MVC

Web Dev, Spring 2021

Last week

JavaScript code to manipulate the DOM in response to events

- we'll practice this on homework 3

The example we saw felt very *ad hoc*

- there was no real process, no real structure
- easy to imagine this leading to unmaintainable code

How can we structure code?

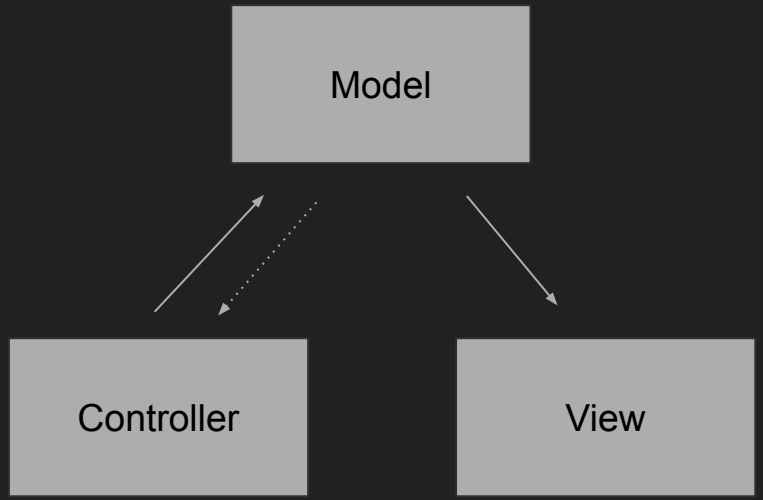
Most systematic approaches to frontend code are based on the Model-View-Controller design pattern

Design pattern: *a general, reusable solution to a commonly occurring problem within a given context in software design [...] a description or template for how to solve a problem that can be used in many different situations*

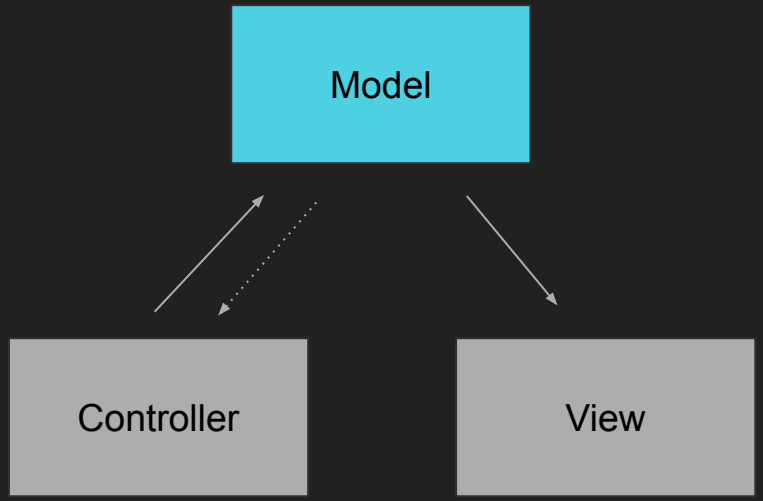
MVC pattern — developed in the 1980s in the context of the SmallTalk development environment for writing GUIs

