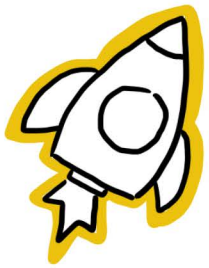
Curso Profesional de

JavaScript JS

códigofacilito

¿QUE ES JAVASCRIPT?

NO NECESITAS CONOCIMIENTOS PREVIOS



INICIAMOS DESDE CERO

JS

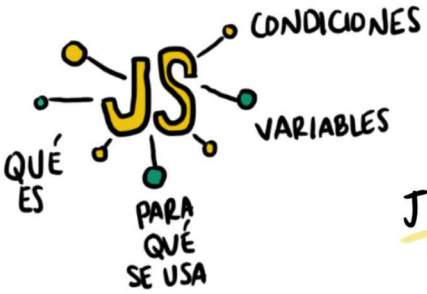
LENGUAJE DE PROGRAMACIÓN MUY POPULAR

MUY USADO POR EMPRESAS

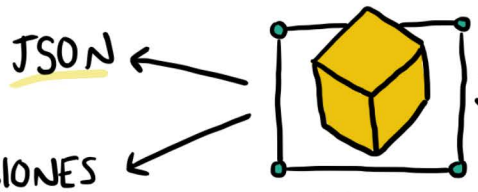
INTERACTIVIDAD

APPS MULTIPLATAFORMA

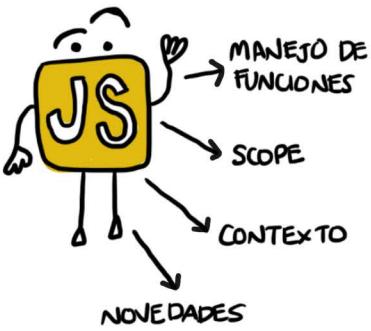
APPS PARA DIFERENTES S.O.

