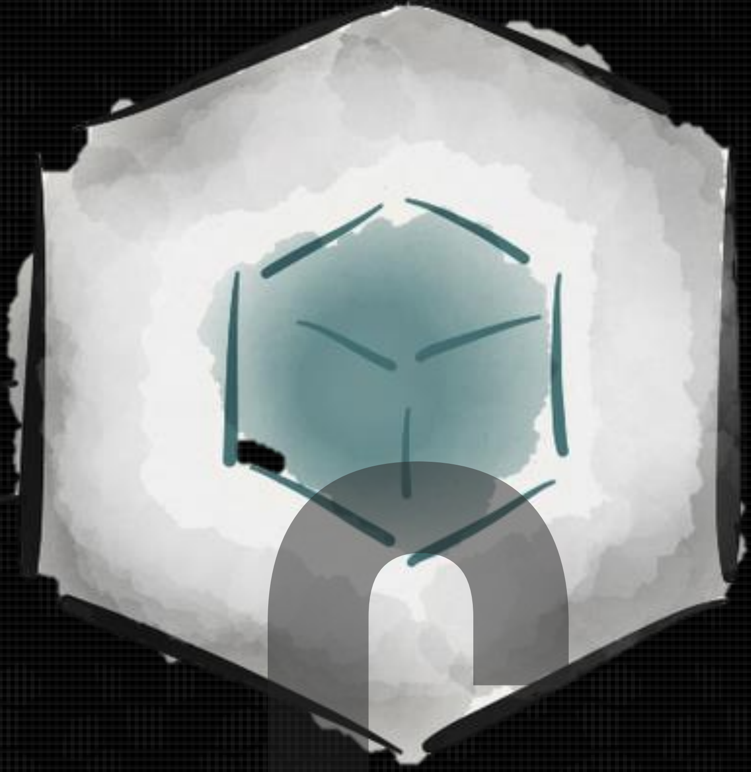
M (odel) V (iew) C (ontroller)

Dinh-Mao Bui

Software Engineering (Fall 2019)

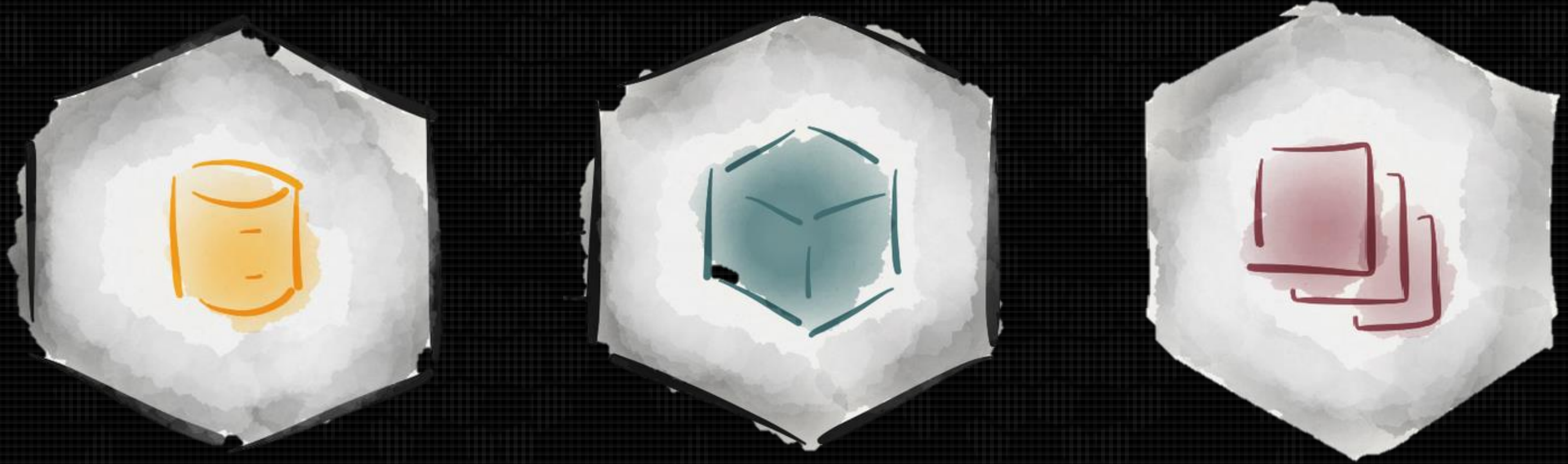
Nazarbayev University



MODEL VIEW CONTROLLER

MODEL VIEW CONTROLLER (MVC)

- Software architecture pattern that separates the model, the user interface and control logic of an application in three distinct components.

