CIS-444

Intro to MVC

Review from Last Class:

- iOS Architecture
- Xcode
- Basic swift principles

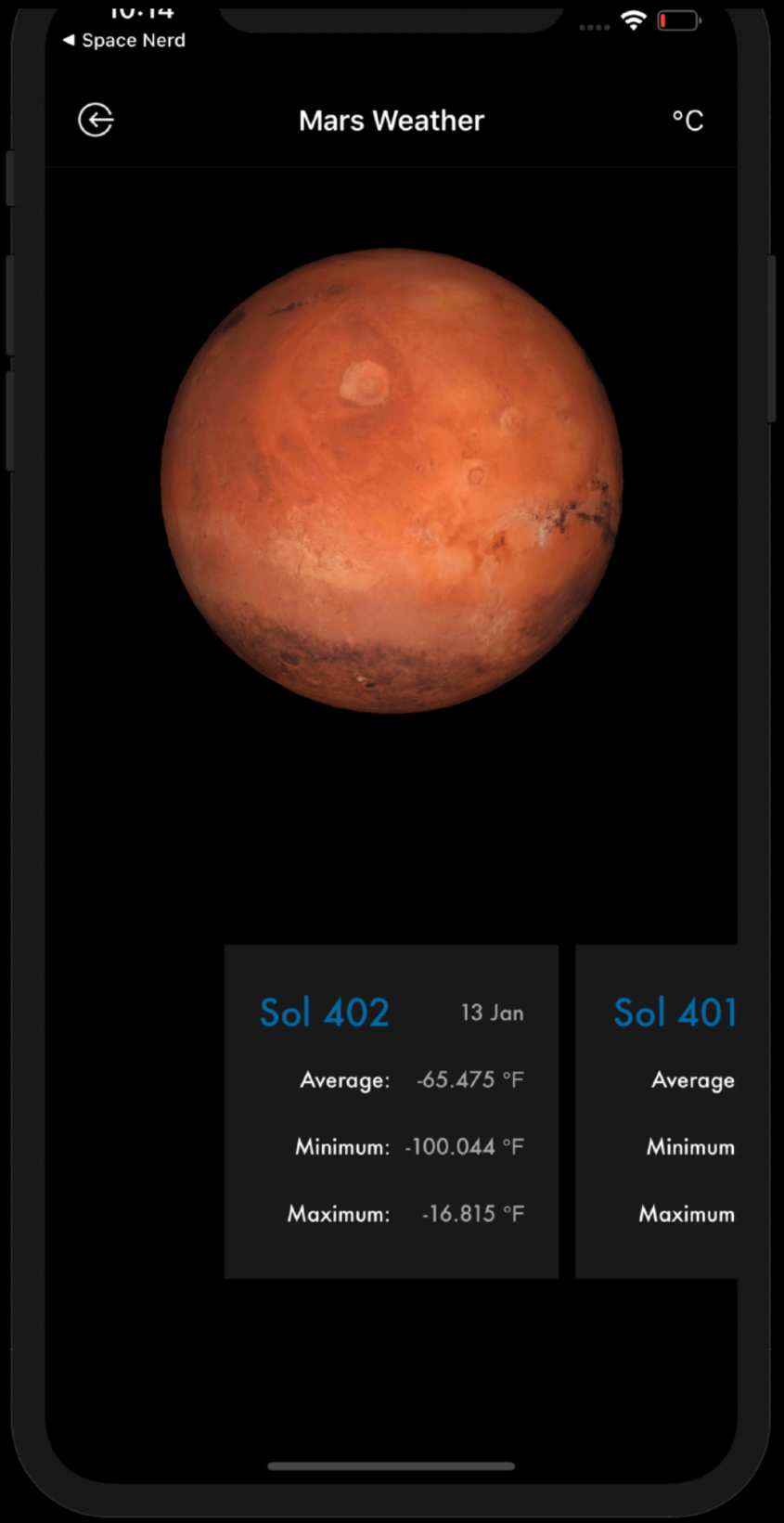
Today

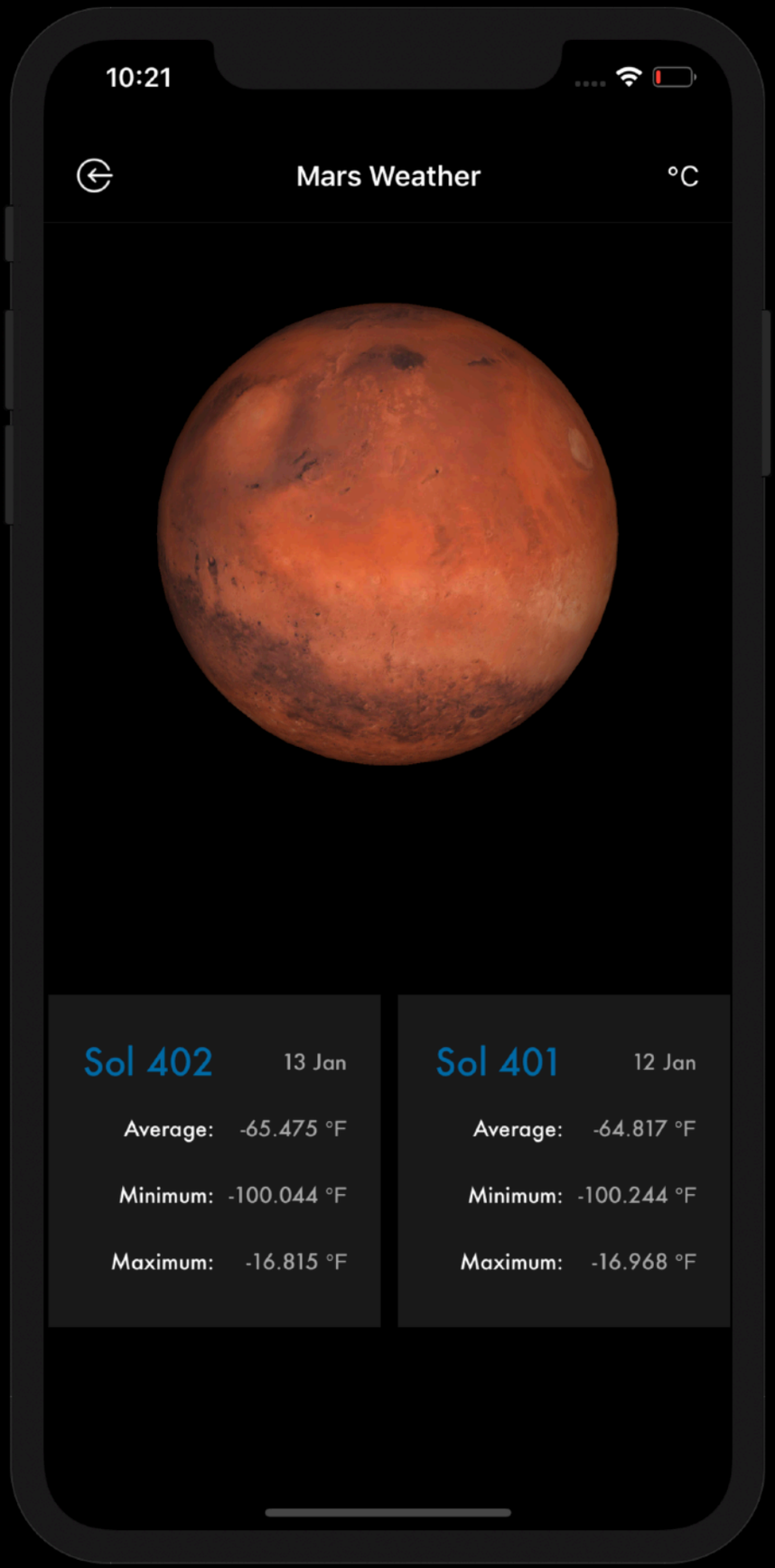
- Introducing: NASA app
- iOS Design Pattern: MVC
- Swift Syntax
- Demo

NASA API


Eye in the sky
Rover Postcard Maker
-70.811 °F | -57.117 °C
Weather on Mars
iPhone 11 Pro Max — 13.1


-70.811 °F | -57.117 °C
Weather on Mars


10:14
Space Nerd
Mars Weather
°C
Sol 402
13 Jan
Average: -65.475 °F
Minimum: -100.044 °F
Maximum: -16.815 °F
Sol 401
Average
Minimum
Maximum



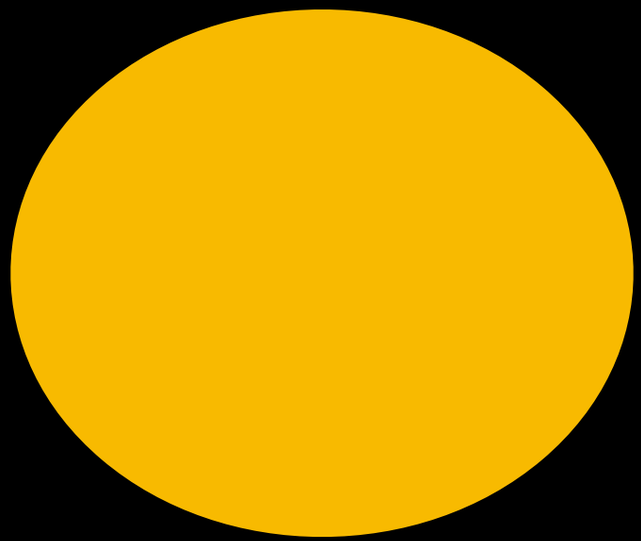
- **Mars Weather Service API** provides per-Sol summary data for each of the last seven available Sols (Martian Days).