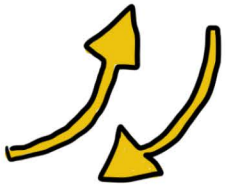


PROGRAMACIÓN

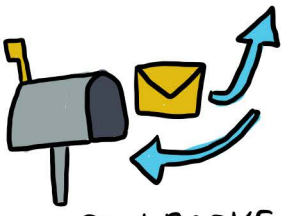


ORIENTADA A OBJETOS A TRAVÉS DE PROTOTIPOS





PROGRAMACIÓN ASÍNCRONA



CALLBACKS Y PROMESAS

BONUS

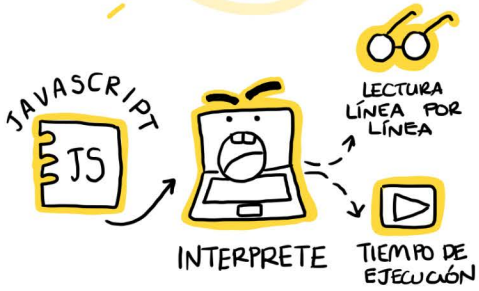
NOVEDADES DEL LENGUAJE

Curso Profesional de JavaScript

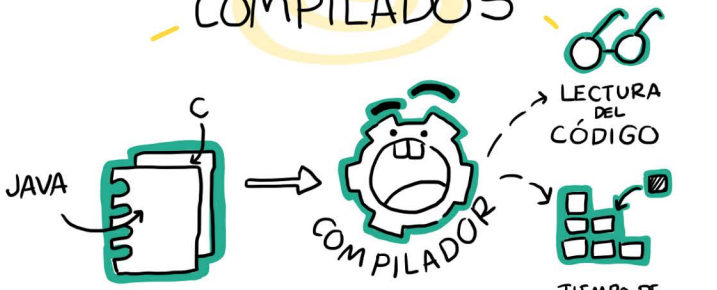
códigofacilito

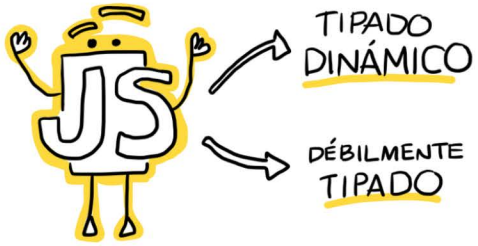
COMO ES JAVASCRIPT TECNICAMENTE

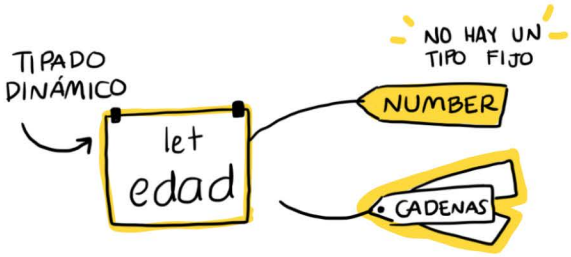
LENGUAJE INTERPRETADO



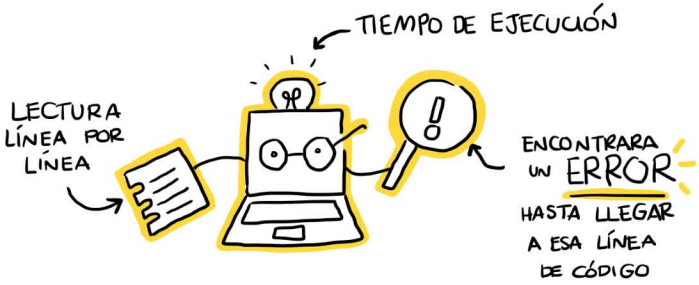
LENGUAJES COMPILADOS



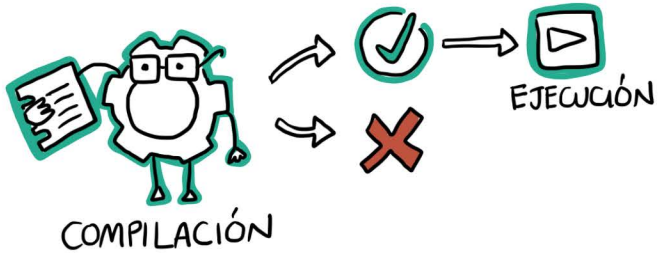


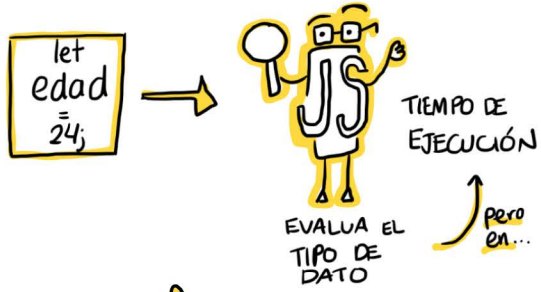


INTÉRPRETE



COMPILADOR





aA

CASE SENSITIVE

Curso Profesional de

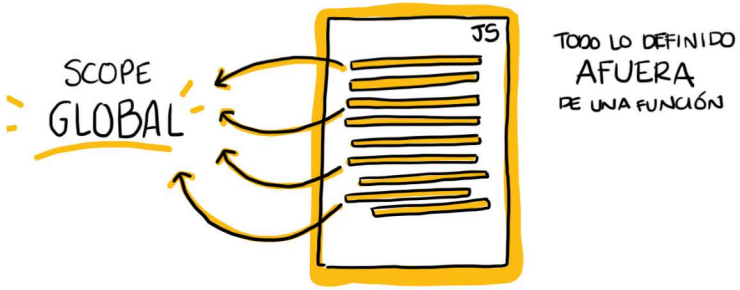
JavaScript

códigofacilito

¿QUE ES EL SCOPE?

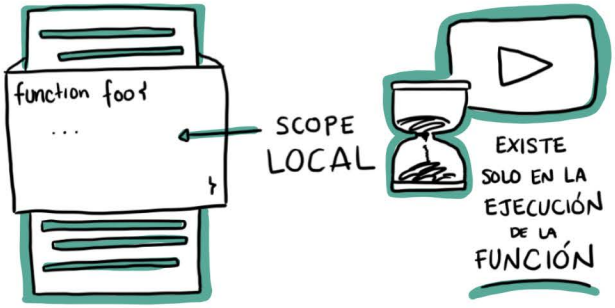
LIMITAR EL ALCANCE A LO QUE LE ES ÚTIL

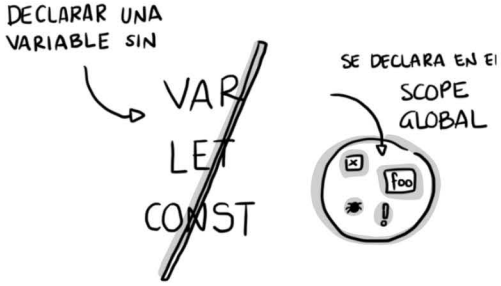
SCOPE GLOBAL

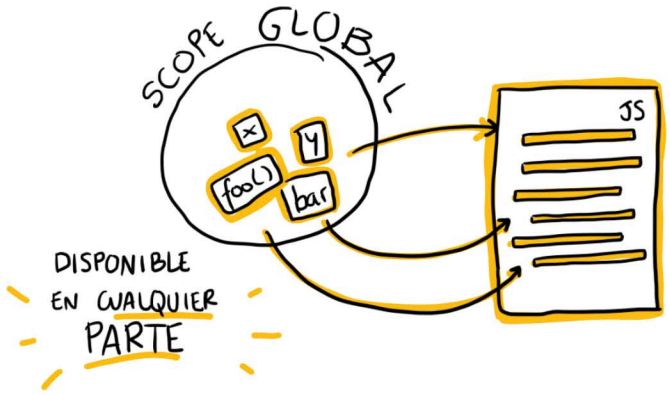


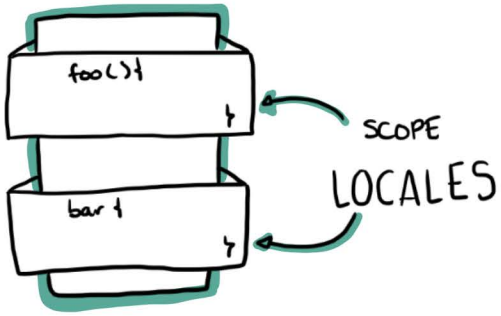
TODO LO DEFINIDO AFUERA DE UNA FUNCIÓN

SCOPE LOCAL









Curso Profesional de

JavaScript

códigofacilito

CUÁNDO USAR LET, CONST y VAR

var ← ALCANCE EN LA FUNCIÓN MAS PRÓXIMA

let, const → ALCANCE EN EL BLOQUE MÁS PRÓXIMO

VARIABLE LET → RECOMENDABLE

CONSTANTE CONST

SCOPE GLOBAL VAR

VAR
ALCANCE EN LA FUNCION A TRAVÉS DEL SCOPE

```
function mayor_de_edad(edad){
    if(edad <= 18){
        var resultado = "Eres mayor de edad";
    } else {
        let resultado = "Eres menor de edad";
    }

    console.log(resultado);
}

mayor_de_edad();
```

LET
SOLO EXISTE EN ESTE BLOQUE

Curso Profesional de

JavaScript

códigofacilito

EL CONTEXTO

this

CONTEXTO DEL OBJETO EN EJECUCIÓN

CAMBIO DE PARADIGMA

IMPLEMENTAR PATRONES DE DISEÑO

CONTEXTO

FLEXIBLE

DINÁMICO

¿CÓMO CAMBIA DE VALOR?
¿COMO CONSERVAR SU VALOR?

