



Argentina programa



#YoProgramo

(Programador Full Stack Web Jr.)



Ministerio de
Desarrollo Productivo
Argentina



Instituto
Nacional
de Tecnología
Industrial



Argentina
programa

Módulo 3 - Front End - Desarrollo Web Dinámico

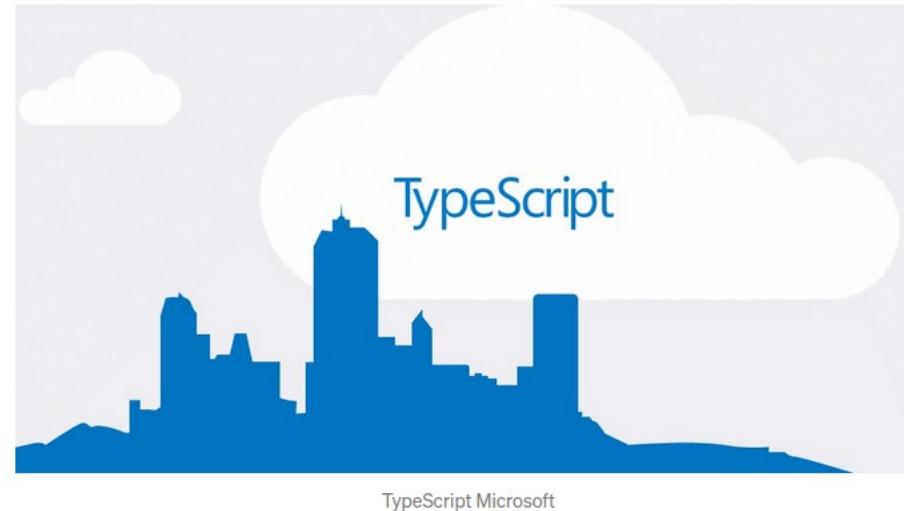
Tema: TypeScript

- Introducción a TypeScript
- Instalar NodeJs
- Instalar TSC
- Crear y compilar Archivo TS
- Tipado Estático (Tipos y Subtipos)
- Funciones
- Programación Orientada a Objetos

Módulo 3 - Front End - Desarrollo Web Dinámico

TypeScript -Introducción

Es un lenguaje de programación de código abierto creado por el equipo de Microsoft como una solución al desarrollo de aplicaciones de gran escala con Javascript dado que este último carece de clases abstractas, interfaces, genéricos, etc. y demás herramientas que permiten los lenguajes de programación tipados.



Módulo 3 - Front End - Desarrollo Web Dinámico

TypeScript -Introducción

JavaScripts	TypeScripts
JavaScripts se ejecuta en el navegador. Es un lenguaje de programación interpretado.	TypeScript necesita ser “transpilado ¹ ” a JavaScript, que es el lenguaje entendido por los navegadores.
JavaScript se ejecuta en el navegador, es decir en el lado del cliente únicamente.	TypeScript se ejecuta en ambos extremos. En el servidor y en el navegador.
Es débilmente tipado.	Es fuertemente tipado (tipado estático)
Basado en prototipos.	Orientado a objetos.

Tabla comparativa de los lenguajes JavaScripts y TypeScripts

Módulo 3 - Front End - Desarrollo Web Dinámico

Tipado estático

Una de las principales características de typescript es que es fuertemente tipado por lo que, no sólo permite identificar el tipo de datos de una variable mediante una sugerencia de tipo sino que además permite validar el código mediante la comprobación de tipos estáticos. Por lo tanto, TypeScript permite detectar problemas de código previo a la ejecución cosa que con Javascript no es posible.

```
let t: string =  
  'Tipado Estático';
```

TS

Módulo 3 - Front End - Desarrollo Web Dinámico

Tipado estático

Se agrega además que Typescript permite describir mucho mejor el fuente ya que, además de ser tipado, es verdaderamente orientado a objetos lo que permite a los desarrolladores un código más legible y mantenible.

```
function calculatePropertyValue(  
    tax: Boolean,  
    amount: number,  
    area: string): number {  
    let PropertyValue  
  
    // some code  
  
    return PropertyValue  
}
```

TypeScript

Módulo 3 - Front End - Desarrollo Web Dinámico

Variable

Es un espacio de memoria que se utiliza para almacenar un valor durante un tiempo (scope) en la ejecución del programa. La misma tiene asociado un tipo de datos y un identificador.

var: Es el tipo de declaración más común utilizada.

let: Es un tipo de variable más nuevo (agregado por la ECMAScript 2015). Este cambio permite declarar variables con ámbito de nivel

varias veces.

```
let name: string = `Javier Ruiz`;
let age: number = 28;
let sentence: string = `Hola, mi nombre es ${ name }.
Este año voy a cumplir ${ age + 1 } años.`;

// Esto sería equivalente a
let sentence: string = "Hola, mi nombre es " + name + ".\n\n" + "Este
año voy a cumplir " + (age + 1) + " años.";
```

```
// Dato de tipo string
var name: string = 'Tu nombre';
name = 'Otro nombre'; // Es correcto
name = 2 // Es incorrecto
```

```
// Dato de tipo number
var age: number = 29;
age = 0xf00d; // Es hexadecimal y es correcto
age = '3'; // Es un string e incorrecto
```

```
// Dato de tipo boolean
var havePets: boolean = true;
havePets = false; // Es correcto
havePets = 3 // Es incorrecto
```

Módulo 3 - Front End - Desarrollo Web Dinámico

Variable

const: Es un tipo constante ya que, al asumir un valor no puede modificarse.

```
const ivaProducto = 0.10;
```

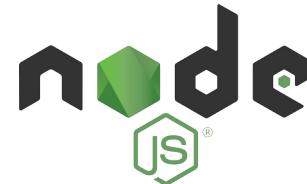
La diferencia entre las declaraciones **let** es que se pueden realizar sin inicialización, mientras que las declaraciones **const** siempre se inicializan con un valor. Y las declaraciones **const**, una vez asignadas, nunca se pueden volver a asignar.

