

Templates

Multimedia Design 2
Week 9

Templates

- Where do you put your stuff?

Templates

- Where do you put the **content**?
- Where do you put the **code**?

Templates

- Where do you put the **content**?
 - Sandwich style
 - Monolithic style
- Where do you put the **code**?
 - Two-layer approach
 - MVC approach



Sandwich style

Sandwich style

- Simple: header and footer
- Everything in-between is your page content
- WordPress uses a variation on this

header.php

```
<!DOCTYPE html>
<html>
  <head>
    <title>Sandwich style</title>
  </head>
  <body>
    <div id="page">
```

footer.php

```
        </div>  
    </body>  
</html>
```

index.php

```
<?php  
  
include "header.php";  
  
?>  
<h1>Home</h1>  
<p>...</p>  
<?php  
  
include "footer.php";  
  
?>
```

about.php

```
<?php  
  
include "header.php";  
  
?>  
<h1>About me</h1>  
<p>...</p>  
<?php  
  
include "footer.php";  
  
?>
```

photos.php

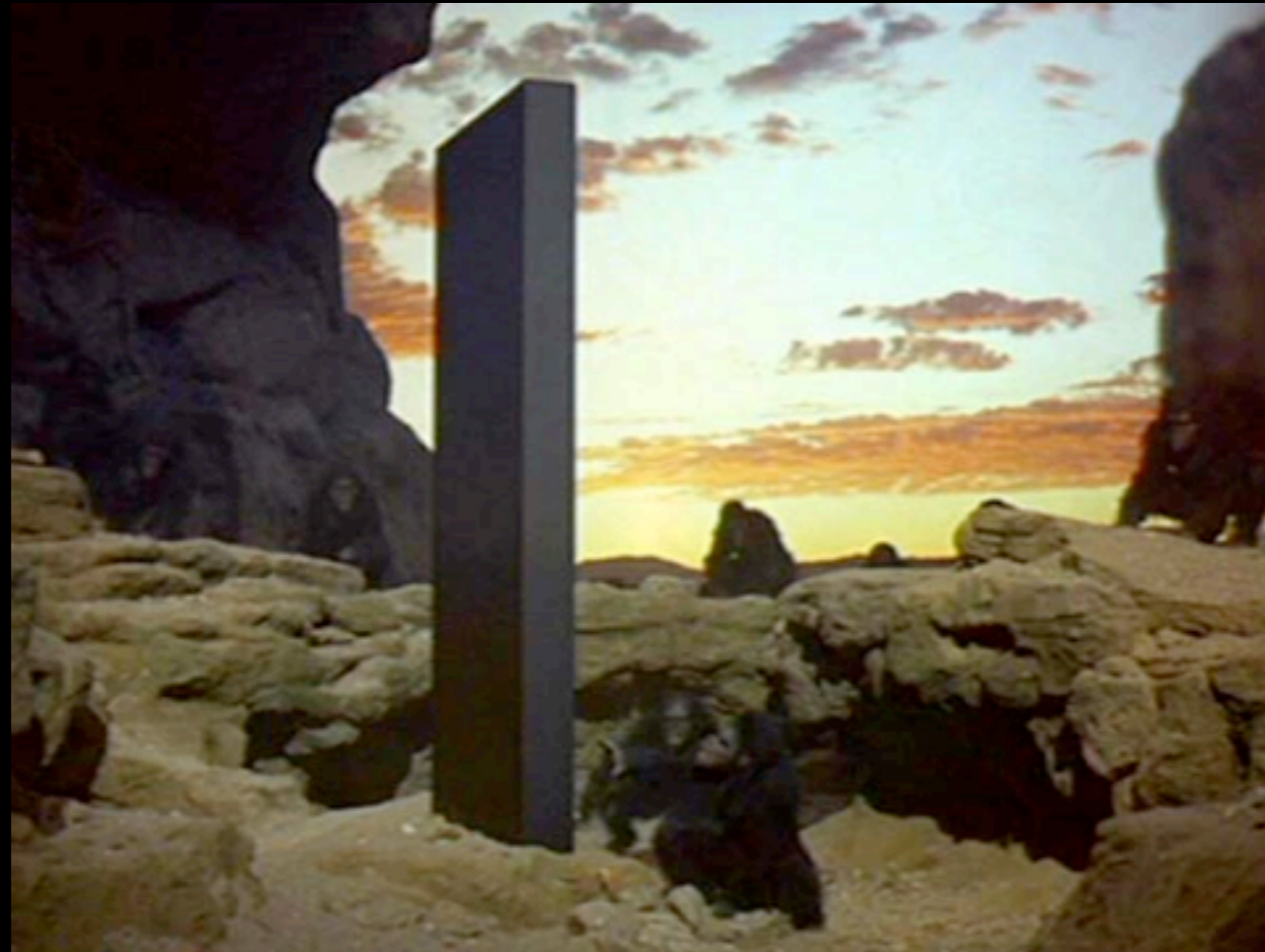
```
<?php  
  
include "header.php";  
  
?>  
<h1>My photos</h1>  
  
<?php  
  
include "footer.php";  
  
?>
```