SCOPE GLOBAL
JS

this
CONTEXTO

WINDOW
OBJETO GLOBAL

```
let objeto = {
    demo: function(){
        console.log(this);
    }
}

objeto.demo();
```

REFERENCIA AL OBJETO EN EJECUCIÓN

SCOPE LOCAL

SCOPE LOCAL

this
CONTEXTO LOCAL

THIS
TOMA EL VALOR DEL OBJETO GLOBAL

```
let objeto = {
    demo: function(){
        console.log(this);
    }
}

let executor = {
    execute: function(f){ f(); }
}

executor.execute(objeto.demo);
```

FUNCIÓN COMO ARGUMENTO

EJECUCIÓN

Curso Profesional de JavaScript

códigofacilito

ARROW FUNCTIONS

LO MISMO PERO DIFERENTE SINTÁXIS

```
function suma(a,b){
    return a + b;
}

let suma = (a, b) => a + b ;
```

ARGUMENTOS

CUERPO

ARROW FUNCTIONS

- ✓ ELEGANTE
- ✓ REDUCIR LINEAS
- ✓ MANEJO DEL CONTEXTO

Curso Profesional de JavaScript

códigofacilito

ARROW FUNCTIONS Y EL CONTEXTO

```
let tutor = {
    nombre: "Uriel",
    apellido: "Hernandez",
    nombreCompleto: function(){
        setTimeout(function(){
            console.log(this.nombre + " " + this.apellido)
        }, 1000);
    }
}
```

THIS
CONTEXTO
DE LA FUNCIÓN

NO EXISTEN
DENTRO

PODEMOS
USAR

()=>

FAT ARROW
PARA CONSERVAR
EL CONTEXTO DEL OBJETO

SINTÁXIS
CORTA

NO
MODIFICA
"THIS"

Curso Profesional de

JavaScript

códigofacilito

CALL APPLY AND BIND

MÉTODOS PARA ASIGNAR VALOR AL CONTEXTO THIS

- CALL
- APPLY
- BIND

EJECUTAN LA FUNCIÓN INMEDIATA

CONSTRUYE UNA FUNCIÓN PARA EJECUTARSE DESPUÉS

① CALL

```
function.call(arg1, arg2, ...)
```

valor de THIS

argumentos de la función

```
function executor(function){
    function.call(tutor);
}

let tutor = {
    nombre: "Uriel",
    apellido: "Hernandez",
    nombreCompleto: function(){
        console.log(this.nombre + " " + this.apellido);
    }
}

executor(tutor.nombreCompleto);
```

ES UN MÉTODO DE UNA FUNCIÓN

ESTE VALOR SE ASIGNARÁ A THIS

SE ENVÍA ESTA FUNCIÓN COMO ARGUMENTO

② APPLY

SIMILAR A CALL

```
function saluda(nombre){ console.log("hola " + nombre); }

saluda.apply(window, ["Uriel"]);
```

VALOR A ASIGNAR A THIS

ARREGLO DE ARGUMENTOS DE LA FUNCIÓN A EJECUTAR

③ BIND

```
function executor(function){
    function();
}

executor(tutor.nombreCompleto.bind(tutor));
```

VALOR QUE SE ASIGNARÁ A THIS

Curso Profesional de

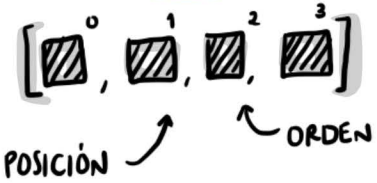
JavaScript

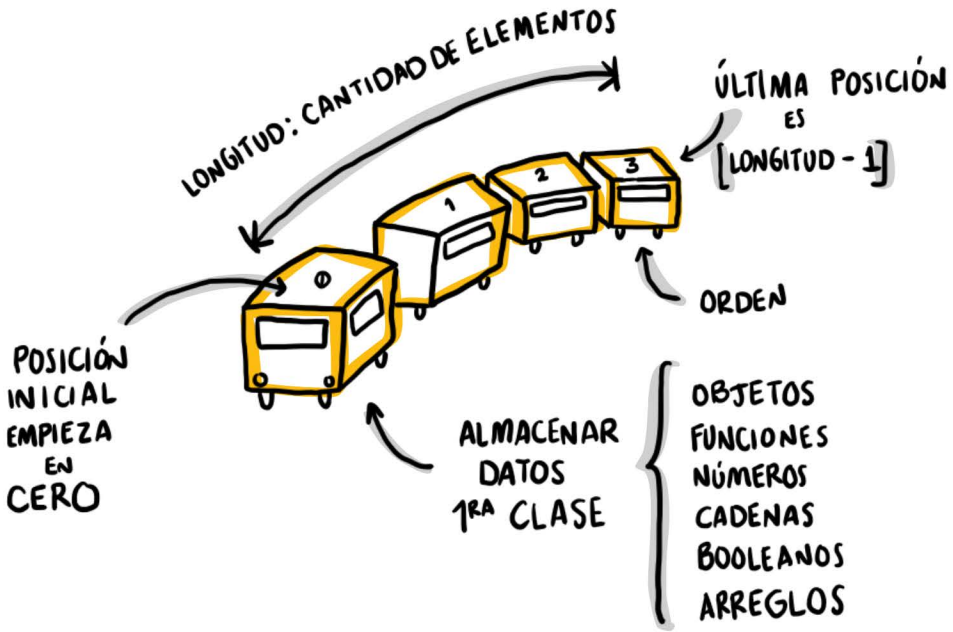
códigofacilito

ARREGLOS Y SUS USOS

ARREGLO

COLECCIÓN DE DATOS AGRUPADOS EN UNA SOLA ESTRUCTURA





SE PUEDE DECLARAR UN ARREGLO PARA ALMACENAR ESTOS NÚMEROS

```
let num1 = 1;
let num2 = 2;
let num3 = 3;

let numeros = [1,2,3]:

let arreglo = [10, 15, 14]
```