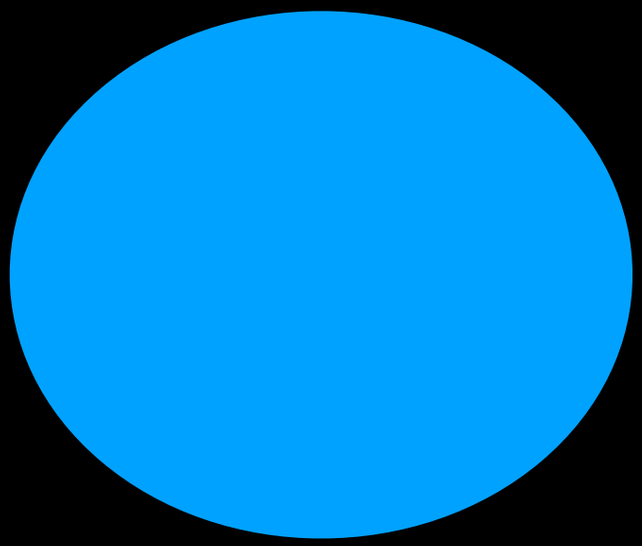
MVC

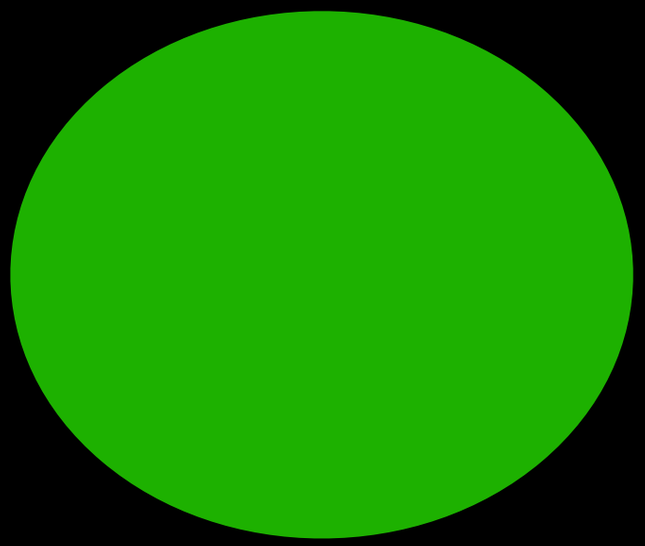
MVC

Model View Controller

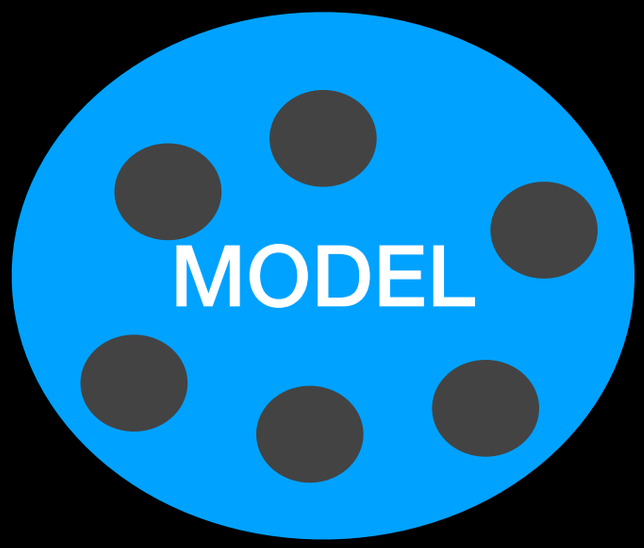
MVC

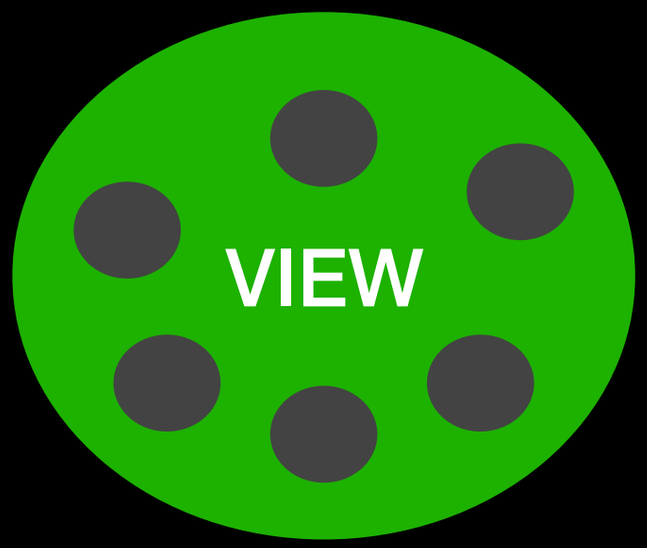




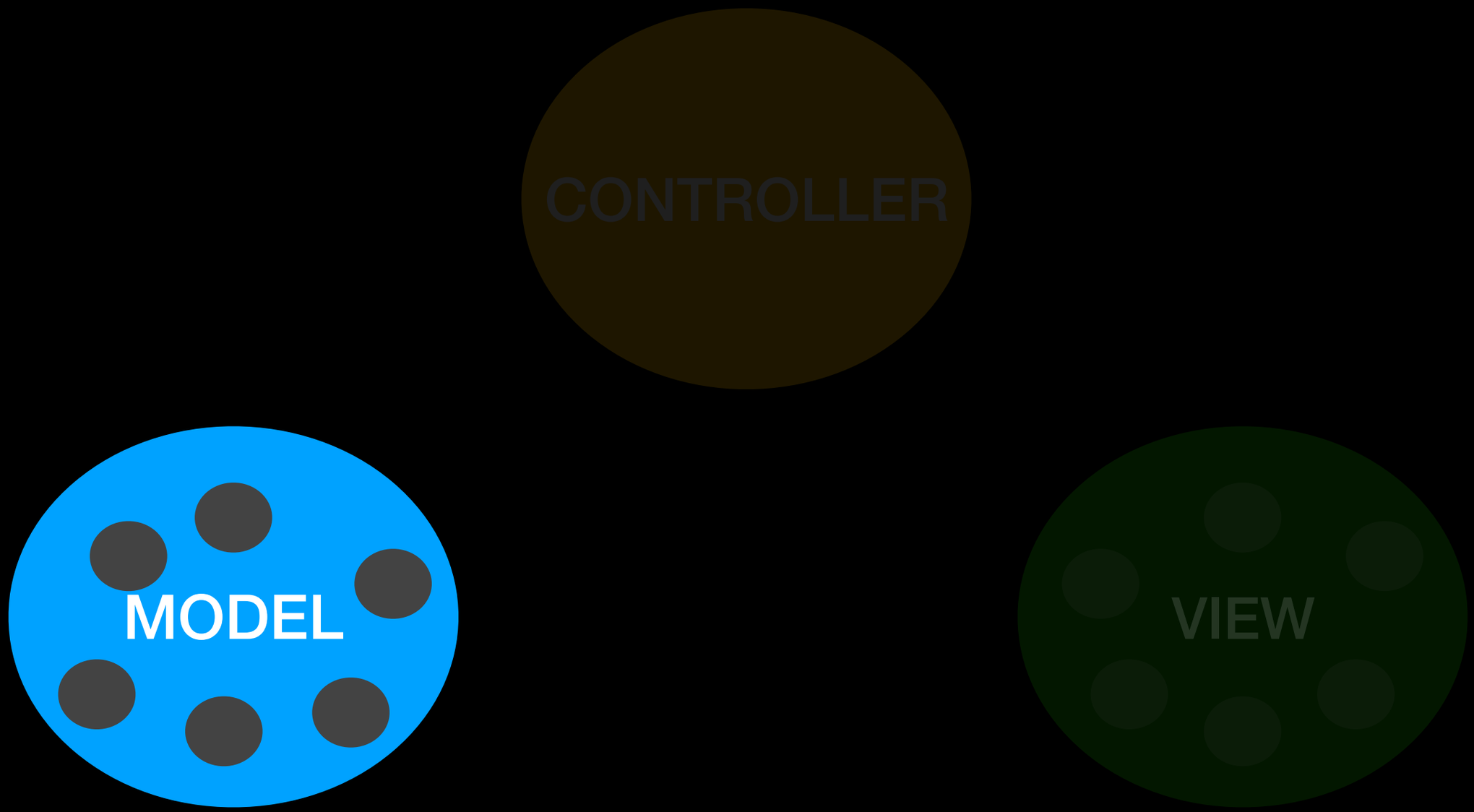


MVC



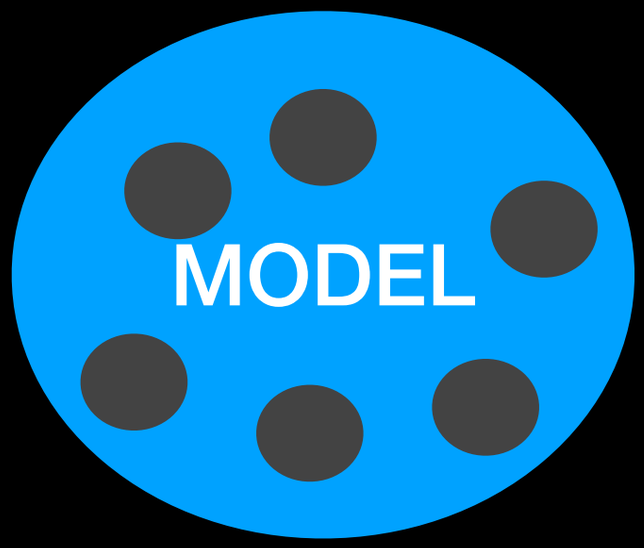


Model: The “What” of your app



Model = what your model IS (but not how it is displayed)

Model: The “What” of your app




```
EXAMPLE:

class MarsSol: {

    /// Properties
    var name: String = ""
    var temperature: MarsTemperature
    let earthDate: String
}

class MarsTemperature {

    /// Properties
    var averageTemperature: Double
    var minTemperature: Double
    var maxTemperature: Double
    var averageTemperatureFahrenheit: Double
    var minTemperatureFahrenheit: Double
    var maxTemperatureFahrenheit: Double
}
```

Model = what your model IS (but not how it is displayed)

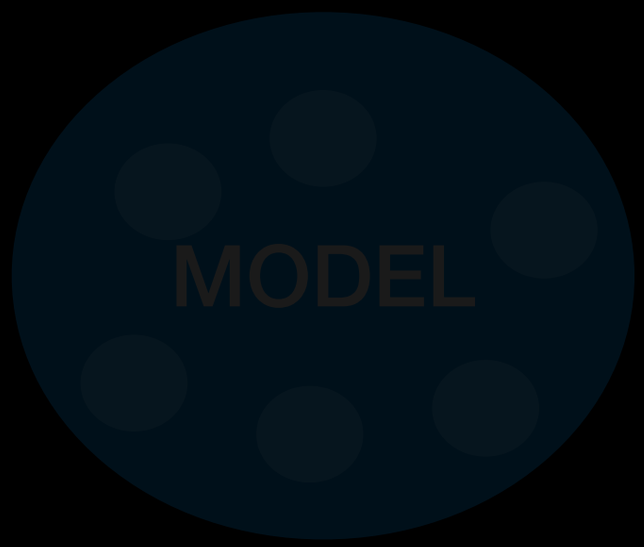
Model: The “WHAT” of your app

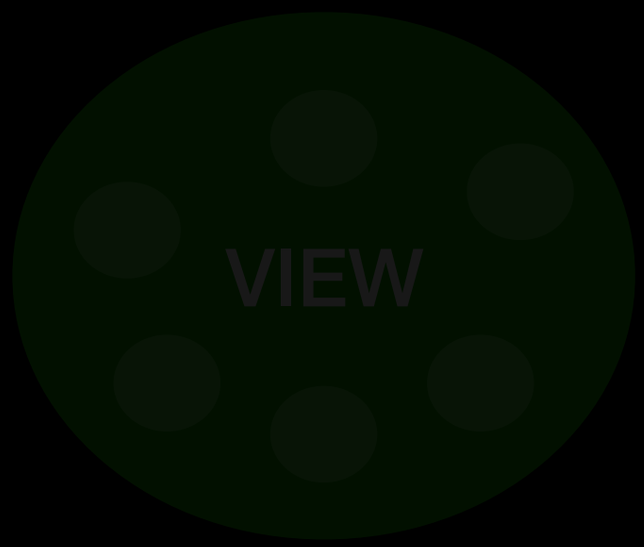
```
JSON Snippet:
{
  "396": {
    "AT": {
      "av": -70.811,
      "ct": 203315,
      "mn": -100.133,
      "mx": -18.183
    },
    "First_UTC": "2020-01-07T02:27:28Z",
    "HWS": {
      "av": 6.006,
      "ct": 92762,
      "mn": 0.205,
      "mx": 21.945
  },
  "Last_UTC": "2020-01-08T03:07:02Z",
  "PRE": {
    "av": 641.809,
    "ct": 103720,
    "mn": 624.5289,
    "mx": 656.2379
  },
    }
},…..
```

```
EXAMPLE:

class MarsSol: {

    /// Properties
    var name: String = ""
    var temperature: MarsTemperature
    let earthDate: String
}

class MarsTemperature {

    /// Properties
    var averageTemperature: Double
    var minTemperature: Double
    var maxTemperature: Double
    var averageTemperatureFahrenheit: Double
    var minTemperatureFahrenheit: Double
    var maxTemperatureFahrenheit: Double
}
```

Model = what your model IS (but not how it is displayed)

MVC





Controller = <u>HOW</u> your Model is presented to the User (UI Logic)

MVC

```swift
class MainViewController: UIViewController {

    // MARK: - Outlets
    @IBOutlet weak var startStackView: UIStackView!
    @IBOutlet weak var eyeInTheSkyView: UIView!
    @IBOutlet weak var roverPostcardMakerView: UIView!
    @IBOutlet weak var marsWeatherView: UIView!
    @IBOutlet weak var marsTemperatureLabel: UILabel!

    // MARK: - Properties
    let client = NASAClient()

    override var prefersStatusBarHidden: Bool {
        return true
    }

    // MARK: - View Life Cycle

    override func viewDidLoad() {  }

    override func viewDidAppear(_ animated: Bool) {}

    override func viewWillLayoutSubviews() {}


    /// Check if internet connection is available
    func checkForInternetConnection() {}

    /// Gets and displays the latest temperature information avalable from the NASA Insight API
    func getLatestMarsTemperature() {}
}
```

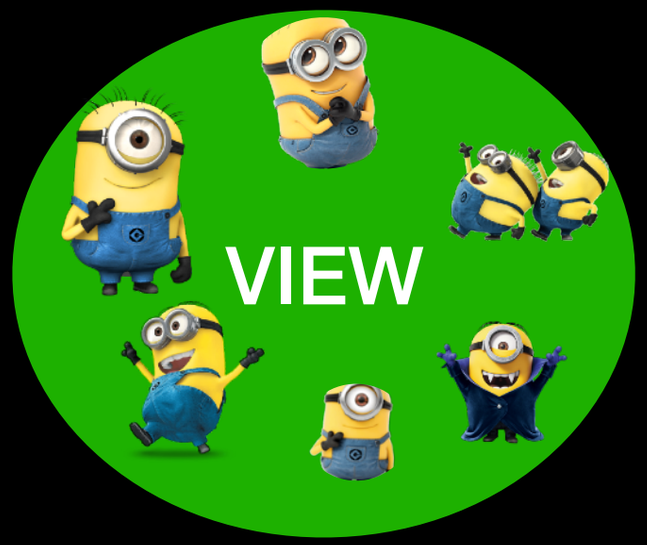
CONTROLLER

Controller = HOW your Model is presented to the User (UI Logic)

MVC

CONTROLLER

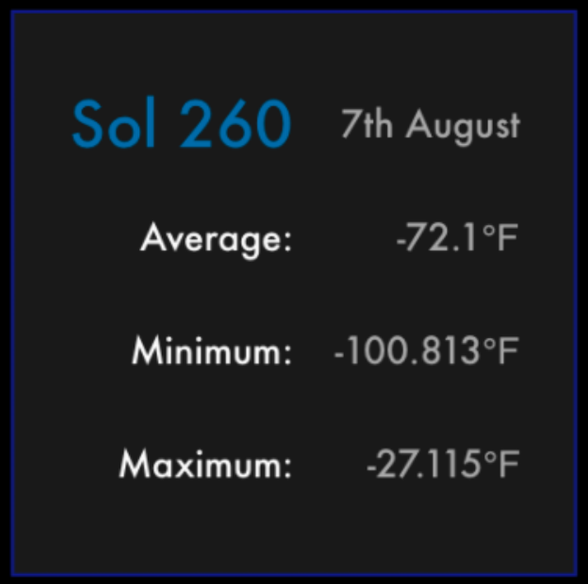
MODEL



VIEW = Your controller’s minions

MVC

View Example:

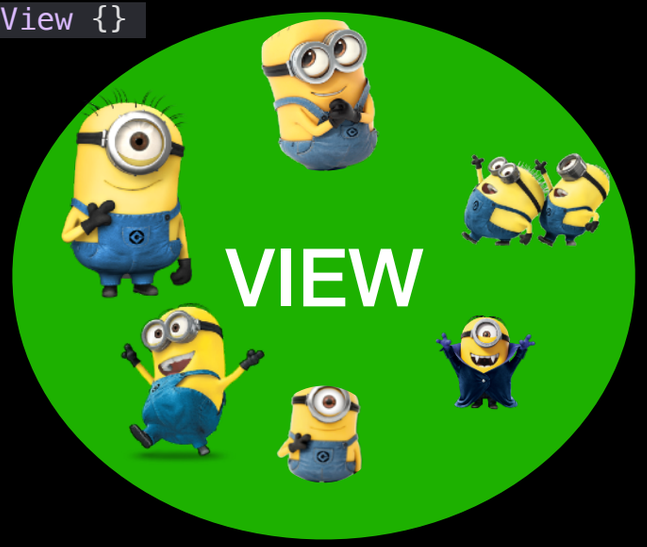


View Example:

```
class MarsWeatherCell: UICollectionViewCell {

    // MARK: - Outlets
    @IBOutlet weak var solLabel: UILabel!
    @IBOutlet weak var earthDateLabel:
UILabel!
    @IBOutlet weak var averageTempLabel:
UILabel!
    @IBOutlet weak var minTempLabel: UILabel!
    @IBOutlet weak var maxTempLabel: UILabel!
}
```