Módulo 3 - Front End - Desarrollo Web Dinámico

Iniciando con TypeScript - ¿Cómo instalar TypeScript?

Para ejecutar código en Typescript necesitamos instalar:

- 1 - NodeJS, dado que el compilador de Typescript está desarrollado en NodeJS.



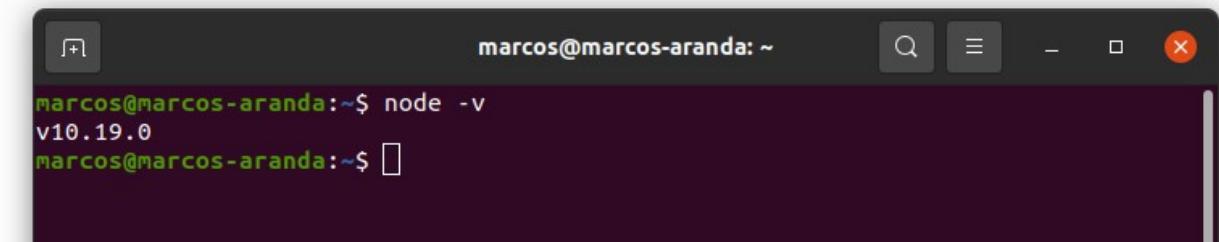
- 2- TSC (Command-line TypeScript Compiler), herramienta que permite compilar un archivo TypeScript a Javascript nativo.



Módulo 3 - Front End - Desarrollo Web Dinámico

Instalar NodeJS

1. Descargar del sitio oficial el instalador acorde a su sistema operativo.
2. Instalar NodeJs.
 - a. Si tienes Windows o Mac, simplemente ejecuta el instalador y ¡listo!
 - b. Si tienes Linux, la página de instalación de NodeJS ofrece los comandos para instalar en Linux.
3. Evaluar que la instalación fue exitosa. Para ello, ir a la línea de comandos del sistema e introducir el comando: **node -v**. A continuación, el sistema mostrará por pantalla la versión de NodeJS instalada.



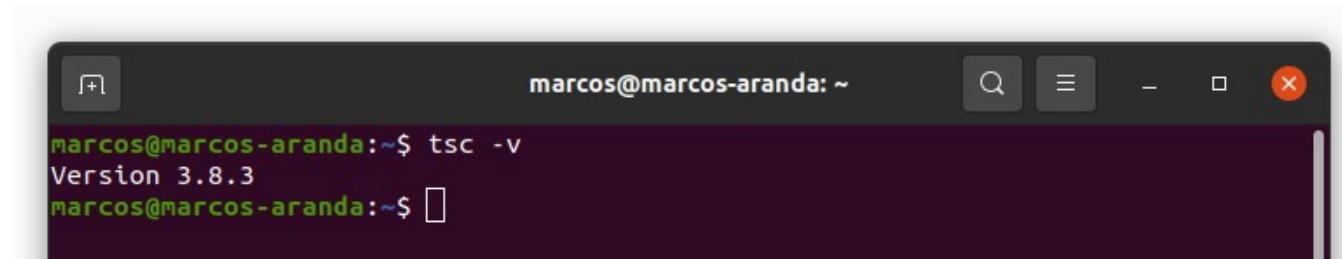
```
marcos@marcos-aranda:~$ node -v
v10.19.0
marcos@marcos-aranda:~$
```

Módulo 3 - Front End - Desarrollo Web Dinámico

¿Cómo instalar TSC (Command-line TypeScript Compiler)?

La misma se realizará vía comando npm como sigue:

