

# Desarrollo de Aplicaciones I

# Clase 5

- Herencia
- Interfaces
- Listener HTML
- Ajax - GET



# Herencia

```
class Animal {  
  move(dist: number = 0) {  
    console.log(`Animal moved ${dist}.`);  
  }  
}  
  
class Dog extends Animal {  
  bark() {  
    console.log('Woof! Woof!');  
  }  
}  
  
let dog: Dog = new Dog();  
dog.bark();  
dog.move(10);  
dog.bark();
```

- Herencia simple
- Múltiples interfaces



# Super

```
class Person {  
  firstName: string;  
  lastName: string;  
  constructor (fName: string, lName: string) {  
    this.firstName = fName;  
    this.lastName = lName;  
  }  
  class Employee extends Person {  
    empID: string;  
    designation: string;  
    constructor (fName: string, lName: string, eID: string, desig: string) {  
      super(fName, lName);  
      this.empID = eID;  
      this.designation = desig;  
    }  
  }  
}
```



# Métodos de la clase

```
class Animal {  
  move(dist: number = 0) {  
    console.log(`Animal moved ${dist}.`);  
  }  
}  
  
class Dog extends Animal {  
  bark() {  
    console.log('Woof! Woof!');  
  }  
  beDog() {  
    super.move(10);  
    this.bark();  
  }  
}
```



# Template Strings

- Literales de texto con expresiones incrustadas.
- Permiten más de una línea.
- Usan la tilde invertida `

```
let tpl:string = `hola mundo`;  
let tpl:string = `hola ${variable} mundo`;  
//let tpl:string = "hola "+ variable +"mundo";
```



# Interfaces

- Definen tipos de datos.
- Permite que un objeto sea de más de un tipo.
- Obligan a las clases a tener ciertos atributos y/o métodos.



# Interfaces

```
interface Hablador {  
    hablar():void;  
}  
  
class Cat implements Hablador {  
    hablar():void {  
        console.log("Miau");  
    }  
}  
  
class Humano implements Hablador {  
    hablar():void{  
        console.log("Hola mundo");  
    }  
}
```





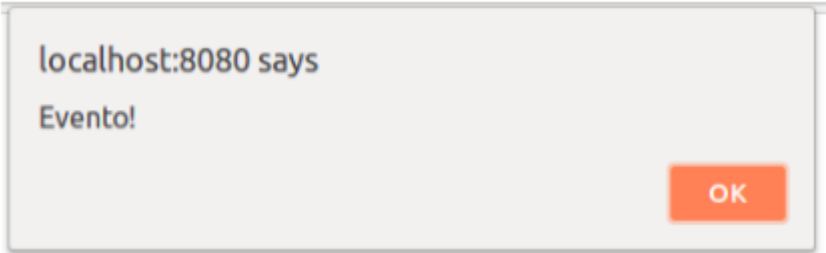
# Listener

- HTML

```
<input type="button" id="boton"/>
```

- TS

```
let b:HTMLElement = document.getElementById("boton");  
b.addEventListener("click",()=>{  
    alert("Evento!");  
});
```

An alert dialog box with a light gray background and a thin border. It contains the text "localhost:8080 says" on the first line and "Evento!" on the second line. In the bottom right corner, there is an orange button with the text "OK" in white.

localhost:8080 says  
Evento!

OK

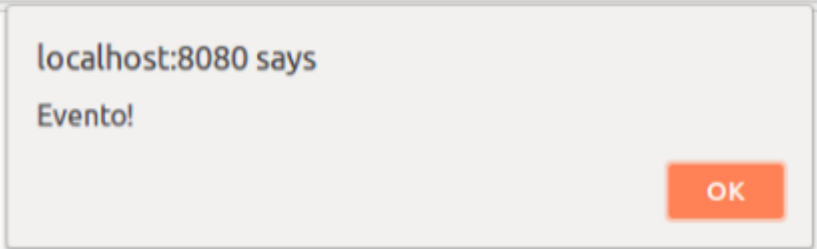
# Listener

- HTML

```
<input type="button" id="boton"/>
```

- TS

```
class MyClass implements EventListenerObject{  
    msg:string="evento!";  
    handleEvent(evt:Event):void{  
        alert(this.msg);  
    }  
}  
  
let b:HTMLElement = document.getElementById("boton");  
b.addEventListener("click",new MyClass());
```

An alert dialog box with a light gray background and a thin border. It contains the text "localhost:8080 says" in a dark gray font, followed by "Evento!" in a slightly lighter gray font. In the bottom right corner, there is an orange button with the text "OK" in white.

localhost:8080 says  
Evento!

OK

# Ajax

- **AJAX** significa JavaScript asíncrono y XML (Asynchronous JavaScript and XML)
- Es un conjunto de técnicas de desarrollo web que permiten que las aplicaciones web funcionen de forma asíncrona
- Procesa cualquier solicitud al servidor en segundo plano



# Ajax – Ejemplo

```
let xhr = new XMLHttpRequest();
xhr.onreadystatechange = function () {
  if (xhr.readyState == 4) {
    if (xhr.status == 200) {
      console.log(xhr.responseText);
    }
  }
}
xhr.open("GET", url, true);
xhr.send();
```