Advantages

- Site-wide code is isolated to header.php and footer.php
- Easy to understand

Disadvantages

- It is awkward matching HTML tags between header and footer
- Some code still needs to go before the header (e.g., page title), which is slightly weird



Monolithic style

Monolithic style

- header.php + footer.php = template.php
- Which parts are common to all pages and which parts are dynamic?

template.php

```
<!DOCTYPE html>
<html>
  <head>
    <title><?php echo $title; ?></title>
  </head>
  <body>
    <div id="page">
      <?php include $main; ?>
    </div>
  </body>
</html>
```

index.php

```
<?php
```

```
$title = "Monolithic style";  
$page = "pages/home.php";  
include "template.php";
```

```
?>
```

index.php

```
<?php

$title = "Home";
if ($user->loggedIn()) {
    $page = "pages/restricted.php";
} else {
    $page = "pages/normal.php";
}
include "template.php";

?>
```

Advantages

- Easier to modify the template in one file
- Decouples the script URL from what ends up being displayed

Disadvantages

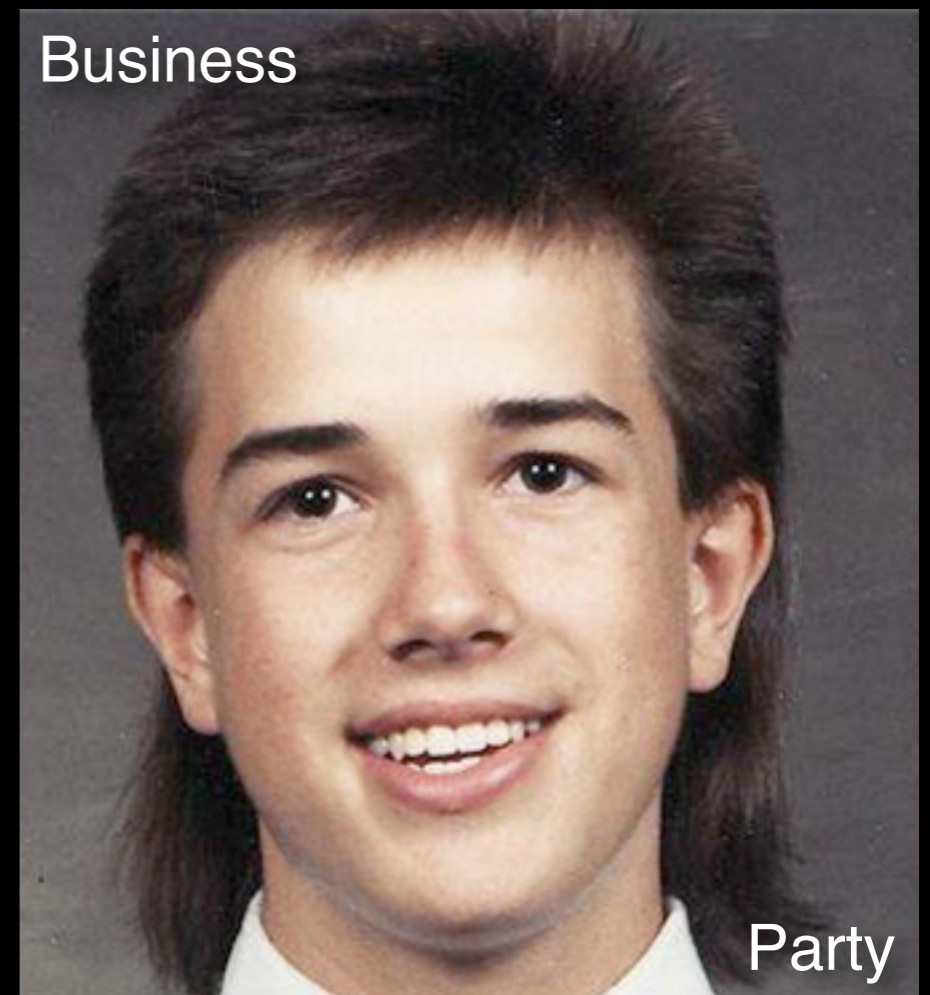
- More complex than sandwich style
- Less delicious



Two-layer approach

Two-layer approach

- Kind of a reverse-mullet
- Business in the back(-end)
- Party in the front(-end)
- Separates business logic from presentation layer



