Demystifying the MVC (with AngularJS!)

Brandon Smith

Introduction

- A *Very* Basic Website
- Case Study: Modern Websites using AngularJS
- The Model View Controller
- AngularJS
- How AngularJS implements an MVC model
- Examples
- Questions

A Basic Website

This is a website

There are no scripts or external Javascript libraries

Just some CSS and static HTML to make this page

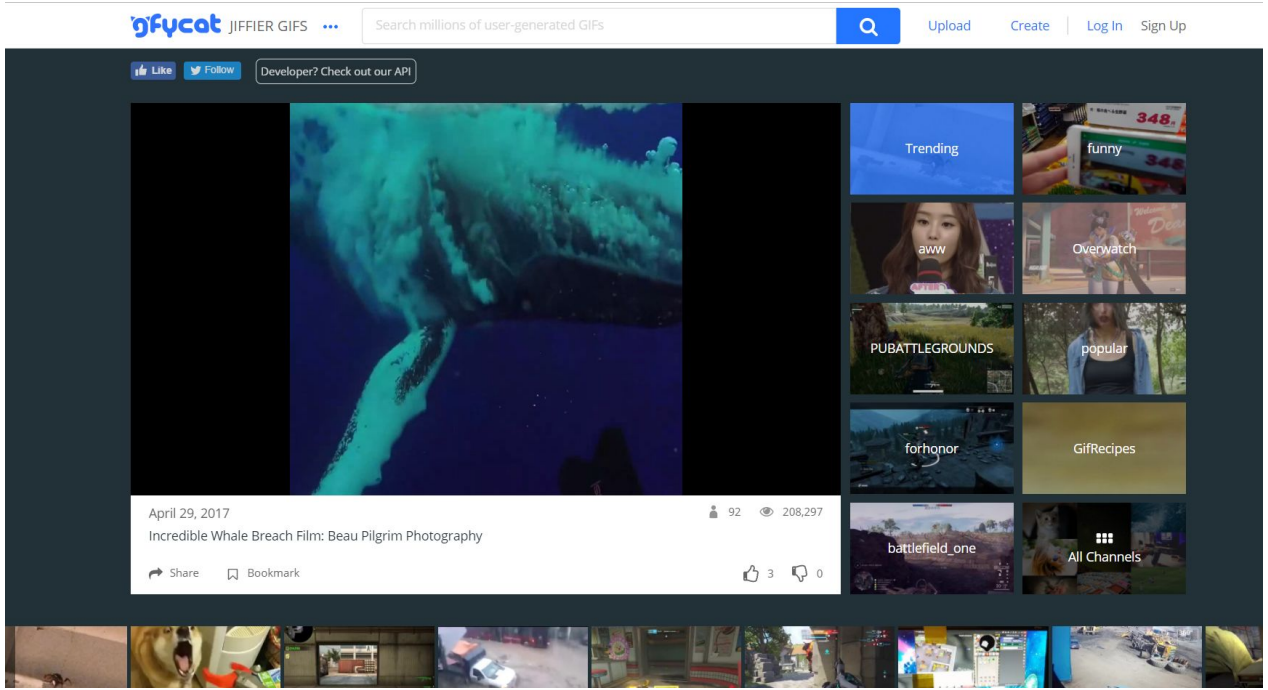
Nothing Special Going on Here

Lame Text Input

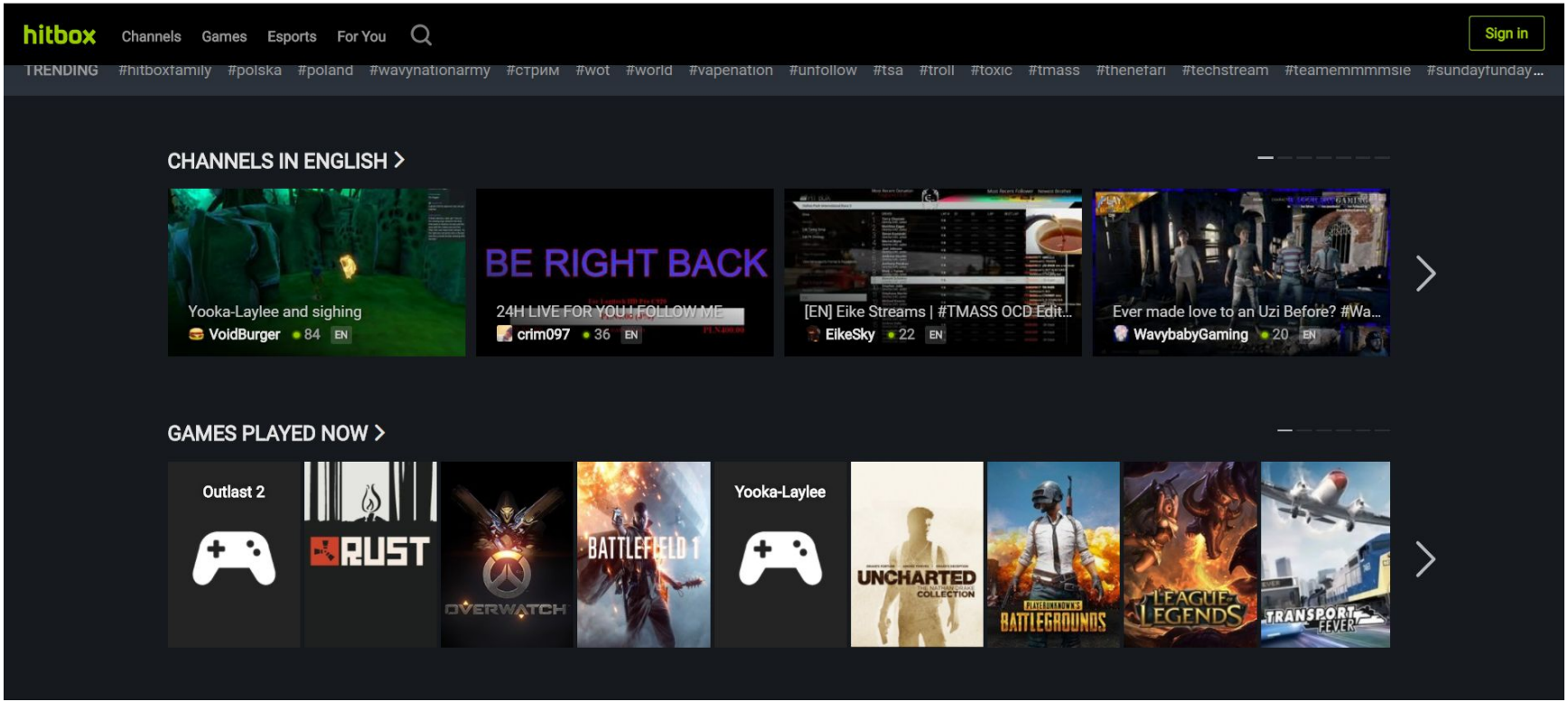
- Static HTML/CSS Only.
- Written by hand.
- Any action requires rendering another page or re-rendering entire DOM.
- No fun.

Case Study: Modern Website Designs

- Example 1: gfycat, GIF sharing website.

