.getElementById('result');

divResult.innerHTML = templateHTML;

// Hide Spinner

document.querySelector('.spinner img').remove();

}, 3000 );

}

// Prints the spinner

showSpinner() {

const spinnerGif = document.createElement('img');

spinnerGif.src = 'img/spinner.gif';

document.querySelector('.spinner').appendChild(spinnerGif);

}

}

app.js

// Instanciate both classes

const cryptoAPI = new CryptoAPI();

const ui = new UI();

// Get the form

const form = document.getElementById('form');

// Execute form when submitted

form.addEventListener('submit',(e) => {

e.preventDefault();

// read currency

const currencySelect = document.getElementById('currency').value;

// read cryptocurrency

const cryptoCurrencySelect = document.getElementById('cryptocurrency').value;

//console.log(currencySelect + ':' + cryptoCurrencySelect);

if(currencySelect === '' || cryptoCurrencySelect === ''){

// Some data is missing print a message

ui.printMessage('All fields are mandatory', 'deep-orange darken-4 card-panel');

} else {

cryptoAPI.queryAPI(currencySelect, cryptoCurrencySelect)

.then(data => {

ui.displayResult( data.result[0], currencySelect.toLowerCase() );

})

}

});

 PROJECT: EVENTS WITH EVENT BRITE API

// eventbrite.js

class EventBrite {

// Constructor when instanciate

constructor() {

this.token\_auth = '';

this.orderby = 'date';

}

// Get the events from API

async queryAPI(eventName, category) {

const eventsResponse = await fetch(`https://www.eventbriteapi.com/v3/events/search/?q=${eventName}&categories=${category}&sort\_by=${this.orderby}&token=${this.token\_auth}`);

// Wait for response, then return as json

const events = await eventsResponse.json();

return {

events

}

}

// get categories from API

async getCategoriesAPI() {

// Query the API

const categoriesResponse = await fetch(`https://www.eventbriteapi.com/v3/categories/?&token=${this.token\_auth}`);

// Wait for response and return as JSON

const categories = await categoriesResponse.json();

return {

categories

}

}

}

// ui.js

class UI {

constructor(){

// App Initialization

this.init();

}

// Method when the app starts

init() {

// Display categories on <select>

this.printCategories();

// Select the results

this.result = document.getElementById('result');

}

// Prints the categories in the <select>

printCategories() {

const categoriesList = eventbrite.getCategoriesAPI()

.then(categories => {

const categoriesList = categories.categories.categories;

const categoriesSelect = document.querySelector('select#category');

categoriesList.forEach(category => {

// Create Options

const option = document.createElement('option');

option.value = category.id;

option.appendChild(document.createTextNode(category.name\_localized));

// Append to <select>

categoriesSelect.appendChild(option);

});

})

}

// Display events from the API

displayEvents(events) {

// Read events and assign into a variable

const eventList = events.events;

// Build the Template

let htmlTemplate = '';

// Loop events and print the result

eventList.forEach(eventInfo => {

this.result.innerHTML += `

<div class="col-md-4 mb-4">

<div class="card">

<div class="card-body">

<img class="img-fluid mb-2" src="${eventInfo.logo !== null ? eventInfo.logo.url : '' }">

</div>

<div class="card-body">

<div class="card-text">

<h2 class="text-center">${eventInfo.name.text}</h2>

<p class="lead texto-info">Event Information:</p>

<p>${eventInfo.description.text.substring(0,200) } ... </p>

<span class="badge badge-primary">Capacity: ${eventInfo.capacity}</span>

<span class="badge badge-secondary">Date & Time: ${eventInfo.start.local}</span>

<a href="${eventInfo.url}" target="\_blank" class="btn btn-primary btn-block mt-4">Get Tickets</a>

</div>

</div>

</div>

</div>`;

});

}

// clear the previous results

clearResults() {

this.result.innerHTML ='';

}

// Remove the message

removeMessage() {

const alert = document.querySelector('.alert');

if(alert) {

alert.remove();

}

}

// Displays a message

printMessage(message, className) {

this.limpiarmessage();

const div = document.createElement('div');

div.className = className;

// Add Text

div.appendChild(document.createTextNode(message));

// insert into the search form

const searchDiv = document.querySelector('#search-events');

searchDiv.appendChild(div);

// Remove alert after 3 seconds

setTimeout(() => {

this.removeMessage();

},3000);