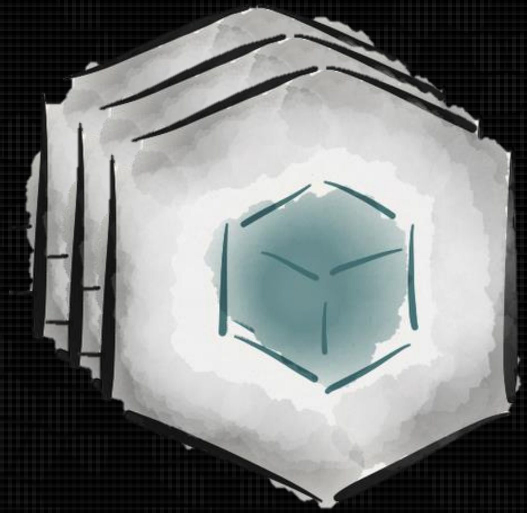


THE MVC PATTERN

- MVC proposes the construction of three distinct components. One side for the representation of information, and on the other hand for user interaction.

One model

Many Views


Many Controllers

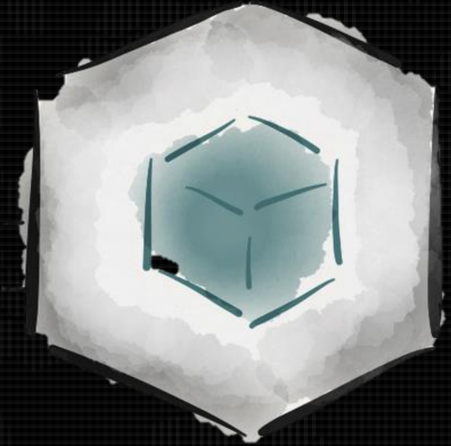
THE MVC PATTERN

The Model

Representation of domain data.
Business logic
Persistence mechanism

The View

User Interface
Interaction elements



The Controller

Intermediary between Model and View
It maps user actions – model actions
Select the view and provide
information to itself

THE MODEL

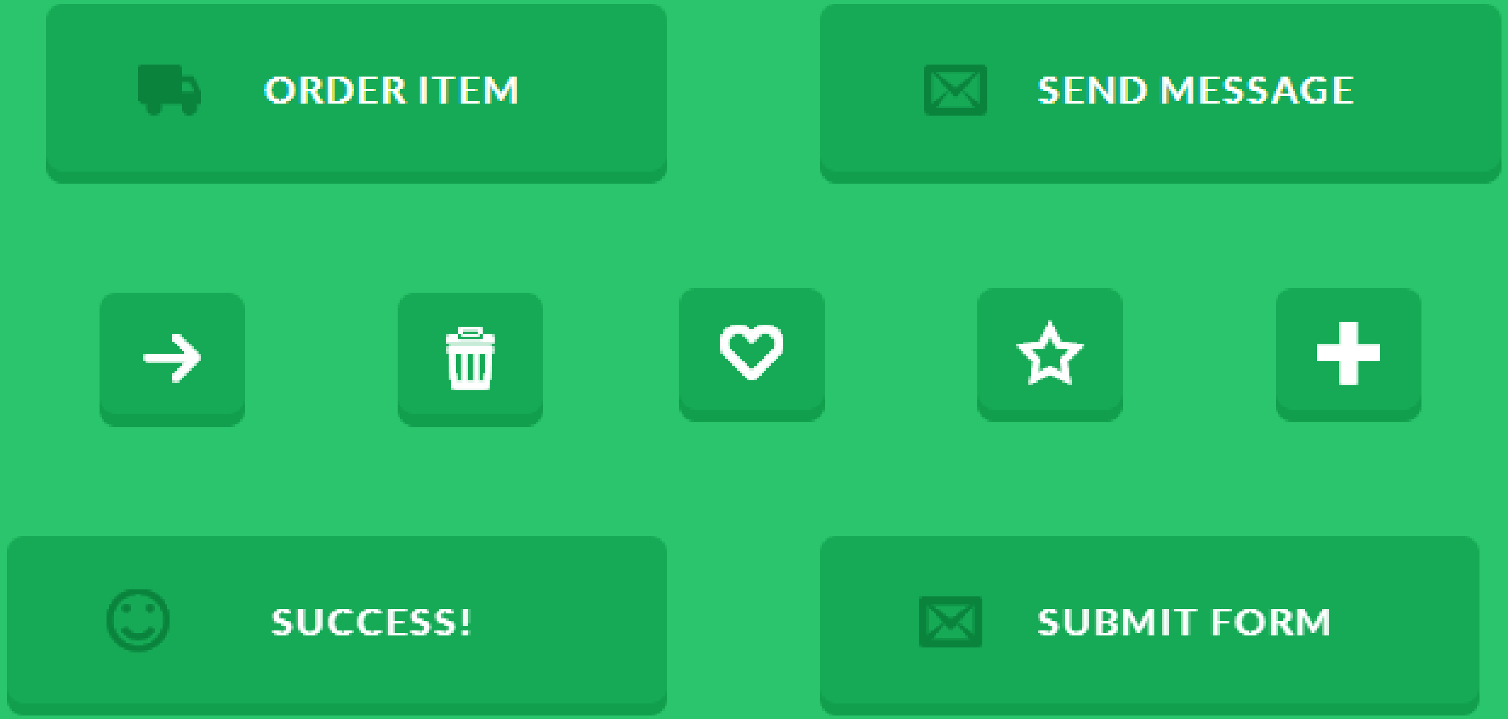
It is the specific representation of the information with which the system operates. Logic ensures the integrity of data and allows to derive it.