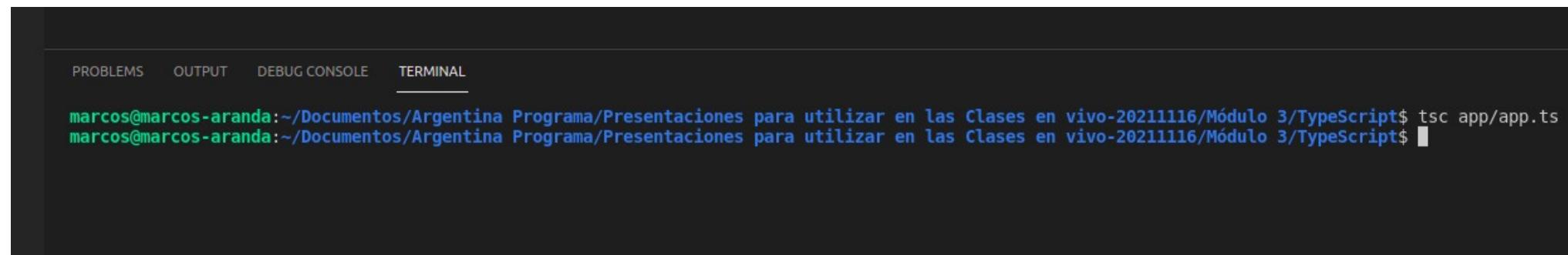
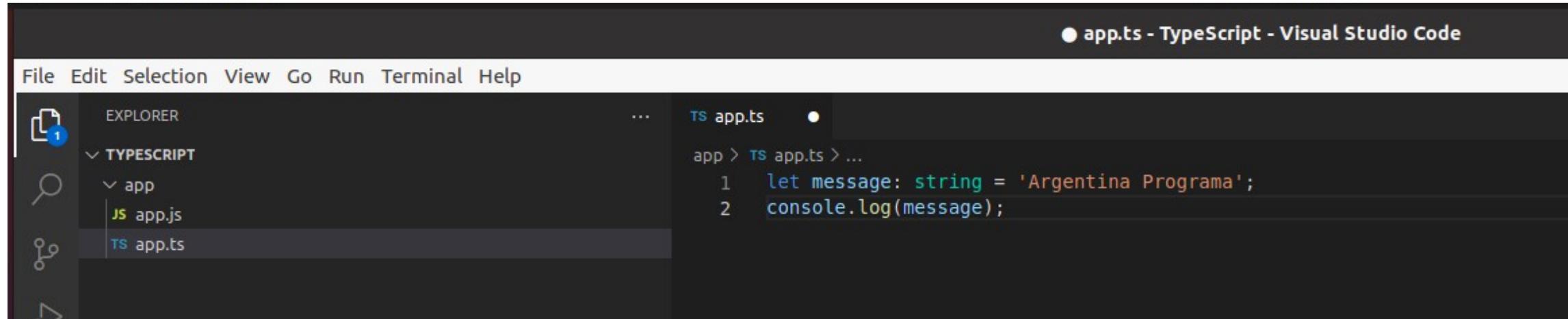
1. Abrir la línea de comandos del sistema.
2. Ejecutar el siguiente comando: **npm install -g typescript**
3. Evaluar que la instalación fue exitosa. Para ello ejecutar el comando: **tsc -v**



```
marcos@marcos-aranda:~$ tsc -v
Version 3.8.3
marcos@marcos-aranda:~$
```

Módulo 3 - Front End - Desarrollo Web Dinámico

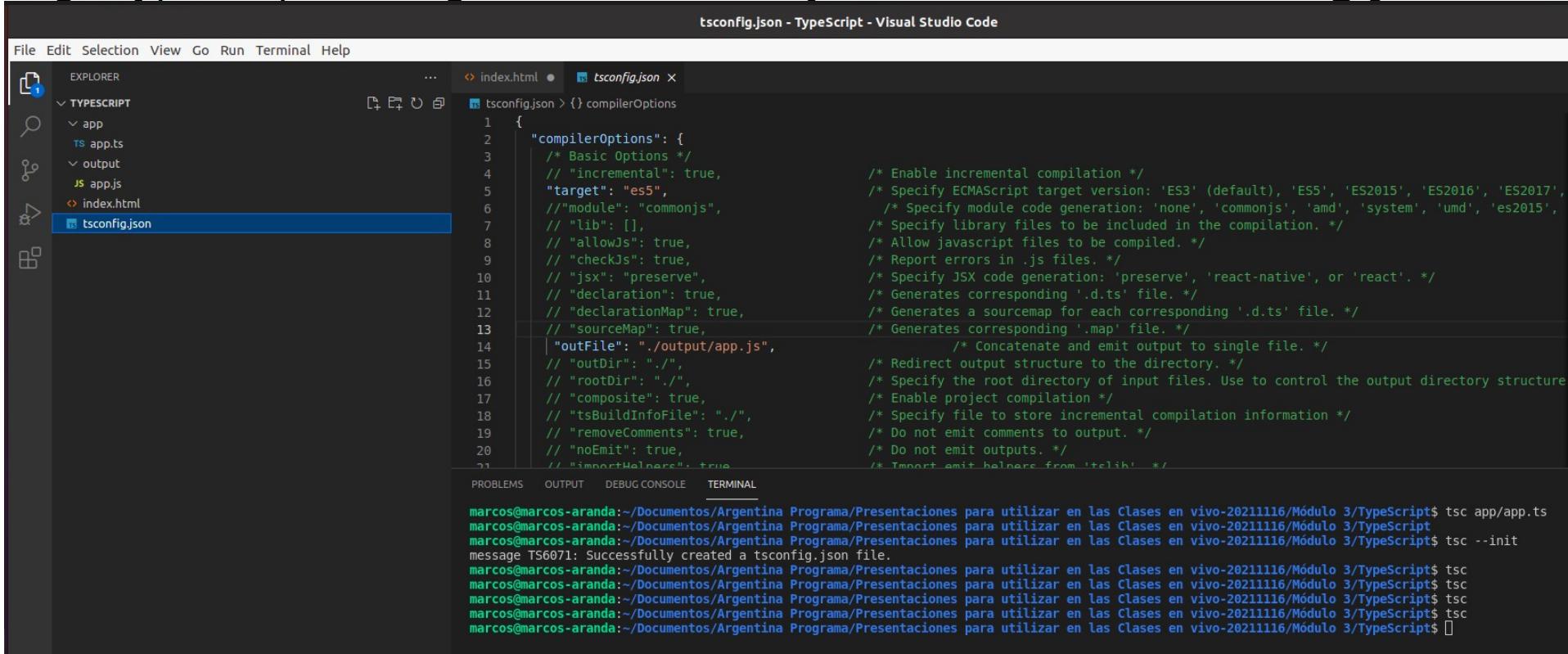
¿Cómo crear y compilar un archivo Typescript en VSCode?