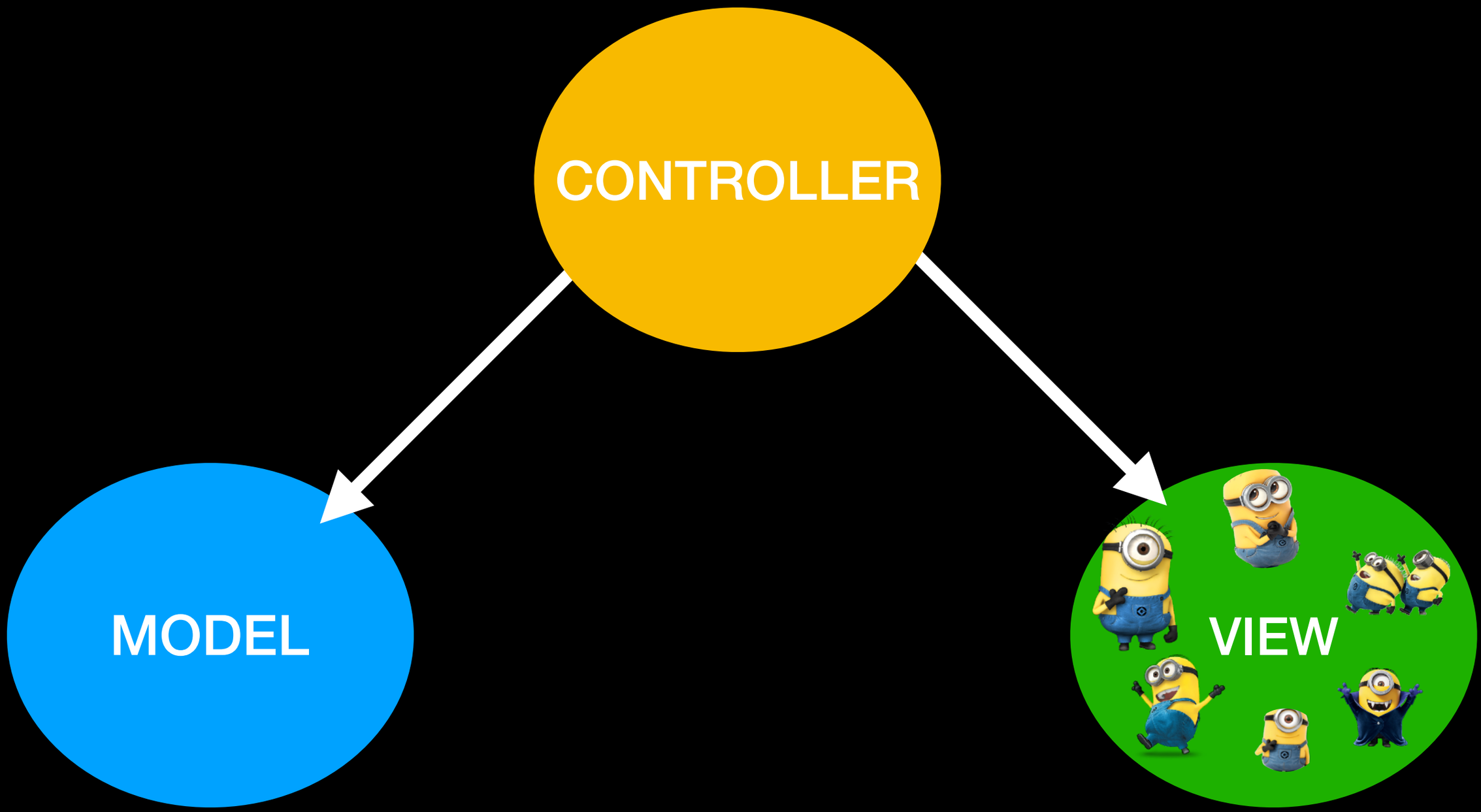
View Example:

```
class MarsSceneView: SCNView {}
```