Mullet

index.php again

```
<?php

// It's all business in here
if ($user->loggedIn()) {
    // These guys get to party hard
    $page = "pages/restricted.php";
} else {
    // A slightly lesser party over here
    $page = "pages/normal.php";
}

// Everyone is invited to this party
include "template.php";
```

Code responsibility

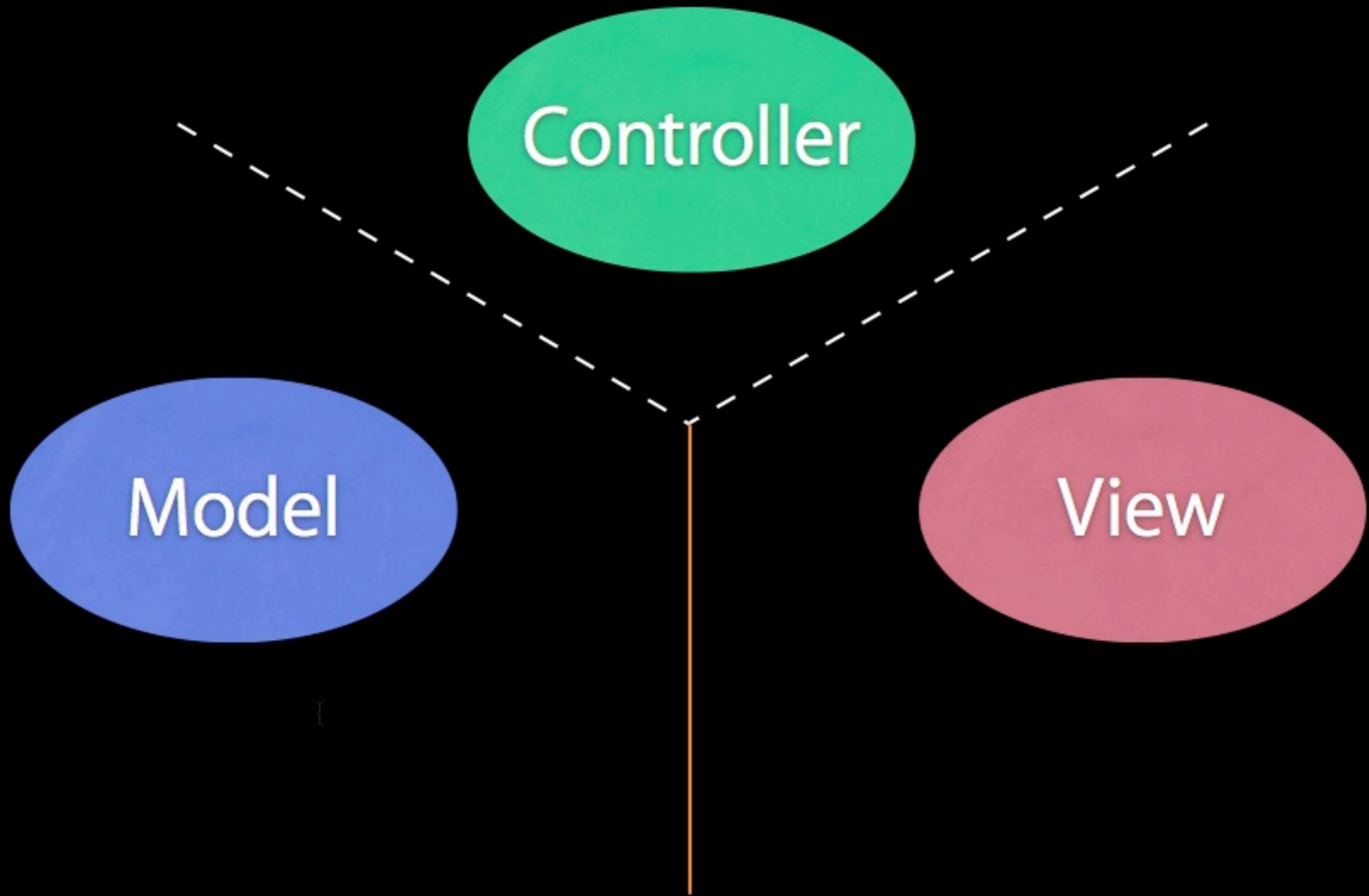
- **Business logic:** which content should be shown? How is it assembled?
- **Presentation layer:** how should it be displayed?

