Τεχνολογίες Εφαρμογών Ιστού

MVC

Example

- Assume a webpage that is meant to present some announcements
- When a new announcement is created, it is stored in a datastore
- The content of the datastore must be reflected in the webpage, perhaps as soon as it changes

Problems to solve

- How the information from the datastore should be relayed to the webpage?
  - Data model?
  - Data exchange protocol?
- How the information that arrives on the webpage is depicted on the webpage?
  - Presentation? How each announcement should look like?
  - Mechanism for dynamic content handling? How the webpage content is updated at runtime?
- How new announcements are added in the datastore?

How the information from the datastore should be relayed to the webpage?

- Data model
  - Information from the datastore can be extracted by a piece of code as an array of objects. One object per announcement. E.g.

```
[
    {
        "title":"announcement#1",
        "body":"lorem ipsum",
        "timestamp":123456
    },
    {
        "title":"announcement#2",
        "body":"lorem ipsum",
        "timestamp":123567
    }
]
```

How the information from the datastore should be relayed to the webpage?

- Data model
  - Information from the datastore can be extracted by a piece of code as an array of objects. One object per announcement. E.g.,

```
[
   {
      "title":"announcement#1",
      "body":"lorem ipsum",
      "timestamp":123456
   },
   {
      "title":"announcement#2",
      "body":"lorem ipsum",
      "timestamp":123567
   }
]
```

Model

How the information from the datastore should be relayed to the webpage?

- Data exchange protocol
  - Through an API that the webpage can use
    - Perhaps a RESTful API so data can be pulled (e.g. http://www.mydomain.xyz/API/getAnnouncements)
    - Perhaps a Websocket-based API so data can be pushed

How the information from the datastore should be relayed to the webpage?

- Data exchange protocol
  - Through an API that the webpage can use
    - Perhaps a RESTful API so data can be pulled (e.g. http://www.mydomain.xyz/API/getAnnouncements)
    - Perhaps a Websocket-based API so data can be pushed

Model

How the information that arrives on the webpage is depicted on the webpage?

- Presentation? How each announcement should look like?
  - HTML templates and some code that generates and updates them. E.g.,

```
<div class= “announcements”>
   <p id=”title” class= “title”>(Announcement title here)</p>
   <p id= “body” class= “body”>(Announcement body here)</p>
   <p id= “time” class= “time”>(Time/date)</p>
<div>
```

How the information that arrives on the webpage is depicted on the webpage?

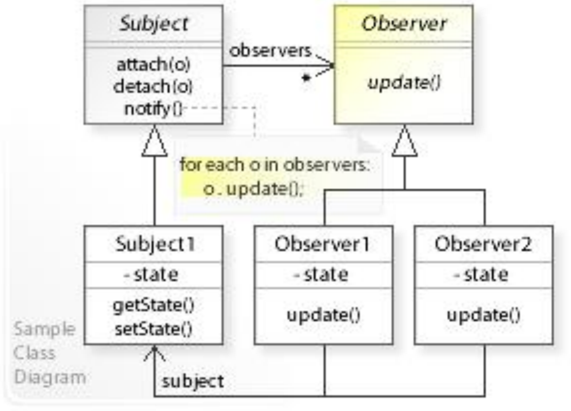
- Presentation? How each announcement should look like?
  - HTML templates and some code that generates and updates them. E.g.,

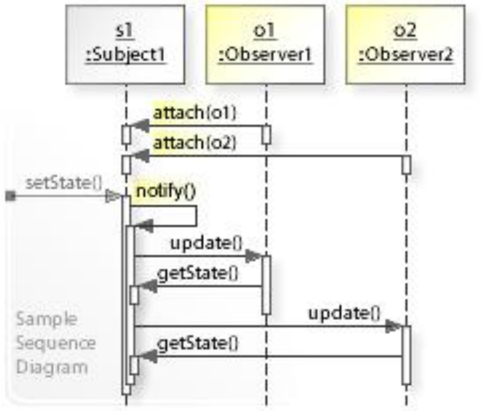
```
<div class= “announcements”>
   <p id=”title” class= “title”>(Announcement title here)</p>
   <p id= “body” class= “body”>(Announcement body here)</p>
   <p id= “time” class= “time”>(Time/date)</p>
<div>
```

View

How the information that arrives on the webpage is depicted on the webpage?

