01. Introducción a la programación

1. Introducción a programación (elegir lenguaje de programación)
    1. Escribiendo código
    2. Condicionales (if, else)
    3. Loops (while, for, repeat)
    4. Reutilizando código (funciones)
    5. Detalles Typescript
    6. Clases en Typescript
    7. Usando clases externas
    8. Corriendo **npm**

Escribiendo código

Programar es como armar una receta de cocina:

- Se tienen instrucciones.
- Se las sigue al pie de la letra.
- Se obtiene un resultado.
- La receta se la puede modificar y compartir.

Nuestro primer "Hola Mundo" en TS

Escribimos un `console.log("hello, world")` y ejecutamos desde el **NPM**.

Tipos de datos en TS

- **number**: valores numéricos.
- **string**: cadenas de caracteres.
- **boolean**: tipos booleanos con valores *"true"* o *"false"*.
- **undefined**: Tipo no definido.

Condicionales (if, else)

Podemos esperar condiciones para ejecutar determinada instrucción.

Por ejemplo:

```
let valor1 = 9;
if (valor1 > 10) {
        console.log("es mayor que 10...");
} else {
        console.log("Es menor que 10...");
}
```

Loops (while,for)

Vamos a tener la necesidad de ejecutar instrucciones una cierta cantidad de veces (definida o indefinida).

Ciclos "For"

Nosotros sabemos la cantidad de veces que se tiene que ejecutar una determinada instrucción.

Por ejemplo:

```
for (let i = 0; i < 10; i++) {
        console.log("este mensaje se repite!");
}
```

Ciclos "While"

Nosotros no sabemos la cantidad de veces que se tiene que ejecutar algunas instrucciones, pero sabemos que se tiene que cumplir cierta condición para que se ejecute.

```
let valor = 5;
let contador = 0;
while (contador < valor) {
        console.log("este mensaje se repite!");
        contador++;
}
```

Funciones en Typescript

Podemos reutilizar código para hacerlo más legible y mantenible.

```
let valor1 = 9;
let valor2 = 11;

if (valor1 > 10) {
        console.log("es mayor a 10");
```

```
} else {
        console.log("es menor o igual a 10");
}

if (valor2 > 10) {
        console.log("es mayor a 10");
} else {
        console.log("es menor o igual a 10");
}
```

Usando funciones, podemos reutilizar parte del código:

```
function chequearNumero(valor: number) {
        if (valor > 10) {
                console.log("es mayor a 10");
        } else {
                console.log("es menor o igual a 10");
        }
}

chequearNumero(9);
chequearNumero(11);
```

Operador "flecha"

Clases en Typescript

Desde el punto de vista conceptual, una clase es una abstracción de la realidad. Dicha abstracción tiene un estado y un comportamiento según ese estado.

```
class Hello {
        message: string;
        constructor(message: string) {
                this.message = message;
        }

        saludar() {
                return "saludos, " + this.message;
        }
}

let hello = new Hello("typescript");
console.log(hello.saludar());
```

Herencia

Podemos extender el comportamiento de una clase haciendo uso de la palabra reservada `extends` indicando la clase que queremos extender.

```
class Animal {
        name: string;
        constructor(name: string) {
                this.name = name;
        }

        saludar() {
                console.log("Me llamo " + this.name);
        }
}

class Snake extends Animal {
        skill: string;
        constructor(name: string) {
                super(name);
                this.skill = "poison";
        }
        mostrarHabilidad() {
                console.log("la habilidad es " + this.skill);
        }
}

let snake = new Snake("serpiente");
snake.saludar();
snake.mostrarHabilidad();
```

Importando y exportando clases

Podemos definir nuestras clases en archivos separados y usar las palabras reservadas `export` para indicar que una clase se exporta, e `import` para importar la clase en cuestión.

```
export class Animal {
        name: string;
        constructor(name: string) {
                this.name = name;
        }

        saludar() {
                console.log("hola, " + this.name);
```

```
        }
}
```

```
import Animal from './animal';

let animal = new Animal("dog");
animal.saludar();
```