Level 2 - Section 1

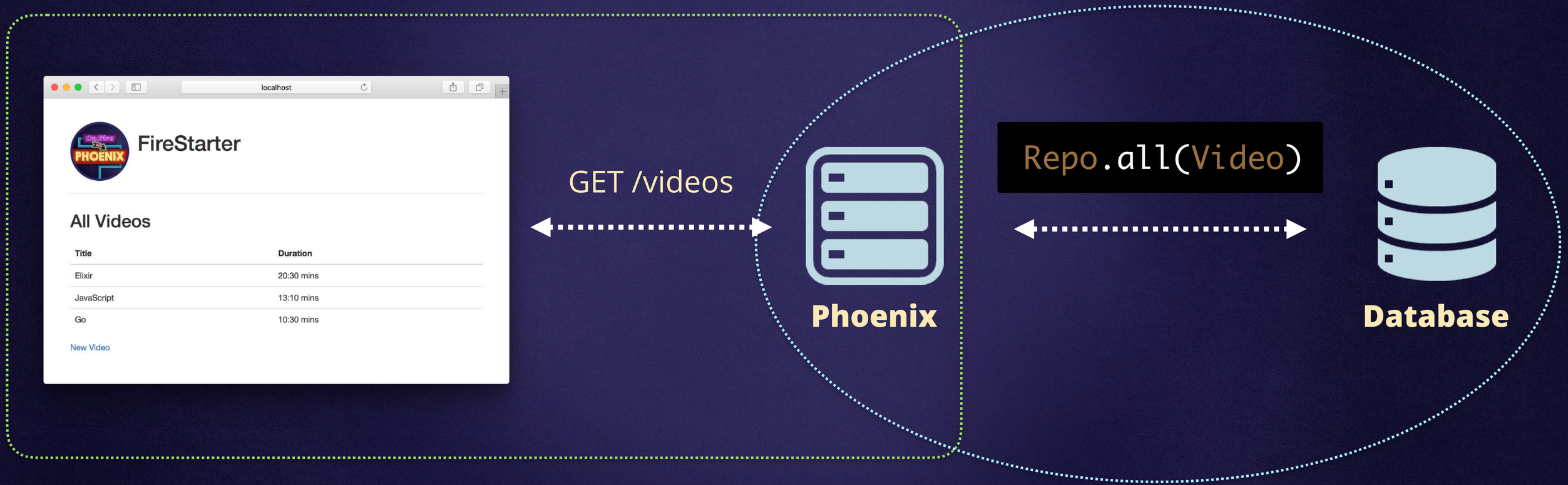
Responding with Data

Sending Data Back and Rendering HTML

Listing Videos

We can read from the database. Now let's learn how to return this data in a **response.**

Step 1 - Read from database



Step 2 - Return response to client (browser)

The C in MVC

The **Controller** part of **MVC** in *Phoenix* includes **Controller Modules** and the **Router.**

Browser

Request

Response

Controller

- Router Module
- Controller Modules

…now we'll write code over here

We've seen how to write code here…

Model

- Schema Modules
- Context Modules

View

- View Modules
- Templates

Database

- Migration Modules

Routing Requests in the Router

A route is composed of 4 things:

1. **HTTP method**
2. **URL path**
3. **Controller name**
   (module name)
4. **Action name**
   (a function from the controller module)

Notice we are at the "`_web`" folder now

lib/fire_starter_web/router.ex

```
defmodule FireStarterWeb.Router do
  ...
  scope "/", FireStarterWeb do
    ...

    get "/videos", VideoController, :index
  end
end
```

The *Controller* module name...

...and the function which will be invoked

The index **action**

Requests to */videos* **are routed to the** index() **function in the** VideoController**.**

lib/fire_starter_web/controllers/video_controller.ex

```
defmodule FireStarterWeb.VideoController do


  def index(conn, _) do

  end
end
```

The first argument passed
by the router is the **connection.**

The second argument is a *Map* with
parameters, but we'll ignore this for now.