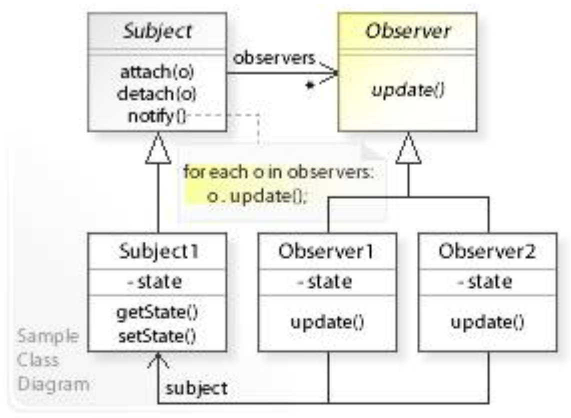
- Mechanism for dynamic content handling? How the webpage content is updated at runtime?
  - DOM: We can write up some code in Javascript.
  - Observer pattern

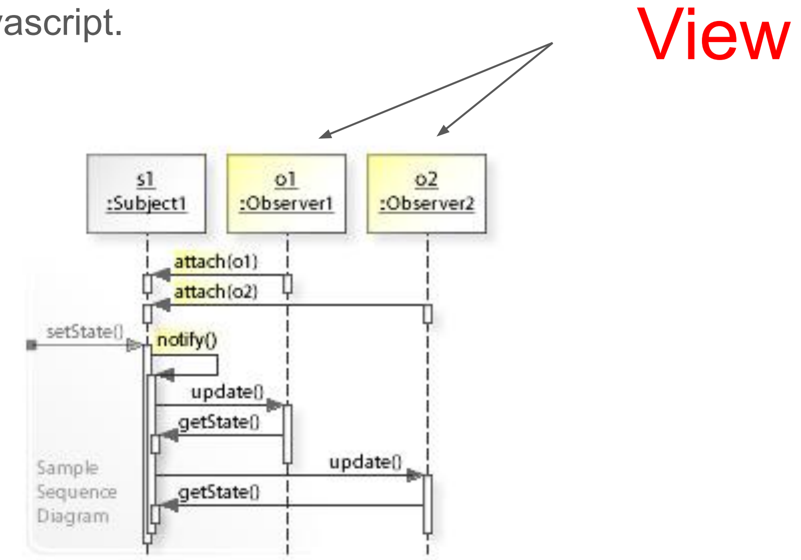




How the information that arrives on the webpage is depicted on the webpage?

- Mechanism for dynamic content handling? How the webpage content is updated at runtime?
    - DOM: We can write up some code in Javascript.
    - Observer pattern





How new announcements are added in the datastore?

- How new announcements are added in the datastore?
  - The UI may allow for the user to insert new data
  - Such changes are transmitted to a piece of code that informs the state, i.e. uses the datastore API (part of the model, remember?) to store the object in the datastore

How new announcements are added in the datastore?

- How new announcements are added in the datastore?
  - The UI may allow for the user to insert new data
  - Such changes are transmitted to a piece of code that informs the state, i.e. uses the datastore API (part of the model, remember?) to store the object in the datastore

Controller

Model-View-Controller (MVC)

- MVC (Model-View-Controller) is a pattern in software design commonly used to implement user interfaces, data, and controlling logic
- It emphasizes a separation between the software's business logic and display.
- This "separation of concerns" provides for a better division of labor and improved maintenance.
- Some other design patterns are based on MVC, such as MVVM (Model-View-Viewmodel), MVP (Model-View-Presenter), and MVW (Model-View-Whatever).

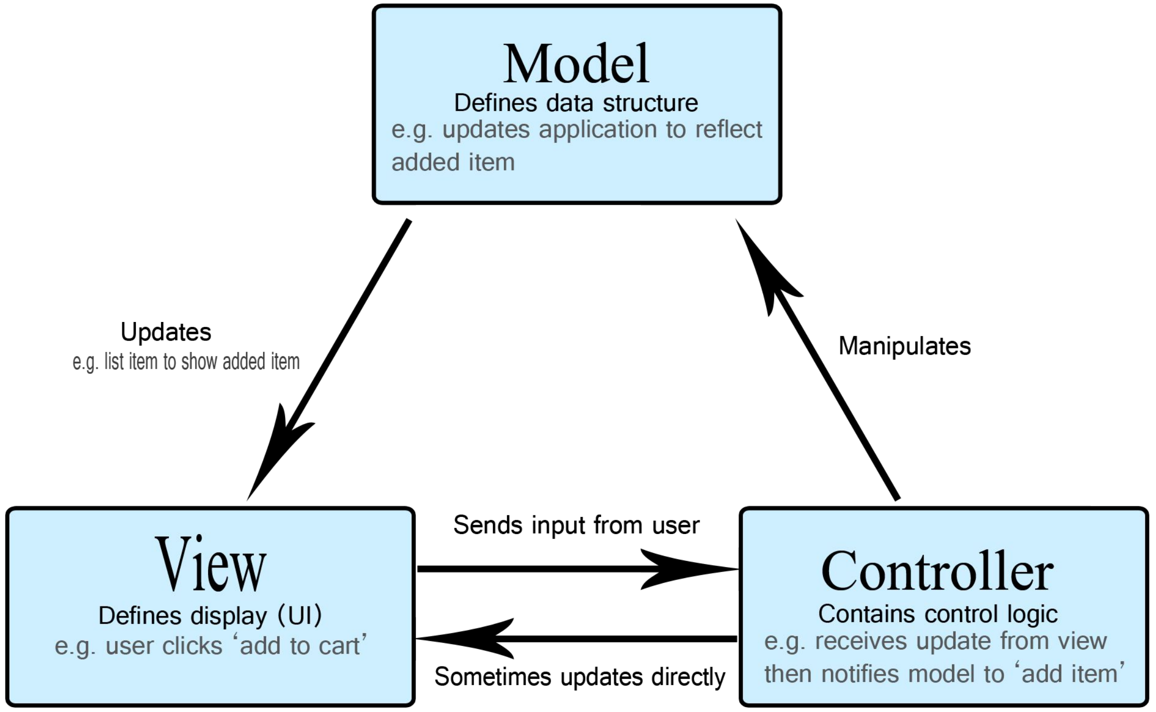
Source: https://developer.mozilla.org/en-US/docs/Glossary/MVC