THE VIEW

Represents the model in a suitable format to interact and access the data, usually called "User Interface" (GUI Java, HTML, XML).

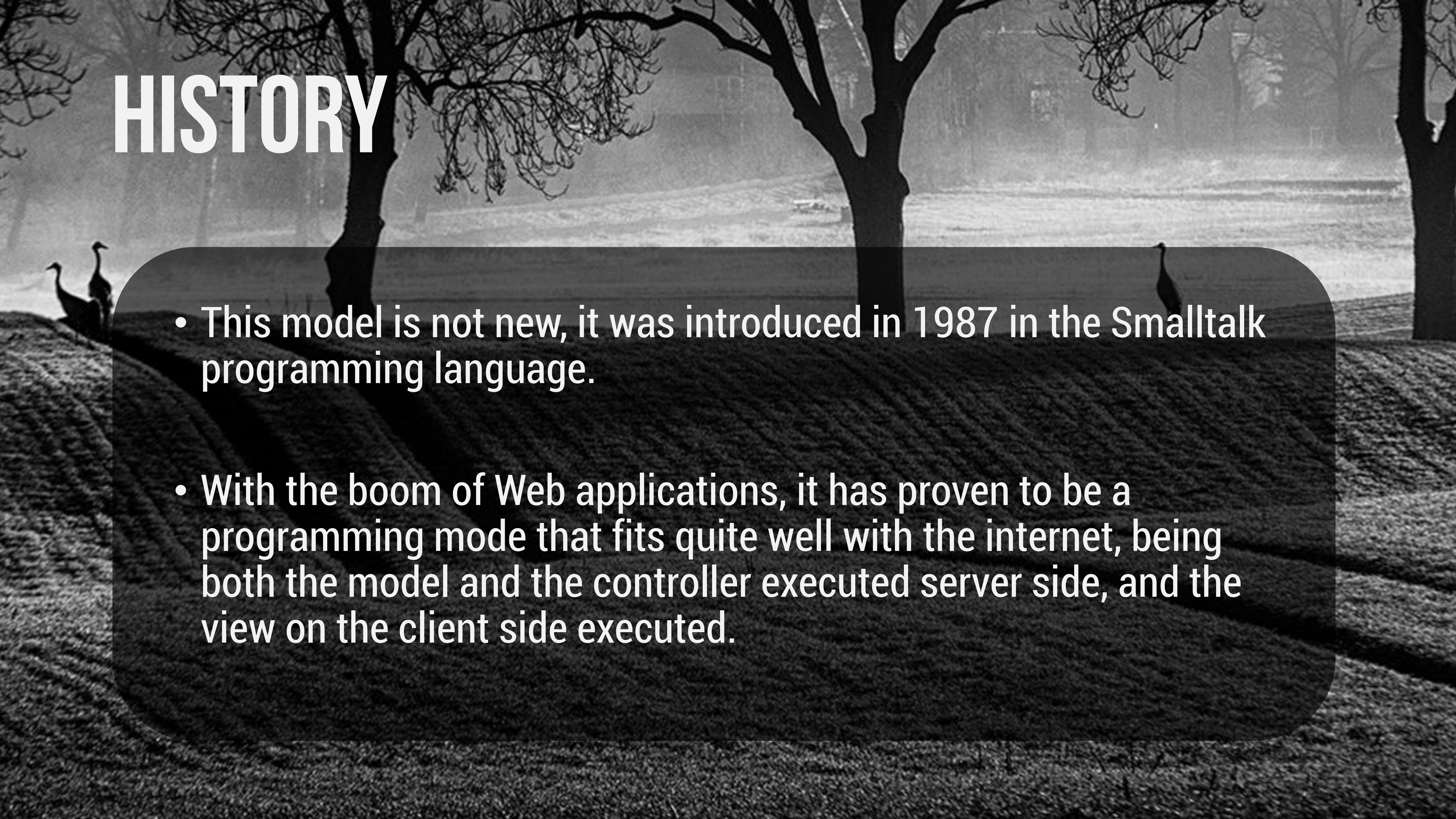
THE CONTROLLER

It is the link between the view and the model, is responsible for receiving and responding to events, typically user actions and invokes changes on the model and probably in the view.



BENEFITS

- Organization
- Rapid Application Development
- Reusing Code
- Parallel development
- It presents the same information in different ways.
- The views and application behavior should reflect the manipulations of the data immediately.
- It allows different user interface standards or port it to other environments where the application code should not be affected.



