

# Desarrollo de Aplicaciones I

# Clase 2

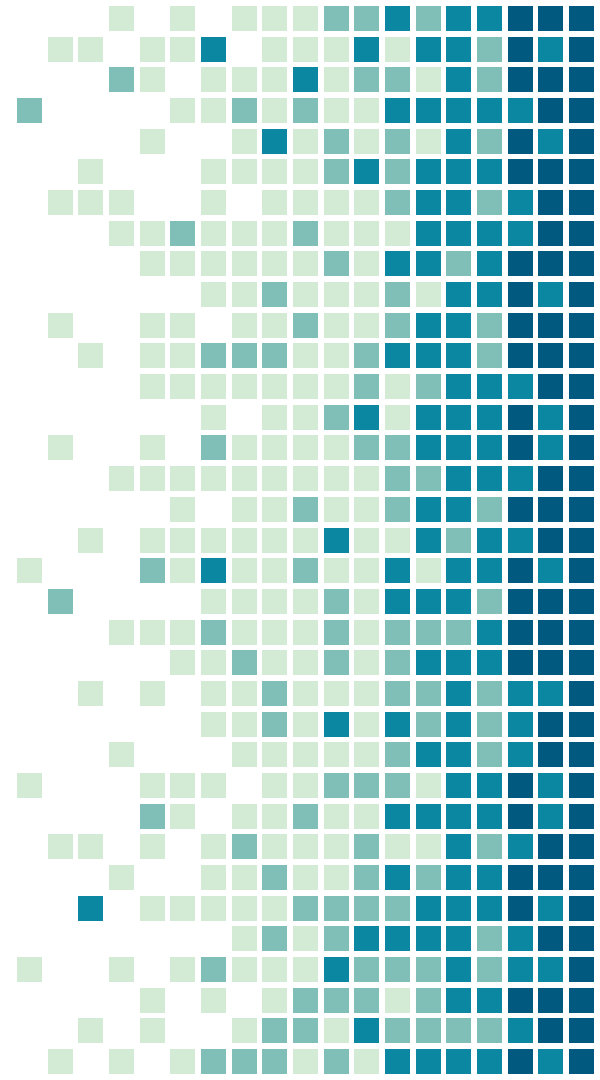
- Introducción Typescript



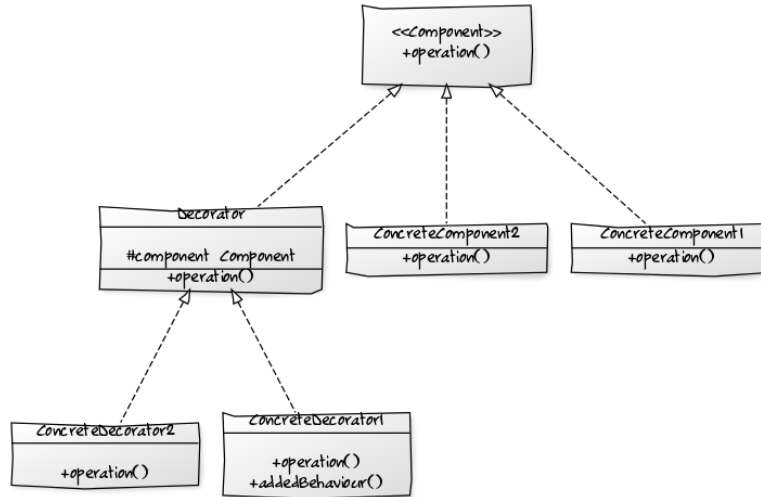
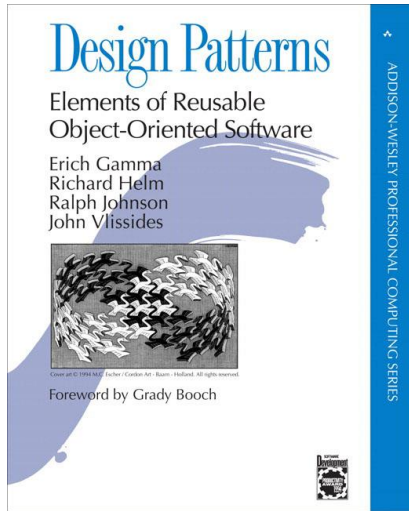
# JAVA



- Imperativo
- Funcional (lambda)
- Recolector de basura
- Tipado estático
- Orientado a Objetos (con clases)
- Modularidad con paquetes.



