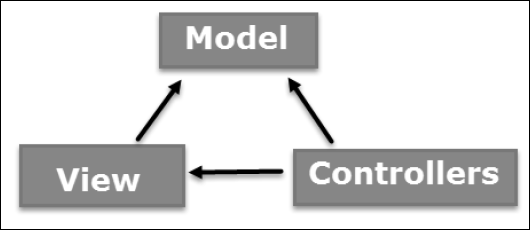
**MVC Pattern – Model- View – Controller Pattern**

MVC is a pattern for organising code in an application to improve maintainability.



# The Model: The Application Object

* The model defines what data the app should contain
* The model is responsible for managing the program’s data.
* A model could be a single object or it could be some structure of objects
* Model contains NO view/Controller code
* Model NEVER tells View/Controller what to do
* The model in To-Do app, it defines tasks or the list of Task

# The View: Screen Presentation

* View is a visual representation of app and defines how the app’s data should be displayed.
* This part deals with UI and trigger action in controller for DOM Event Handling

- e.g. user clicks ‘add task’ button

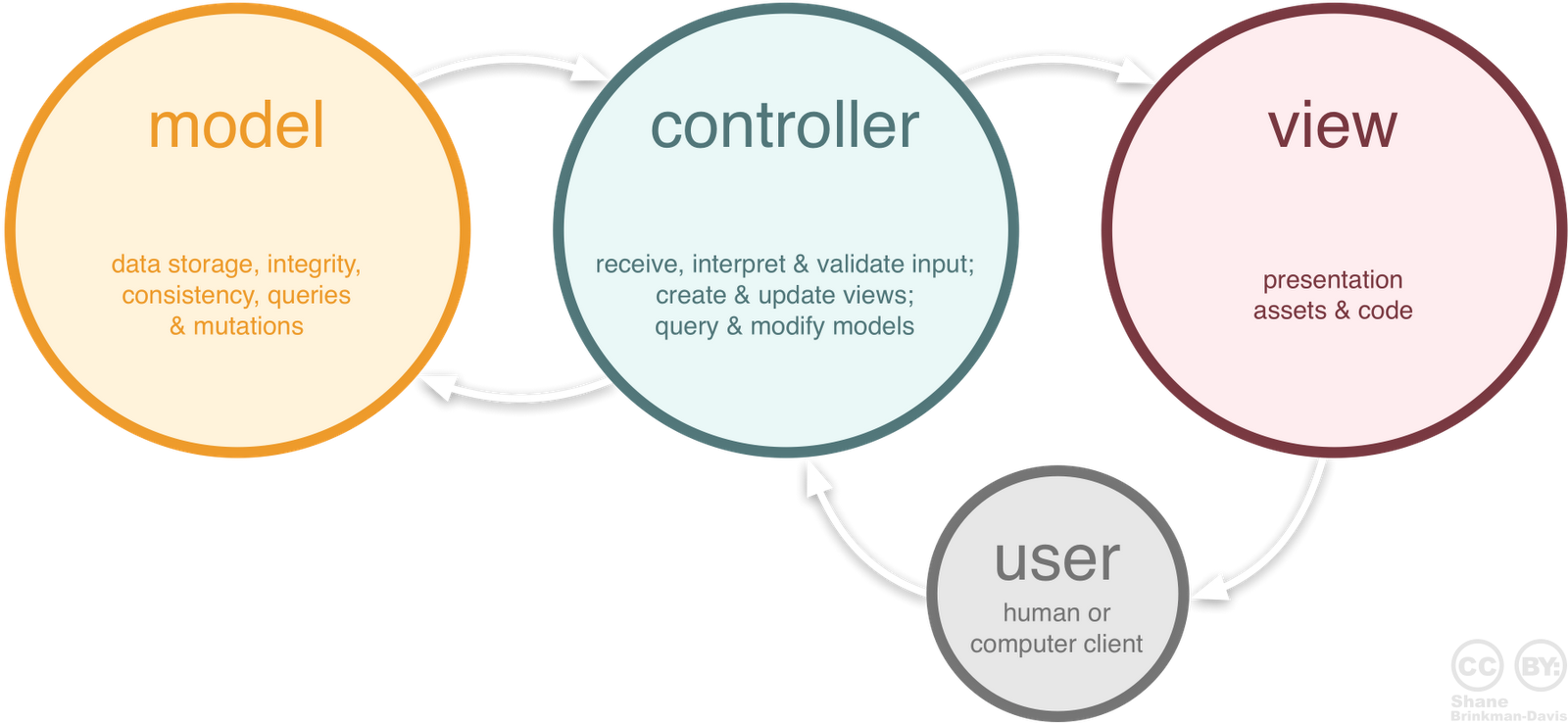
- e.g. a new task appears on screen (data displayed from the model)

# The Controller: the way UI acts to user input

* This is the middle person acting on both model and view.
* The Controller is responsible for handling request and connecting between Model and View.
* The Controller handles external input to the system invoking modification of Model.
* It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

# The Router

* Selects the right controller to handle a request



# **Flow Steps**

**Step 1** − The client browser sends request to the MVC Application.

**Step 2** – app.ts receives this request, and pass the request to the appropriate controller.

**Step 3** − The Controller processes the data using Model and invokes the appropriate method using ControllerActionInvoker object

**Step 4** − The processed Model is then passed to the View, which in turn renders the final output.

# **MVC in Action**

<https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm>

Observer pattern can be implemented – between model and view

MVC decouples views and models by establishing a subscribe/notify protocol between them. A view must ensure that its appearance reflects the state of the model. Whenever the model’s data changes, the model notifies views that depend on it. In response, each view gets an opportunity to update itself.