}

}

// app.js

// Instanciate Both Classes

const eventbrite = new EventBrite();

const ui = new UI();

// Listener for the submit button

document.getElementById('submitBtn').addEventListener('click', (e) => {

e.preventDefault();

// get values from form

const eventName = document.getElementById('event-name').value;

const category = document.getElementById('category').value;

// console.log(eventName + ' ' + category);

// Check something is in the input

if(eventName !== '') {

// into the console

// console.log(eventName);

// console.log('success');

eventbrite.queryAPI(eventName, category)

.then(data => {

// console.log(data.events);

if(data.events.events.length > 0) {

// Print the Events in case there're

ui.clearResults();

ui.displayEvents(data.events);

} else {

// There're no results

ui.printMessage('No Results Found', 'alert alert-danger mt-4');

}

})

} else {

// Print alert

ui.printMessage('Add an Event Name or City', 'alert alert-danger mt-4');

// console.log('failed');

}

});

 TRY CATCH

// When a function doesn't exists..

try {

something();

} catch (error) {

console.log(error);

} finally {

console.log('It will execute anyways!');

}

// Function that does exists

function getClients() {

console.log('Download...');

setTimeout(function() {

console.log('Complete');

}, 3000);

}

getClients();

 DESTRUCTURING

// Destructuring

// Destructuring will extract values from a javascript object

// This code has the deavantage that if you have multiple properties you have to create a bunch of variables

// Example using normal javascript

const client = {

name : 'Alexa',

membership: 'Premium'

}

let name = client.name,

membership = client.membership;

console.log(name);

console.log(membership);

// Destructuring

const client = {

name : 'Alexa',

membership: 'Premium'

}

// Assignt the variables

let {name, membership} = client;

console.log(name);

console.log(membership);

// Object values

const client = {

name : 'Alexa',

membership: 'Premium'

};

name = 'Mary',

membership = 'Platinum';

({name, membership} = client);

console.log(name);

console.log(membership);

// Extract object that's inside another object...

const client = {

membership: 'Premium',

name : 'Paul',

data: {

clientLocation: {

city: 'Mexico',

country: 'Mexico'

},

acount: {

memberSince: '10-12-2012',

balance: 4000

}

}

};

// Read data from object

let { data: {clientLocation}} = client;

console.log(clientLocation.city);

console.log(clientLocation.country);

let { data: {account}} = client;

console.log(account.memberSince);

console.log(account.balance);

// Default values

let client = {

name : 'Peter',

membership : 'Premium',

balance : 3000

};

let {name, membership, balance = 0} = client;

console.log(name);

console.log(membership);

console.log(balance);

// Destructuring an array

let cities = ['London', 'New York', 'Madrid', 'Paris'];

const [

firstCity,

secondCity

] = cities;

console.log(firstCity);

console.log(secondCity);

// Add an space to skip that value

const [ , , , paris] = cities;

console.log(paris);

// More in Depth example

let client = {

membership: 'Premium',

balance: 30000,

data: {

name: 'Paul',

lastName: 'Banks',

living: {

city: 'Mexico',

country: 'Mexico'

}

},

lastMovements: ['12-03-2018', '10-03-2018', '08-03-2018']

};

let {

data: {living},

lastMovements: [first]

} = client;

console.log(living);

console.log(living.city);

console.log(first);

//Destructuring functions old method

function reservation(completo, options) {

options = options || {};

let payment = options.paymentMethod,

amount = options.amount,

days = options.days;

console.log(payment);

console.log(amount);

console.log(days);

}

//2do argument are the options

reservation(

true,

{

paymentMethod: 'creditCard',

amount: 2000,

days: 3

}

);

// Destructuring functions new method

function reservation(complete, options) {

let {paymentMethod, amount, days} = options;

console.log(paymentMethod);

console.log(amount);

console.log(days);

}

// Destructuring functions with default parameters

function reservation(cancel,

{

paymentMethod = 'cash',

amount = 0,

days = 0

} = {}

) {

console.log(paymentMethod);

console.log(amount);

console.log(days);

}

//2nd argument are the options as an object

reservation(

false,

{

paymentMethod: 'card',

amount: 2000,

days: 3

}

);

 SYMBOLS

// Symbol

// Symbols are new in ES6, They will create a unique value in JavaScript

// Creating a symbol

const sym = Symbol();

const sym2 = Symbol('sym');

// Symbols are always different

// console.log( Symbol() === Symbol() );