HISTORY

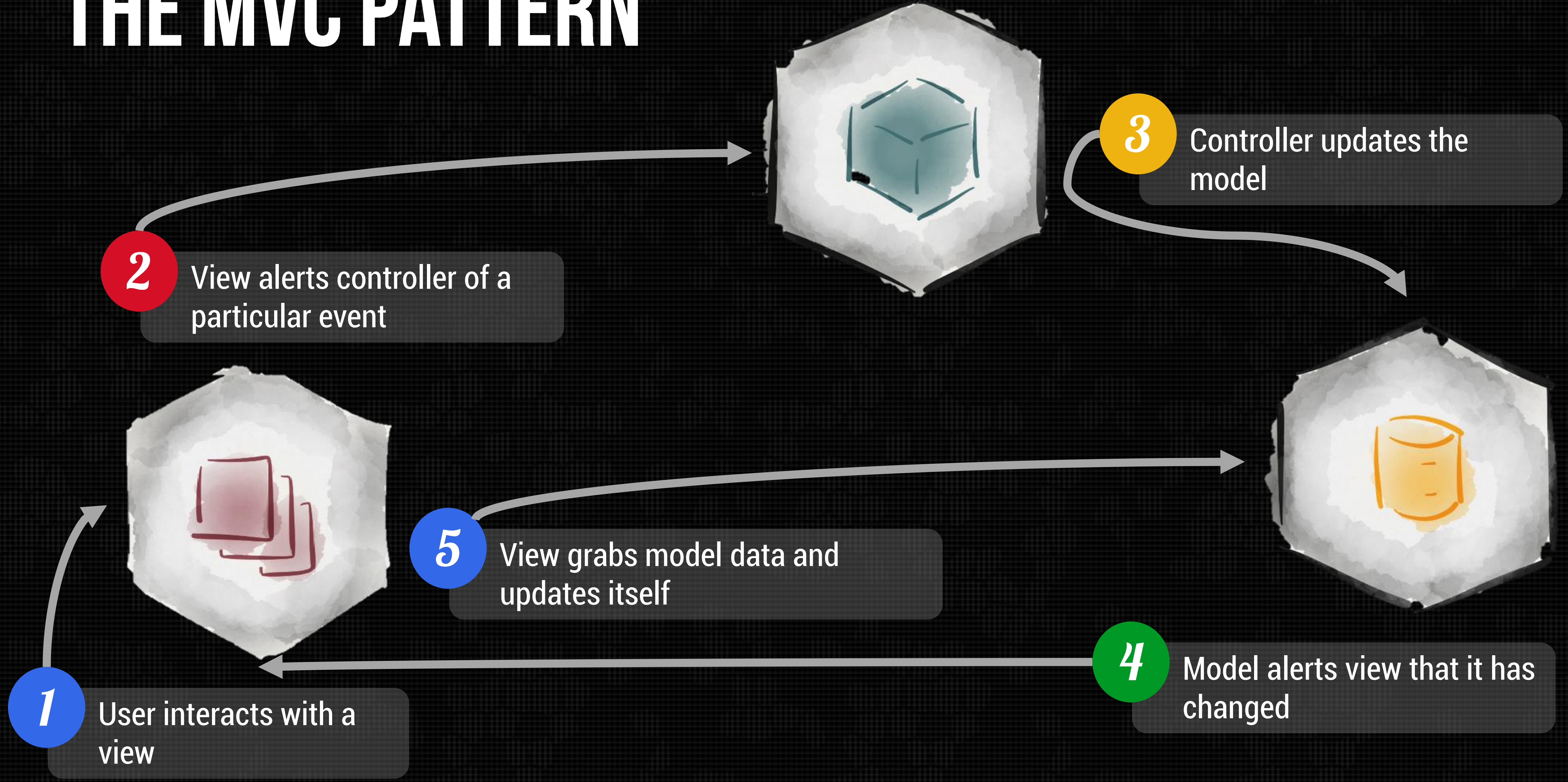
- This model is not new, it was introduced in 1987 in the Smalltalk programming language.
- With the boom of Web applications, it has proven to be a programming mode that fits quite well with the internet, being both the model and the controller executed server side, and the view on the client side executed.

THE MVC PATTERN

Control Flow

1. The user performs an action on the interface.
2. The controller takes the input event.
3. The controller notifies the user action to the model, which may involve a change of state of the model.
4. It generates a new view. The view takes the data model.
5. The user interface waits for another user interaction, which starts a new cycle.

THE MVC PATTERN



1 User interacts with a view

2 View alerts controller of a particular event

3 Controller updates the model

4 Model alerts view that it has changed

5 View grabs model data and updates itself

WHERE CAN I USE IT?