Módulo 3 - Front End - Desarrollo Web Dinámico

Comandos y opciones del compilador CLI de TSC

Las opciones del compilador permiten controlar cómo se genera el código JavaScript a partir del código TypeScript de origen en un archivo JSON denominado **tsconfig.json**.



The screenshot shows a Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Explorer Panel:** Shows a project structure with **TYPESCRIPT** folder containing **app** (with **app.ts**) and **output** (with **app.js** and **index.html**). The **tsconfig.json** file is selected and highlighted in blue.
- Code Editor:** Displays the **tsconfig.json** file content:

```
1  {
2    "compilerOptions": {
3      /* Basic Options */
4      // "incremental": true,
5      "target": "es5",
6      // "module": "commonjs",
7      // "lib": [],
8      // "allowJs": true,
9      // "checkJs": true,
10     // "jsx": "preserve",
11     // "declaration": true,
12     // "declarationMap": true,
13     // "sourceMap": true,
14     | "outFile": "./output/app.js",
15     // "outDir": "./",
16     // "rootDir": "./",
17     // "composite": true,
18     // "tsBuildInfoFile": "./",
19     // "removeComments": true,
20     // "noEmit": true,
21     // "importHelpers": true
22   }
23   /* Enable incremental compilation */
24   /* Specify ECMAScript target version: 'ES3' (default), 'ES5', 'ES2015', 'ES2016', 'ES2017',
25   /* Specify module code generation: 'none', 'commonjs', 'amd', 'system', 'umd', 'es2015',
26   /* Specify library files to be included in the compilation. */
27   /* Allow javascript files to be compiled. */
28   /* Report errors in .js files. */
29   /* Specify JSX code generation: 'preserve', 'react-native', or 'react'. */
30   /* Generates corresponding '.d.ts' file. */
31   /* Generates a sourcemap for each corresponding '.d.ts' file. */
32   /* Generates corresponding '.map' file. */
33   /* Concatenate and emit output to single file. */
34   /* Redirect output structure to the directory. */
35   /* Specify the root directory of input files. Use to control the output directory structure */
36   /* Enable project compilation */
37   /* Specify file to store incremental compilation information */
38   /* Do not emit comments to output. */
39   /* Do not emit outputs. */
40   /* Import emit helpers from 'tslib' */
41 }
```
- Terminal:** Shows command-line history:

```
marcos@marcos-aranda:~/Documentos/Argentina Programa/Presentaciones para utilizar en las Clases en vivo-20211116/Módulo 3/TypeScript$ tsc app/app.ts
marcos@marcos-aranda:~/Documentos/Argentina Programa/Presentaciones para utilizar en las Clases en vivo-20211116/Módulo 3/TypeScript
marcos@marcos-aranda:~/Documentos/Argentina Programa/Presentaciones para utilizar en las Clases en vivo-20211116/Módulo 3/TypeScript$ tsc --init
message TS6071: Successfully created a tsconfig.json file.
marcos@marcos-aranda:~/Documentos/Argentina Programa/Presentaciones para utilizar en las Clases en vivo-20211116/Módulo 3/TypeScript$ tsc
marcos@marcos-aranda:~/Documentos/Argentina Programa/Presentaciones para utilizar en las Clases en vivo-20211116/Módulo 3/TypeScript$ tsc
marcos@marcos-aranda:~/Documentos/Argentina Programa/Presentaciones para utilizar en las Clases en vivo-20211116/Módulo 3/TypeScript$ tsc
marcos@marcos-aranda:~/Documentos/Argentina Programa/Presentaciones para utilizar en las Clases en vivo-20211116/Módulo 3/TypeScript$ tsc
marcos@marcos-aranda:~/Documentos/Argentina Programa/Presentaciones para utilizar en las Clases en vivo-20211116/Módulo 3/TypeScript$ tsc
```

Módulo 3 - Front End - Desarrollo Web Dinámico

¿Cómo crear un servidor de pruebas en VSCode?

1. Ejecutar el siguiente comando “**npm install --global http-server**”. A continuación, se creará la carpeta node_modules. Linux **sudo npm install --save http-server**
2. Ejecutar el comando “**npm init**”. A continuación, se creará un archivo **package.json**
3. Configurar la entrada “**scripts**”
4. Finalmente, ejecutar el comando “**npm start**”

Módulo 3 - Front End - Desarrollo Web Dinámico

TypeScript Programación Orientada a Objetos

La programación orientada a objetos (abreviada de ahora en más como POO), es un conjunto de reglas y principios de programación (o sea, un paradigma de programación) que busca representar las entidades u objetos del dominio (o enunciado) del problema dentro de un programa, de la forma más natural posible.

En el paradigma de programación tradicional o estructurado, que es el que hemos usado hasta aquí, el programador busca identificar los procesos (en forma de subproblemas o módulos) a fin de obtener los resultados deseados. Y esta forma de proceder no es en absoluto incorrecta: la estrategia de descomponer un problema en subproblemas es una técnica elemental de resolución de problemas que los programadores orientados a objetos siguen usando dentro de este nuevo paradigma. Entonces, ¿a qué viene el paradigma de la POO?

Módulo 3 - Front End - Desarrollo Web Dinámico

La POO significa una nueva visión en la forma de programar, buscando aportar claridad y naturalidad en la manera en que se plantea un problema. Ahora, el objetivo primario no es identificar procesos sino identificar actores: las **entidades** u **objetos** que aparecen en el escenario o dominio del problema, tales que esos objetos tienen no sólo datos asociados sino también algún comportamiento que son capaces de ejecutar. Por ejemplo, pensemos en un objeto como en un robot virtual: el programa tendrá muchos robots virtuales (objetos de software) que serán capaces de realizar eficiente y prolijamente ciertas tareas en las que serán expertos. Además, serán capaces de interactuar con otros robots virtuales (objetos...) con el objeto de resolver el problema que el programador esté planteando.