- Application as a whole
- Various components of an application

Basics of MVC



Basics of MVC



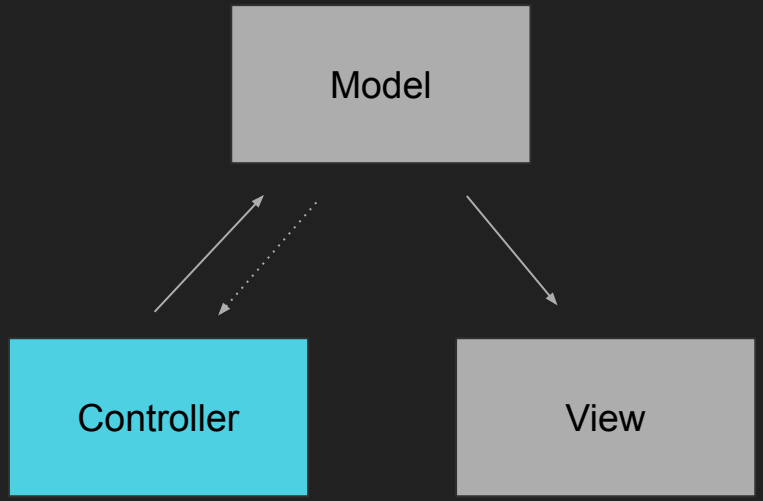
Model

Some interesting application state

Exports a set of actions that controllers can use to modify that state

Can inform views of changes to that state

Basics of MVC



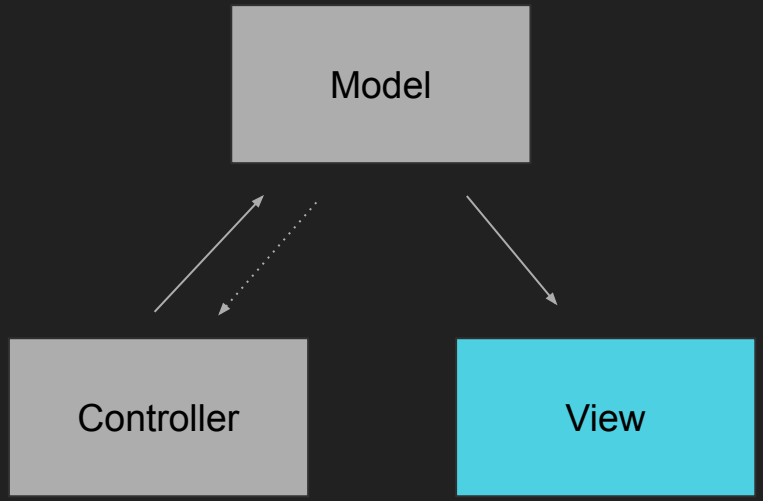
Controller

UI elements that cause changes in the model

Calls model actions to change the model

Can also be informed of model changes

Basics of MVC

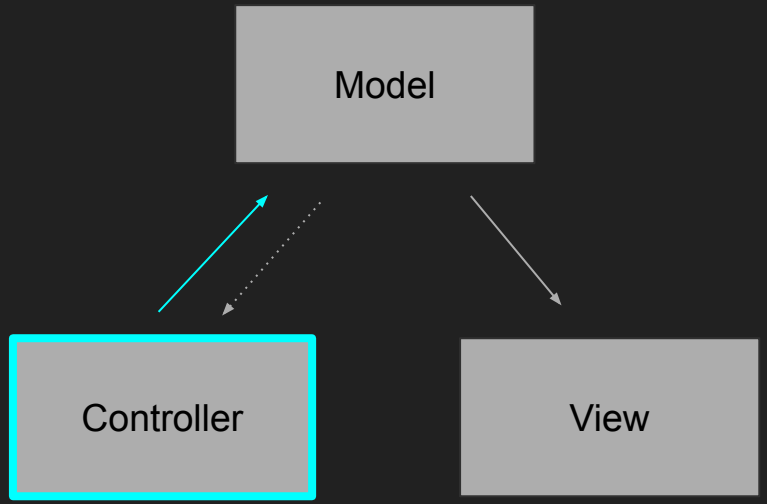


View

Visualizations based on the model

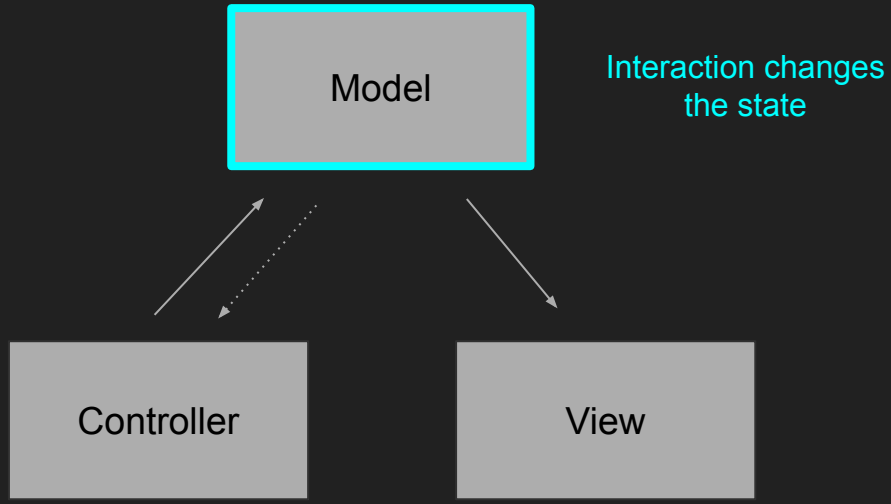
Informed of model changes

Basics of MVC

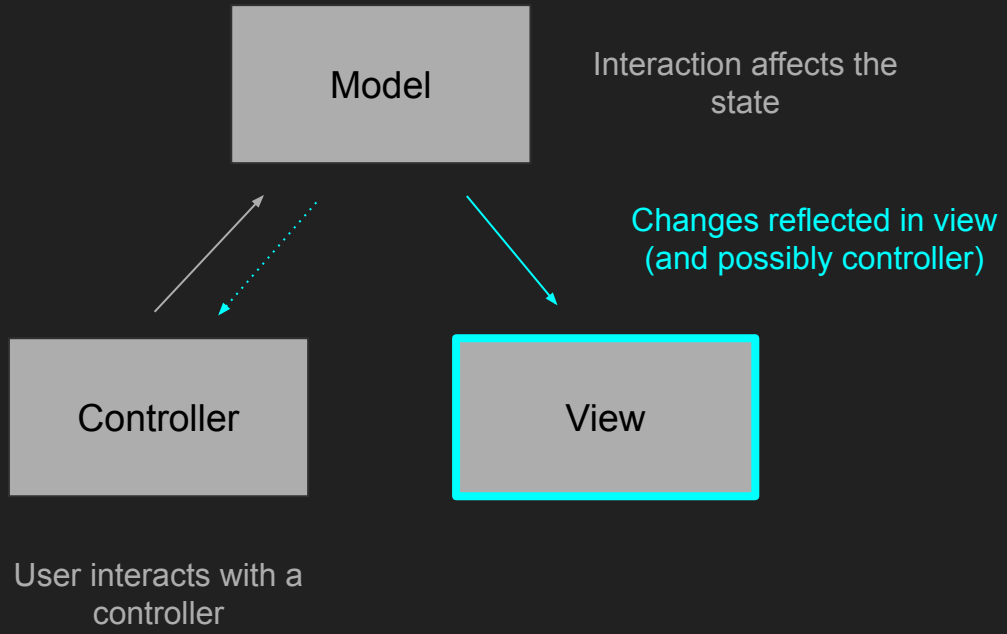


User interacts with a controller

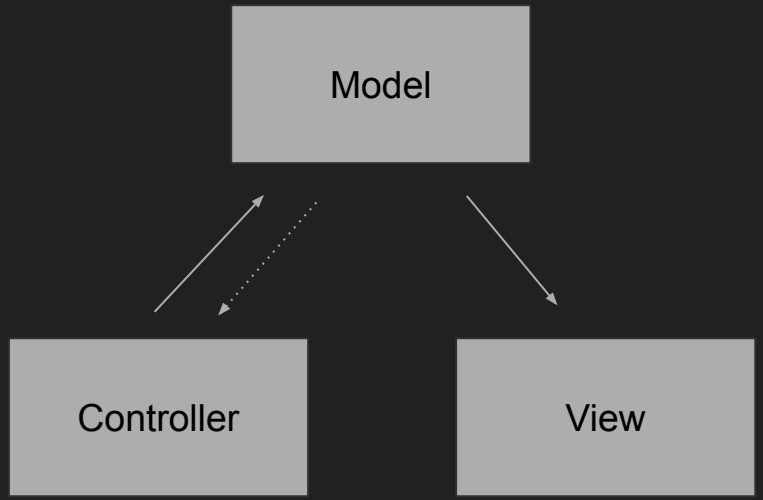
Basics of MVC



Basics of MVC



Basics of MVC



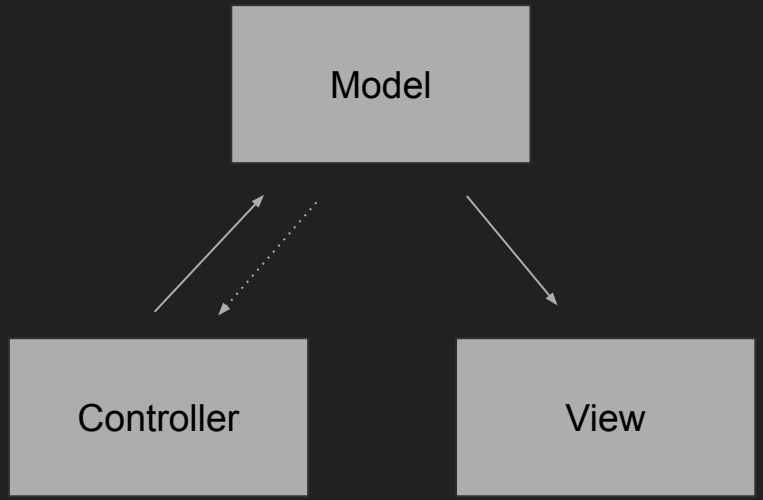
Many controllers/views can be hooked up to a model

Model is independent of controllers and views

Controllers invoke model actions

What about views?

Basics of MVC



Views are informed of changes in the model by subscribing to the model and getting told when the model changes