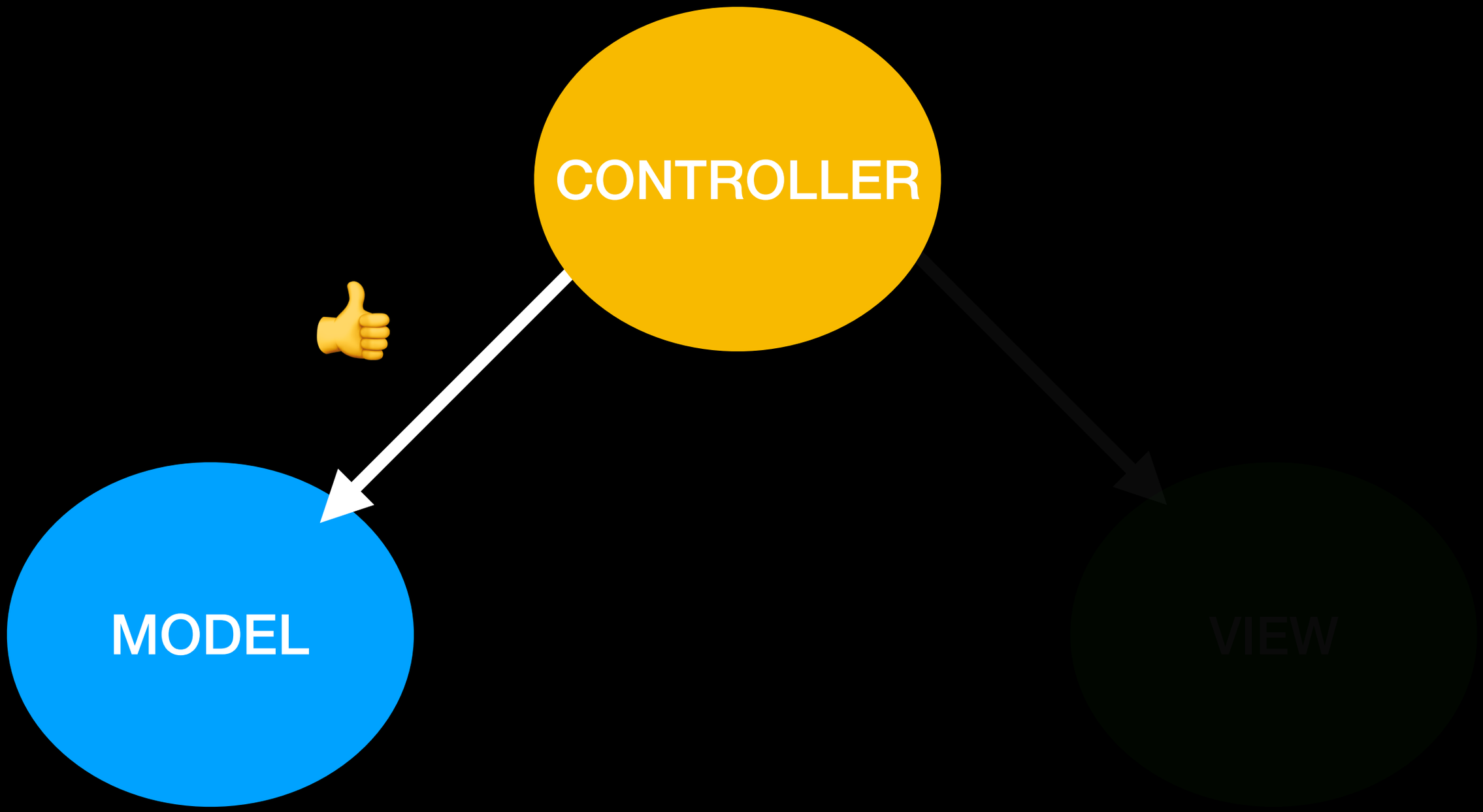
VIEW = Your controller’s minions

MVC



Managing communication between layers

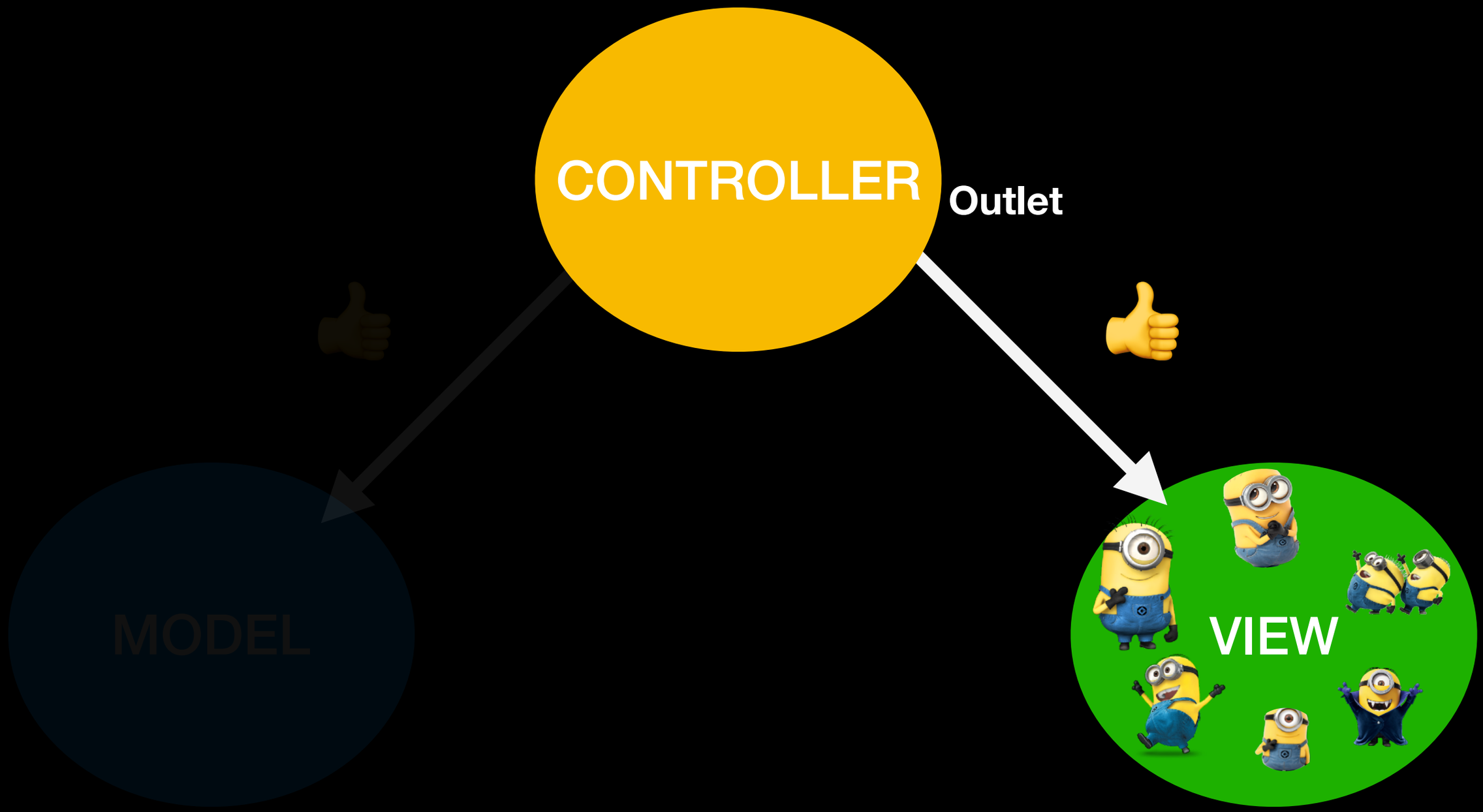
MVC



CONTROLLERs can always talk directly to their **MODELs**

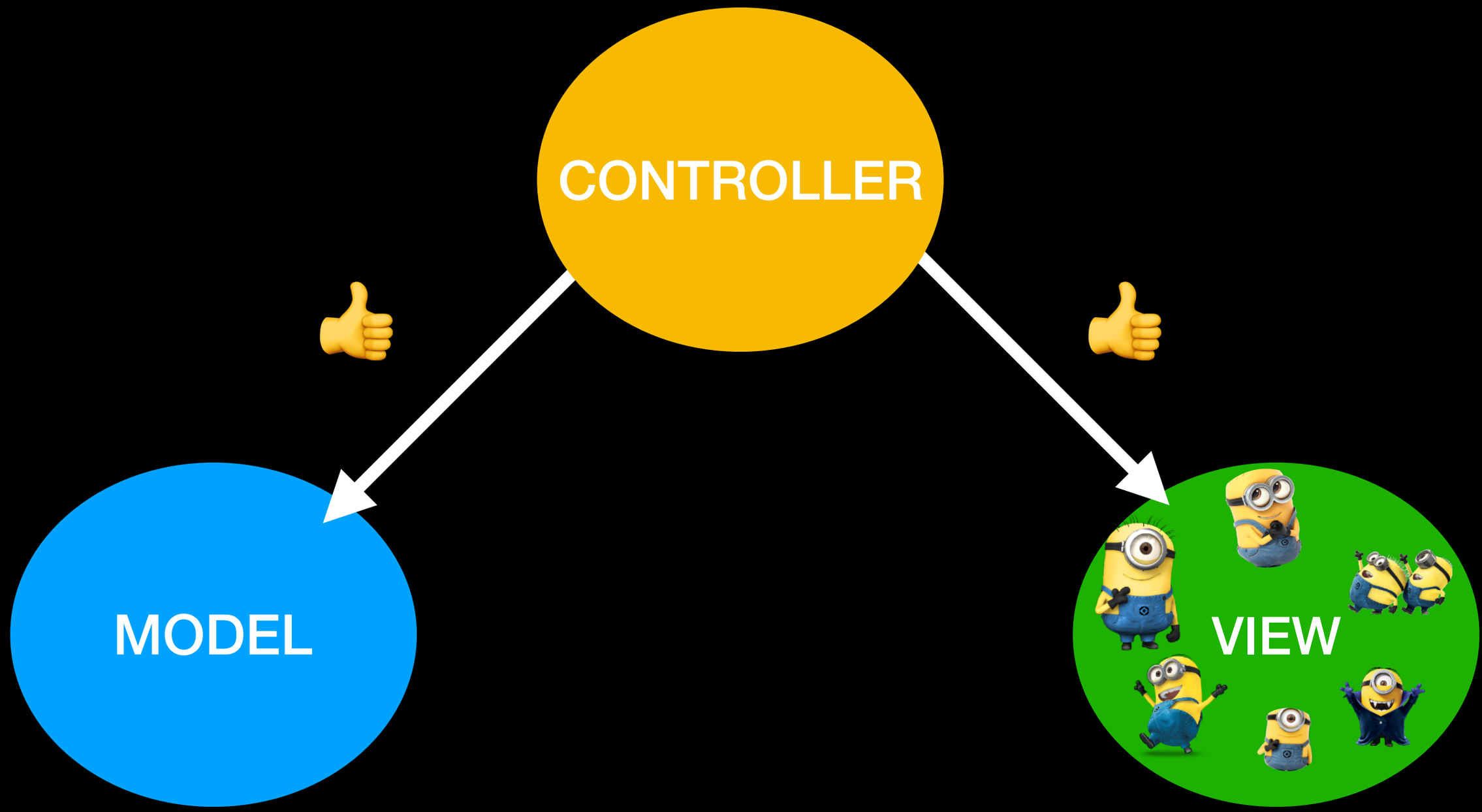
MVC

Managing communication between layers

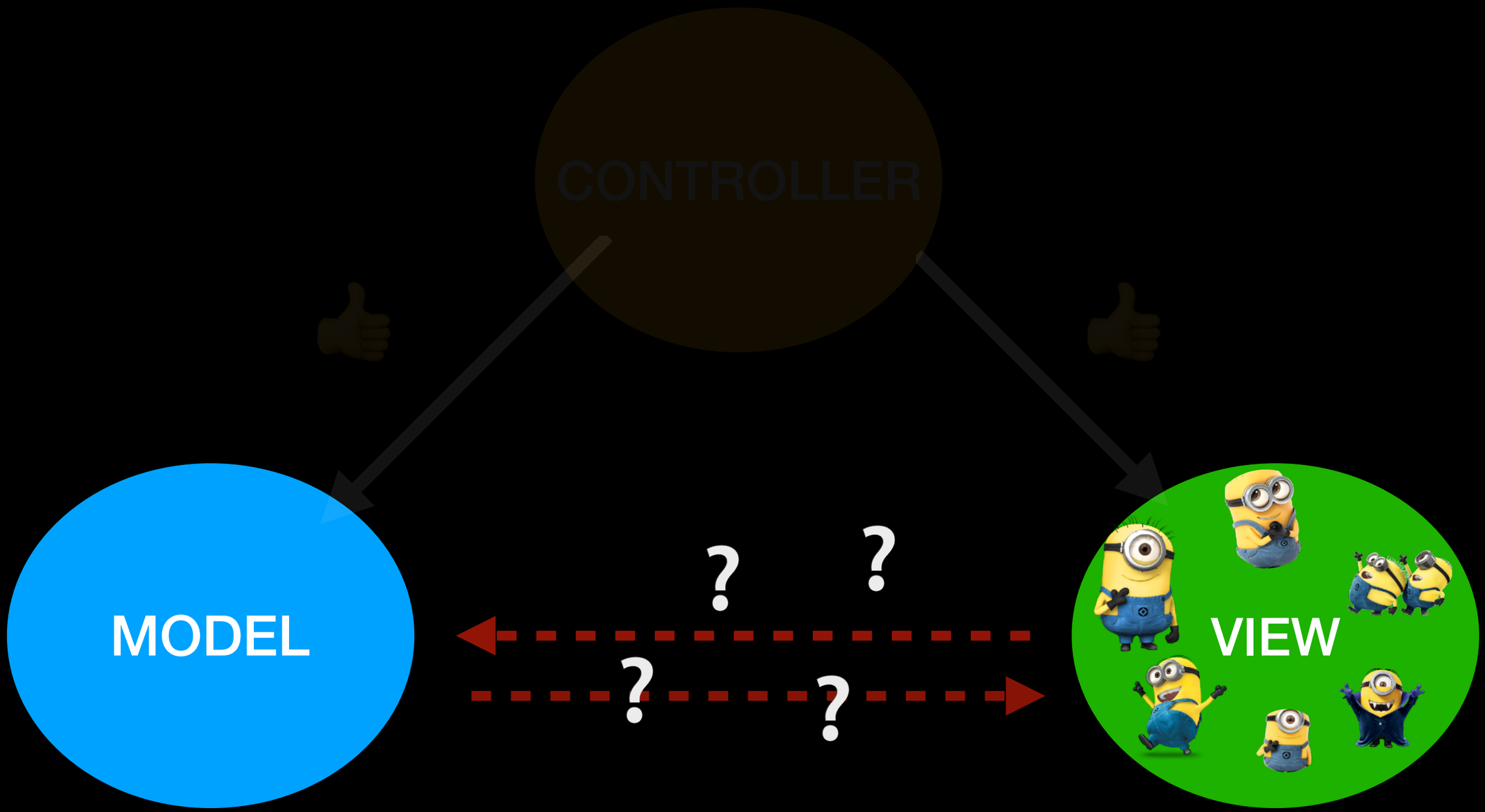


Controllers can talk directly to their **VIEW**S

MVC



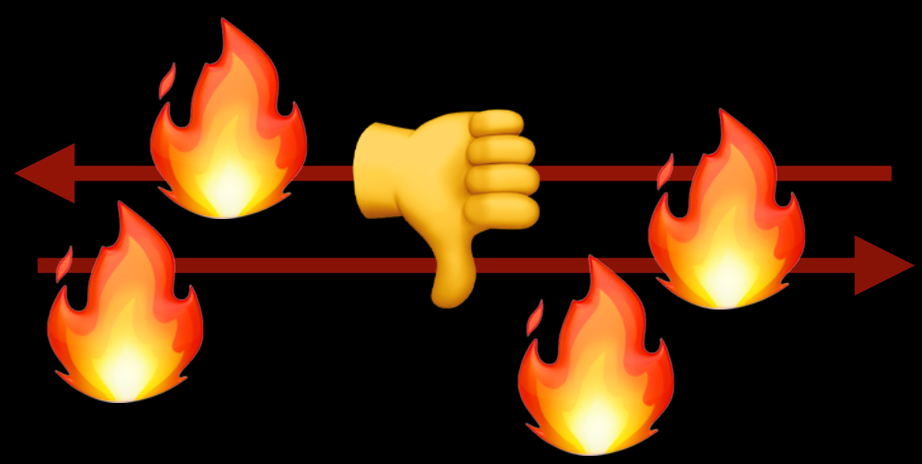
MVC



MVC

CONTROLLER

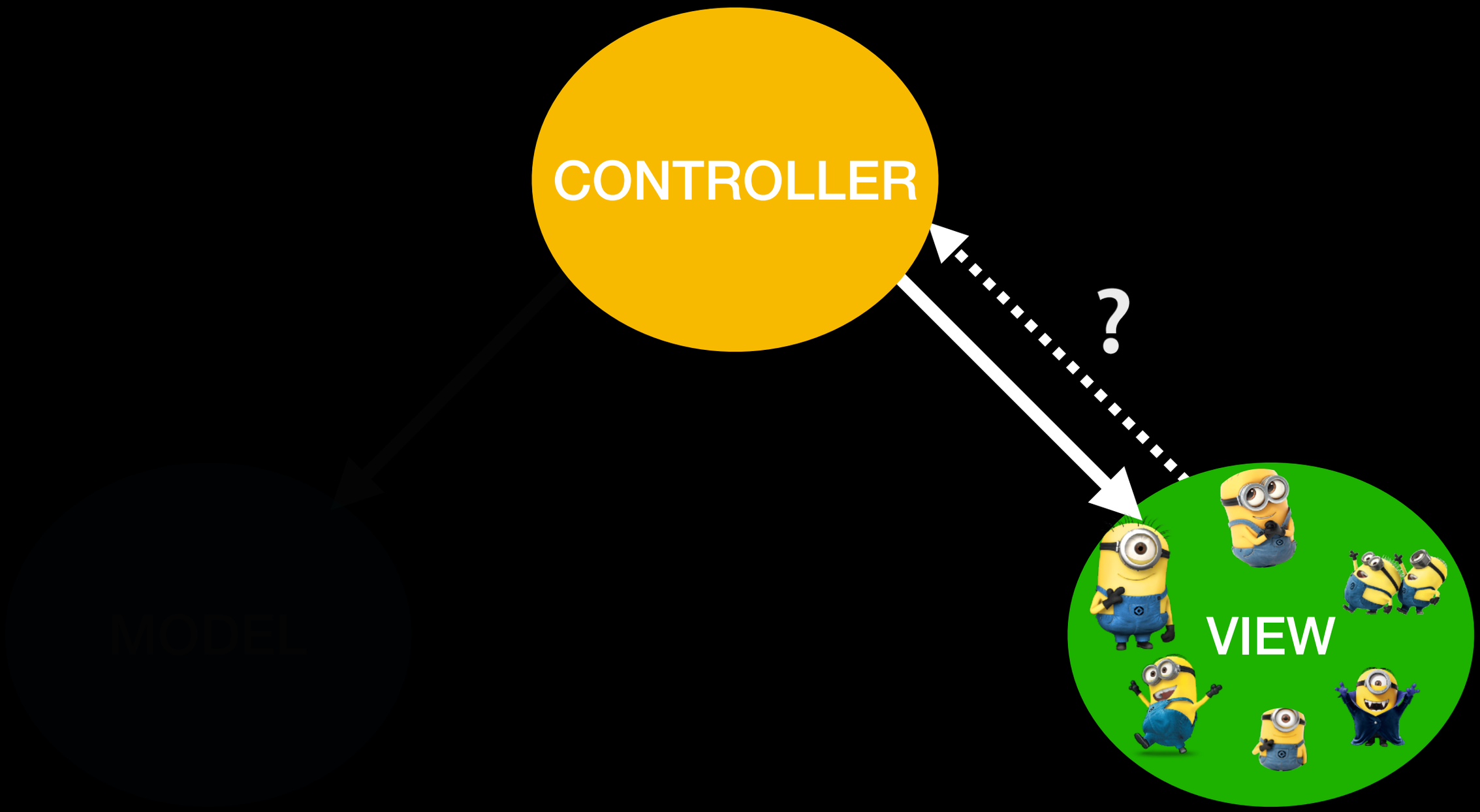
MODEL



VIEW

The MODEL and VIEW should never speak to each other

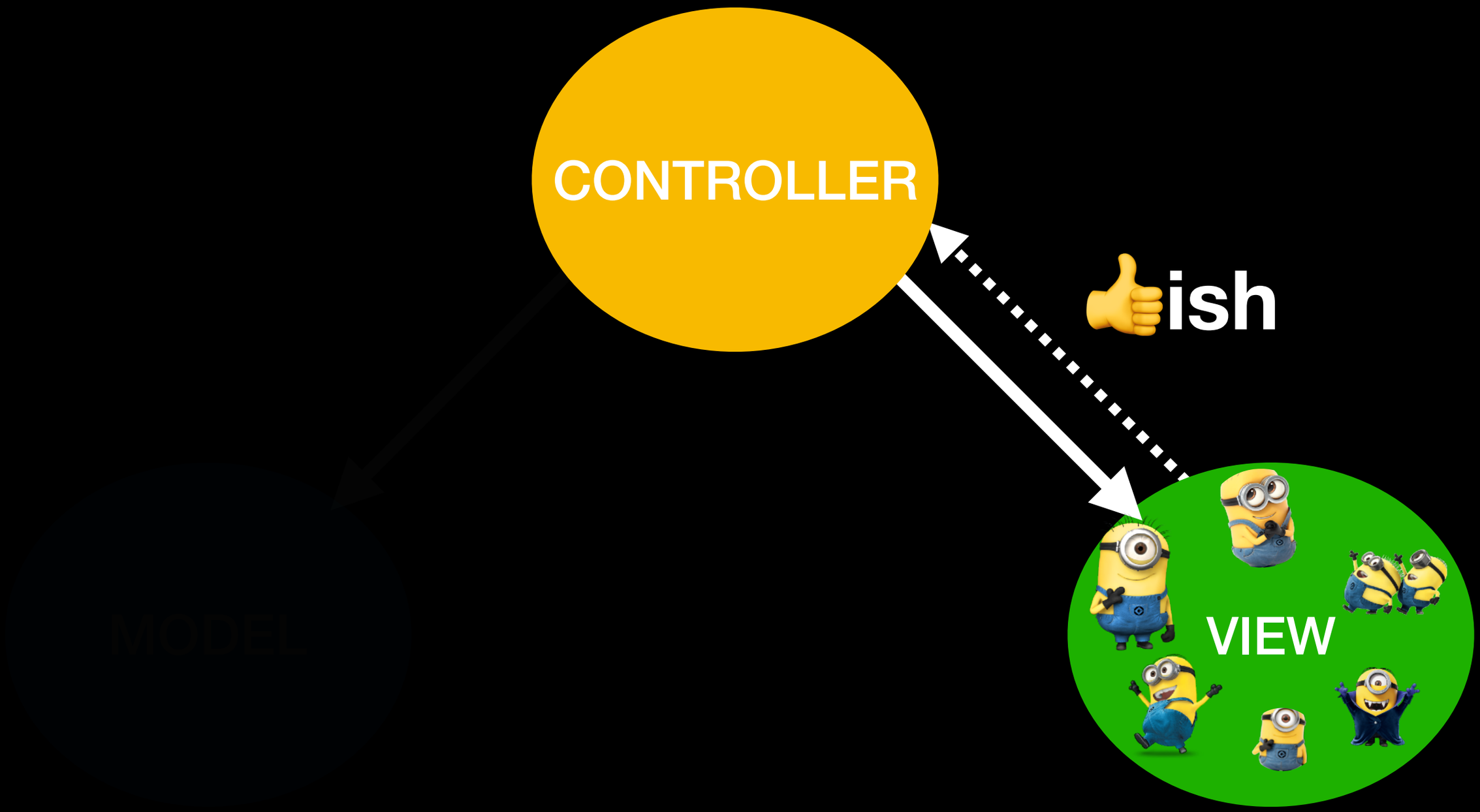
MVC



Should the VIEW be able to talk back to the CONTROLLER?

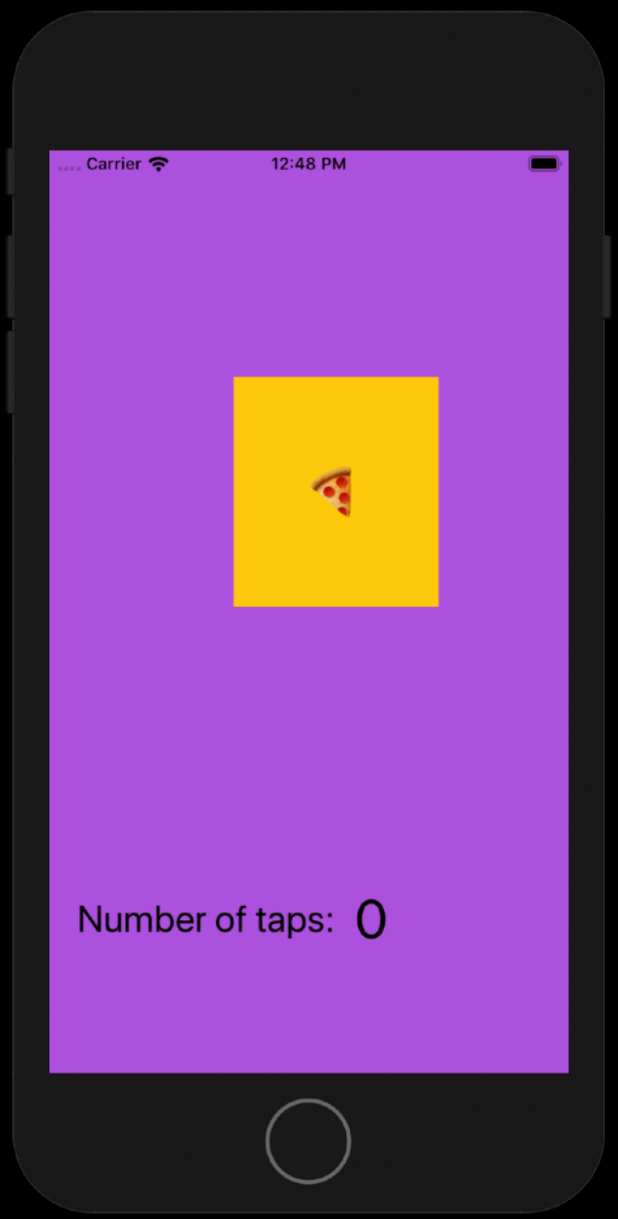
The MODEL and VIEW should never speak to each other.