# Orientación a objetos







# Tipados estáticos



```
public class MainTest {  
    public static void main(String[] args) {  
        miBoolean = true;  
    }  
}
```

 miBoolean cannot be resolved to a variable

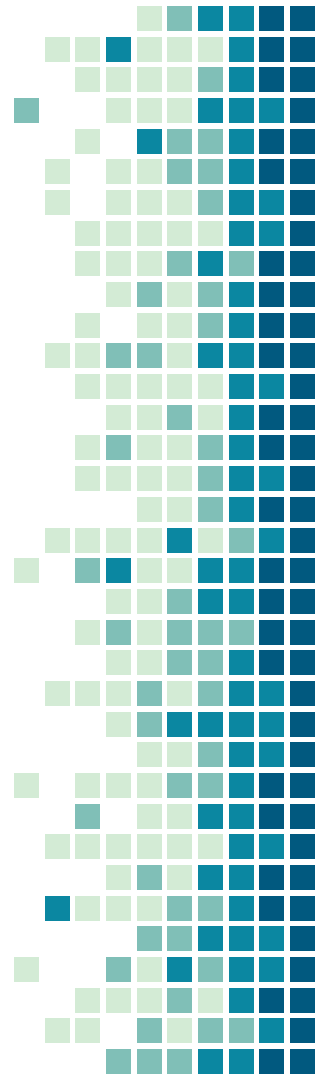
4 quick fixes available:

-  [Create local variable 'miBoolean'](#)
-  [Create field 'miBoolean'](#)
-  [Create parameter 'miBoolean'](#)
-  [Remove assignment](#)

# Características (Es5)



- Imperativo
- Funcional
- Recolector de basura
- **Tipado dinámicos**
- Orientado a Objetos (**con prototipos**)
- **Sin modularidad.**



# Tipos dinámicos (Es5)



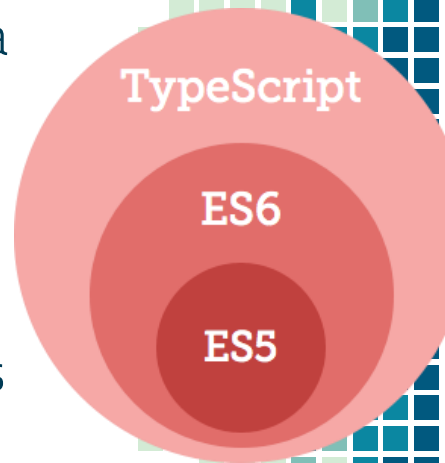
- El compilador no te ayuda
  - Hay que ejecutar los test (si se tienen)
- El IDE tampoco te ayuda
  - No se puede refactorizar de forma automática.
  - El auto completado es muy limitado.
  - No se puede navegar a la implementación.



# ¿Qué es TypeScript?



- TypeScript es un lenguaje de programación de código abierto desarrollado y mantenido por Microsoft.
- Es un superset estricto de JavaScript, y añade, de manera opcional, tipado estáticos y herramientas que ayudan a desarrollar a través de metodologías de programación orientada a objetos basado en clases.
- puede ser utilizado para desarrollar aplicaciones en JavaScript para el lado del cliente o en el servidor a través de Node.js.

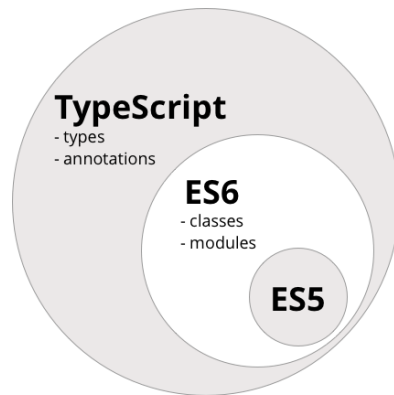




# Características

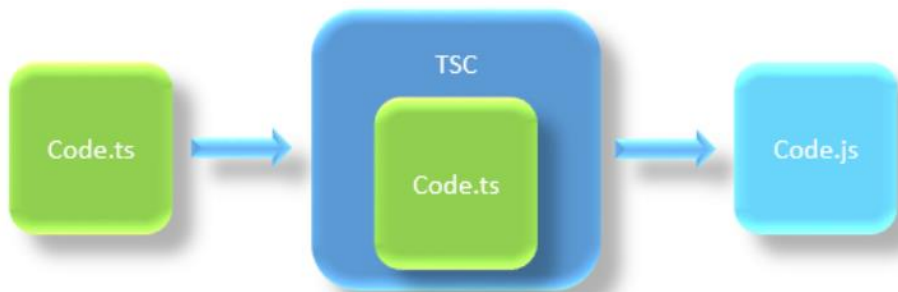


- Añade tipos estáticos a JavaScript ES6
  - Inferencias de tipos
  - Tipos opcionales
- El compilador genera código JavaScript ES5 (Navegadores actuales)
- Orientado a Objetos con clases. (No como ES5)
- Anotaciones (ES7)