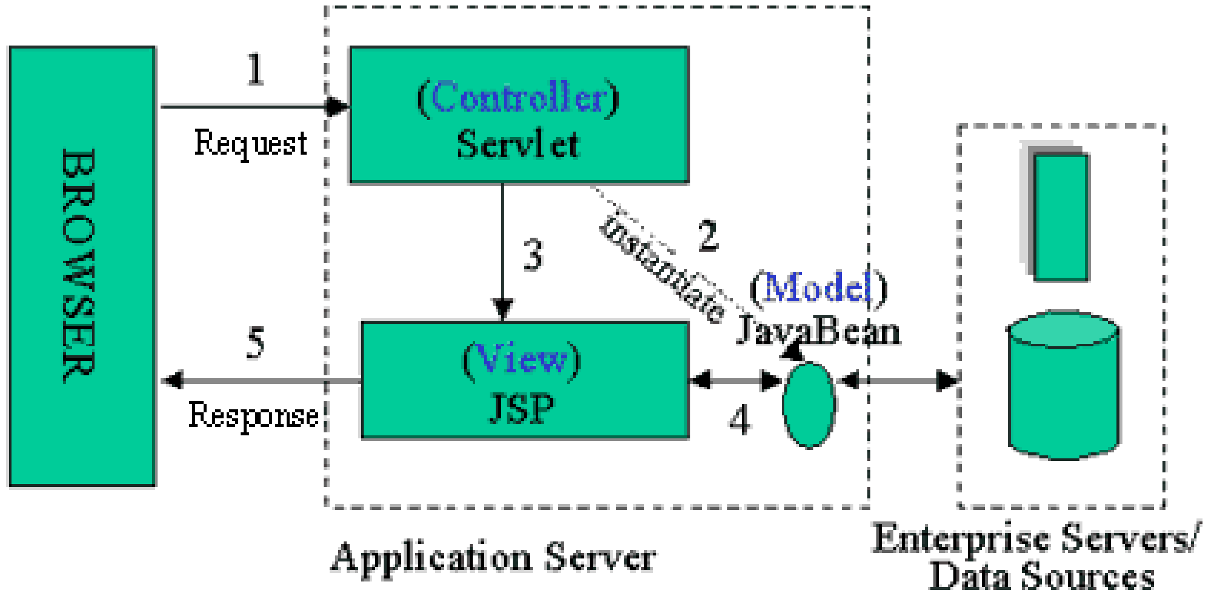
- It applies to all types of systems
- And technologies (Java, Ruby, Python, Perl, Flex, SmallTalk,. Net, etc..)

MVC IN JAVA

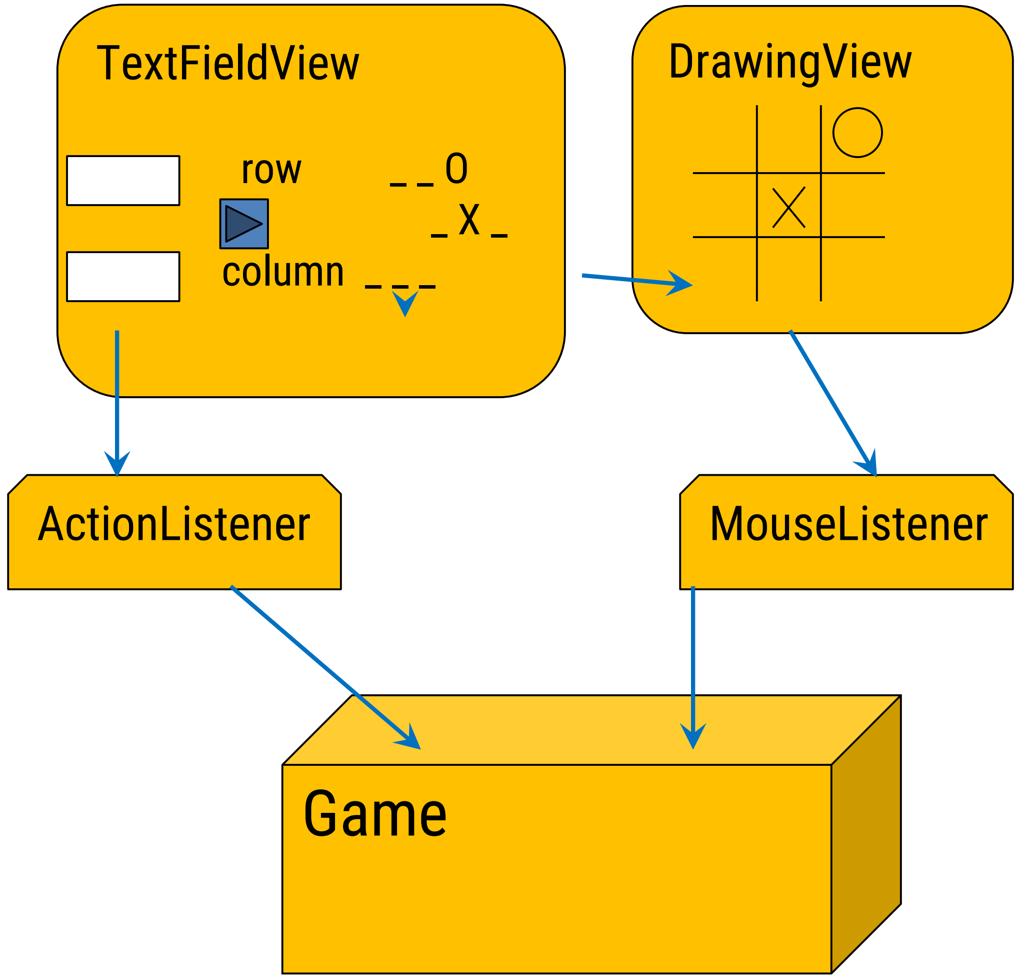
- **Model:** The model is made by the developer.
- **View:** Set of objects of classes that inherit from java.awt.Component.
- **Controller:** The controller is the event processing thread, which captures and propagates the event to the view and the model. Treatment classes of events (sometimes as anonymous classes) that implement EventListener type interfaces (ActionListener, MouseListener, WindowListener, etc..).

