Programación Funcional

Diseño con **funciones**

Agenda

- Un poco de historia...
- ¿Que es una función?
- Funciones como componentes de primera clase
- Pensando funcionalmente (funcional vs OOP)
- Predictibilidad
- Inmutabilidad

Agenda

- **Un poco de historia...**
- ¿Que es una función?
- Funciones como componentes de primera clase
- Pensando funcionalmente (funcional vs OOP)
- Predictibilidad
- Inmutabilidad

Un poco de historia...

- Proviene de la necesidad de crear un **modelo matemático** de **computación**: Calculo Lambda
  - Contraparte imperativa: Maquina de Turing

- Uso original: Investigación de lenguajes

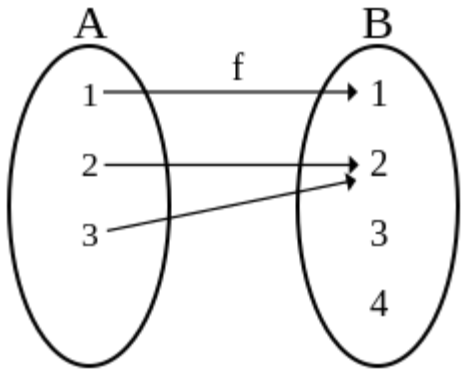
- Uso moderno: Sistemas con cambios de estado complejos
  - Concurrencia
  - Distribucion

Agenda

- Un poco de historia...
- **¿Que es una función?**
- Funciones como componentes de primera clase
- Pensando funcionalmente (funcional vs OOP)
- Predictibilidad
- Inmutabilidad

¿Que es una función?

Para cada elemento del dominio (A)



Asocia exactamente UN elemento del codominio (B)

Agenda

- Un poco de historia...
- ¿Que es una función?
- **Funciones como componentes de primera clase**
- Pensando funcionalmente (funcional vs OOP)
- Predictibilidad
- Inmutabilidad

Funciones como componentes de primera clase

Las funciones se tratan como cualquier otro valor:

- Pueden ser asignadas a variables
- Pueden ser argumentos a funciones
- Pueden ser retornadas por otras funciones
- Tienen expresiones literales

Funciones como componentes de primera clase

Funciones de orden 1:

Valor → Valor

Funciones de orden superior:

Función → Función

Son formas de **combinar funciones**

Funciones como componentes de primera clase

Funciones **importantes**:

Filter

Map

Reduce

Funciones como componentes de primera clase (**Filter**)

```
const persons = [
  { firstName: "John", lastName: "Smith", age: 21 },
  { firstName: "Charles", lastName: "William", age: 32 },
  { firstName: "Joseph", lastName: "Wagner", age: 45 }
];

persons.filter(person => person.age > 25);
// [ { firstName: "Charles", lastName: "William", age: 32 },
//   { firstName: "Joseph", lastName: "Wagner", age: 45 }
// ]
```

Funciones como componentes de primera clase (**Map**)

```
const persons = [
  { firstName: "John", lastName: "Smith", age: 21 },
  { firstName: "Charles", lastName: "William", age: 32 },
  { firstName: "Joseph", lastName: "Wagner", age: 45 }
];

persons.map(person => person.age);
// [ 21, 32, 45 ]
```

Funciones como componentes de primera clase (**Reduce**)

```
const persons = [
  { firstName: "John", lastName: "Smith", age: 21 },
  { firstName: "Charles", lastName: "William", age: 32 },
  { firstName: "Joseph", lastName: "Wagner", age: 45 }
];

persons.reduce((accumulator, currentValue) => accumulator + currentValue.age, 0);
// 98
```

Funciones como componentes de primera clase (**Combined**)

```
const persons = [
  { firstName: "John", lastName: "Smith", age: 21 },
  { firstName: "Charles", lastName: "William", age: 32 },
  { firstName: "Joseph", lastName: "Wagner", age: 45 }
];

persons
  .filter(person => person.age > 25)
  .map(person => person.age)
  .reduce((accumulator, currentValue) => accumulator + currentValue, 0);
// 77
```

Agenda

- Un poco de historia...
- ¿Que es una función?
- Funciones como componentes de primera clase
- **Pensando funcionalmente (PF vs OOP)**
- Predictibilidad
- Inmutabilidad

Pensando funcionalmente (**PF** vs **OOP**)

```
f(o)
```

```
o.f()
```

Pensando funcionalmente (PF vs OOP)

```
class User {
    constructor(firstName, lastName, yearOfBirth) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.yearOfBirth = yearOfBirth;

        this.getFullName = function(){
            return "User's name: " + this.firstName + " " + this.lastName;
        }
    }
}

const user = new User("John", "Smith", 1982);
const fullName = user.getFullName();
```

Pensando funcionalmente (**PF** vs **OOP**)

```
const user = { firstName: "John", lastName: "Smith", yearOfBirth: 1982 };

const getFullName = (user) => {
  return "User's name: " + user.firstName + " " + user.lastName;
}

getFullName(user);
```

Agenda

- Un poco de historia...
- ¿Que es una función?
- Funciones como componentes de primera clase
- Pensando funcionalmente (funcional vs OOP)
- **Predictibilidad**
- Inmutabilidad

Predictibilidad

Ejecutamos una subrutina repetidas veces:

1ra ejecución: update() → 0

2da ejecución: update() → 1

¿Qué podemos esperar de una 3ra ejecución?

a) 0	b) 1	c) 2
d) -2	e) 4	f) ?

Predictibilidad

Predecible: Siempre produce la misma salida para una determinada entrada.

Algunas dependencias que rompen predictibilidad:

- Aleatoriedad
- Operaciones Entrada/Salida
  - Red
  - Sistema de archivos
  - Usuario
- Estado variable
  - Tiempo
  - Ubicación

Agenda

- Un poco de historia...
- ¿Que es una función?
- Funciones como componentes de primera clase
- Pensando funcionalmente (funcional vs OOP)
- Predictibilidad
- **Inmutabilidad**

Inmutabilidad

Inmutabilidad: Evaluar una función no modifica el estado del programa.
En lugar de actualizar valores, se crean nuevos valores.

Programación Funcional = Programación sin Estado Mutable

- Inmutabilidad ⇒ Predictibilidad
  No hay que usar para variar los resultados
- Predictibilidad ⇒ Inmutabilidad
  No hay utilidad en mutar valores si no se pueden usar