LONGITUD: 3

0 1 2 POSICIONES

Curso Profesional de

JavaScript

códigofacilito

MÉTODOS Y OPERACIONES SOBRE ARREGLOS

OPERACIONES SOBRE ARREGLOS → FOR EACH, FILTER, FIND, MAP

ARREGLO.MÉTODO(ARG)

- MÉTODO: forEach, filter, map
- ARG: FUNCIÓN COMO ARGUMENTO

① FOR EACH

```
let arreglo = ['ruby', 'python', 'java', 'javascript']

arreglo.forEach(function(elemento){
    console.log(elemento);
})
```

- RECORRE 1 a 1 CADA ELEMENTO DEL ARREGLO
- FUNCIÓN QUE SE EJECUTA POR CADA ELEMENTO

② FILTER

```
let numeros = [14, 23, 123, 2]

numeros.filter(element => element != 2);
```

- CONDICIÓN QUE DEBE CUMPLIR CADA ELEMENTO
- GENERA UN NUEVO ARREGLO → [14, 23, 123]

③ FIND

```
let arreglo = ['ruby', 'python', 'java', 'javaScript']

let el = arreglo.find(el => el == 'javascript')
```

- CONDICIÓN
- REGRESA EL PRIMER VALOR QUE CUMPLA LA CONDICIÓN

④ MAP

```
let numeros = [2, 35, 6, 20];

let cuadrados = numeros.map(numero => numero * numero);
```

- FUNCIÓN APLICADA A CADA ELEMENTO
- GENERA UN NUEVO ARREGLO → [4, 1225, 36, 400]

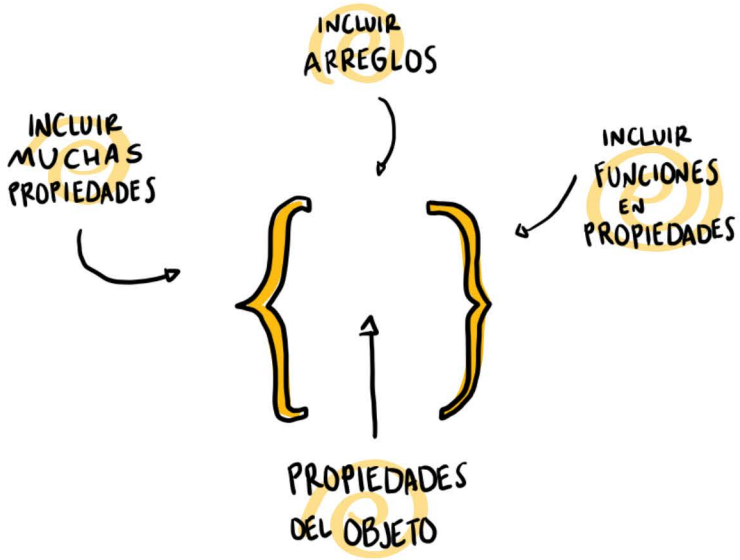
Curso Profesional de

JavaScript

códigofacilito

DECLARAR UN OBJETO CON JSON

JAVASCRIPT OBJECT NOTATION **JSON**



REASIGNACIÓN

curso.titulo = "otro titulo"

```
let curso = {
    titulo: 'Curso Profesional de JS',
    duracion: 2.2,
    formato: 'video',
    bloque: ['Introduccion', 'Funciones']
}
```

CONSULTAR OBJETO

curso.titulo (titulo: PROPIEDAD)

curso['titulo'] (titulo: PROPIEDAD)

curso.____() ← EJECUCIÓN

____: MÉTODO

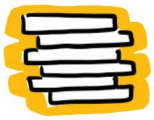
Curso Profesional de

JavaScript

códigofacilito

FUNCIONES CONSTRUCTORAS

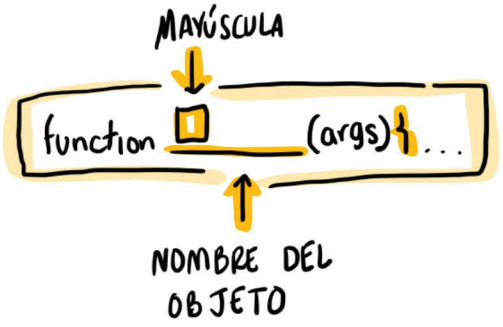
FUNCIONES PARA DECLARAR UN OBJETO



LA FUNCIÓN CONTIENE LA ESTRUCTURA DEL OBJETO

JSON

ES UNA ALTERNATIVA A CREAR OBJETOS JSON



NOMBRE DEL OBJETO

ARGUMENTOS

PROPIEDADES

MÉTODOS

```
function Curso(titulo){
    this.titulo = titulo;

    this.inscribir = function(usuario){
        console.log(usuario + "Se ha inscrito");
    }
}

let cursoJavaScript = new Curso("Curso Profesional de JS");
let cursoRuby = new Curso("Curso de Ruby");
```