MVC the old days

- Model: Enterprise Beans with data in the DBMS
  - JavaBean: a class that encapsulates objects and can be displayed graphically
- Controller: Servlets create beans, decide which JSP to return
  - Servlet object do the bulk of the processing
- View: The JSPs generated in the presentation layer (the browser)

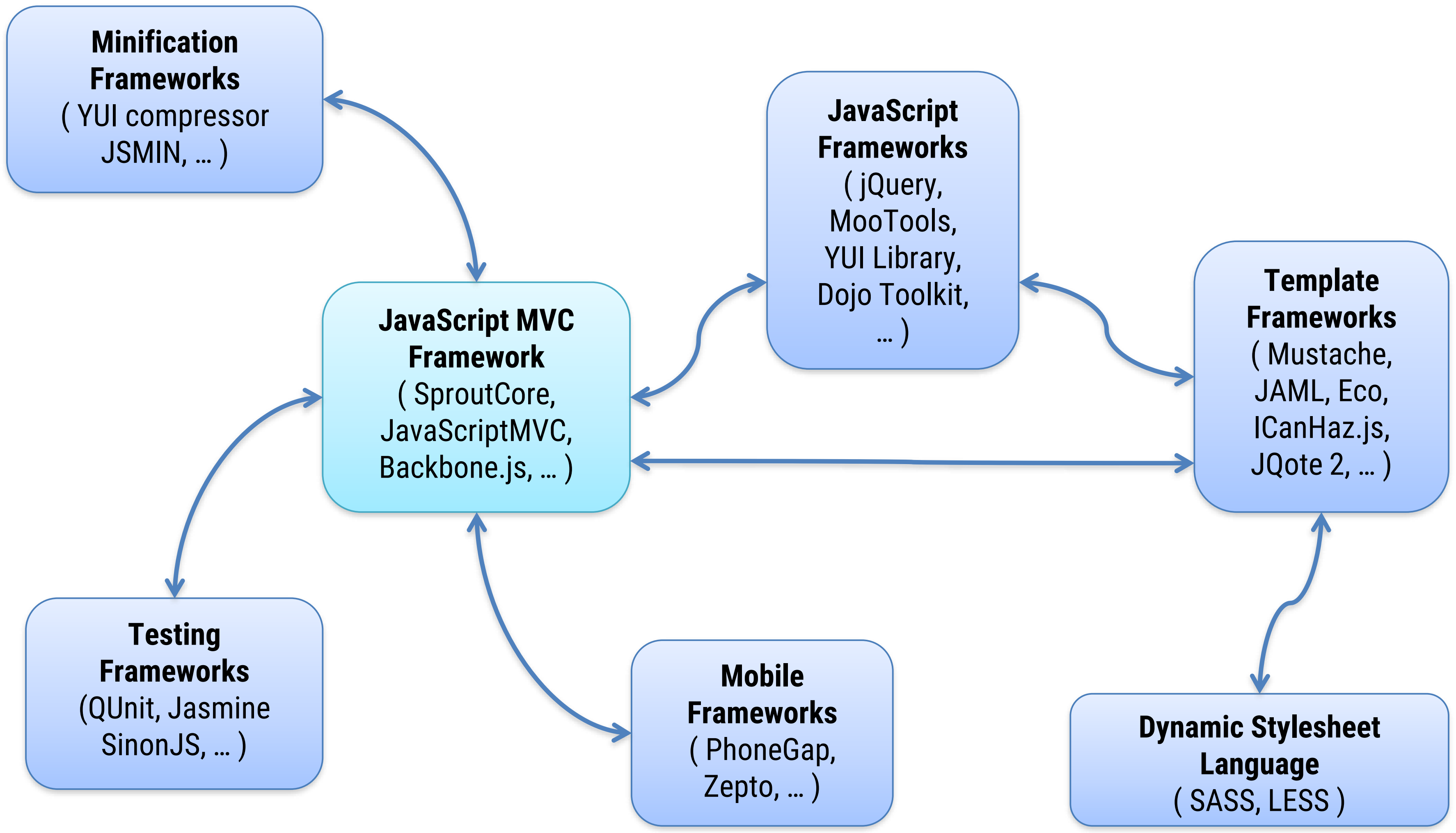


MVC in TicTacToe



1. Name the controllers
2. Name the model
3. Do the controllers have a user interface?
4. Does a view allow user input?
5. Can a view send messages to the model?
6. Can a view send messages to the controller?
7. Can a system have more than one view?
8. Can a system have more than one controller?