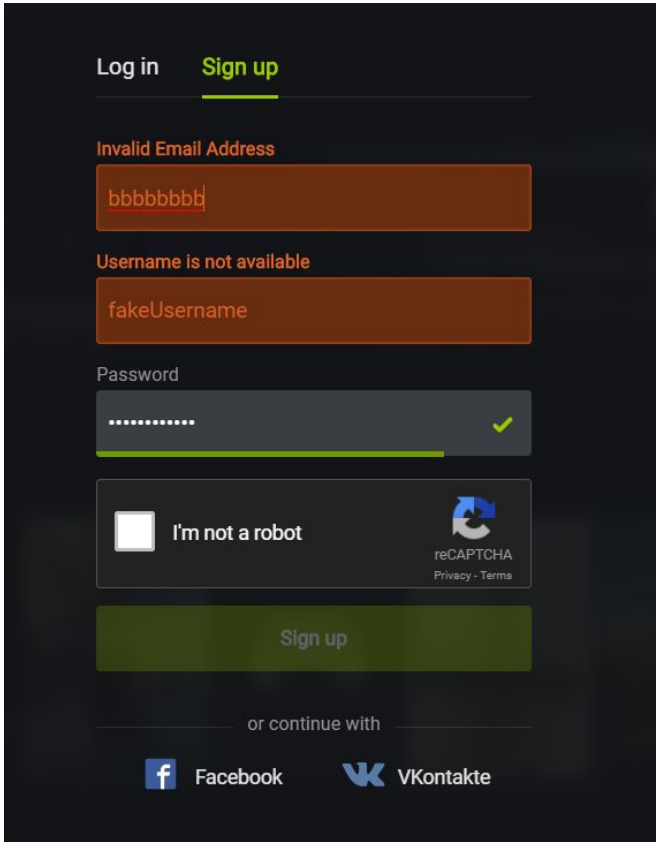


Example 2: hitbox.tv, game streaming website



Case Study: Modern Form Inputs

- Example: The Hitbox.tv signup dialog.
- Input fields that validate as you type
- Compare to older forms that only validate AFTER you submit information.



Animated Photo Carousels

- You've probably seen these things *everywhere*
- Typically written with jQuery or AngularJS

What are we noticing here?

- Repeated containers of dynamic content
- A UI that is responsive to user interactions
- Animated and content that can move around on the page

However….

Isn’t HTML supposed to be static? How does the data get sent to the webpage?

Where do we go from here?

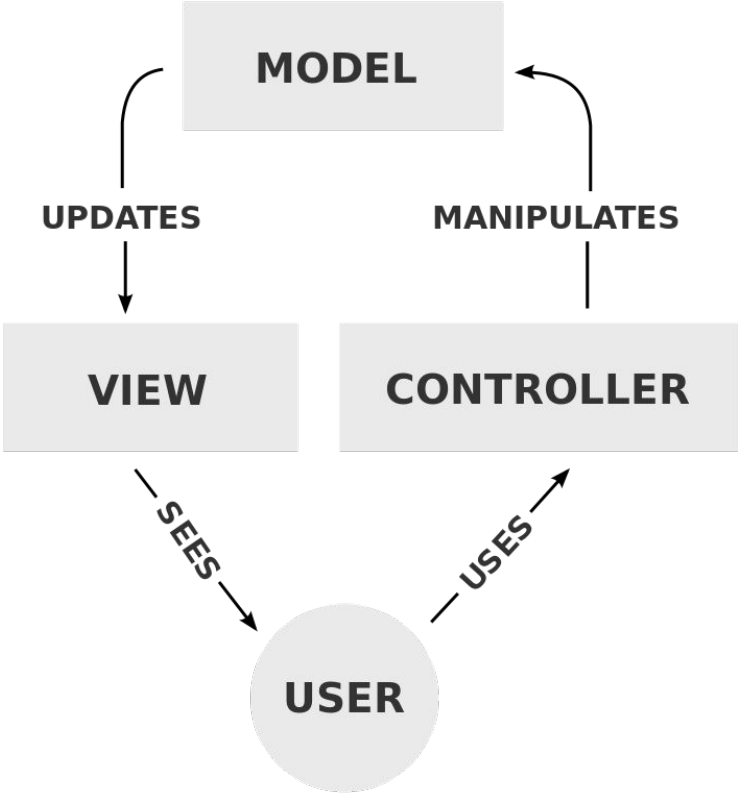
- Obviously, nobody is manually writing HTML pages with this dynamic info.
- Writing Javascript *within* the HTML is generally bad practice.

Problem: How do we automate content generation for our webpage and how do we decouple our Javascript from our HTML?