Pass a function to the model to be called when the state changes

- Publish-Subscribe pattern

Example: Picture Selector from last time

MODEL

State
list of pictures
current picture

Actions
changePicture
addPicture

Subscriptions
changedPicture
addedPicture

Example: Picture Selector from last time

SELECT CONTROLLER

on <SELECT> change:
action changePicture

subscribed to addedPicture:
add new <OPTION>

MODEL

State
list of pictures
current picture

Actions
changePicture
addPicture

Subscriptions
changedPicture
addedPicture

Example: Picture Selector from last time

SELECT CONTROLLER

on <SELECT> change:
action changePicture

subscribed to addedPicture:
add new <OPTION>

ADD PICT CONTROLLER

on <BUTTON> click:
read picture info
clear info
action addPicture

MODEL

State
list of pictures
current picture

Actions
changePicture
addPicture

Subscriptions
changedPicture
addedPicture

Example: Picture Selector from last time

SELECT CONTROLLER

on <SELECT> change:
action changePicture

subscribed to addedPicture:
add new <OPTION>

ADD PICT CONTROLLER

on <BUTTON> click:
read picture info
clear info
action addPicture

MODEL

State
list of pictures
current picture

Actions
changePicture
addPicture

Subscriptions
changedPicture
addedPicture

IMAGE VIEW

subscribed to changedPicture:
update name
update image

Example: Picture Selector from last time

SELECT CONTROLLER

on <SELECT> change:
action changePicture

subscribed to addedPicture:
add new <OPTION>

ADD PICT CONTROLLER

on <BUTTON> click:
read picture info
clear info
action addPicture

MODEL

State
list of pictures
current picture

Actions
changePicture
addPicture

Subscriptions
changedPicture
addedPicture

IMAGE VIEW

subscribed to changedPicture:
update name
update image

ADDED VIEW

on <IMG> hover:
show time added

subscribed to changedPicture:
update time added

MVC as design guide

We are *not* going to follow the MVC pattern in any sort of dogmatic way

It's going to be an inspiration and a guide when we seek to structure code

When we look at frontend web frameworks, we are going to see the kind of split that MVC advocates arise naturally

There isn't a single right way to structure code inspired by MVC —
many choices I'll make in my examples can be debated

“A foolish consistency is the hobgoblin of little minds”