MVC Ecosystem in JS


Minification Frameworks (YUI compressor JSMIN, …)
JavaScript Frameworks (jQuery, MooTools, YUI Library, Dojo Toolkit, …)
JavaScript MVC Framework (SproutCore, JavaScriptMVC, Backbone.js, …)
Template Frameworks (Mustache, JAML, Eco, ICanHaz.js, JQote 2, …)
Testing Frameworks (QUnit, Jasmine SinonJS, …)
Mobile Frameworks (PhoneGap, Zepto, …)
Dynamic Stylesheet Language (SASS, LESS)

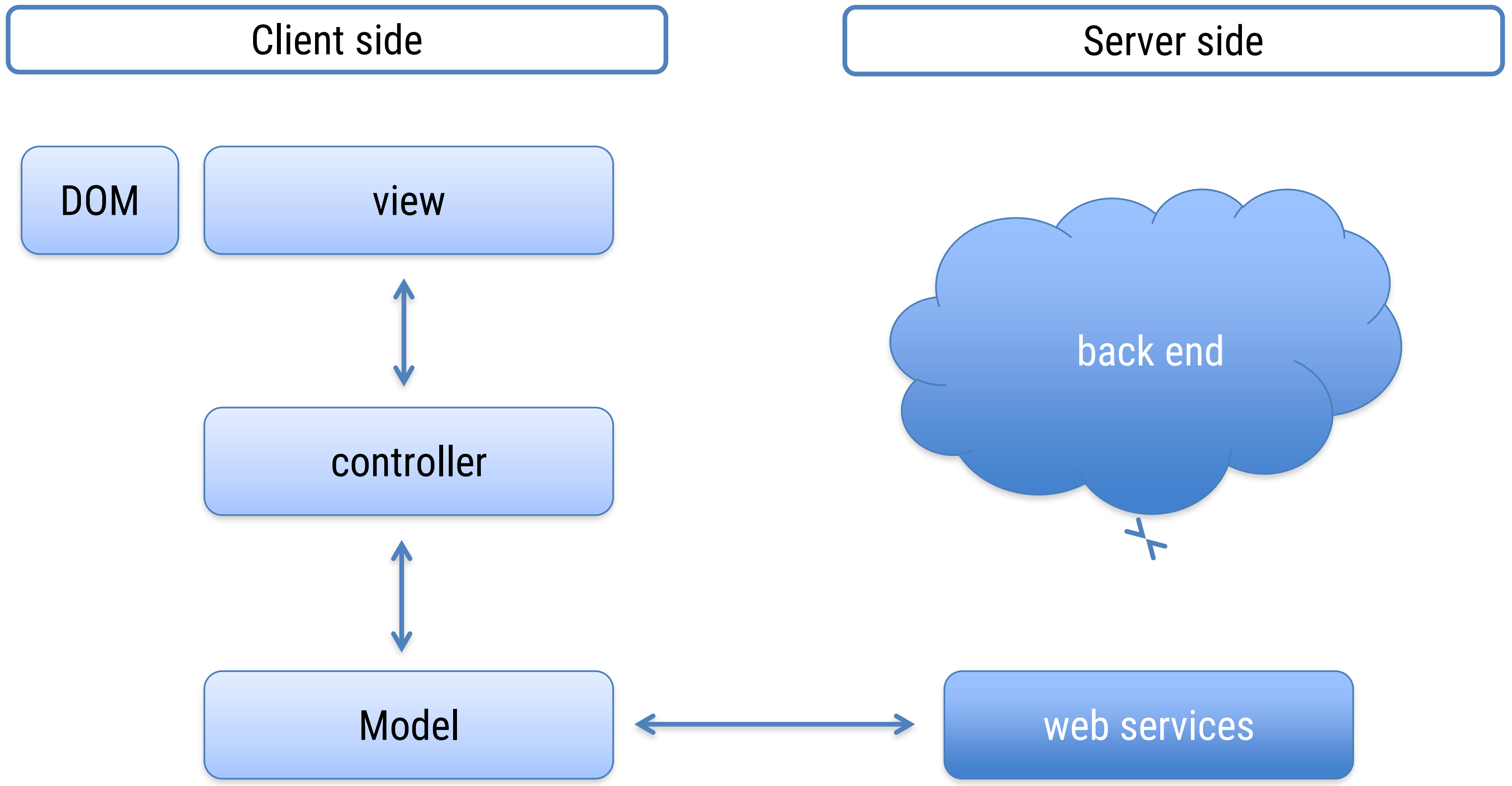
Why MVC JavaScript ?

- Front end developers know JavaScript
- Better organization to front end applications
- Abstract complexity
- Good integration with JavaScript frameworks
- An easier way to do tests

Documentation

- ***SproutCore:*** good documentation, examples, blog, mailing list, irc, videos at Vimeo.
- ***JavaScriptMVC:*** excellent documentation, examples, blog, forum.
- ***Backbone.js:*** documentation ok, mailing list, irc, videos at PeepCode.