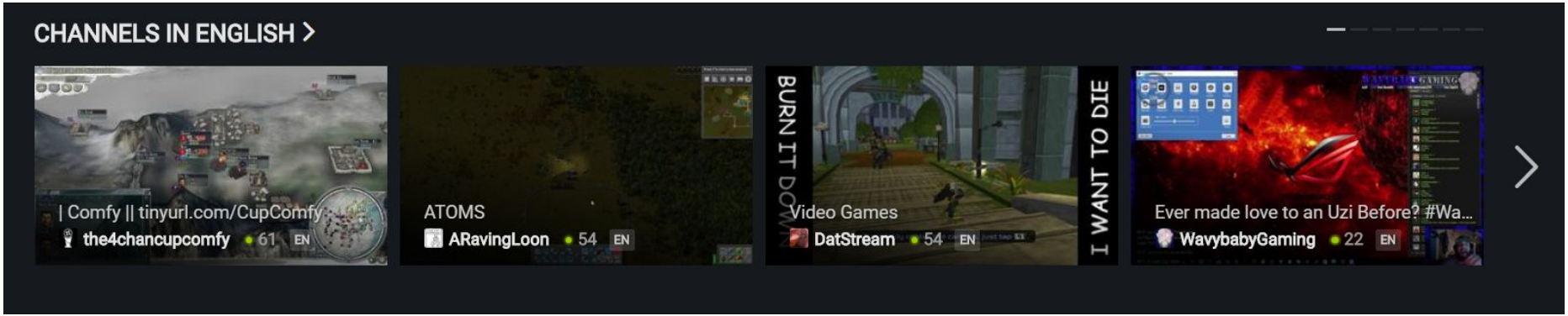
Solution: The Model-View-Controller programming pattern.

The Model-View-Controller

- A software-architecture design pattern used for user-interfaces.
- The **View** displays information stored **Models**, which are managed by back-end **Controllers**.
- These three elements are highly **cohesive** (work well together) while being minimally **coupled** (interdependent).

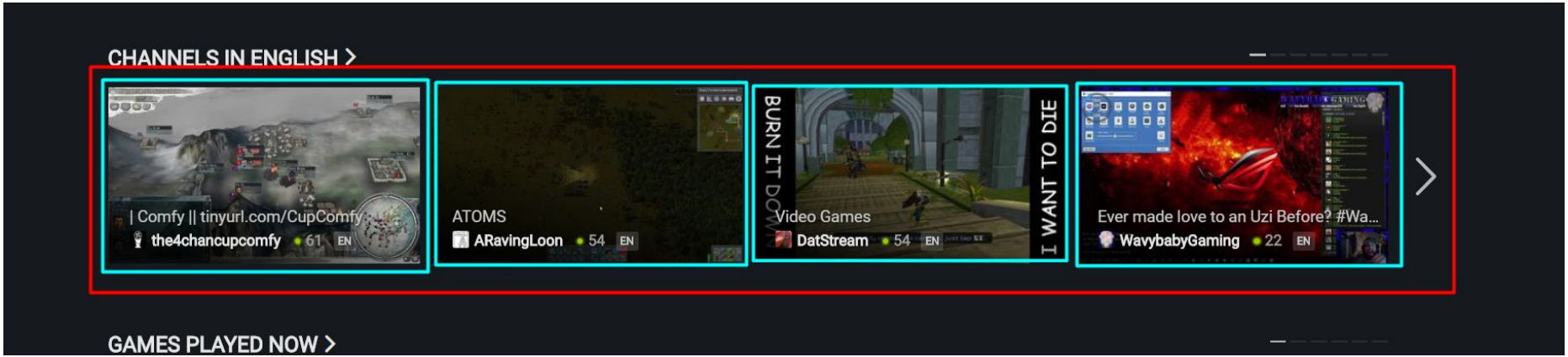


Case Study, Revisited: hitbox.tv



What are the Views here?