# ¿Pero el browser ejecuta Typescript?

- Compilador de Typescript
  - Convierte el código TS a código JS.
  - Empaqueta todos los archivos JS en uno.
  - Minifica el archivo JS generado.
  - Lo deja listo para incluir en index.html)



# Tipos básicos

```
let m:number;  
let s:string;  
let b:boolean = false;  
let o:Persona; // no primitivo  
let list: number[] = [1, 2, 3];  
let list: Array<number> = [1, 2, 3];  
let notSure: any = 4;  
let list: any[] = [1, true, "free"];
```



# Conversiones

```
let i:number;  
let s:string;  
i = 9;  
s = i.toString(); // numero a string  
i = +s; // string a numero  
i = parseInt(s); // string a numero  
i = parseFloat(s); // string a numero
```



# Arrays

```
let lista: Array<number>;  
lista = [];  
lista = new Array<number>();  
lista.push(2);  
lista.push(3);  
lista.push(5);  
for(let i in lista){  
    console.log(lista[i]);  
}  
console.log("size:"+lista.length);
```

The TypeScript logo, consisting of a blue square with the white letters "TS" inside.

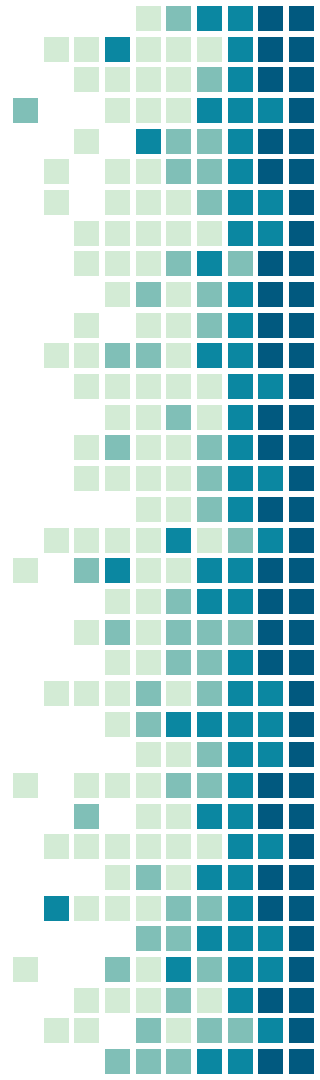
# Funciones

JS:

```
function add(x, y) {  
  return x + y;  
}
```

TS:

```
function add(x:number, y:number):number {  
  return x + y;  
}
```



# Declaraciones var



```
function f(shouldInitialize: boolean) {  
  if (shouldInitialize) {  
    var x = 10;  
  }  
  return x;  
}  
f(true); // returns '10'  
f(false); // returns 'undefined'
```



# Declaraciones let



```
function f(shouldInitialize: boolean) {  
  if (shouldInitialize) {  
    let x = 10;  
  }  
  return x;  
}
```

ERROR DE COMPILACIÓN: “Cannot find name 'x'.





# Funciones como argumentos



```
function pot(x:number):number{  
    return x*x;  
}  
  
function calcular(n:number, fun:any):number{  
    return fun(n);  
}  
  
calcular(2,pot); // Retorna '4'
```



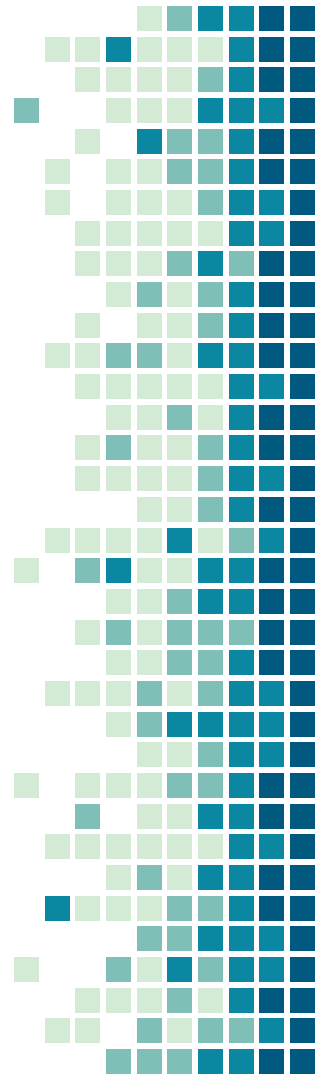
# Classes



```
class Main{  
  constructor(){  
    console.log("constructor");  
  }  
}  
  
let m:Main = new Main();
```

1 solo constructor





```
class Main{  
  private mensaje:string;  
  constructor(b:string){  
    this.mensaje=b;  
  }  
  miMetodo(a:number,):number{  
    console.log(this.mensaje);  
    return a+1;  
  }  
}  
  
let m:Main = new Main("mensaje");  
let r:number = m.miMetodo(2);  
console.log(r);
```