Sending text **from the Controller**

The simplest way to respond with plain text from a *Controller* is using the text() **function.**

lib/fire_starter_web/controllers/video_controller.ex

```
defmodule FireStarterWeb.VideoController do
  use FireStarterWeb, :controller

  def index(conn, _) do
    text conn, "Hello from VideoController"
  end
end
```

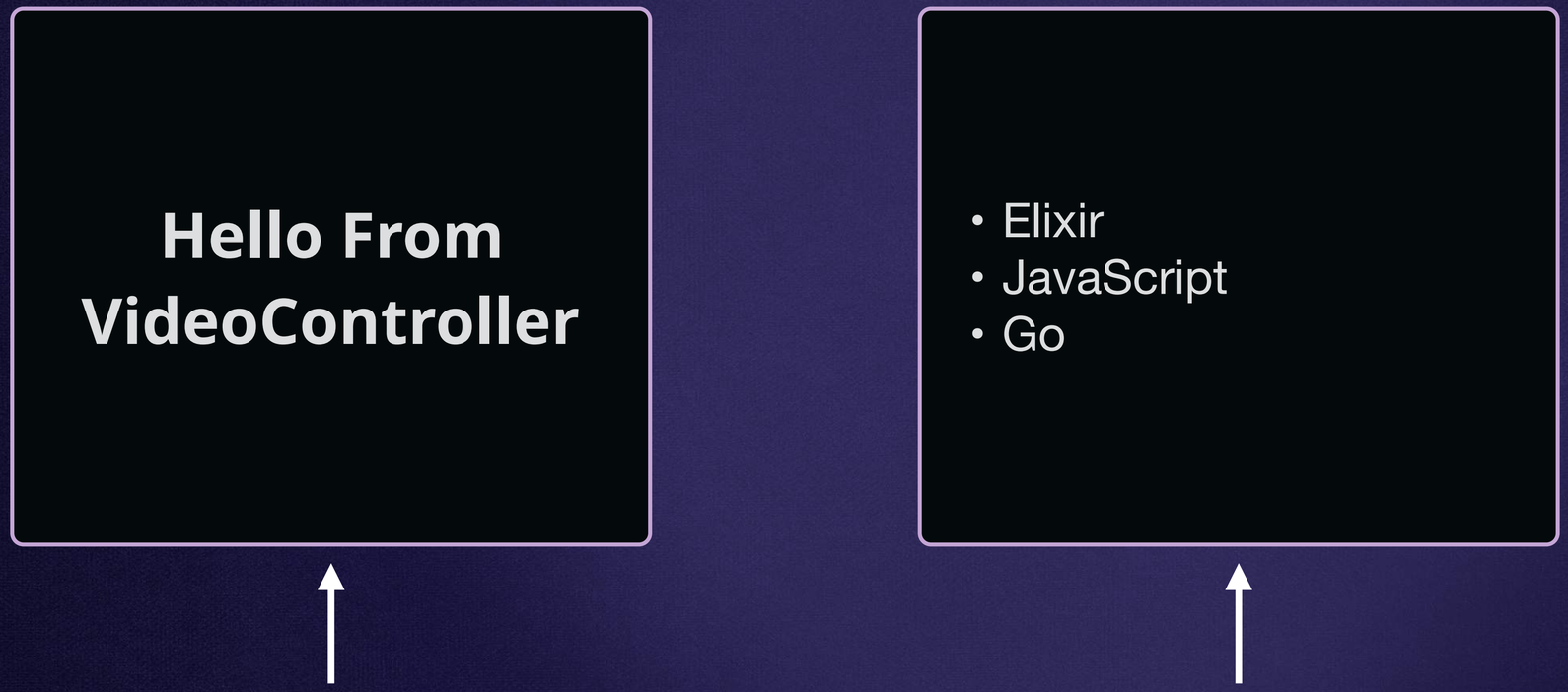
available from here

The **connection** is the first argument...

