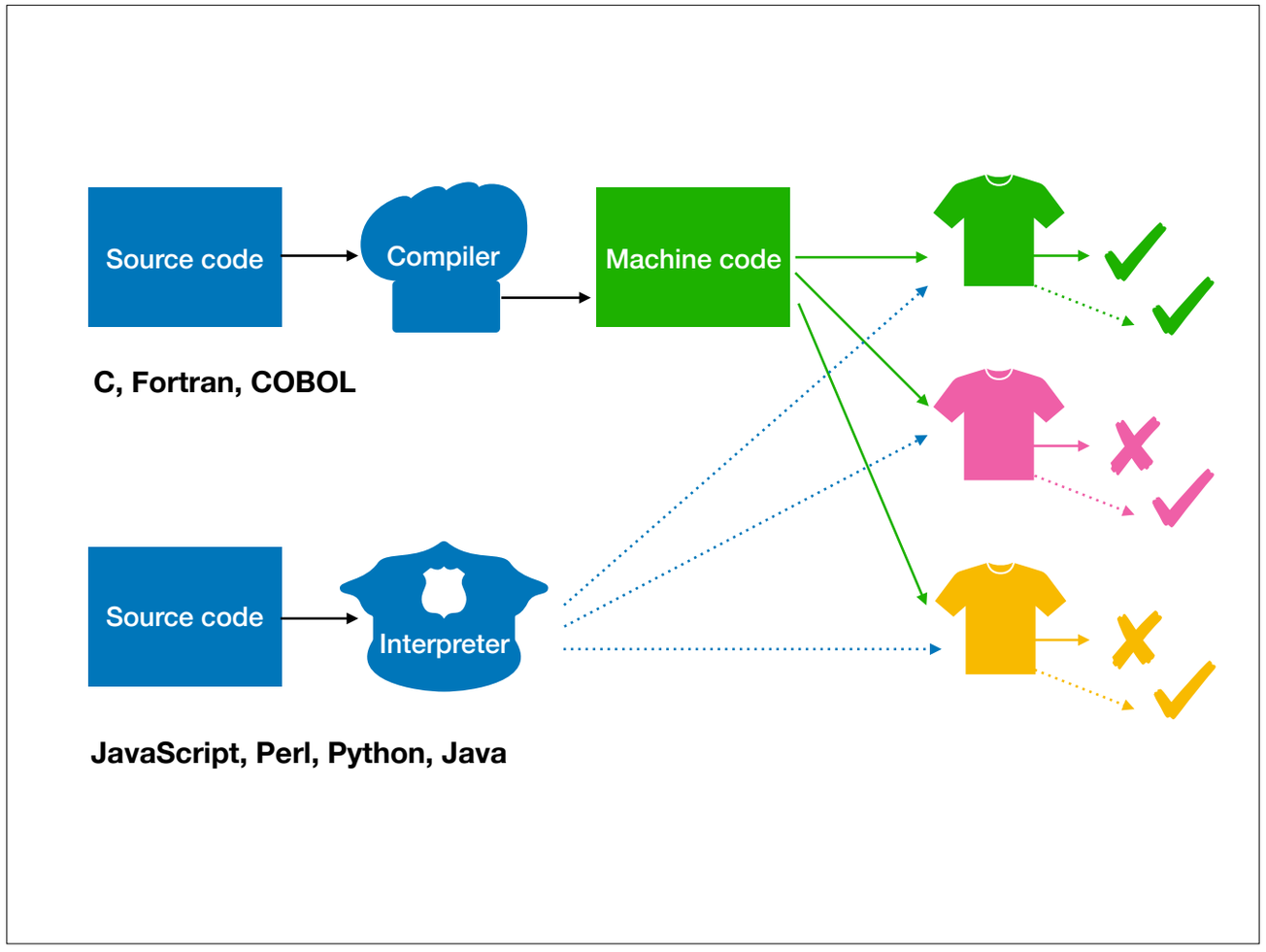
JavaScript

JavaScript (JS) - Brendan Eich. Dic, 1995

- Lenguaje orientado a objetos, ligero e **interpretado**
- Mejor conocido cómo el lenguaje para la Web, aunque es utilizado también fuera del navegador cómo en node.js
- Lenguaje multi paradigma, soporta diferentes estilos de programación: orientado a objetos, imperativo y funcional