OBJETOS DIFERENTES

INSTANCIACIÓN

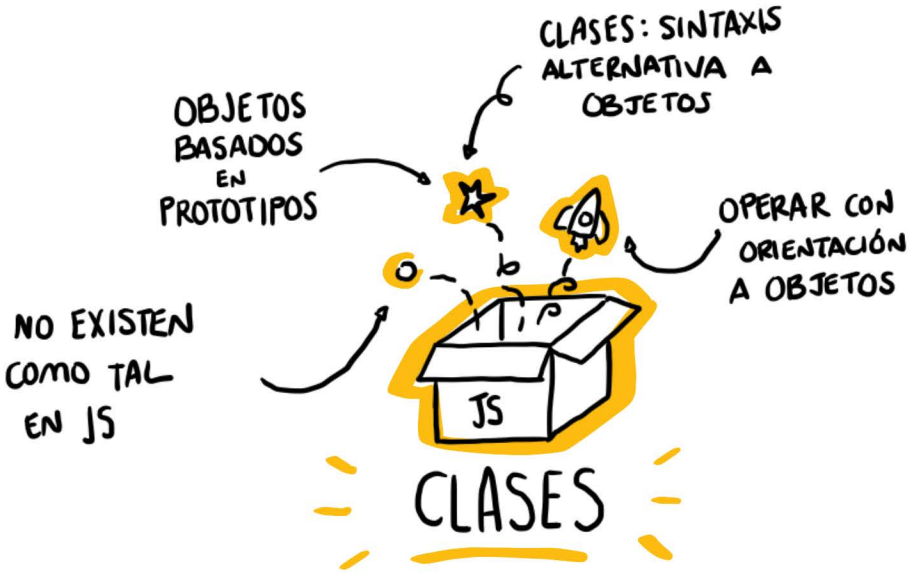
new + función

Curso Profesional de

JavaScript JS

CLASES

códigofacilito



DECLARACIÓN

```
class Curso{}

let Curso = class{}

let Usuario = class Usuario{}
```

CLASS DECLARATION

CLASS EXPRESSIONS

PALABRA RESERVADA

CONSTRUCTOR DE LA CLASE

PROPIEDAD

MÉTODO

INSTANCIACIÓN

PROPIEDAD

EJECUTAR MÉTODO

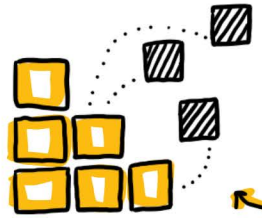
```
class Curso{
    constructor(titulo){
        this.titulo = titulo;
    }

    inscribir(usuario){
        console.log(usuario + "se ha inscrito");
    }
}

let jsCurso = new Curso("Curso Profesional de JS");

console.log(jsCurso.titulo);

jsCurso.inscribir("Uriel");
```

Curso Profesional de JavaScript

códigofacilito

CONSTRUCTOR

CONSTRUCTOR:



INSTANCIAR OBJETOS

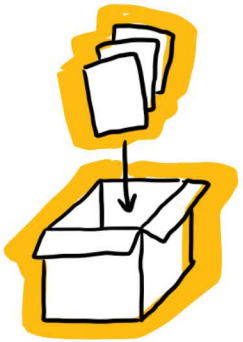
MÉTODO PARA CREAR UN NUEVO OBJETO

PALABRA RESERVADA

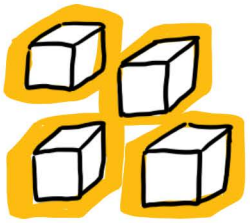
ARGUMENTOS PARA INSTANCIAR

```
class Curso{

    constructor(titulo, duracion, color='yellow'){
        this.titulo = titulo;
        this.duracion = duracion;
        this.color = color;
    }

}
```

ASIGNACIÓN



ASIGNAR VALORES INICIALES AL OBJETO



ENLISTAR LOS PARÁMETROS QUE EL OBJETO NECESITA AL CREARSE

ASIGNAR VALORES POR DEFECTO

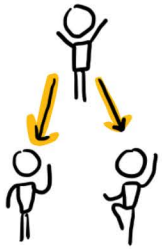
new _____ (args...)

CLASE

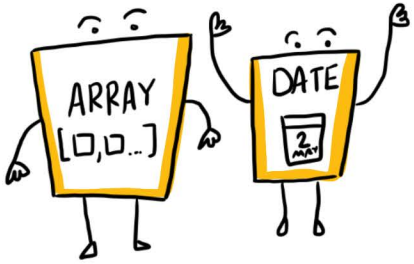
Curso Profesional de JavaScript

códigofacilito

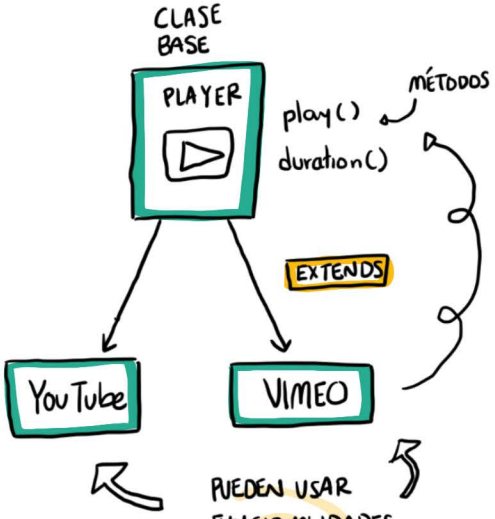
HERENCIA



REHUSAR FUNCIONALIDAD USANDO ORIENTACIÓN A OBJETOS



JS TIENE OBJETOS PREDEFINIDOS DE LOS CÚALES TAMBIÉN SE PUEDE HEREDAR



CLASE BASE

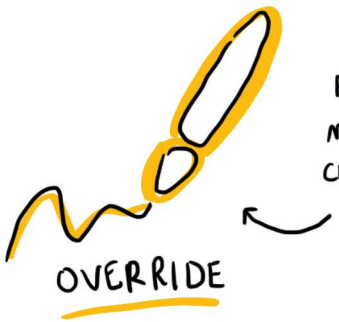
```
class User{
    saludar(){
        console.log("Hola usuario");
    }
}

class Admin extends User{
    saludar(){
        super.saludar();
        console.log("Hola, soy Admin");
    }
}

let admin = new Admin();

admin.saludar();
```

HERENCIA

MÉTODO DE CLASE BASE

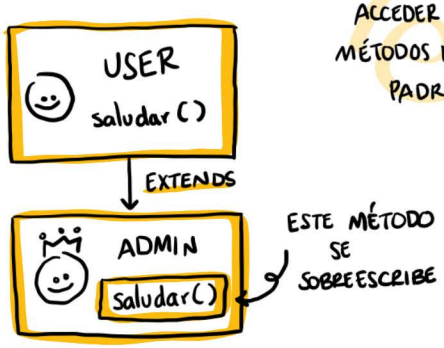
SUPER: OBJETO PARA ACCEDER A LOS MÉTODOS DE LA CLASE PADRE