MVC



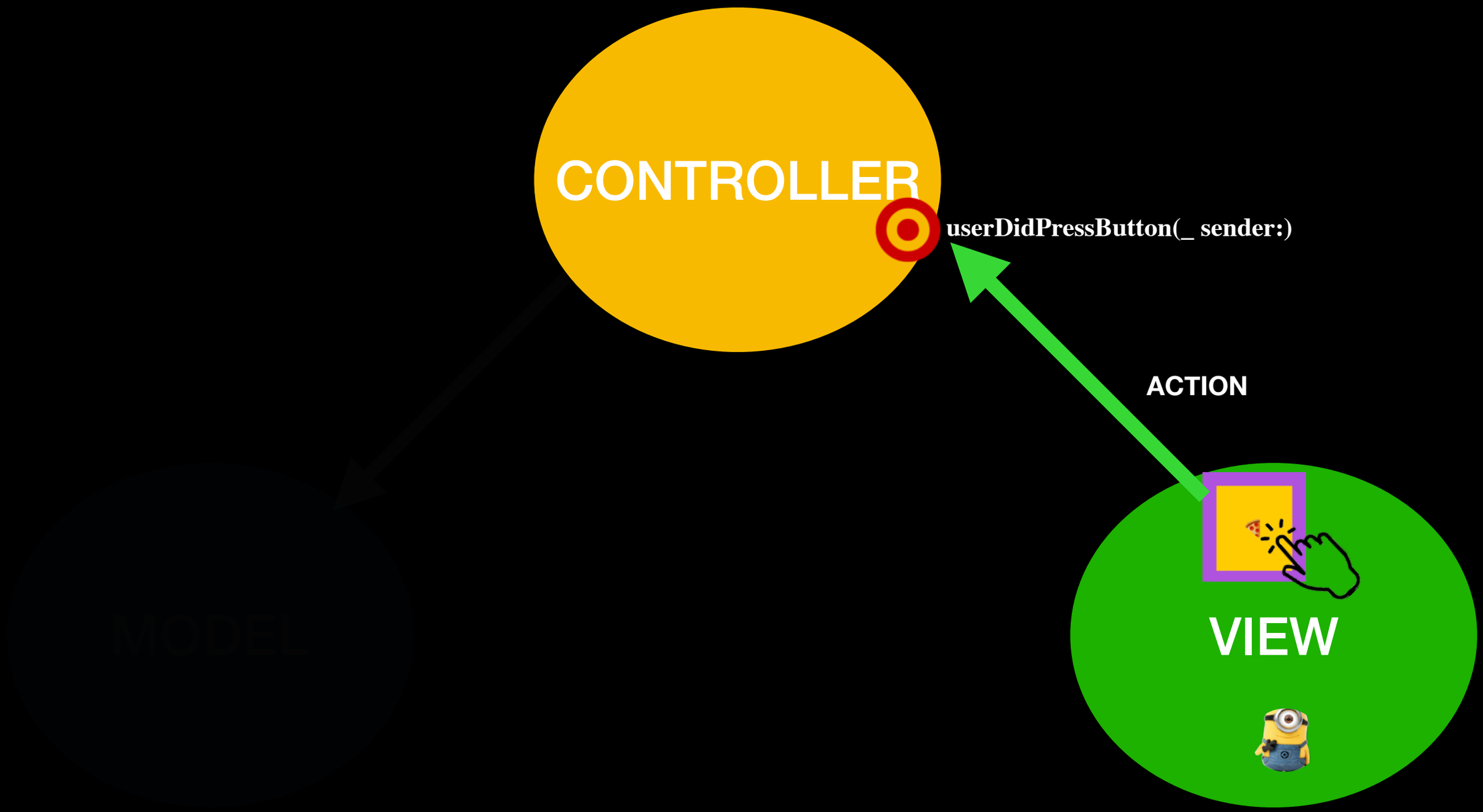
Sort of. Communication must be blind and structured.

MVC



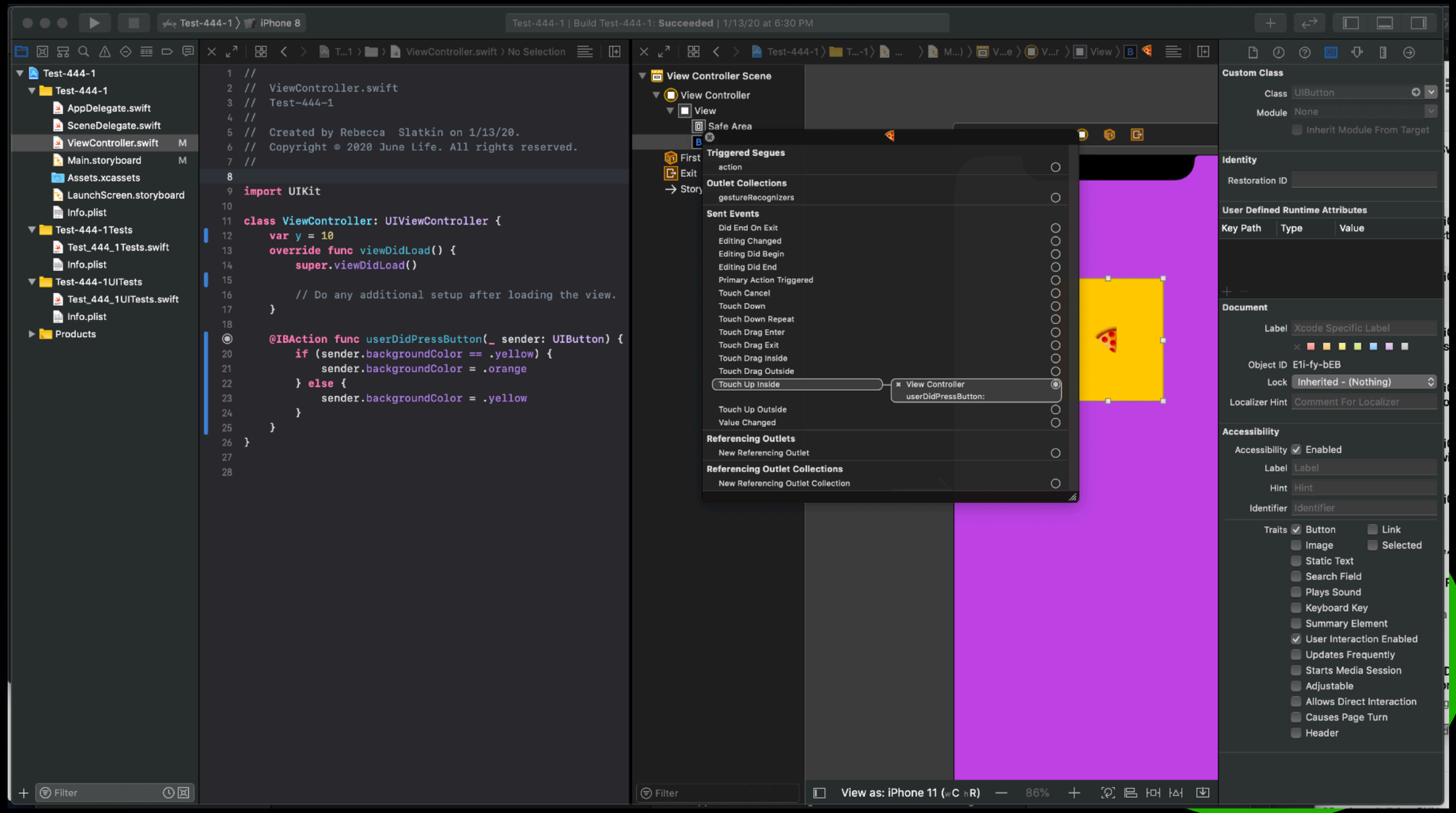
The MODEL and VIEW should never speak to each other

MVC



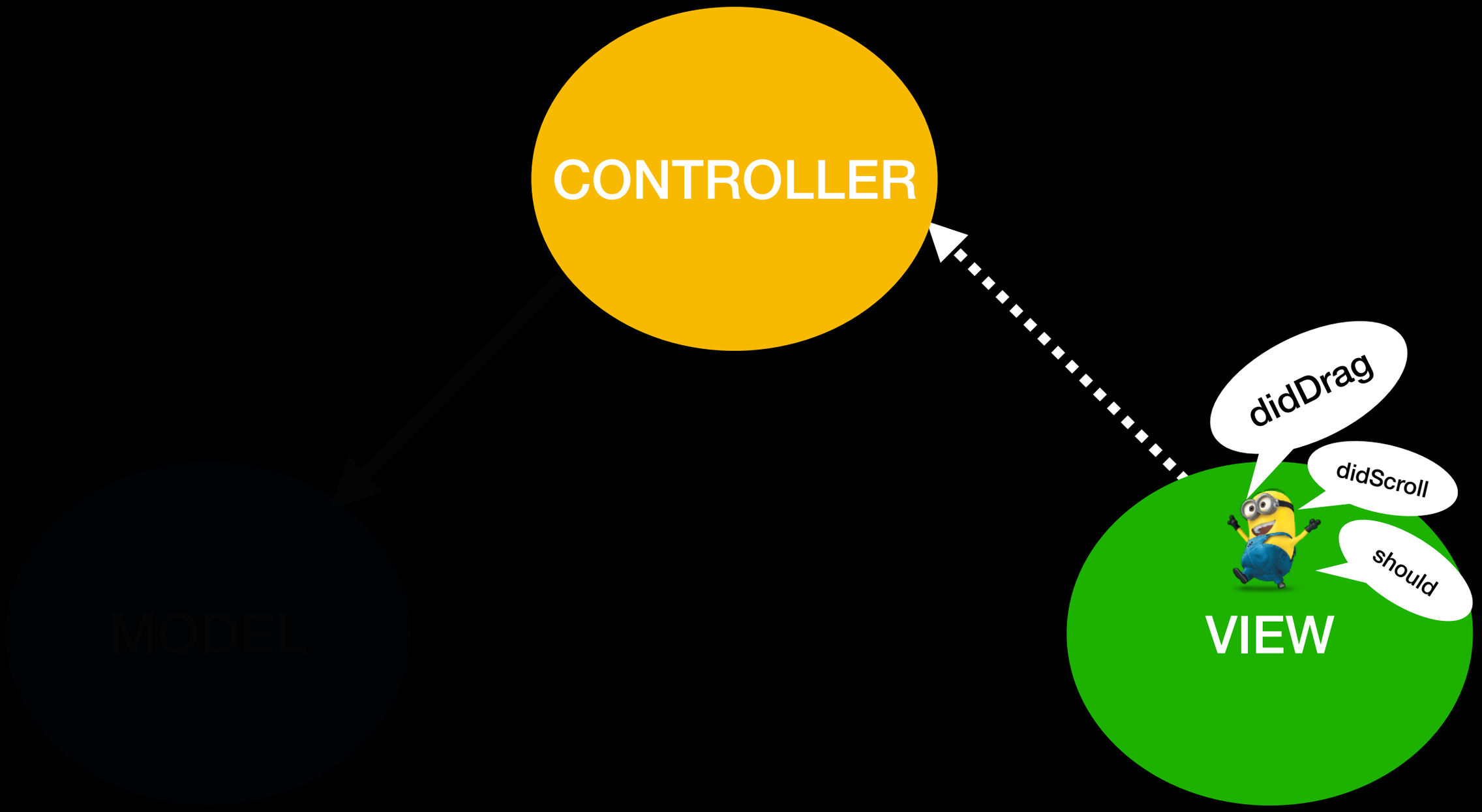
Sort of. Communication must be blind and structured.

MVC



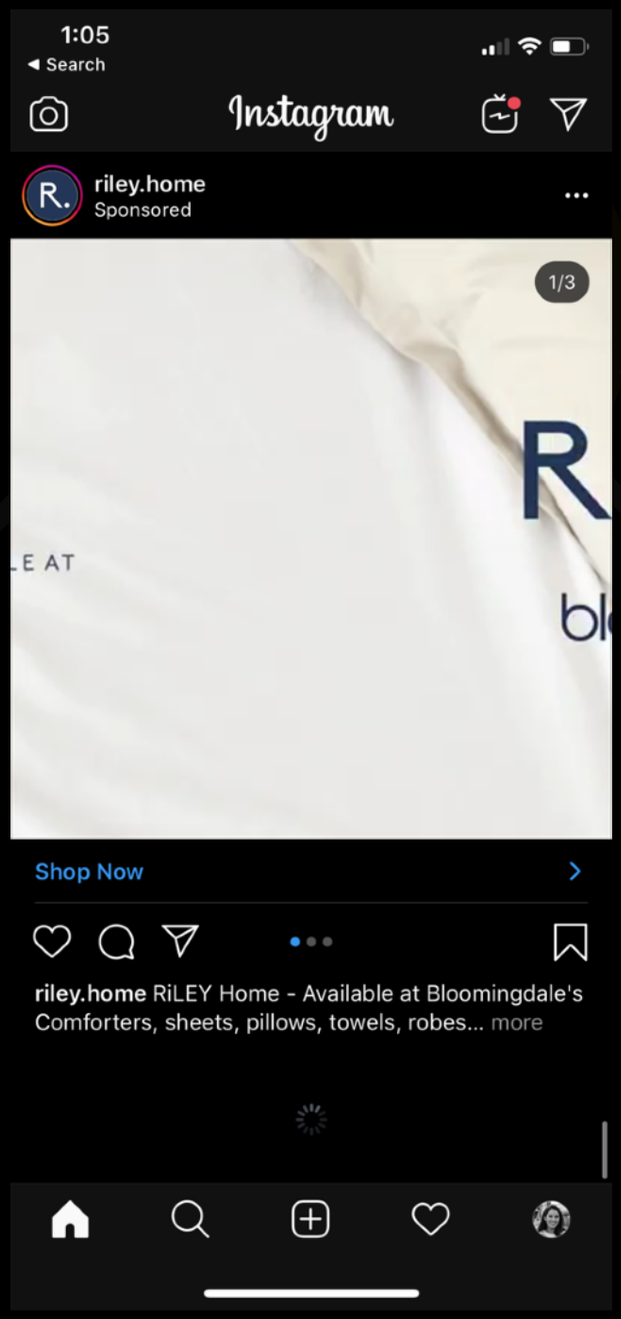
The **VIEW** sends the action when things happen in the **UI**.