Ventajas de la POO

- Es una forma más natural de modelar.
- Permite manejar mejor la complejidad.
- Facilita el mantenimiento y extensión de los sistemas.
- Es más adecuado para la construcción de entornos GUI.

Módulo 3 - Front End - Desarrollo Web Dinámico

Objetos

Para comprenderlo veamos la siguiente imagen y luego intentemos responder: ¿Cuáles son los objetos que se pueden abstraer para ver televisión? ¿Cómo podrías describirlos?



En el problema planteado se pueden identificar tres elementos: la persona, el televisor y el control remoto. Cada elemento posee sus propias características y comportamientos. En la POO a estos elementos se los conoce bajo el nombre de **OBJETOS**, a las características o estados que identifican a cada objeto **ATRIBUTOS** y, a los comportamientos o acciones, **MÉTODOS**.

Módulo 3 - Front End - Desarrollo Web Dinámico

Un objeto es una entidad (tangible o intangible) que posee características propias (atributos) y acciones o comportamientos (métodos) que realiza por sí solo o interactuando con otros objetos.

ELEMENTO	DESCRIPCIÓN
Persona	Tiene sus propios atributos: Apellido, Nombre, Altura, género, Color ojos, Cabello, etc. Y tiene un comportamiento: Ver , escuchar, hablar, correr, saltar, etc.
Control Remoto	Tiene sus propios atributos: Tamaño, color, tipo, batería, etc. Y tiene un comportamiento: Enviar señal, codificar señal, cambiar canal, aumentar volumen, ingresar a menú, prender TV, etc.
Televisor	Tiene sus propios atributos: pulgadas, tipo, número parlantes, marca , etc. Y tiene un comportamiento: Decodificar señal, prender, apagar, emitir señal, emitir audio, etc.

Módulo 3 - Front End - Desarrollo Web Dinámico

Las características generales de los objetos en POO:

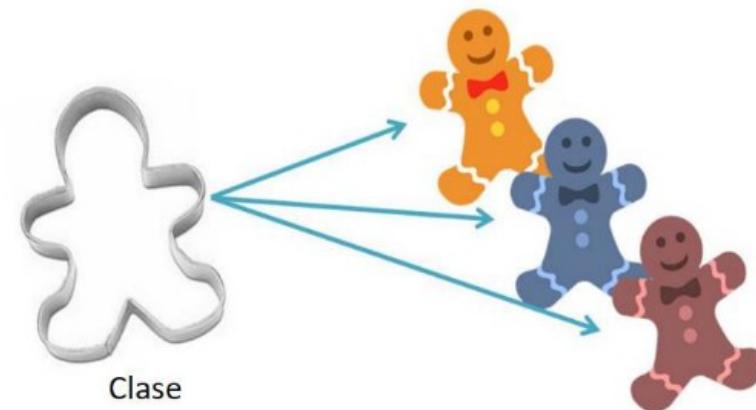
- Se identifican por un nombre o identificador único que lo diferencia de los demás.
- Poseen estados.
- Poseen un conjunto de métodos.
- Poseen un conjunto de atributos.
- Soportan el encapsulamiento (nos deja ver sólo lo necesario).
- Tienen un tiempo de vida.
- Son instancias de una clase (es de un tipo).

Para hacer eso, los lenguajes de programación orientados a objetos (como TypeScript) usan descriptores (plantillas) de entidades conocidas como **clases**.

Módulo 3 - Front End - Desarrollo Web Dinámico

Clases

Es la descripción de una entidad u objeto de forma tal que pueda usarse como plantilla para crear muchos objetos que respondan a dicha descripción. Para establecer analogías, se puede pensar que una clase se corresponde con el concepto de tipo de dato de la programación estructurada tradicional, y los objetos creados a partir de la clase (llamados instancias en el mundo de la POO) se corresponden con el concepto de variable de la programación tradicional. Así como el tipo es uno solo y describe la forma que tienen todas las muchas variables de ese tipo, la clase es única y describe la forma y el comportamiento de los muchos objetos de esa clase.



Módulo 3 - Front End - Desarrollo Web Dinámico

Características generales de las clases en POO.