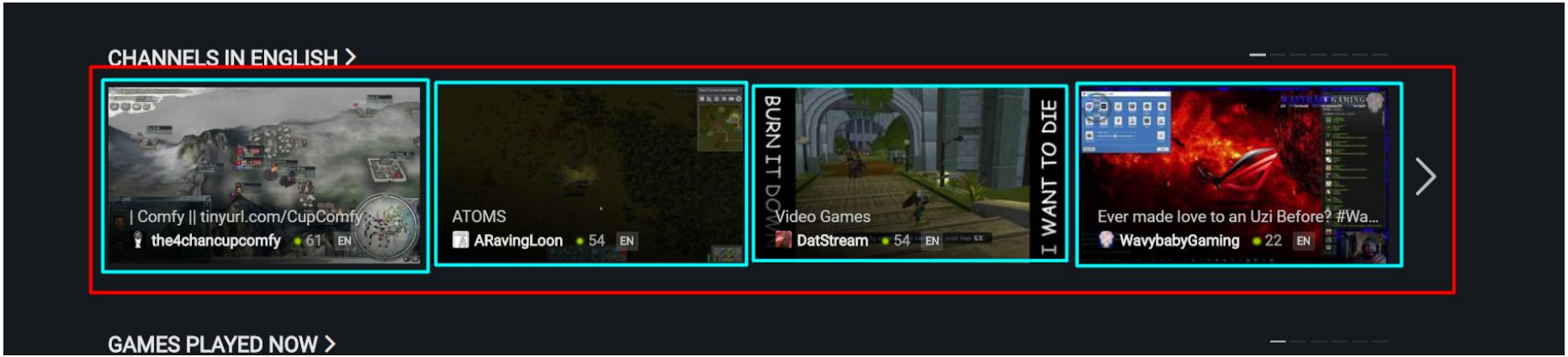
Case Study, Revisited: hitbox.tv



What are the Views here?

- Both red and blue boxes can be classified as Views. The whole page is a View. Anything a user sees and interacts with counts as a view.
- Red box contains a list of channel previews (blue boxes).
- Channel previews have a stream preview, stream name, number of viewers, and stream link.

Case Study, Revisited: hitbox.tv

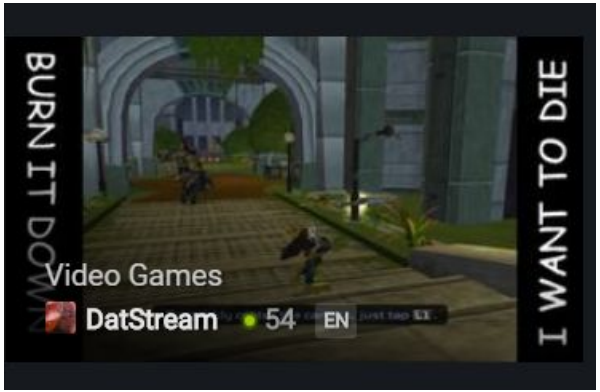


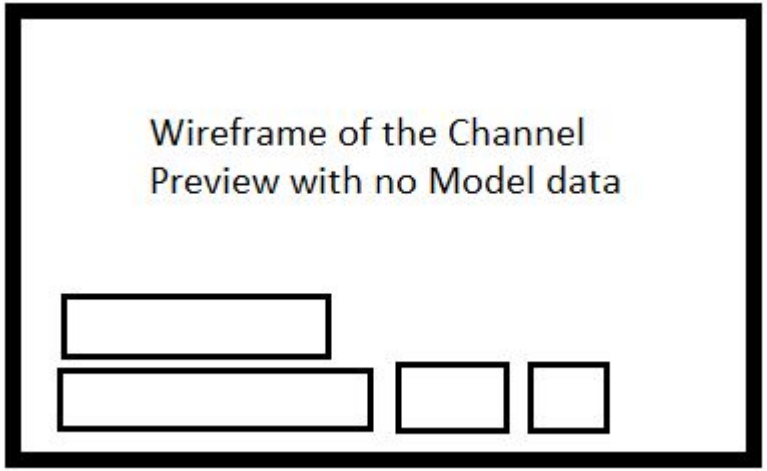
What are the Views here?