MVC



Sometimes the **VIEW** needs to synchronize with the **Controller**

MVC



MODEL

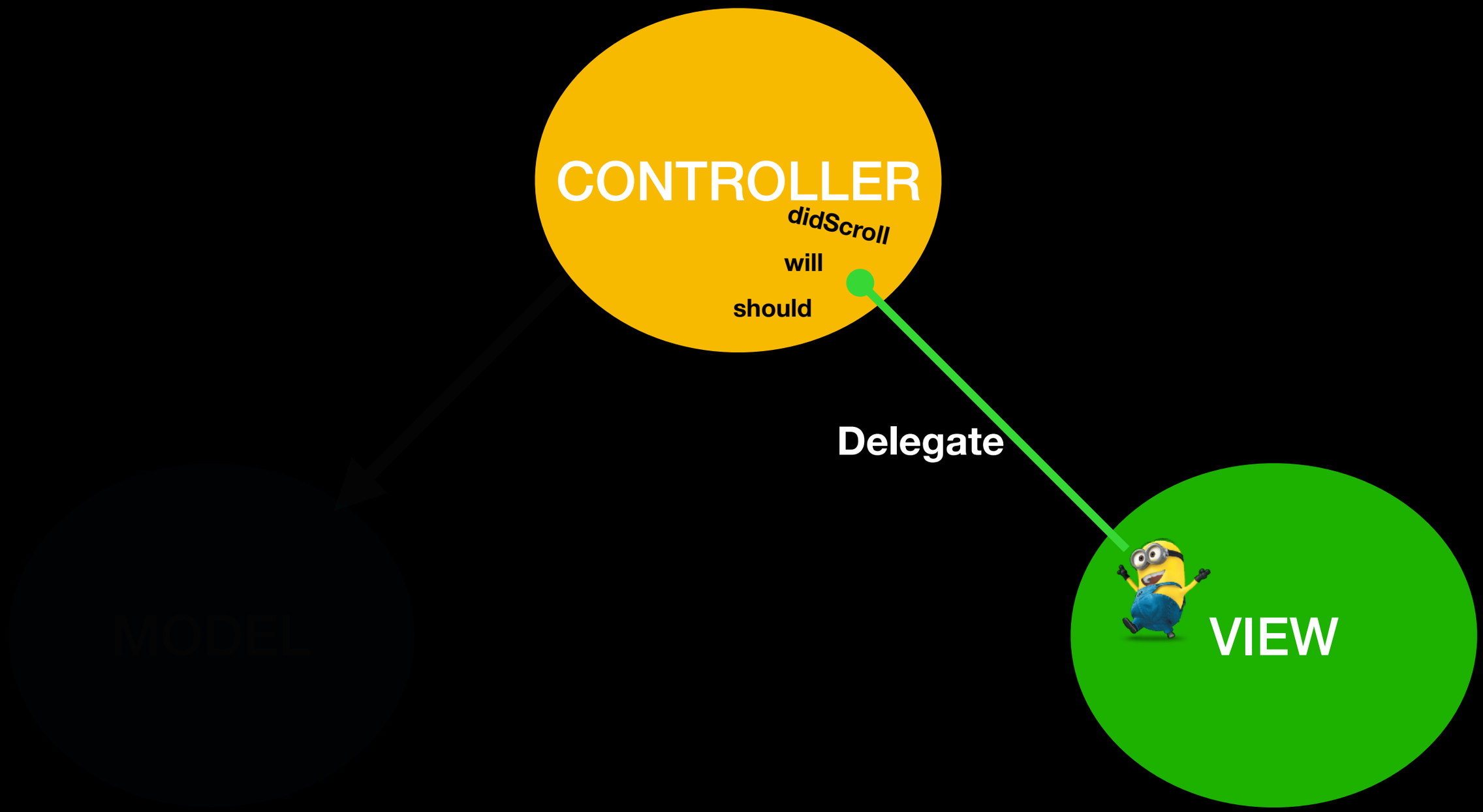
didDrag
didScroll
should

VIEW

Sometimes the VIEW needs to synchronize with the Controller

The MODEL and VIEW should never speak to each other

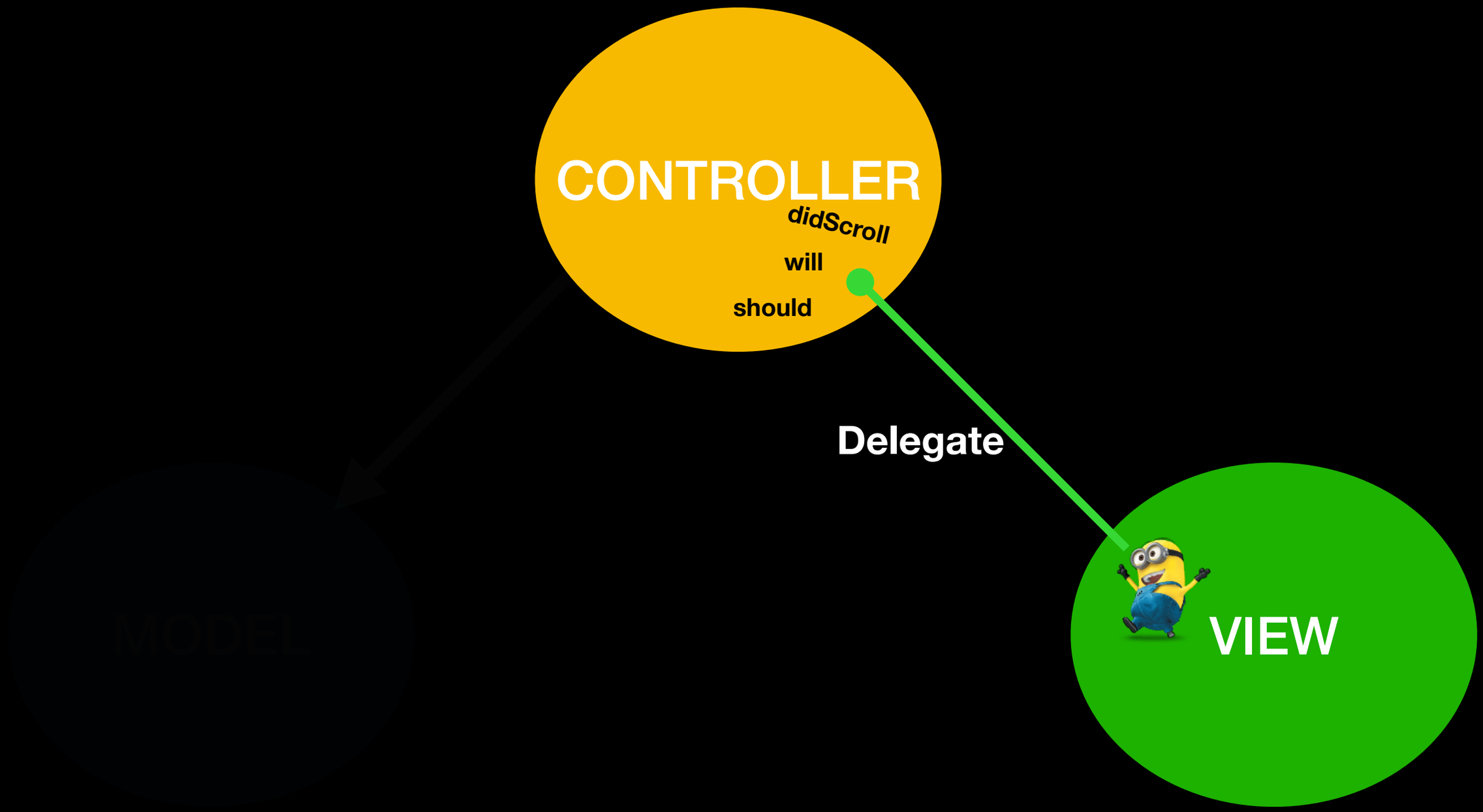
MVC



The **Controller** sets itself as the **View's** delegate

The MODEL and VIEW should never speak to each other

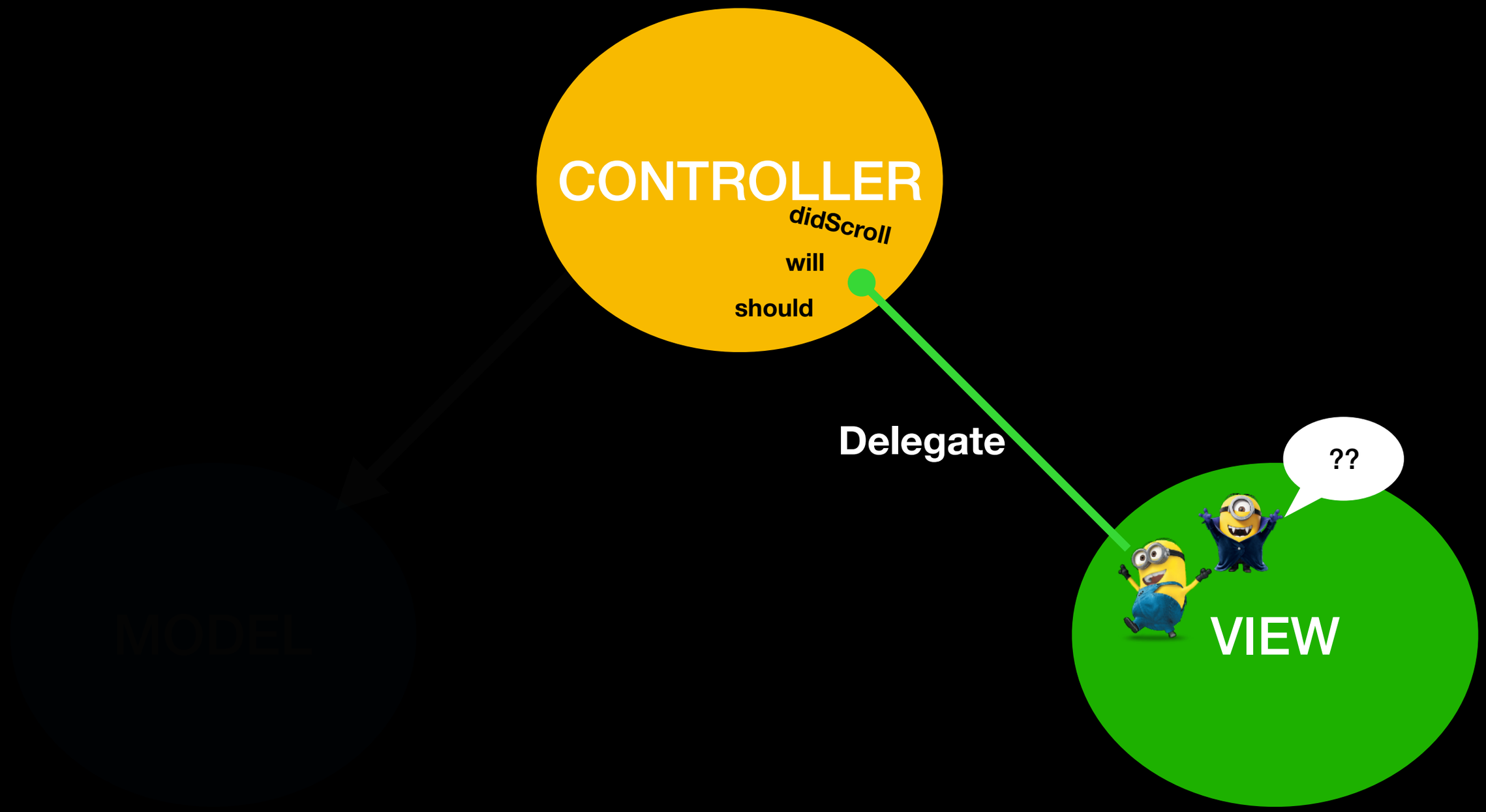
MVC



The **DELEGATE** is set via a protocol

The MODEL and VIEW should never speak to each other

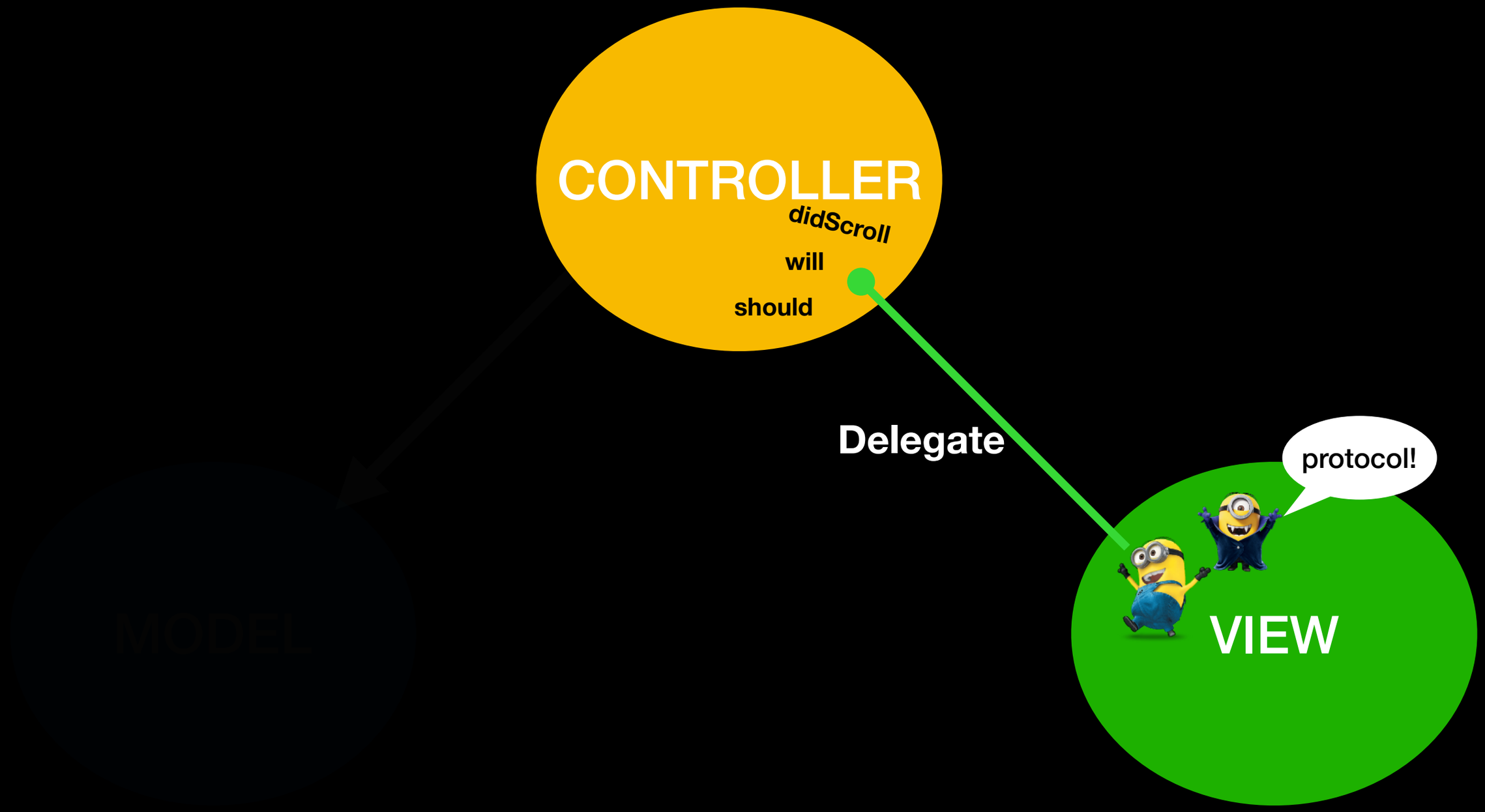
MVC