The three parts MVC

- Model: Manages data and business logic.
- View: Handles layout and display.
- Controller: Routes commands to the model and view parts.

Source: https://developer.mozilla.org/en-US/docs/Glossary/MVC

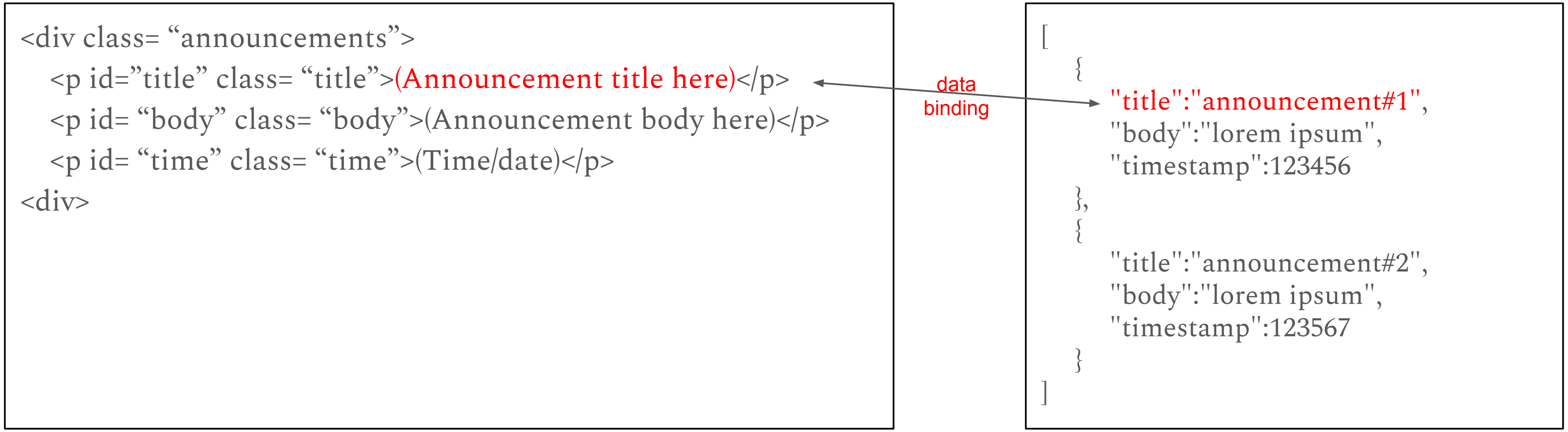
Model-View-Controller (MVC)


Model
Defines data structure
e.g. updates application to reflect added item
Updates
e.g. list item to show added item
Manipulates
View
Defines display (UI)
e.g. user clicks 'add to cart'
Sends input from user
Sometimes updates directly
Controller
Contains control logic
e.g. receives update from view then notifies model to 'add item'

Source: https://developer.mozilla.org/en-US/docs/Glossary/MVC

Interesting concept

- When we want an item in the View to be informed by the Model but also to inform the model when changes happen to it by the user...
  - ...we call this a “data binding”



```
<div class= “announcements”>
   <p id=”title” class= “title”>(Announcement title here)</p>
   <p id= “body” class= “body”>(Announcement body here)</p>
   <p id= “time” class= “time”>(Time/date)</p>
<div>
```

```
[
   {
      "title":"announcement#1",
      "body":"lorem ipsum",
      "timestamp":123456
   },
   {
      "title":"announcement#2",
      "body":"lorem ipsum",
      "timestamp":123567
   }
]
```

MVC Vs Web MVC

- MVC is a pattern that assumes a single system and programming language upon which an application is executed and created respectively.
- In a web app, there are with two distinct systems (front-/back-end)
- So, it doesn’t feel like 100% fit…
- ...but we can always focus on the front-end
    - Push most of the logic to the client by using client-side data stores, and XMLHttpRequest to allow partial page updates as required