// Unique object keys

let firstName = Symbol();

let lastName = Symbol();

// create empty object

let person = {}

// Esto no va a servir

persona.datos;

// Attach symbol into Object

person[firstName] = 'Juan';

person[lastName] = 'De la torre';

// Standard properties

person.membership = 'Premium';

person.amount = 500;

console.log(person);

console.log(person[firstName]);

// You cannot access a symbol in a for loop

for(let i in person) {

console.log(`${i} : ${person[i]}`);

}

// You can also a symbol description

/\*

let clientName = Symbol('Client Name');

let client = {};

client[clientName] = 'Peter';

// Test

console.log(client);

console.log(client[clientName]);

console.log(clientName);

\*/

 SETS

// CReating a set

// A set is going to a set values without duplicates

let shoppingCart = new Set();

shoppingCart.add('Shirt');

shoppingCart.add('Album #1');

shoppingCart.add('Album #2');

shoppingCart.add('Album #3');

shoppingCart.add('Album #3');

shoppingCart.add('Guitar');

console.log(shoppingCart.size);

// In an aray

let numbers = new Set([1,2,3,4,5,6,7,3,3,3,3]);

console.log(numbers.size);

let shoppingCart = new Set();

shoppingCart.add('Shirt');

shoppingCart.add('Album #1');

shoppingCart.add('Album #2');

shoppingCart.add('Album #3');

shoppingCart.add('Album #3');

shoppingCart.add('Guitar');

console.log(shoppingCart.size);

// Checking a value exists in the set

console.log( shoppingCart.has('Shirt') );

// Delete item from set

console.log( shoppingCart.delete('Shirt') );

console.log(shoppingCart);

// Clean a set

shoppingCart.clear();

console.log(shoppingCart);

// Foreach in a set

shoppingCart.forEach(product => {

console.log(product);

})

// Foreach in a set

shoppingCart.forEach((product, index, isPartOf) => {

console.log(`${index} : ${product}`);

console.log(isPartOf === shoppingCart);

})

// Convert a SET Into an array

const shoppingCartArray = [...shoppingCart];

console.log(shoppingCartArray);

 MAPS

// MAPS

// Ordered lists with a key and a value, can hold any value

// cualquier tipo.

let client = new Map();

client.set('name', 'Karen');

client.set('membership', 'Premium');

client.set('balance', 3000);

console.log(client);

// access the values

console.log(client.get('name'));

console.log(client.get('membership'));

console.log(client.get('balance'));

// Map Methods

// Map Size

console.log(client.size);

// Check if value exists

console.log(client.has('membership'));

console.log(client.get('membership'));

// Delete

client.delete('name');

console.log(client.has('name'));

console.log(client.get('name'));

console.log(client.size);

// Delete Map

client.clear();

console.log(client);

// Default values into map

const patient = new Map([['name', 'patient Name'], ['room', 'not defined']]);

patient.set('name', 'Paul');

// patient.set('room', 400);

console.log(patient);

// For each into map

client.forEach((data, index) => {

// console.log(data);

console.log(`${index}: ${data}`);

});

 ITERATOR

// Iterators

function createIterator(cart) {

let i = 0;

return {

nextProduct: function() {

let end = (i >= cart.length);

let value = !end ? cart[i++] : undefined;

return {

end: end,

value: value

};

}

};

}

const cart = ['Product 1', 'Product 2', 'Product 3', 'Product 4'];

const shoppingCart = createIterator(cart);

console.log(shoppingCart.nextProduct() );

console.log(shoppingCart.nextProduct() );

console.log(shoppingCart.nextProduct() );

console.log(shoppingCart.nextProduct() );

console.log(shoppingCart.nextProduct() );

 GENERATORS

// Instead of creating iterators by hand you can use a generator

// You indicate a generator with the asterisk before the function name

// generator

function \*createGenerator() {

// Yield is a new keyword in ES6

yield 1;

yield 'Name of th eperson';

yield 3 + 3;

yield true;

}

// They're executed as standard functions but the return value is a iterator

const iterator = createGenerator();

console.log(iterator.next().value);

console.log(iterator.next().value);

console.log(iterator.next().value);

console.log(iterator.next().value);

console.log(iterator.next().value);

// Create a second generator

function \*newGenerator(cart) {

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

yield cart[i];

}

}

// Shopping cart

const cart = ['Product 1', 'Product 2', 'Product 3', 'Product 4'];