Views do **NOT** own the data they display

The MODEL and VIEW should never speak to each other

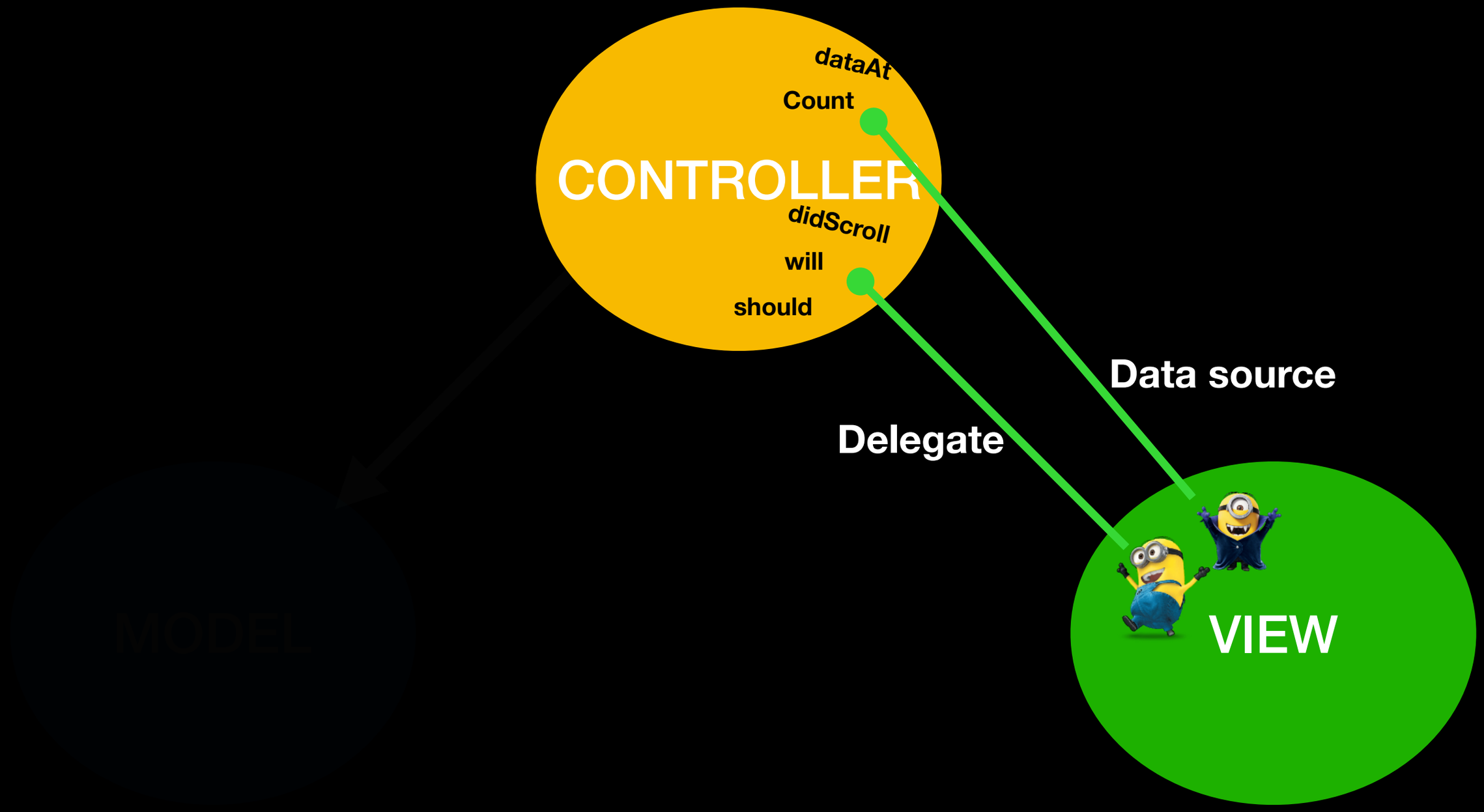
MVC



So, if needed, they have a protocol to acquire it

The MODEL and VIEW should never speak to each other

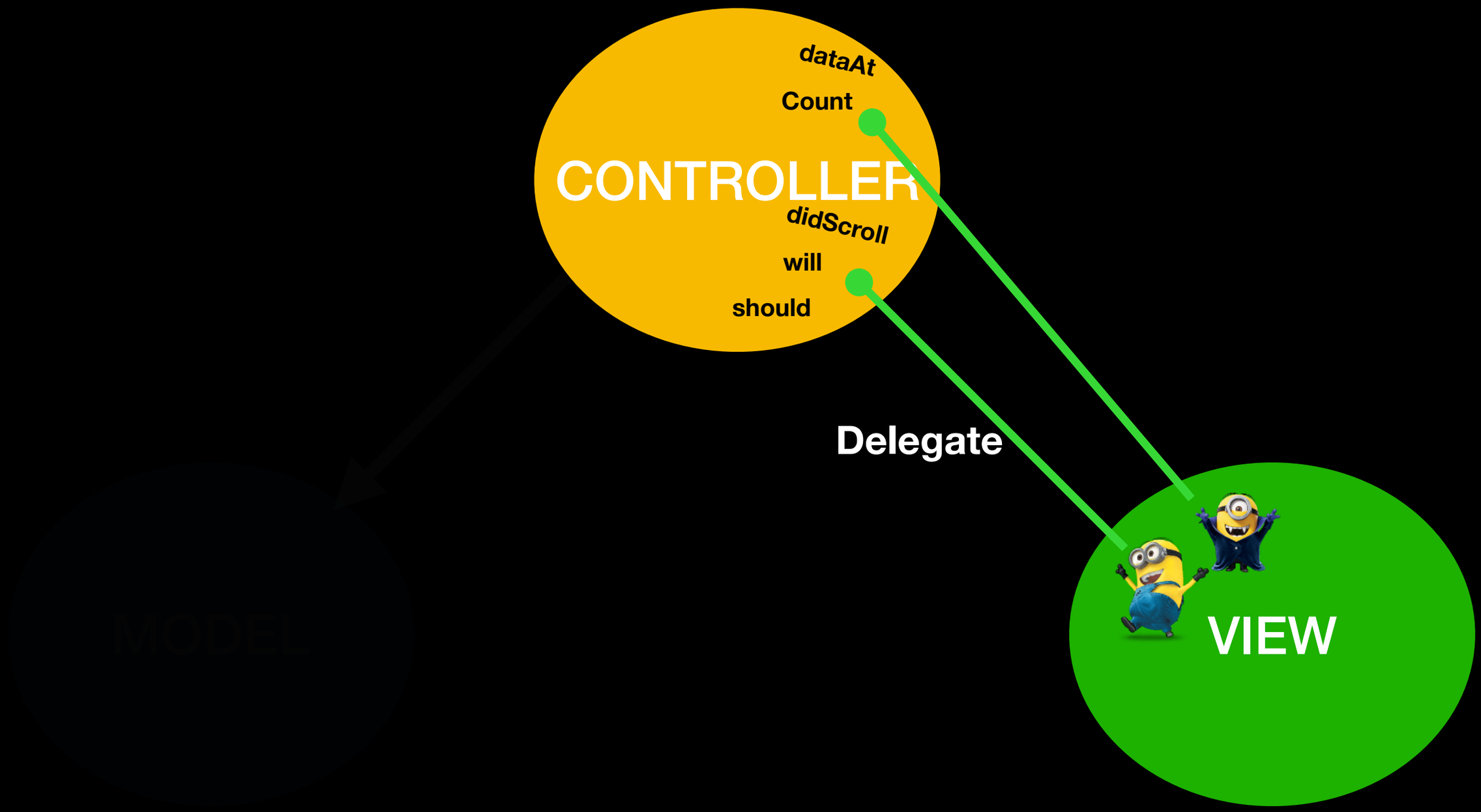
MVC



Controllers are almost always that data source (not MODEL)

The MODEL and VIEW should never speak to each other

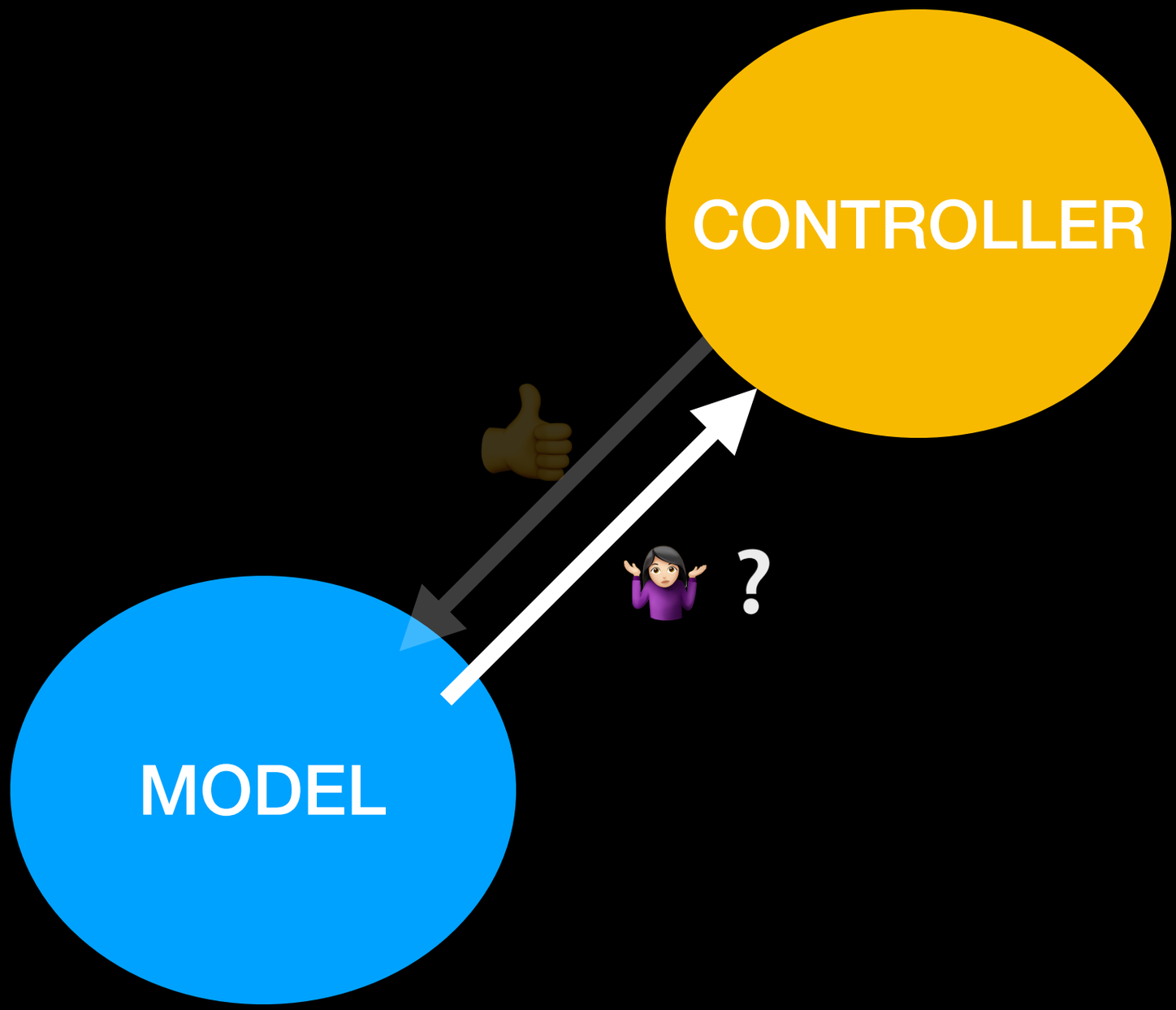
MVC



Controllers interpret/format **Model** information for the **View**

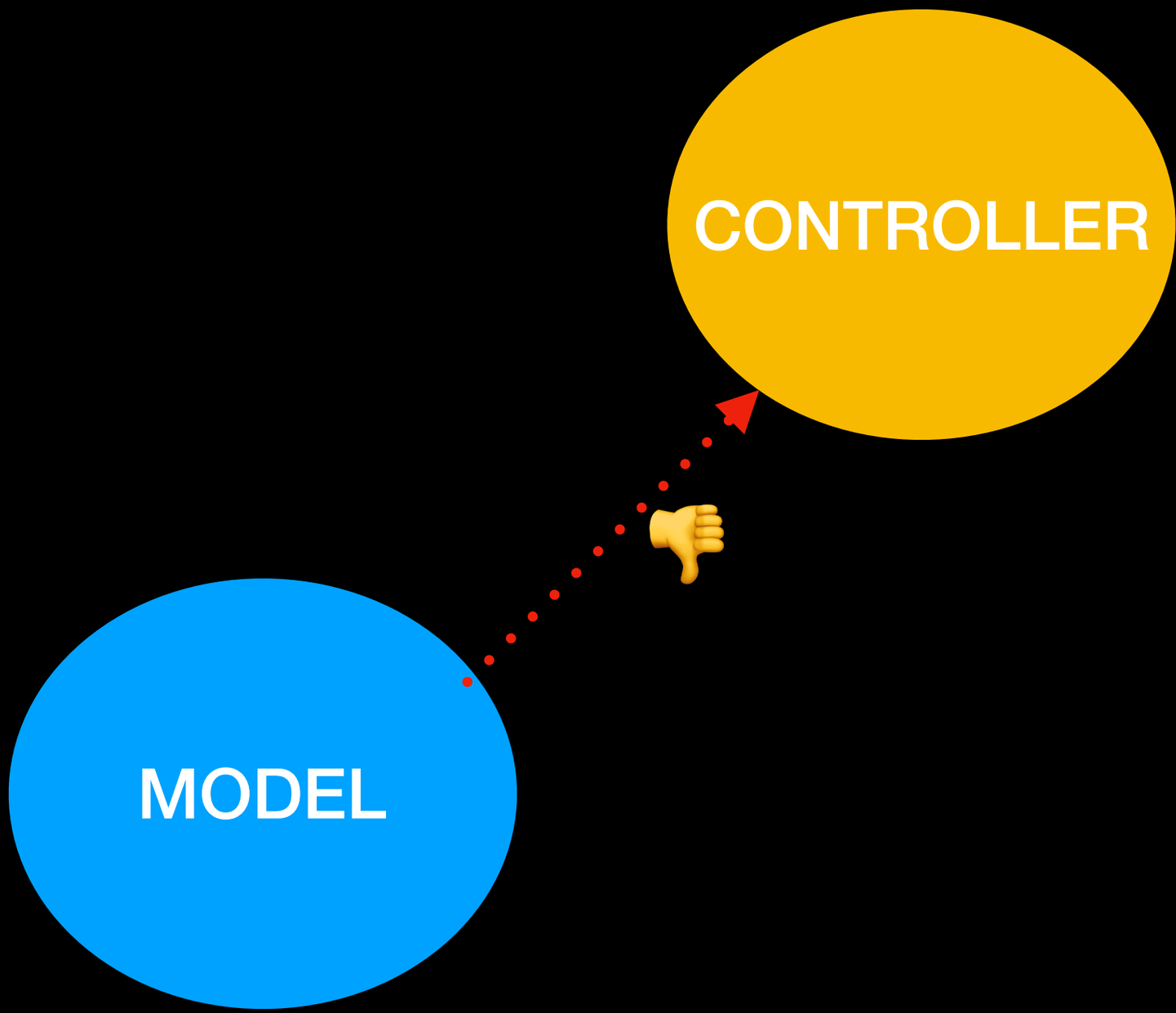
The MODEL and VIEW should never speak to each other