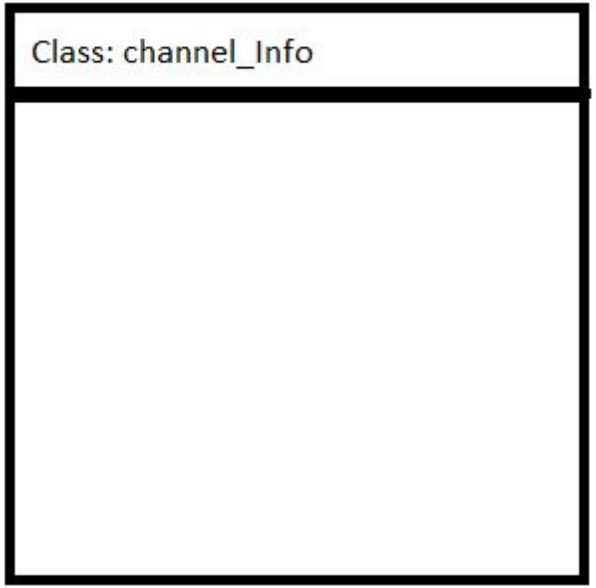
- Both red and blue boxes can be classified as Views. The whole page is a View. Anything a user sees and interacts with counts as a view.
- Red box contains a list of channel previews (blue boxes).
- Channel previews have a stream preview, stream name, number of viewers, and stream link.
- A View is typically a template that is populated with data from a **Model.** If the Model changes, our view changes.

Case Study, Revisted: hitbox.tv

- As an abstract example, how should we define a class or object to populate this Channel View?
- What did we mention a Channel Preview had earlier?

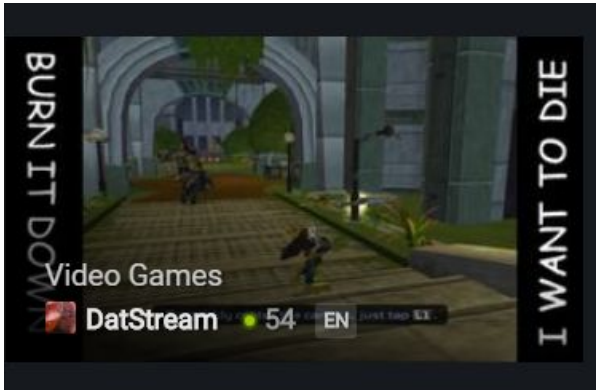


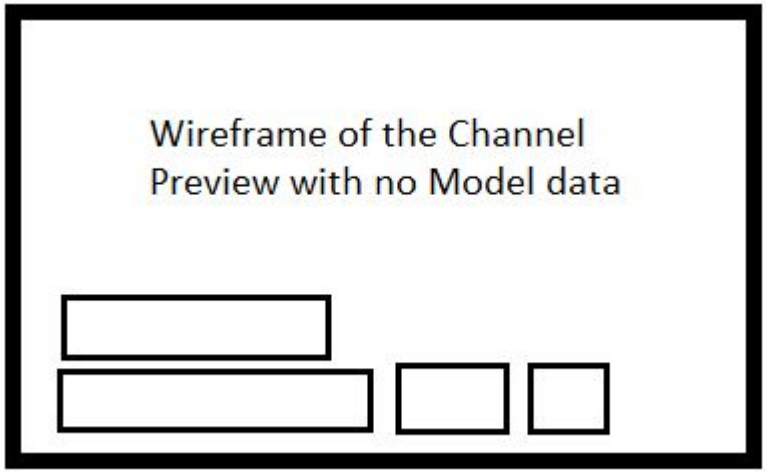




Case Study, Revisted: hitbox.tv

- As an abstract example, how should we define a class or object to populate this Channel View?
- What did we mention a Channel Preview had earlier?
- We can use Models to populate View templates and dynamically generate data on our pages.