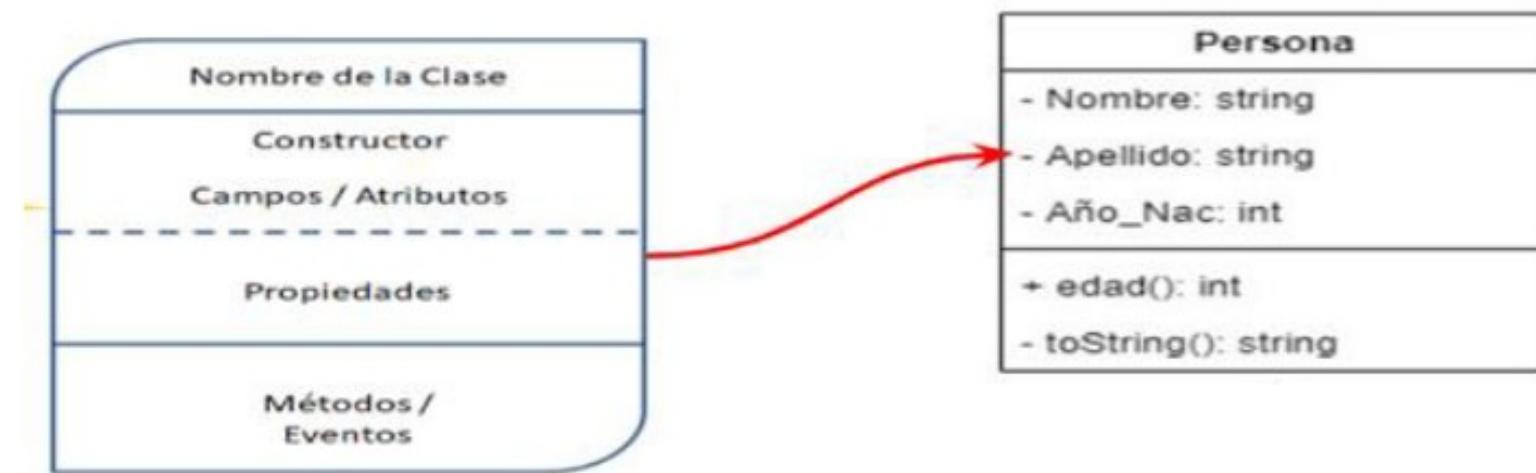
- Poseen un alto nivel de abstracción.
- Se relacionan entre sí mediante jerarquías.
- Los nombres de las clases deben estar en singular.

Representación gráfica de clases

La representación gráfica de una o varias clases se realiza mediante los denominados **Diagramas de Clase**. Para ello, se utiliza la notación que provee el Lenguaje de Modelación Unificado (UML), a saber:

- Las clases se denotan como rectángulos divididos en tres partes. La primera contiene el nombre de la clase, la segunda contiene los atributos y la tercera los métodos.
- Los modificadores de acceso a datos y operaciones, a saber: público, protegido y privado; se representan con los símbolos +, # y - respectivamente, al lado derecho del atributo. (+ público, # protegido, - privado).

Módulo 3 - Front End - Desarrollo Web Dinámico



Estructura de una clase (Diagrama de clases)

Módulo 3 - Front End - Desarrollo Web Dinámico

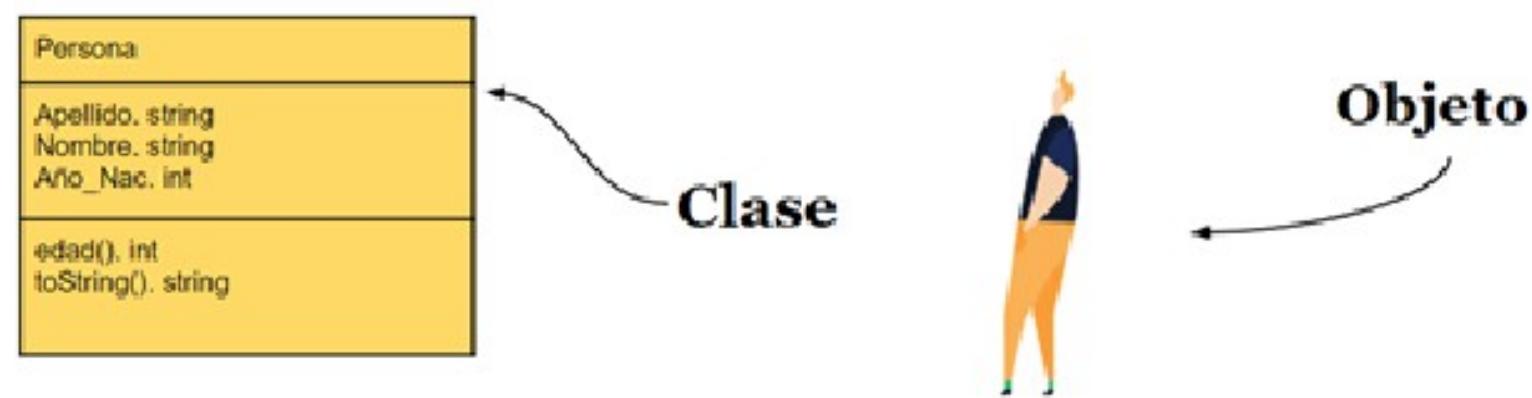
Relación entre clases y objetos

Algorítmicamente, las clases son descripciones netamente estáticas o plantillas que describen objetos. Su rol es definir nuevos tipos conformados por atributos y operaciones. Es decir que, las clases son una especie de molde de fábrica, en base al cual son construidos los objetos. Por el contrario, los objetos son instancias particulares de una clase. Durante la ejecución de un programa sólo existen los objetos, no las clases.

- La declaración de una variable de una clase NO crea el objeto.

La creación de un objeto debe ser indicada explícitamente por el programador (instanciación), de forma análoga a como inicializamos las variables con un valor dado, sólo que para los objetos se hace a través de un método **CONSTRUCTOR**.

Módulo 3 - Front End - Desarrollo Web Dinámico



Relación entre clases y objetos.

Módulo 3 - Front End - Desarrollo Web Dinámico

Fundamentos del enfoque orientado a objetos (EOO)

El Enfoque Orientado a Objeto se basa en cuatro principios que constituyen la base de todo desarrollo orientado a objetos. Estos principios son:

- Jerarquías (herencia, agregación y composición)
- Abstracción
- Encapsulamiento
- Modularidad

Otros elementos para destacar (aunque no fundamentales) son:

- Polimorfismo
- Tipificación
- Concurrency
- Persistencia.