Architecture



Responsibilities

View

- Binding DOM
- Render templates

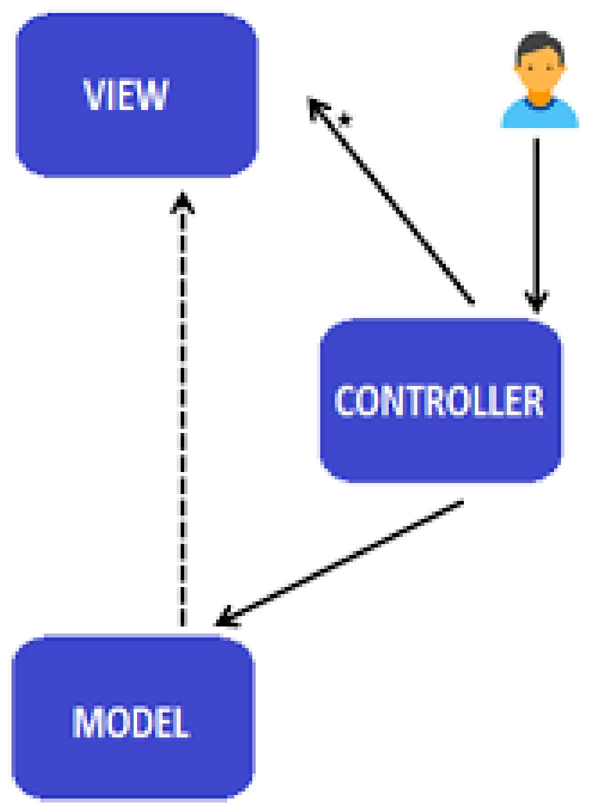
Controller

- Router
- Initialize and create models
- Manipulate DOM

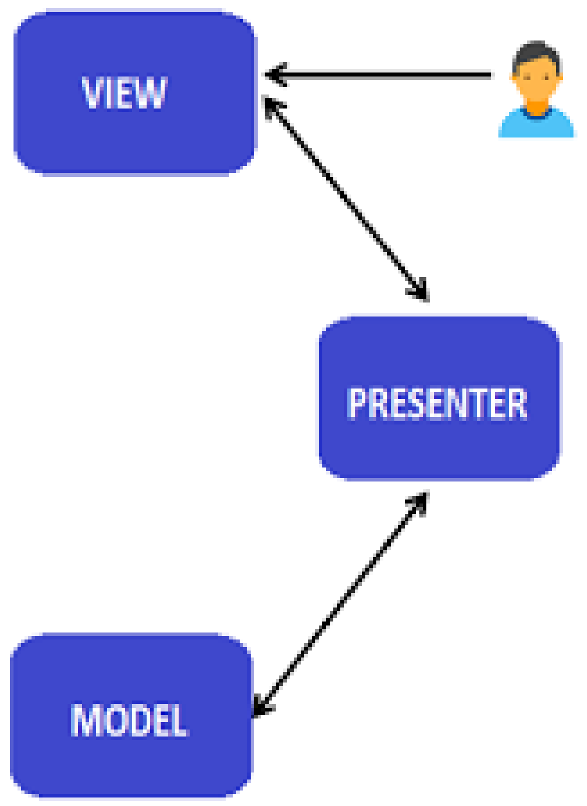
Model

- Data Validation
- Synchronize Data
- Call web services

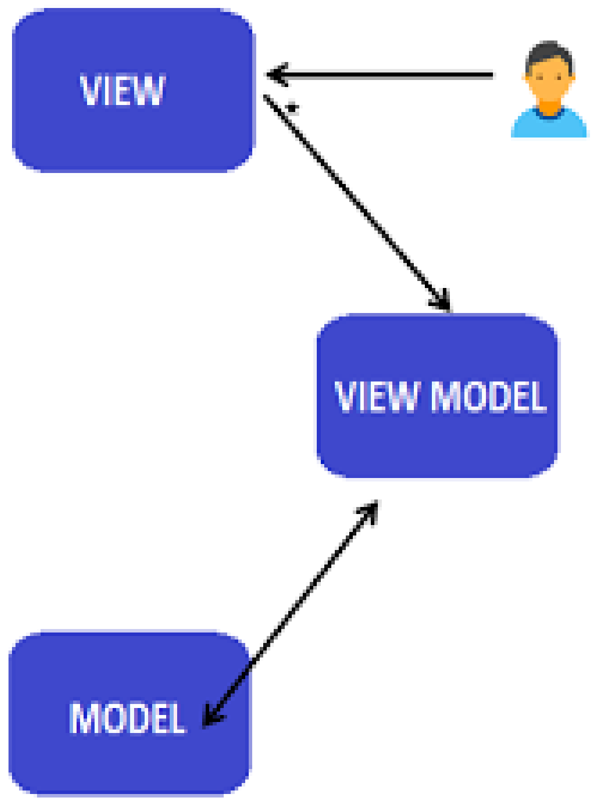
MVC vs MVP vs MVVM



MVC



MVP



MVVM

Credit

Thanks to:

- Javier Humarán (Instituto Tecnológico de Sonora)
- André de Santi Oliveira
- Rick Mercer (the University of Arizona)
- Ankit Sinhal (MasterCard)