

# Swift 101

## Constantes, Variables, y Tipos de Datos

# Constantes y variables

Asociamos un nombre con un valor

Definimos si será constante o variable

- Alojamos espacio en memoria para el o los valores
- Asociamos la constante con el valor asignado

# Constantes

Serán datos que no cambiarán durante la ejecución del programa

Definimos una constante usando la palabra clave `let`

```
let name = "John"
```

```
let pi = 3.14159
```

Una vez que asignamos una constante no le podemos reasignar otro valor

```
let name = "John"
```

```
name = "James"
```



Cannot assign to value: 'name' is a 'let' constant

# Variables

Serán datos que estarán cambiando durante el tiempo de ejecución del programa

Definimos una variable usando `var` keyword

```
var age = 29
```

Podemos reasignar nuevos valores a la variable

```
var age = 29
```

```
age = 30
```

```
let defaultScore = 100
var playerOneScore = defaultScore
var playerTwoScore = defaultScore

print(playerOneScore)
print(playerTwoScore)

playerOneScore = 200
print(playerOneScore)
```

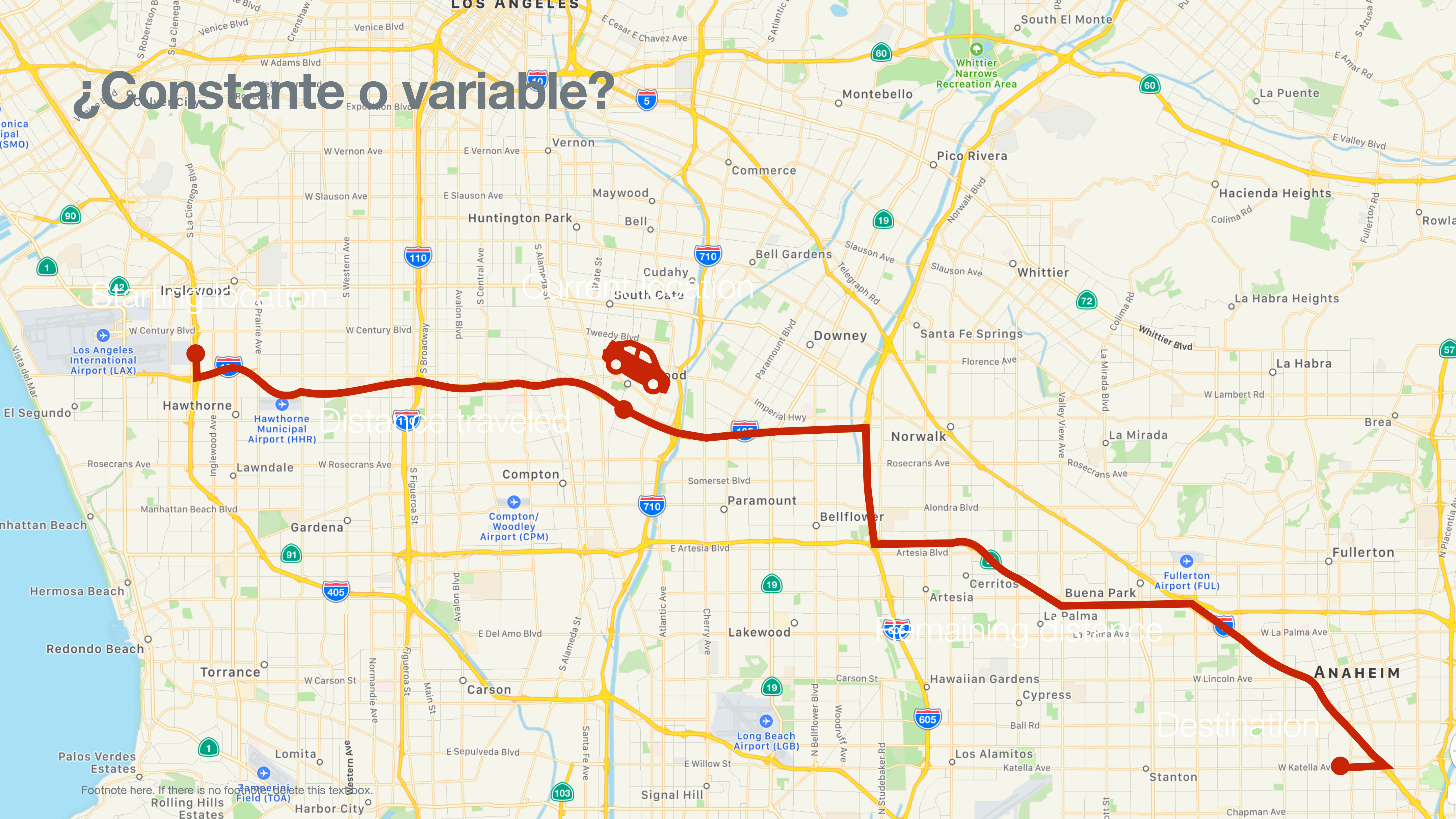
100

100

200



# ¿Constante o variable?



Starting location

Current location

Distance traveled

Remaining distance

Destination

Footnote here. If there is no footnote, delete this text box.



# Nombrando constantes y variables

## Reglas

No símbolos matemáticos

No espacios

No puede empezar con un número

```
let  $\pi$  = 3.14159
```

```
let 一百 = 100
```

```
let 🎲 = 6
```

```
let mañana = "Tomorrow"
```

```
let anzahlDerBücher = 15 //numberOfBooks
```

# Nombrando constantes y variables

## Mejores prácticas

### 1. Ser claro y descriptivo

✗ `n`

✓ `firstName`

### 2. Usar “camel case” cuando existan mas palabras

✗ `firstname`

✓ `firstName`



# Comentarios

```
// Setting pi to a rough estimate
```

```
let  $\pi$  = 3.14
```

```
/* The digits of pi are infinite,  
so instead I chose a close approximation.*/
```

```
let  $\pi$  = 3.14
```

# Tipos de datos

```
struct Person {  
    let firstName: String  
    let lastName: String  
  
    func sayHello() {  
        print("Hello there! My name is \(firstName) \(lastName).")  
    }  
}
```

```
struct Person {  
    let firstName: String  
    let lastName: String  
  