Módulo 3 - Front End - Desarrollo Web Dinámico

Jerarquías

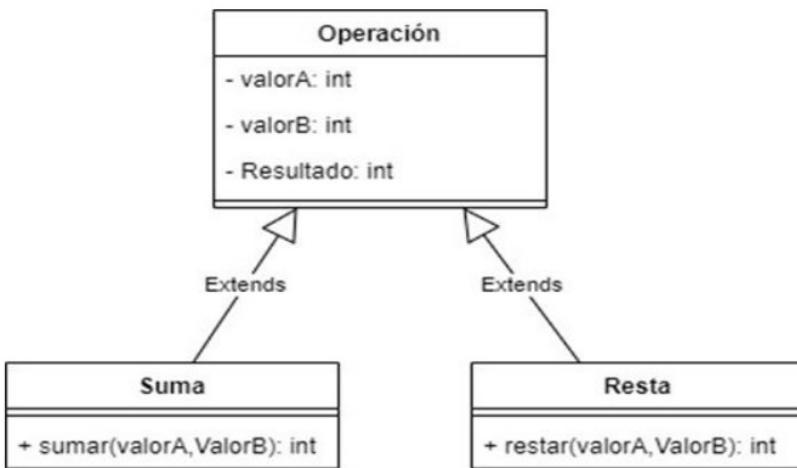
Las clases no se construyen para que trabajen de manera aislada, la idea es que ellas se puedan relacionar entre sí de manera que puedan compartir atributos y métodos sin necesidad de volver a escribirlos y así resolver un problema.

La posibilidad de establecer jerarquías entre las clases es una característica que diferencia esencialmente la programación orientada a objetos de la programación tradicional, ello debido fundamentalmente a que permite extender y reutilizar el código existente sin tener que volver a escribirlo cada vez que se necesite.

Herencia

En Programación Orientada a Objetos se llama herencia al mecanismo por el cual se puede definir una nueva clase B en términos de otra clase A ya definida, pero de forma que la clase B obtiene todos los miembros definidos en la clase A sin necesidad de hacer una redeclaración explícita. El sólo hecho de indicar que la clase B hereda (o deriva) desde la clase A, hace que la clase B incluya todos los miembros de A como propios (a los cuales podrá acceder en mayor o menor medida de acuerdo al calificador de acceso [public, private, protected, "default"] que esos miembros tengan en A).

Módulo 3 - Front End - Desarrollo Web Dinámico



```

class Operacion{
    protected valorA:number;
    protected valorB:number;
    protected resultado:number;
    constructor() {
        this.valorA=0;
        this.valorB=0;
        this.resultado=0;
    }

    set ValorA(value:number){
        this.valorA=value;
    }
    set ValorB(value:number){
        this.valorB=value
    }

    get Resultado():number {
        return this.resultado;
    }
}
  
```

```

class Suma extends Operacion
{
    Sumar ()
    {
        this.resultado=this.valorA+this.valorB;
    }
}
  
```

```

class Restar extends Operacion
{
    Restar ()
    {
        this.resultado=this.valorA-this.valorB;
    }
}
  
```

```

let operacionS= new Suma();
operacionS.ValorA=45;
operacionS.ValorB=10;
operacionS.Sumar();
console.log("El resultado de la suma es " + operacionS.Resultado);
  
```

```

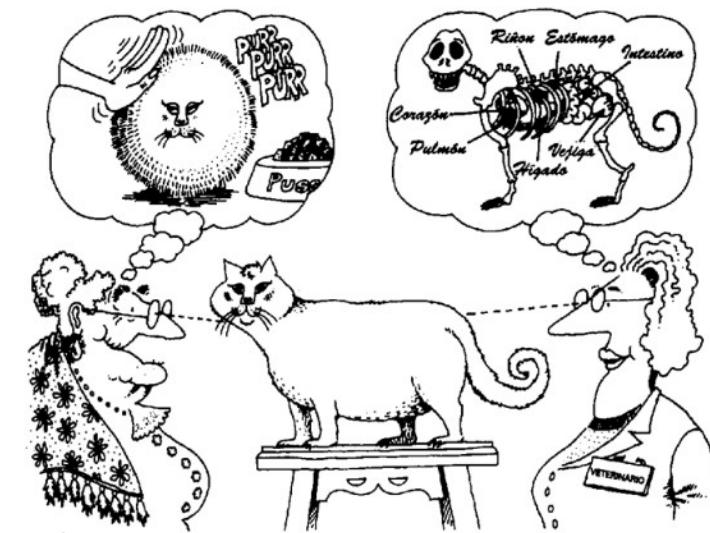
let operacionR= new Restar();
operacionR.ValorA=45;
operacionR.ValorB=10;
operacionR.Restar();
console.log("El resultado de la resta es " + operacionR.Resultado);
  
```

Módulo 3 - Front End - Desarrollo Web Dinámico

Abstracción

Una abstracción denota las características esenciales de un objeto (datos y operaciones), que lo distingue de otras clases de objetos. Decidir el conjunto correcto de abstracciones de un determinado dominio, es el problema central del diseño orientado a objetos.