...and the second argument is a string

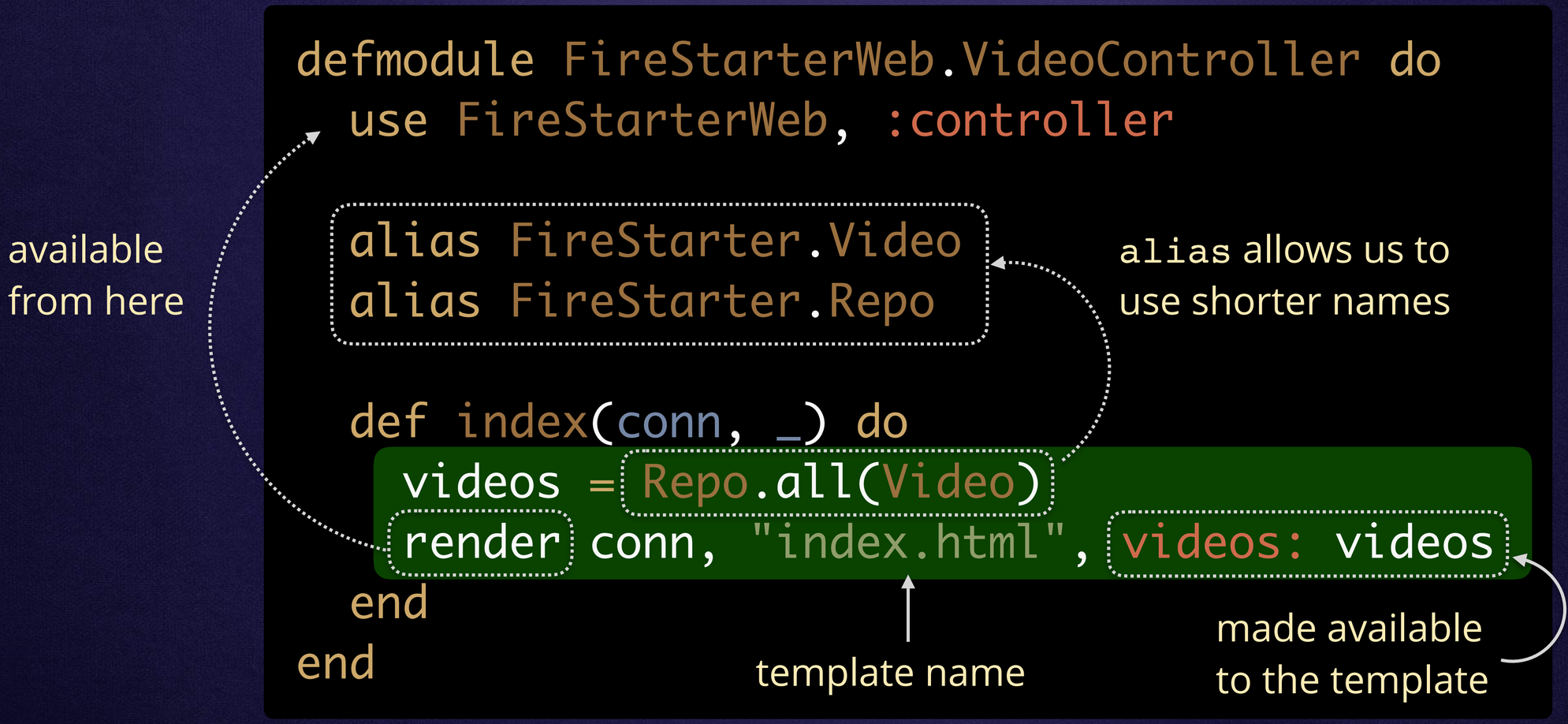
From Text to HTML

Now that we know how to send text back to the client, let’s see how we can respond with HTML.



Setting up Data for the HTML Page

To render HTML with video data from the database we can use the render() function. This function takes three arguments: the **connection,** a **template name** and a **Keyword List.**



Sending Data from *Controller* to *Template*

Controllers use **Schemas** to fetch data from the database which will be read from **Templates.**

Browser

Request

Response

Controller

- Router Module
- Controller Modules

Uses **Schema Modules** to fetch and setup data.