Code responsibility

- **Back-end developer** handles the logic layer (PHP, Ruby, Python, etc.)
- **Front-end developer** handles the presentation layer (HTML, CSS, JS)
- Sometimes this is the same person, but separating the code is still helpful

Git'er done

- The two-layer approach is a quick and easy way to structure websites
- Less object-oriented fussiness
- As code gets more complex this approach becomes cumbersome



MVC approach

Model, View, Controller

- **Models** manage data
- **Views** display data
- **Controllers** choose which models get handed off to which views

Model, View, Controller

- **Models** manage data
- **Views** display data
- **Controllers** choose which models get handed off to which views

Model, View, Controller

- **Models** manage data
- **Views** display data
- **Controllers** choose which models get handed off to which views

MVC is basically the two-layer approach but with **models** thrown in

Models

- Data structures with methods for storing and retrieving
- Using **models** makes object-oriented programming more natural

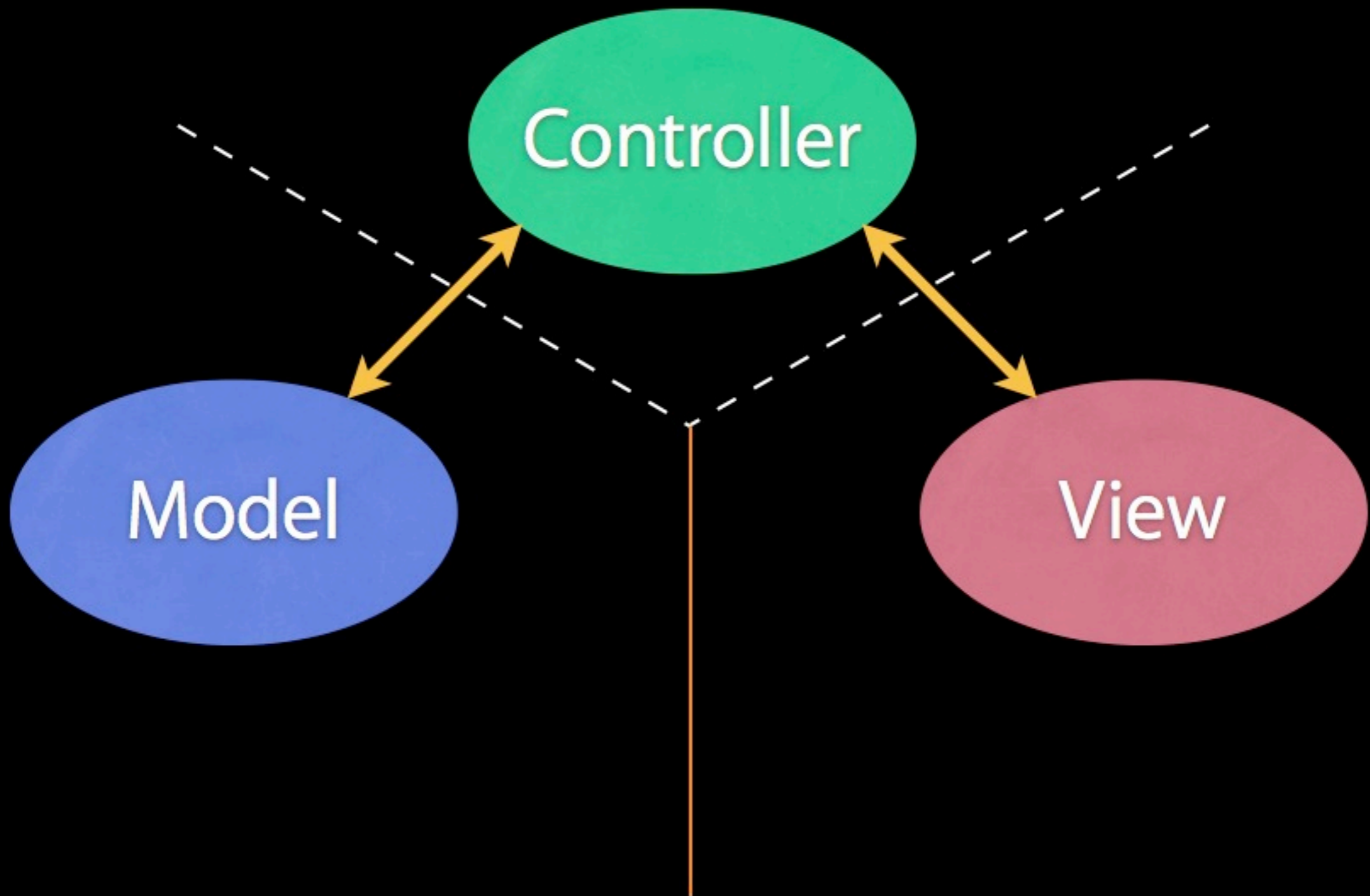
Controllers

- Which page gets shown?
- Which data are loaded from where?
- Which view is used?

Views

- The interface-y part
- HTML, CSS and JavaScript
- Includes, conditionals, loops

Who talks to who



MVC is sensible

- Good for dealing with complex apps
- Ruby on Rails, Django, CakePHP, Symfony, Cocoa development all use an MVC pattern

Next week

```
<?php
```

```
$content = $db->query("
    SELECT * FROM page
");
```

```
?>
```