- Netscape - *best browser!!* en ese momento
- 1990 Java de Sun systems, un lenguaje para “smart appliances”
- Brendan Eich 1995, Netscape’s Mocha como el compañero de Java así como Visual Basic era de C++



What is the difference between a compiled and an interpreted program? https://kb.iu.edu/d/agsz

Paradigmas de programación

- **Programación orientada a objetos:** basado en organizar el código en objetos que encapsulan valor y comportamiento (métodos o funciones)

- **Programación imperativa:** describe explícitamente *cómo* conseguir un resultado, el programa se compone de una serie de pasos para cambiar el estado. Al contrario de la **programación declarativa** que describe *qué* debe hacer el programa

- **Programación funcional:** basado en organizar el código en funciones, pero las funciones son entidades de primera clase que se pueden componer de varias maneras según el *calculo lambda*. Evita cambiar el estado y mutar los datos.

First-class object: entidades u objetos que pueden ser usados en el lenguaje en cuestión, como cualquier otro objeto o entidad con todas las operaciones disponibles. Por lo general las operaciones disponibles incluyen pasar el objeto como argumento, ser resultado de una función, ser modificado, asignado a una variable, entre otros.
Ej: en C no puedes pasar una función como parámetro a otra función, es decir: second-class objects, pero en JS si es posible, por lo que son first-class objects.

Variables

Comentarios

```
var myVariable;
myVariable = 'Bob'; // 10, true, [ 1, 'a', false ]
/*
    Esto también es un comentario
*/
```

Operadores

```
6 + 9;
“Hello ” + “world!”;
9 - 3;
8 * 2;
9 / 3;

myVariable = ‘Bob';

myVariable === 4;
myVariable !== 3;

// ||, &&, !
```

Scope

var
- Pueden ser declaradas y actualizadas 🤯 incluso dentro de un scope más profundo
- *Hoisted to the top* como *undefined*

let
- Block scoped: en diferentes scopes se toma cómo diferentes variables
- Pueden ser actualizadas pero no re declaradas
- *Hoisted to the top* pero no es inicializado y mostrará un Reference Error si se intenta utilizar

const
- Block scoped: solo se puede acceder a ellas en el scope dónde fueron declaradas
- No pueden ser actualizadas ni re declaradas
- Debe ser inicializada cuando sea declarada

Truthy and Falsy

```
/* falsy

0
“” or ‘’
null
undefined
NaN

*/
```

Short-circuit evaluation

```
let defaultName = username || ‘John Doe’;
```

Ternary operator

```
 isSnowing ? console.log(‘OMGGGGG!! That’s so cool!’) :
console.log('How sad');
```

Condicionales

```
var iceCream = ‘chocolate’;

if (iceCream === 'chocolate') {
  alert(‘Yay, me encanta el chocolate!');

} else if (iceCream === 'papaya') {
  alert(‘Ay! Que horror’);

} else {
  alert('Awwww, pero yo quería chocolate...');

}
```

Funciones

```
function multiply(num1,num2) {
  var result = num1 * num2;
  return result;
}

multiply(4, 5); // 20
```

Objetos

```
var person = {
   name: 'Yvone',
   favoriteIceCream: 'chocolate',
};

person.name // “Yvone”
person['favoriteIceCream'] // “chocolate”
person.age = 24;
person // {name:”Yvone”, favoriteIceCream:"chocolate", age:24}
```

Arreglos

```
var people = []
people.push(person)

var first = people[0]

people.pop(); // shift()

people.forEach(function(item, index, array) {
   console.log(item, index);
});
```

```
// Concat strings

let newString = `My name is ${obj.name},
nice to meet you!`;

// De structuring properties

const { name, lastName, age } = obj;

// Spread syntax

const newArray = [ …arrA, …arrB ];
```

```
// Reduce

let total = sales.reduce( (acc, cur) => acc + cur);

// Map

let totalWithTaxes = sales.map( v => v * 1.16);

// Filter

let mostProfit = sales.filter( v => v > 100);

// Loop

sales.forEach( sale =>
 console.log(`Total: ${sale.total}`)
);
```

Debug

```
// Computed property names

console.log({ foo, bar })

// Table

console.table([ fooObject, barObject ])

// Adding css

console.log(“c% Hello”, “color: orange;
font-weight: bold;”)
```

Laboratorio 2

- Crea una cuenta en codewars.com
- Completa 3+ katas en JS