Model

- Schema Modules
- Context Modules

View

- View Modules
- Templates

Database

- Migration Modules

Reads data setup by **Controller Modules**

The Video *Template*

Templates are files compiled on the server and which output **HTML responses.**

lib/fire_starter_web/templates/video/**index.html.eex**

EEx is the default template system in *Phoenix*

```
<h2>All Videos</h2>
<ul>
  <%= for video <- @videos do %>
    <li><%= video.title %></li>
  <% end %>
</ul>
```

Set from the **Controller**

creates a list of videos using *list comprehension*

List comprehensions are used to loop through enumerables:

```
output = for letter <- ["a","b","c"] do
  "Letter: #{letter} "
end
IO.puts output
```

```
Letter: a Letter: b Letter: c
```

Displaying List of Videos in HTML

Now we are successfully displaying a list of videos in HTML

- Elixir
- JavaScript
- Go duration

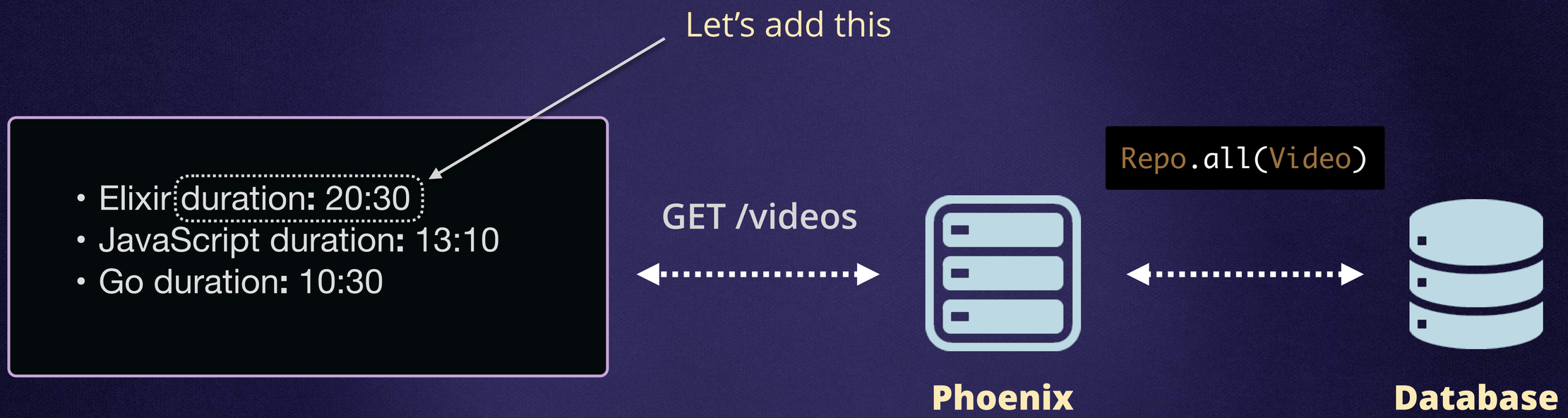
Level 2 - Section 2

Using View Modules

Formatting Data for Templates

Adding *duration* to the list of videos.

We are displaying a list of video titles. Now we want to add the **duration** for each video.



Reading the *duration* property

We can read the *duration* property from each video *Struct* in the template.

lib/fire_starter_web/templates/video/index.html.eex

```
<h2>All Videos</h2>
<ul>
  <%= for video <- @videos do %>
    <li><%= video.title %> duration: <%= video.duration %> </li>
  <% end %>
</ul>
```

Issues with *duration* displayed in seconds

The *duration* is stored in the database as seconds. We need to make this easier to read.

