Constants, Variables, and Data Types

Constants and variables

Associate a name with a value

Defining a constant or variable

- Allocates storage for the value in memory
- Associate the constant name with the assigned value

Constants

Defined using the `let` keyword

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

Defined using the let keyword

```
let pi = 3.14159
```

Can’t assign a constant a new value

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

Cannot assign to value: ‘name’ is a ‘let’ constant

Variables

Defined using the `var` keyword

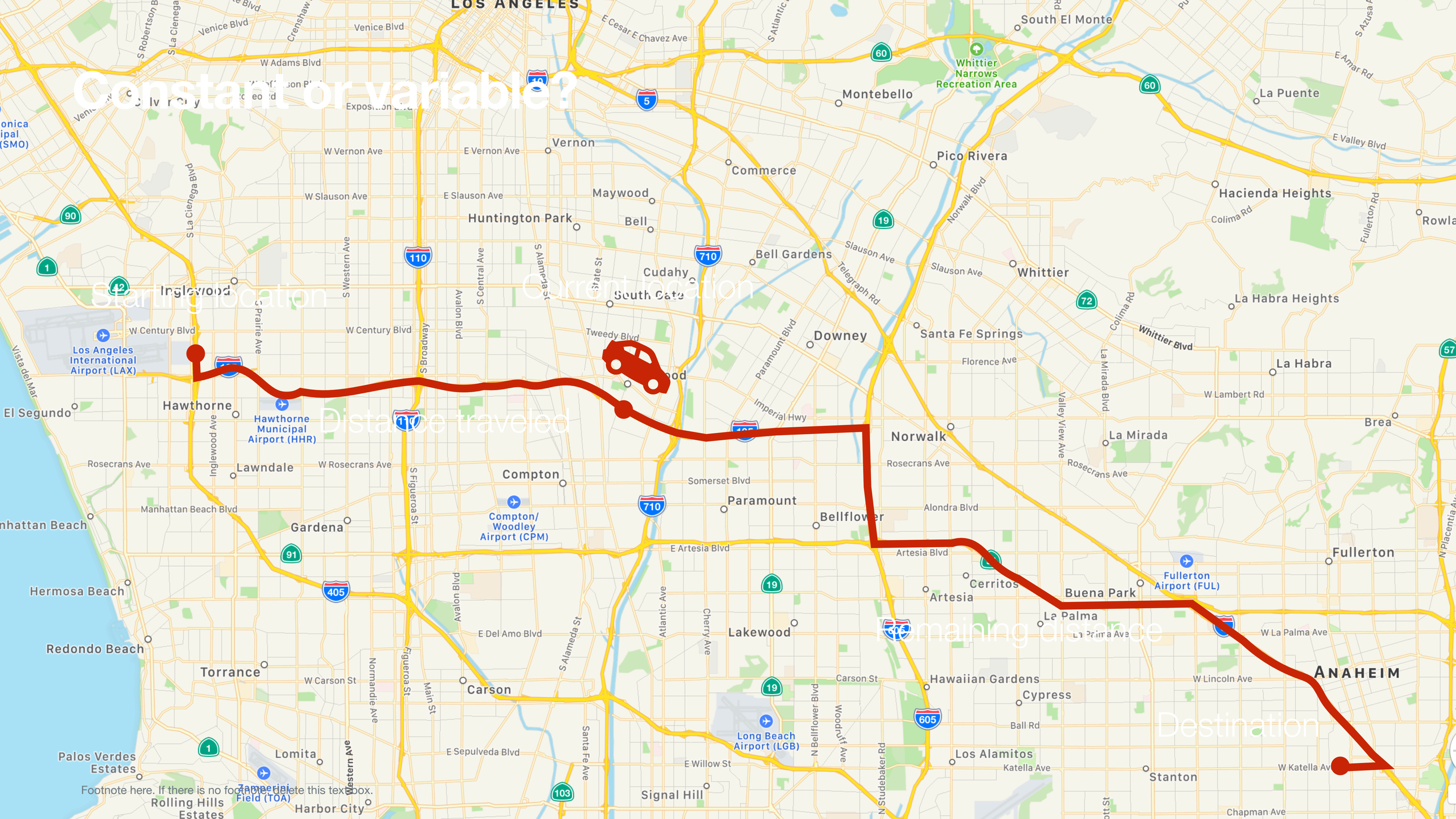
```
var age = 29
```

Can assign a new value to a variable

```
var age = 29

age = 30
```

Constant or variable?



Footnote here. If there is no footnote, delete this textbox.

Naming constants and variables

Rules

No mathematical symbols

No spaces

Can’t begin with a number

```
let π = 3.14159
let 一百 = 100
let 🎲 = 6
let mañana = "Tomorrow"
let anzahlDerBücher = 15 //numberOfBooks
```

Naming constants and variables

Best practices

1. Be clear and descriptive

   ✗ `n` ✓ `firstName`

2. Use camel case when multiple words in a name

   ✗ `firstname` ✓ `firstName`

Comments

```
// Setting pi to a rough estimate
let π = 3.14
```

```
/* The digits of pi are infinite,
so instead I chose a close approximation.*/
let π = 3.14
```

Types

```
struct Person {
  let firstName: String
  let lastName: String

  func sayHello() {
    print("Hello there! My name is \(firstName) \(lastName).")
  }
}
```

```swift
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.
```

Most common types

	Symbol	Purpose	Example
Integer	`Int`	Represents whole numbers	`4`
Double	`Double`	Represents numbers requiring decimal points	`13.45`
Boolean	`Bool`	Represents true or false values	`true`
String	`String`	Represents text	`"Once upon a time…"`

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 = 30.0
```

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
```