MÉTODO DE CLASE QUE ESTA HEREDANDO



REESCRIBIR UN MÉTODO DE LA CLASE PADRE

OVERRIDE



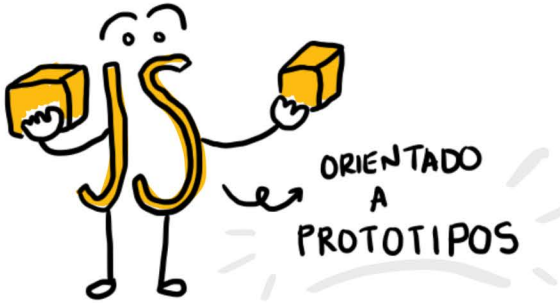
ESTE MÉTODO SE SOBREESCRIBE

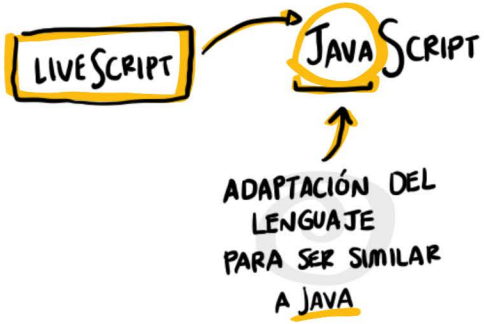
Curso Profesional de

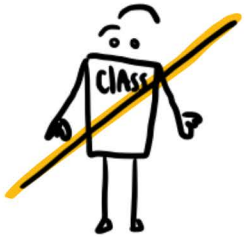
JavaScript

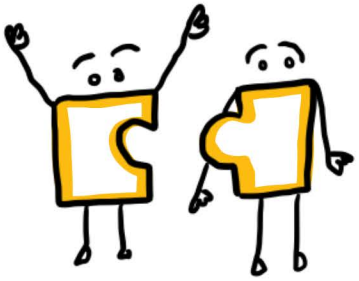
PROTOTIPOS

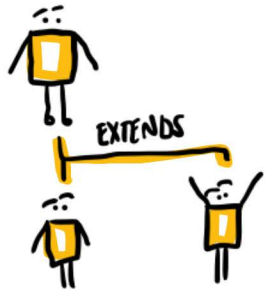
códigofacilito

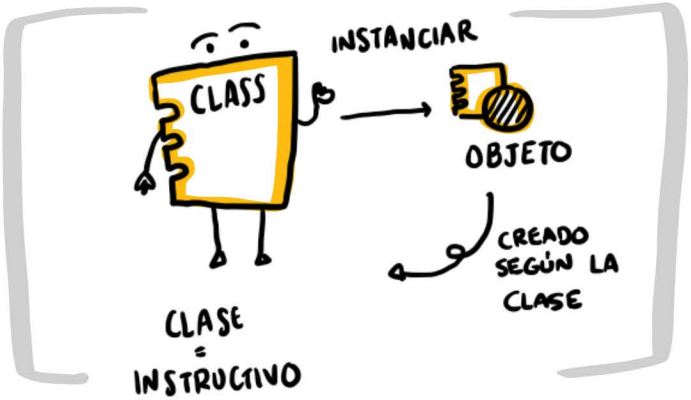


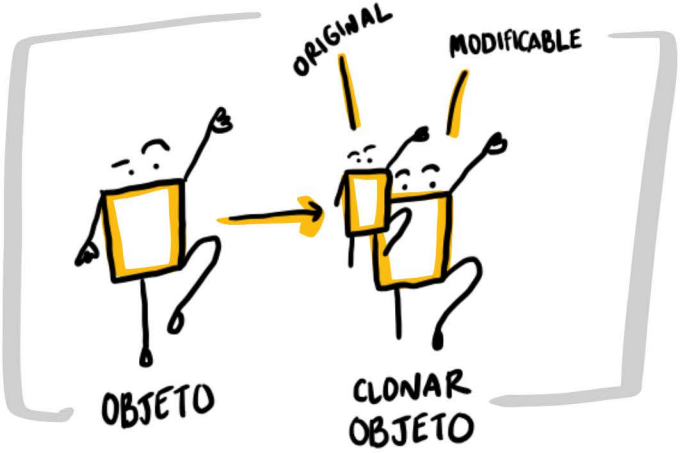








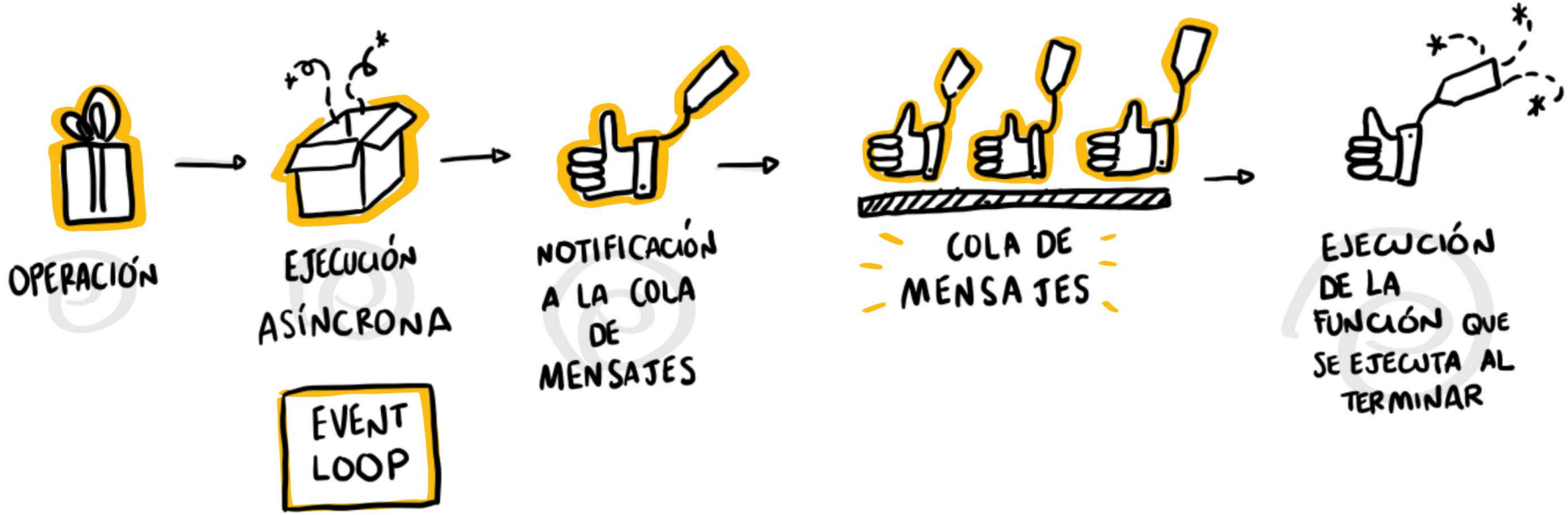




Curso Profesional de JavaScript

códigofacilito

PROGRAMACIÓN ASÍNCRONA



Curso Profesional de

JavaScript JS

PROMESAS

códigofacilito

PROMESAS:

ALTERNATIVA A MEJORAR PROCESOS ASÍNCRONOS