// Loop iterator

let iterator = newGenerator(cart);

console.log(iterator.next() );

console.log(iterator.next() );

console.log(iterator.next() );

console.log(iterator.next() );

console.log(iterator.next() );

 OTHER ITERATORS

// Entries Iterador

const cities = ['London', 'New York', 'Madrid', 'Paris'];

const orders = new Set([123, 231, 131, 102]);

const data = new Map();

data.set('learning', 'JavaScript');

data.set('JSisGreat', true);

// entries

for( let entry of cities.entries() ){

console.log(entry);

}

// entries

for( let entry of orders.entries() ){

console.log(entry);

}

// entries

for( let entry of datos.entries() ){

console.log(entry);

}

// Values iterator

// values

for(let value of cities.values()) {

console.log(value);

}

// values

for( let value of orders.values() ){

console.log(value);

}

// values

for( let value of datos.values() ){

console.log(value);

}

// Keys iterator

// keys

for(let keys of cities.keys() ) {

console.log(keys);

}

// keys

for( let keys of orders.keys() ){

console.log(keys);

}

// keys

for( let keys of datos.keys() ){

console.log(keys);

}

// Default

for(let city of cities) {

console.log(city);

}

for( let order of orders){

console.log(order);

}

for( let info of data){

console.log(info);

}

// Iterate an string

const message = 'Learning JavaScript';

// Old WAY

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

console.log(message[i]);

}

// new way

for( let char of message) {

console.log(char);

}

// Iterate a node list

const anchors = document.getElementsByTagName('a');

for (let anchor of anchors) {

console.log(anchor.href);

}

 REGULAR EXPRESSIONS

/\*

\d Any Number

\w Any number or letter

\s any white space (space, tab or line break)

\D character that's not a digit

\W not alphanumeric character

\S any character but a whitespace

. any character but a line break

\*/

// You can create a regular expression with 2 different methods

const exp1 = new RegExp('/abc/');

const exp2 = /abc/;

// Check if it includes 1992...

console.log(/[0123456789]/.test('1992'));

// Same as above

console.log(/[0-9]/.test('1992'));

// a date following the pattern... 20-10-2018

const dateRegExp = /\d\d-\d\d-\d\d\d\d/ ;

const date = '20-10-2018';

console.log( dateRegExp.test(date) );

//Check the time: 12:00

const TimeRegExp = /\d\d:\d\d/;

const time = '18:03';

console.log( TimeRegExp.test(time) );

// Check time: 08:30 PM

const TimeRegExpComplete = /\d\d:\d\d \D\D/;

const completeTime = '08:30 PM';

const completeTime = '08:30 10';

console.log(TimeRegExpComplete.test(completeTime));

// Check for multiple numbers

const repeteadedNumber = /\d+/;

const digits = 1234;

console.log(repeteadedNumber.test(digits));

// Deny the expression ^

const denyRegExp = /[^0-9]/;

const numbers = 12345;

console.log(denyRegExp.test(numbers));

// The Syntax {1,2} represents that a character can appear between 1 & 2 times

let expReg = /\d{1,2}-\d{1,2}-\d{4}/

const date = '10-20-2018';

const date2 = '10-2-2018';

const date3 = '1-20-2018';

console.log(expReg.test(date));

console.log(expReg.test(date2));

console.log(expReg.test(date3));

// Check for letters or numbers

const messageRegExp = /\w+/;

let message ;

message = 'Test Message';

message = ' ';

message = 1234;

console.log(messageRegExp.test(message));

// Check for numbers

const checkNumbers = /\d+/;

const numbers = 1234;

console.log(checkNumbers.test(numbers));

// Check for only numbers

const checkForNumbers = /([0-9])\w+/;

const numbers = 1234;

console.log(checkForNumbers.test(numbers));

// Check for Uppercase letters only

const uppercaseRegExp = /([A-Z])\w+/;

let message;

message = 'UPPERCASE;

message = 1234;

message = 'message';

console.log(uppercaseRegExp.test(message));

// Check for lowercase only

const lowercaseRegExp = /([a-z])\w+/;

let message;

message = 'lowercase';

message = 1234;

message = 'MESSAGE';

console.log(lowercaseRegExp.test(message));

// A REALLY COMPLEX REGULAR EXPRESSION

const expRegMail = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)\*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;

const email = 'email@email.com';

console.log(expRegMail.test(email));