“Una abstracción se centra en la visión externa de un objeto por lo tanto sirve para separar el comportamiento esencial de un objeto de su implementación. La decisión sobre el conjunto adecuado de abstracciones para determinado dominio es el problema central del diseño orientado a objetos. Se puede caracterizar el comportamiento de un objeto de acuerdo a los servicios que presta a otros objetos, así como las operaciones que puede realizar sobre otros objetos”

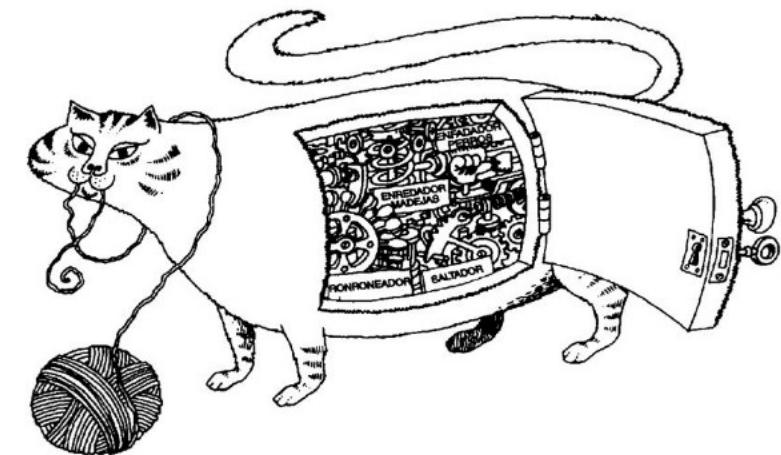


Módulo 3 - Front End - Desarrollo Web Dinámico

Encapsulamiento

Es la propiedad de la POO que permite ocultar los detalles de implementación del objeto mostrando sólo lo relevante. Esta parte de código oculta pertenece a la parte privada de la clase y no puede ser accedida desde ningún otro lugar.

El encapsulamiento sólo los métodos de una clase deberían tener acceso directo a los atributos de esa clase, para impedir que un atributo sea modificado en forma insegura, o no controlada por la propia clase. El Principio de Encapsulamiento es la causa por la cual en general los atributos se declaran como **privados (private)**, y los métodos se definen públicos (**public**). Los calificadores **private** y **public** tienen efecto a nivel de compilación: si un atributo de una clase es privado, y se intenta acceder a él desde un método de otra clase, se producirá un error de compilación.



Módulo 3 - Front End - Desarrollo Web Dinámico

```
class Persona{  
    private nombre:string;  
    private apellido:string;  
    private añoNac:number;  
    constructor(nombre:string, apellido:string)  
{  
        this.nombre = nombre;  
        this.apellido = apellido;  
    }  
  
    get Nombre():string {  
        return this.nombre;  
    }  
  
    get Apellido():string {  
        return this.apellido;  
    }  
    ...  
}
```

La clase Persona encapsula los atributos a fin de que, no tomen valores inconsistentes.

El encapsulamiento da lugar al Principio de Ocultamiento: sólo los métodos de una clase deberían tener acceso directo a los atributos de esa clase, para impedir que un atributo sea modificado en forma insegura, o no controlada por la propia clase.

Módulo 3 - Front End - Desarrollo Web Dinámico

Modularidad

Es la propiedad que permite tener independencia entre las diferentes partes de un sistema. La modularidad consiste en dividir un programa en módulos o partes, que pueden ser compilados separadamente, pero que tienen conexiones con otros módulos. En un mismo módulo se suele colocar clases y objetos que guarden una estrecha relación. El sentido de modularidad está muy relacionado con el ocultamiento de información.



Módulo 3 - Front End - Desarrollo Web Dinámico

Polimorfismo

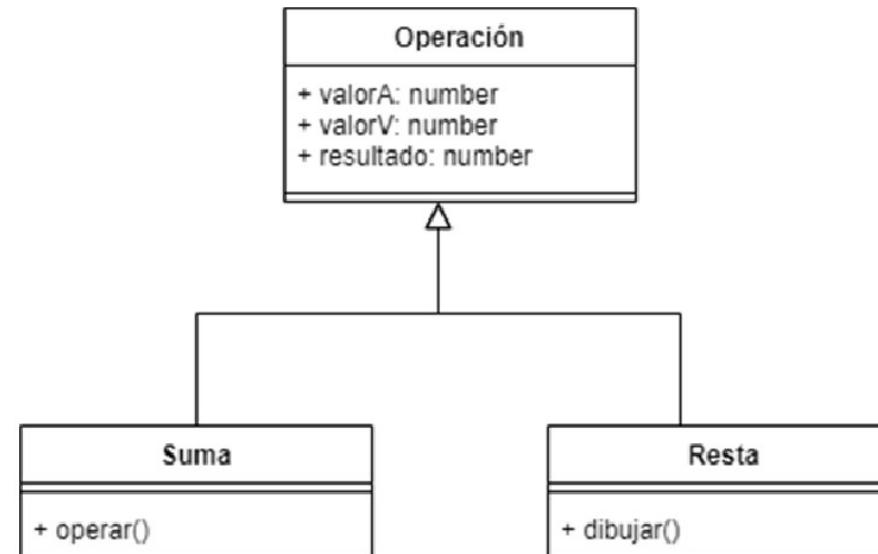
Clases diferentes (polimórficas) implementan métodos con el mismo nombre. En resumen, el polimorfismo permite comportamientos diferentes, asociados a objetos distintos compartiendo el mismo nombre; al llamarlos por ese nombre se utilizará el comportamiento



Módulo 3 - Front End - Desarrollo Web Dinámico

Polimorfismo por herencia

Cuando una subclase hereda de una clase base, obtiene todos los métodos, campos, propiedades y eventos de la superclase sin embargo, quizás necesitemos un comportamiento diferente para las clases derivadas (o subclases).



Muchas gracias.



Ministerio de
Desarrollo Productivo
Argentina



Instituto
Nacional
de Tecnología
Industrial



Argentina
programa