    func sayHello() {  
        print("Hello there! My name is \(firstName) \(lastName).")  
    }  
}
```

```
let aPerson = Person(firstName: "Jacob", lastName: "Edwards")  
let anotherPerson = Person(firstName: "Candace", lastName: "Salinas")
```

```
aPerson.sayHello()  
anotherPerson.sayHello()
```

Hello there! My name is Jacob Edwards.

Hello there! My name is Candace Salinas.

# Tipos mas comunes

	Symbol	Purpose	Example
Integer	<code>Int</code>	Represents whole numbers	<code>4</code>
Double	<code>Double</code>	Represents numbers requiring decimal points	<code>13.45</code>
Boolean	<code>Bool</code>	Represents true or false values	<code>true</code>
String	<code>String</code>	Represents text	<code>"Once upon a time..."</code>



# Type safety

```
let playerName = "Julian"  
var playerScore = 1000  
var gameOver = false  
playerScore = playerName
```



Cannot assign value of type 'String' to type 'Int'

```
var wholeNumber = 30  
var numberWithDecimals = 17.5  
wholeNumber = numberWithDecimals
```



Cannot assign value of type 'Double' to type 'Int'

# Type inference

```
let cityName = "San Francisco"  
let pi = 3.1415927
```

# Type annotation

```
let cityName: String = "San Francisco"  
let pi: Double = 3.1415927
```

```
let number: Double = 3  
print(number)
```

3.0

# Type annotation

## Three common cases

1. When you create a constant or variable before assigning it a value

```
let firstName: String  
//...  
firstName = "Layne"
```



# Type annotation

## Three common cases

2. When you create a constant or variable that could be inferred as two or more different types

```
let middleInitial: Character = "J"  
var remainingDistance: Float = 30
```

# Type annotation

## Three common cases

### 3. When you add properties to a type definition

```
struct Car {  
    let make: String  
    let model: String  
    let year: Int  
}
```

# Required values

```
var x
```



Type annotation missing in pattern


# Required values

```
var x: Int
```



# Required values

```
var x: Int  
print(x)
```

 Variable 'x' used before being initialized

# Required values

```
var x: Int
```

```
x = 10
```

```
print(x)
```

10

# Number formatting

```
var largeUglyNumber = 1000000000  
var largePrettyNumber = 1_000_000_000
```