- Elixir duration: 1230
- JavaScript duration: 790
- Go duration: 630

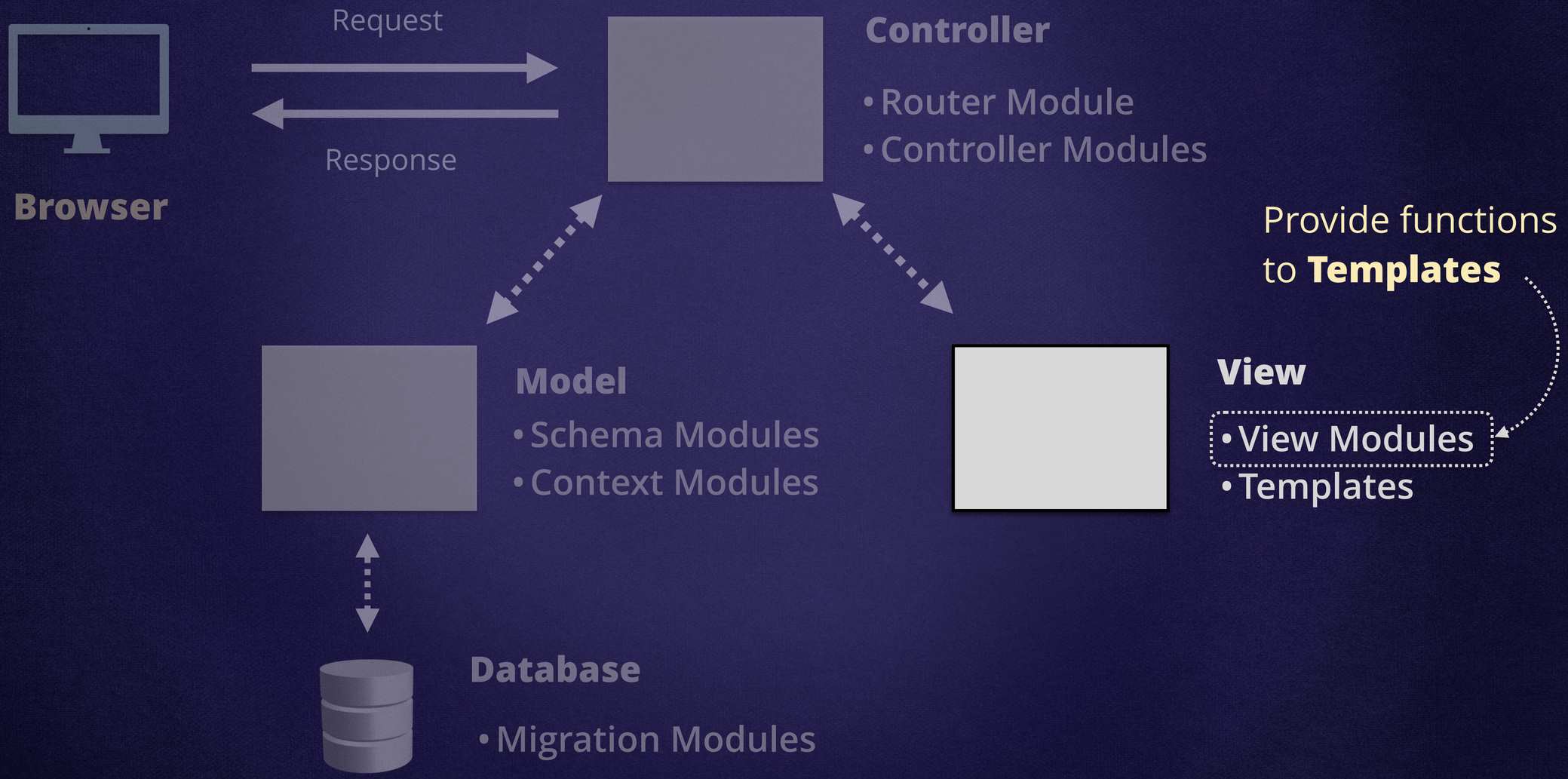
Duration in seconds is hard to understand...

- Elixir duration**: 20:30 mins**
- JavaScript duration**: 13:10 mins**
- Go duration**: 10:30 mins**

It would be much easier to read in this format!

View Modules are the *V* in *MVC*

View Modules are also part of the **V** in **MVC**. They provide **helper functions** for **Templates.**



Functions in the View

Our new helper function expects **one argument** and returns a formatted string.

lib/fire_starter_web/views/video_view.ex

```elixir
defmodule FireStarterWeb.VideoView do
  use FireStarterWeb, :view

  def duration_in_mins(seconds) do
    minutes = div(seconds, 60)
    seconds = rem(seconds, 60)
    "#{minutes}:#{seconds}"
  end
end
```

Performs division and rounds down to the closest integer

Remainder of division by 60

`div` and `rem` are part of the *Kernel* module, automatically imported by Elixir.

Calling a *View* Function

The **Template** can call any function defined in a **View.**

lib/fire_starter_web/templates/video/index.html.eex

```
<h2>All Videos</h2>

<ul>
  <%= for video <- @videos do %>
    <li><%= video.title %>
        duration: <%= duration_in_mins(video.duration) %> mins</li>
  <% end %>
</ul>
```

function is available to the **Template**

The *duration* Is Now Easy to Understand

With the new format, it's easier to understand the duration for each video.