NO NECESITAS DE ALGUNA BIBLIOTECA EXTERNA

REQUEST PROMISE: NOS AYUDA A IMPLEMENTAR LA CONSULTA A UNA URL

INSTALAR PREVIAMENTE PARA ESTE EJEMPLO

```
$ npm install request-promise
```

NECESITAS NPM

BIBLIOTECA PREINSTALADA PARA REALIZAR PROMESAS

```
const rp = require('request-promise');

rp('http.//google.com').then(function(html){
    console.log("Terminó la petición");
}).catch(function(err){
    console.log(err);
})
```

PROMESA
OBJETO `PROMISE'

EJECUCIÓN SI LA PROMESA SE CUMPLE

MÉTODO

SE EJECUTA SI LA PROMESA FALLA

CALLBACKS:
- THEN
- CATCH

ESTADOS DE UNA PROMESA:

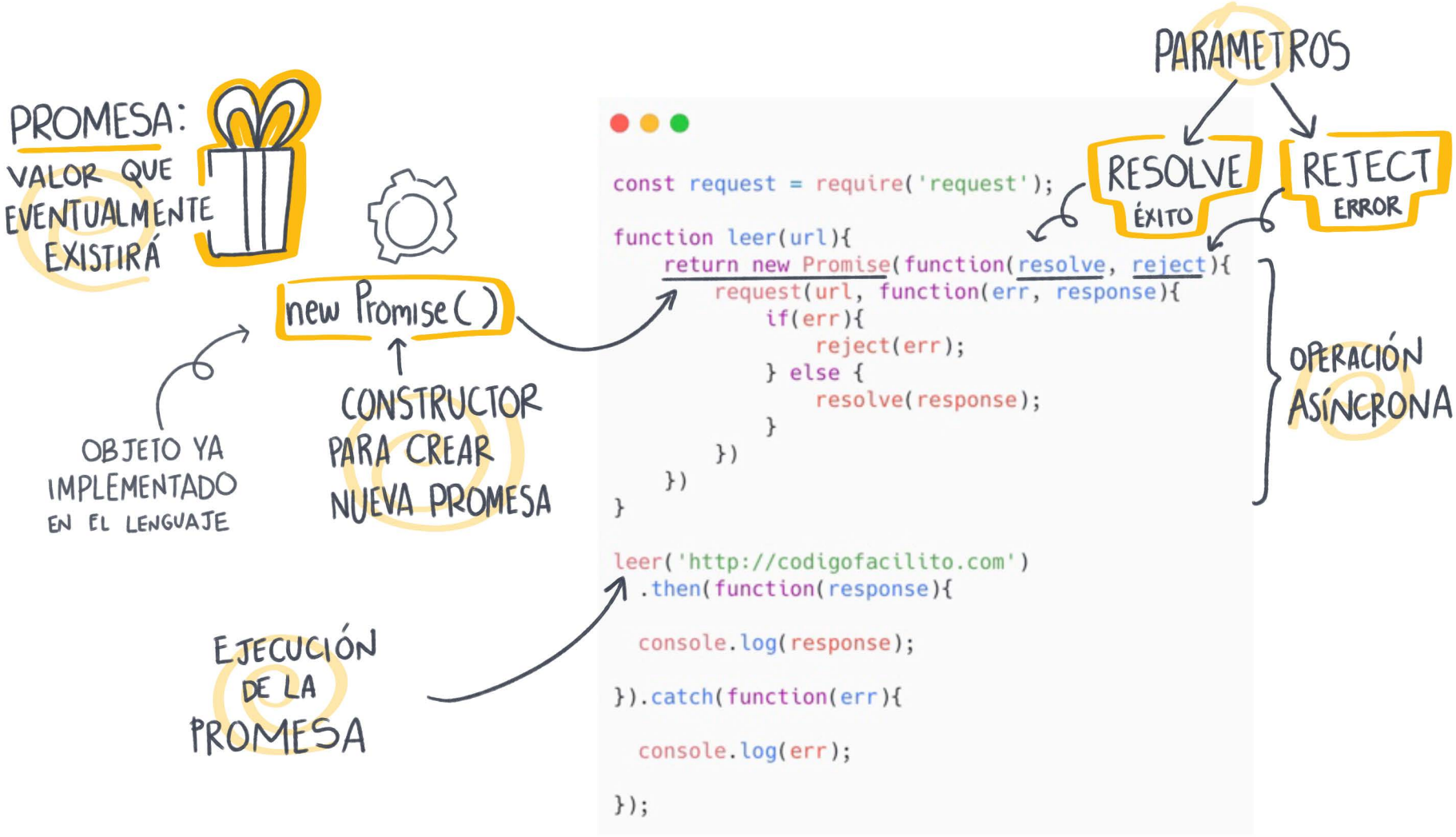
- FULLFILED → ÉXITO
- REJECTED → SIN ÉXITO
- PENDING → PENDIENTE
- SETLED → TERMINÓ

