


## Prototipos

Son un mecanismo central que permite a los objetos compartir propiedades y métodos entre sí. Cada objeto en JavaScript tiene un prototipo asociado, que sirve como un modelo para el objeto y proporciona una cadena de búsqueda para propiedades y métodos que no se encuentran directamente en el objeto. Esta relación de prototipo facilita la herencia y la reutilización de código, lo que resulta en un código más eficiente y modular.

js

 Copiar código

```
// función constructora
function Persona (nombre, edad){
  this.nombre = nombre;
  this.edad = edad;
}

// Método fuera de la función constructora
Persona.prototype.caminar = function (){
  console.log("Debe caminar todos los Dias")
}


const Administrador = new Persona ("Raul Mesenes", 27);
const mecanico = new Persona ("Brayan Dias", 37);

console.log(Administrador);
console.log(mecanico);
```

## Herencia

La herencia en JavaScript permite que los objetos adquieran propiedades y métodos de otros objetos, facilitando la reutilización de código y la organización modular. Esto se logra a través de la cadena de prototipos, permitiendo la creación de relaciones jerárquicas entre objetos de manera dinámica y flexible.

js

 Copiar código

```
// Creamos nuestra clase principal.
class Vehiculo{
  constructor(marca, color, modelo, TipoC, cRuedas, Estado, cPersonas, cilindraje){
    this.marca = marca;
    this.color = color;
    this.modelo = modelo;
    this.TipoC = TipoC;
    this.cRuedas = cRuedas;
    this.Estado = Estado;
    this.cPersonas = cPersonas;
```

```

    this.cilindraje = cilindraje;
  }

  // Definimos sus métodos propios de la función 'vehiculo'
  arrancar(){
    console.log("El vehículo arranco");
  }
}

// Creamos una clase que va a heredar atributos de 'vehiculo'
class Carro extends Vehiculo{
  constructor(marca, color, modelo, TipoC, cRuedas, Estado, cPersonas, cilindraje, nPuertas){

    super(marca, color, modelo, TipoC, cRuedas, Estado, cPersonas, cilindraje);
    this.nPuertas = nPuertas;
  }

  abrirBaul(){
    console.log("Se esta abriendo el baul");
  }
}

const ford = new Carro("scape", "negro", 2024, "Disesel", 4, "nuevo", 5, 2400, 4);
console.log(ford);
ford.arrancar();
ford.abrirBaul();

```

## Ejercicio practico - Herencia

Crea una **clase principal** que herede a dos subclases sus propiedades y que tenga que ver con **contratación** y tipos de empleados.

js - Solución

 Copiar código

```

// Definimos la clase principal:
class Contratacion {
  constructor (nombre, cedula, contrato, salario, termino, empleador ){
    this.nombre = nombre;
    this.cedula = cedula;
    this.contrato = contrato;
    this.salario = salario;
    this.termino = termino;
    this.empleador = empleador;
  }
  // Con su respectivo método
  contratado(){
    console.log("Felicidades " + this.nombre + " estas contratado");
  }
}

// Definimos la primer subclase que heredará parámetros de nuestra clase principal
// 'Contratación' a la cual se le añaden sus propios atributos: 'eps' y 'pension'.
class EmpleadoFijo extends Contratacion{
  constructor(nombre, cedula, contrato, salario, termino, empleador, eps, pension){
    super(nombre, cedula, contrato, salario, termino, empleador);
    this.eps = eps;
    this.pension = pension;
  }

  // También contiene su propio método independiente.
  contratoFijo() {
    console.log("Tu contrato es de tipo: '" + this.contrato + "' y tu salario es: '"
    + this.salario + "'");
  }
}

```

```
// Definimos la segunda subclase que heredar  par metros de nuestra clase principal
// 'Contrataci n' a la cual se le a aden sus propios atributos: 'cargo' y 'practicante'.
class EmpleadoPrestacion extends Contratacion{
  constructor(nombre, cedula, contrato, salario, termino, empleador, cargo, practicante){
    super(nombre, cedula, contrato, salario, termino, empleador);
    this.cargo = cargo;
    this.practicante = practicante;
  }

  // Tambi n cuenta con su propio m todo.
  contratadoPrestacion(){
    console.log("Tu contrato es de tipo: '" + this.contrato + "' con un salario de: '"
    + this.salario + "'" );
  }
}

// Instanciamos nuestro primer objeto para la subclase 'EmpleadoFijo'.
const newEmpleado = new EmpleadoFijo("Bryan", 1022485632, "Indefinido", 1500000, 1,
"JuanIT", "Capital Salud", "Porvenir");
console.log(newEmpleado); // Llamamos a dicho objeto por consola
newEmpleado.contratado(); // Llamamos a el m todo que hereda de 'Contratacion'
newEmpleado.contratoFijo(); // y por  ltimo tambi n llamamos a su propio m todo.

// Instanciamos nuestro segundo objeto para la subclase 'EmpleadoPrestacion'.
const newEmpleado2 = new EmpleadoPrestacion("Carlos", 1135896322, "Prestaci n Servicios",
1300000, 0.6, "DianaIT", "Asistente", true);
console.log(newEmpleado2); // Llamamos a dicho objeto por consola
newEmpleado2.contratado(); // Llamamos a el m todo que hereda de 'Contratacion'
newEmpleado2.contratadoPrestacion(); // y por  ltimo tambi n llamamos a su propio m todo.
```