MVC



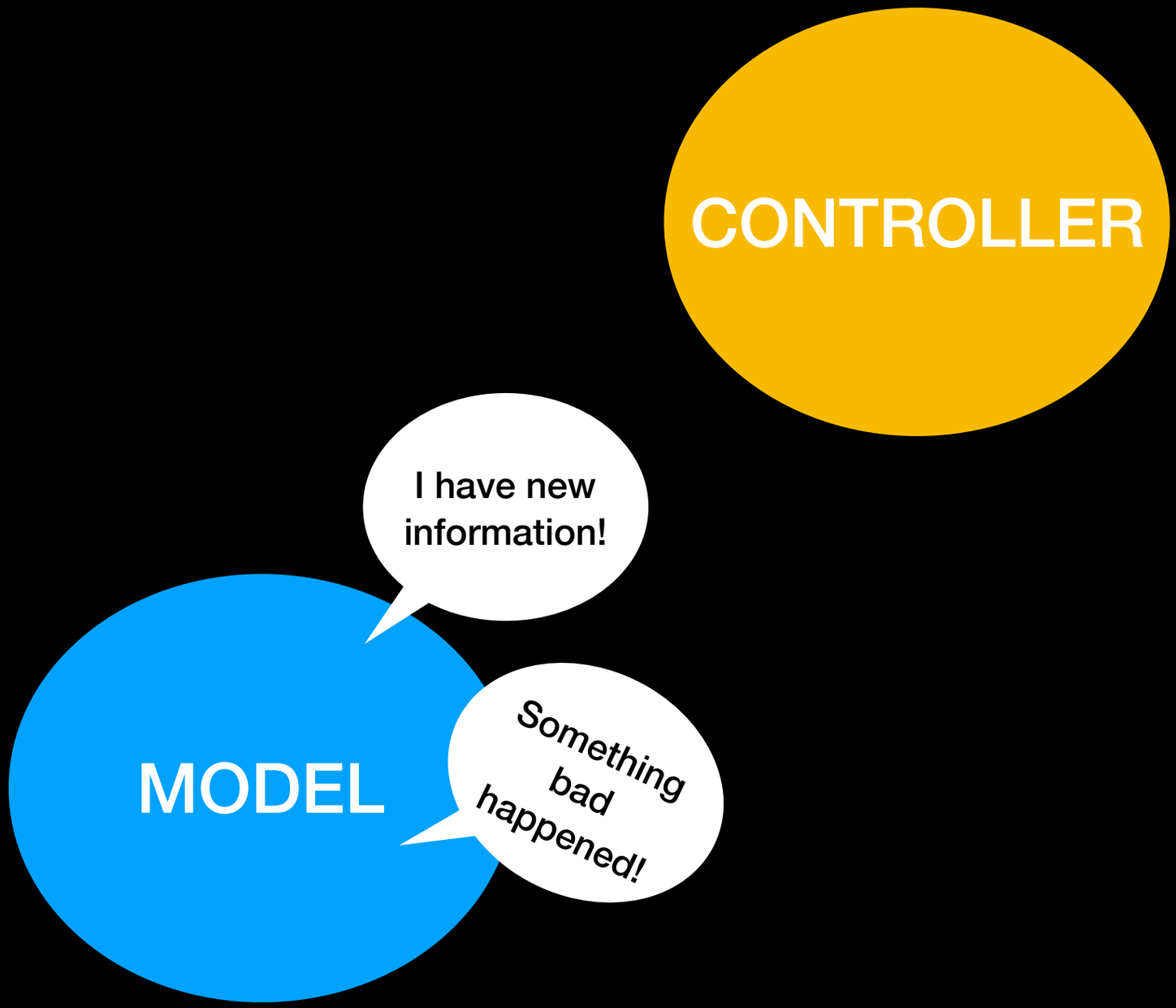
Can the **Model** talk directly to the **Controller**?

MVC



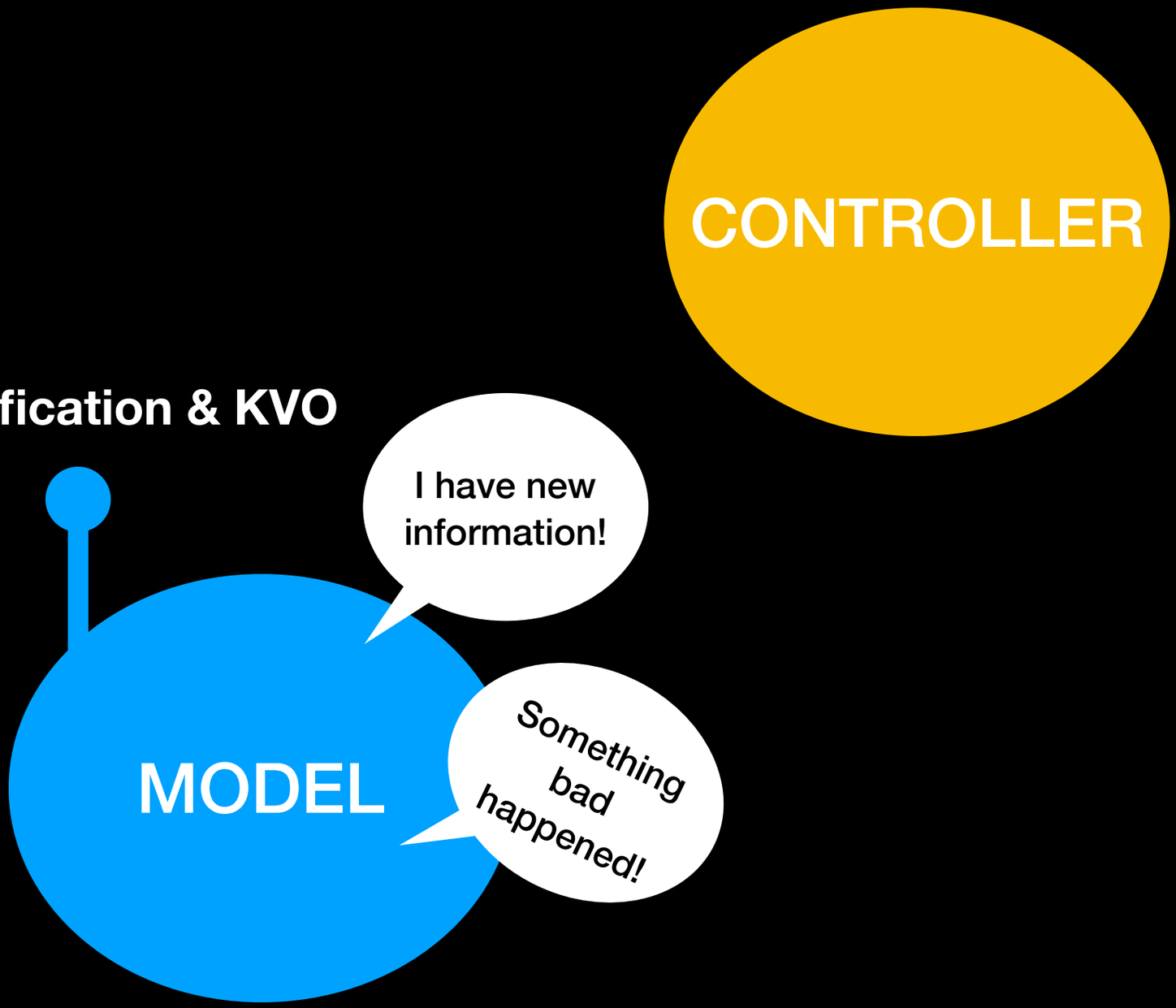
No. The M**odel** should be UI independent

MVC



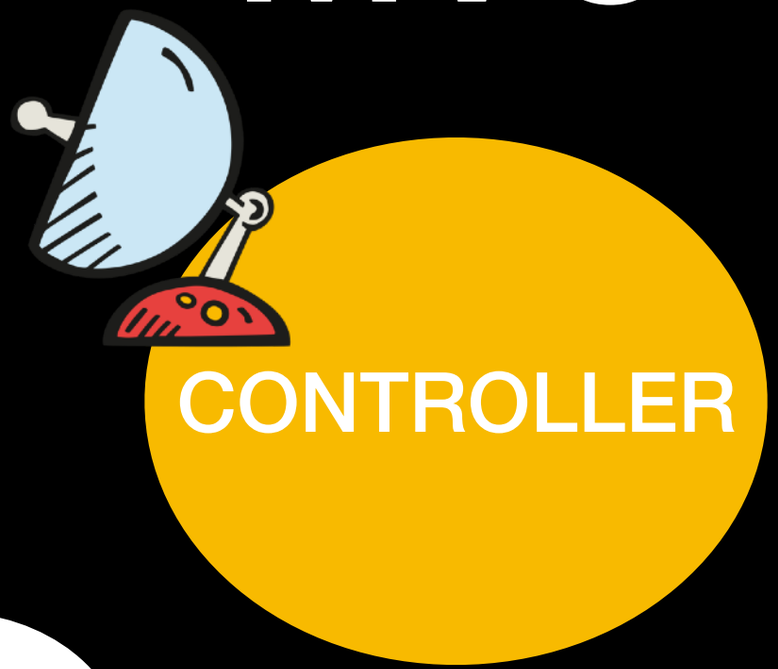
But what if the **Model** has information to update or something?

MVC

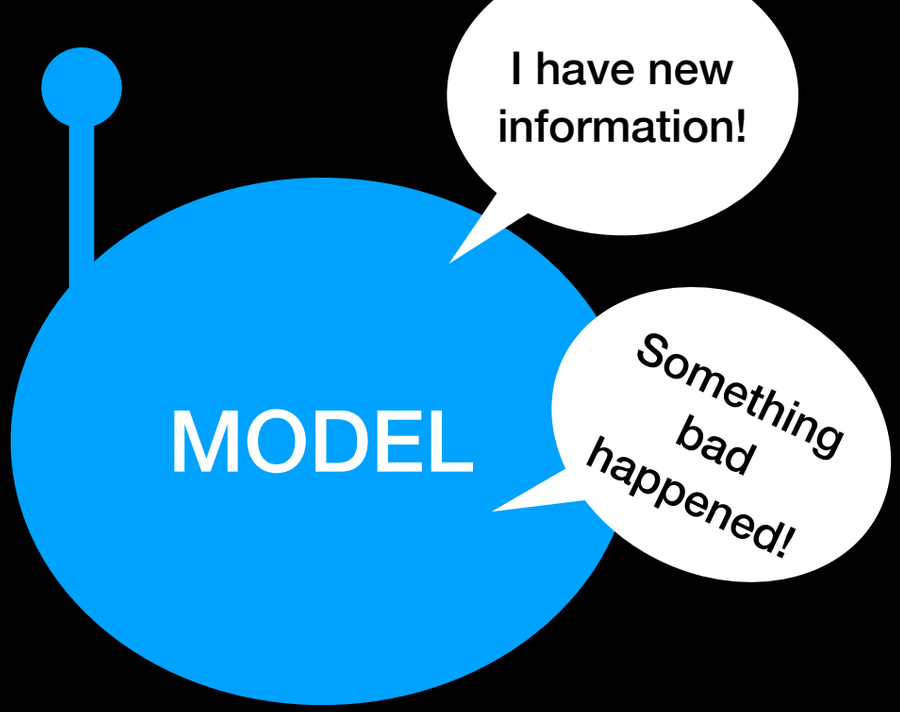


Model uses a “radio station”-like broadcast mechanism

MVC

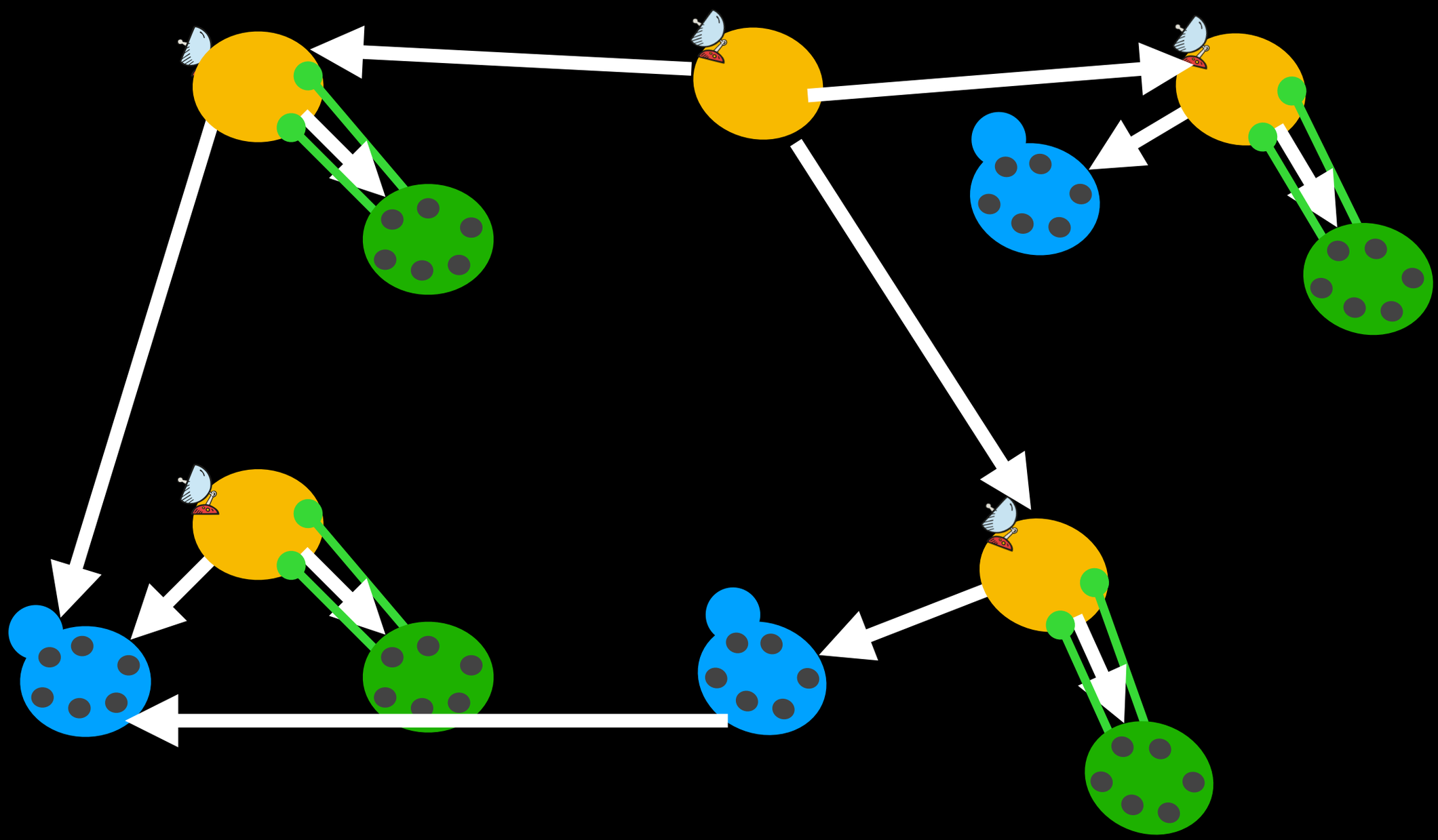


Notification & KVO



Controllers or other **Models** “tune in” to interesting stuff

MVCs working together



Mars Demo