Curso Profesional de

JavaScript

códigofacilito

CREAR PROMESAS



Curso Profesional de

JavaScript

códigofacilito

RESOLVER MULTIPLES PROMESAS

PROMISE.ALL

MÉTODO PARA RESOLVER MULTIPLES OPERACIONES ASÍNCRONAS

```
let p1 = new Promise((resolve, reject) => setTimeout(resolve, 500, "Hola mundo"));

let  p2= new Promise((resolve, reject) => setTimeout(resolve, 600, "Segundo Hola mundo"));

let saluda = () => console.log("Hola a todos");

p1.then(function(){
    p2.then(function(){
        saluda();
    });
});

Promise.all([p1, p2]).then(resultado => {
 saluda();
});
```

P1 y P2

PROMESAS, OPERACIONES ASÍNCRONAS A EJECUTAR

PRIMERO SE EJECUTA P1, LUEGO P2

ES POSIBLE ANIDAR SU EJECUCIÓN

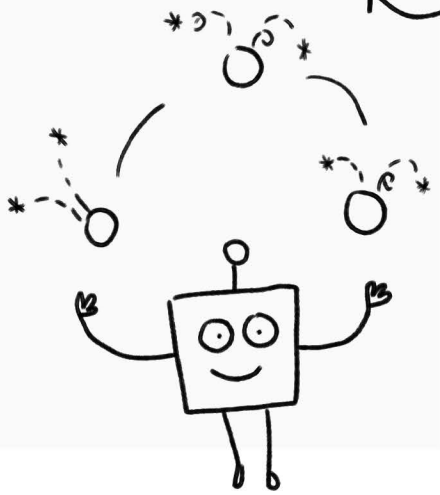
HACEN LO MISMO

SI TODAS SON EXITOSAS, AQUÍ SE CONTIENEN TODOS LOS RESULTADOS

MEJOR SINTAXIS

ARREGLO DE PROMESAS

FALLA SI AL MENOS FALLA UNA PROMESA



Curso Profesional de JavaScript

códigofacilito

ENCADENAR PROMESAS

CASO 1

OPERACIONES ASÍNCRONAS INDEPENDIENTES

CASO 2

SE RESUELVE ENCADENANDO

OPERACIONES DEPENDIENTES

NECESITAN VALORES DE ALGUNA PROMESA

PROMESAS

```
function calcular(){
    return new Promise(
      (resolve, reject) => { setTimeout(resolve, 400, 5); })
}


function segundoCalculo(numero){
    console.log(numero);

    return new Promise(
      (resolve, reject) => { setTimeout(resolve, 400, 'Segunda Promesa'); })
}

calcular().then(segundoCalculo).then(console.log);
```

RECIBE COMO ARGUMENTO EL RESULTADO DE LA PROMESA ANTERIOR

① EJECUCIÓN DE PRIMERA PROMESA

MÉTODO

SE ENVÍA UNA FUNCIÓN QUE SE EJECUTARA ③

② SE ENVÍA SEGUNDA PROMESA COMO ARGUMENTO

Curso Profesional de

JavaScript

códigofacilito

FUNCIONES ASÍNCRONAS

Curso Profesional de

JavaScript JS ASYNC

códigofacilito

async + function...

PUEDE SER SÍNCRONA O ASÍNCRONA

```
async function suma(valor1, valor2){
    return valor1 + valor2;
}

async function calcular(){
    return new Promise((resolve, reject) => { setTimeout(resolve, 400, 5)})
}
```

Curso Profesional de

JavaScript

códigofacilito

AWAIT

AWAIT + FUNCIÓN ASÍNCRONA

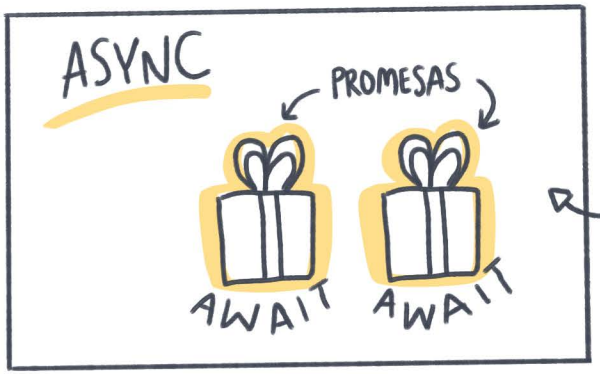
SÓLO SE PUEDE USAR DENTRO DE ASYNC

DECLARACIÓN ASYNC

RESUELVE LA OPERACIÓN ASÍNCRONA

```
(async function(){
    let valor1 = await new Promise((resolve, reject) => { setTimeout(resolve, 400, 5)});
    let valor2 = await new Promise((resolve, reject) => { setTimeout(resolve, 400, 4)});

    console.log(valor1 + valor2);
})();
```

ES POSIBLE TRABAJAR CON LOS VALORES



SE PUEDEN TRATAR COMO CUALQUIER OTRO OBJETO

Curso Profesional de

JavaScript

códigofacilito

MÓDULOS EN JAVASCRIPT