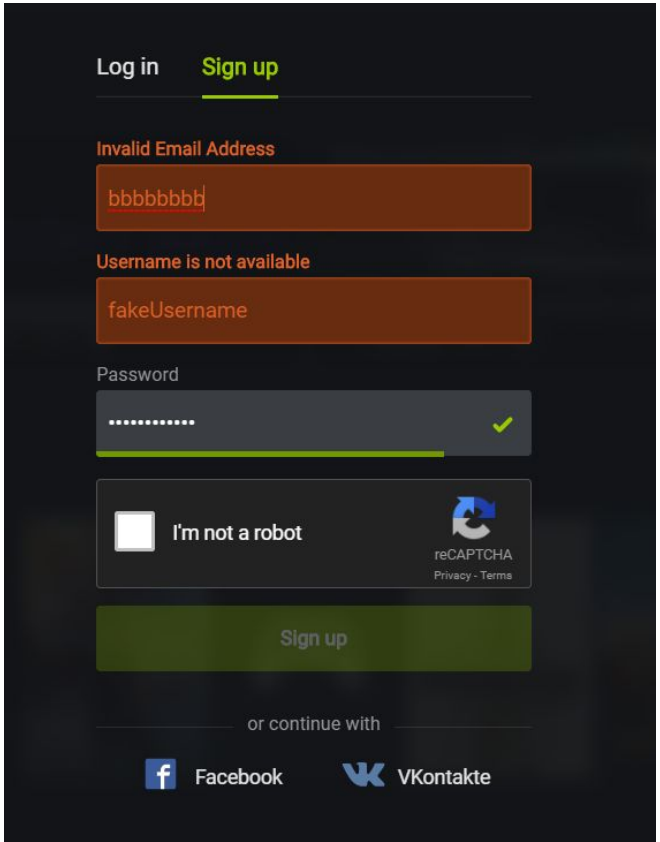
Model Populates View

Class: channel_Info

Public
- Image: Stream Preview
- String: Channel Name
- String: Owner Name
- Number: Viewer Count
- Enum: Language
- Bool: isOnline

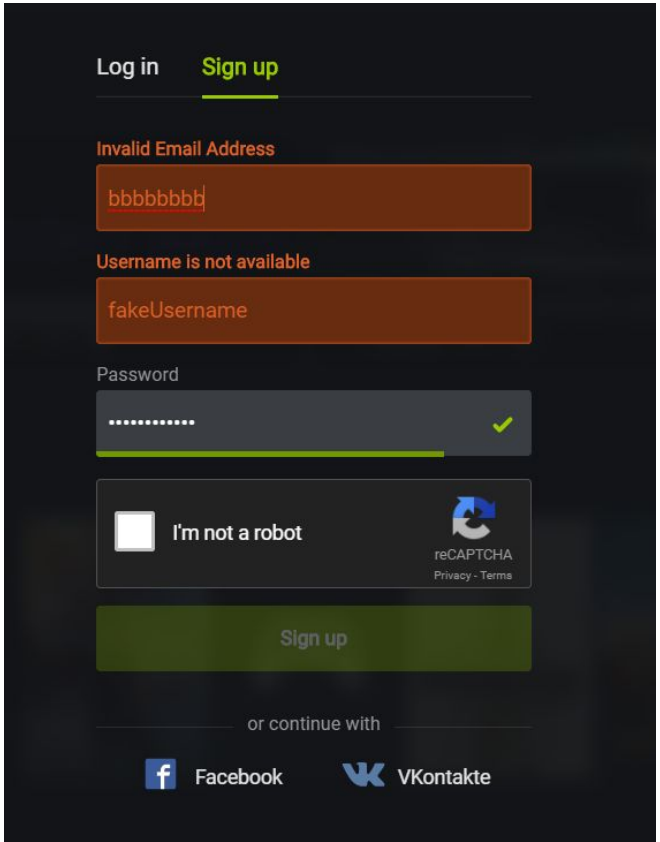
Case Study, Revisited: hitbox.tv (Form Inputs)

- Models and Views are used to display data, so how do we interact with our interface?
  - Answer: A Controller!

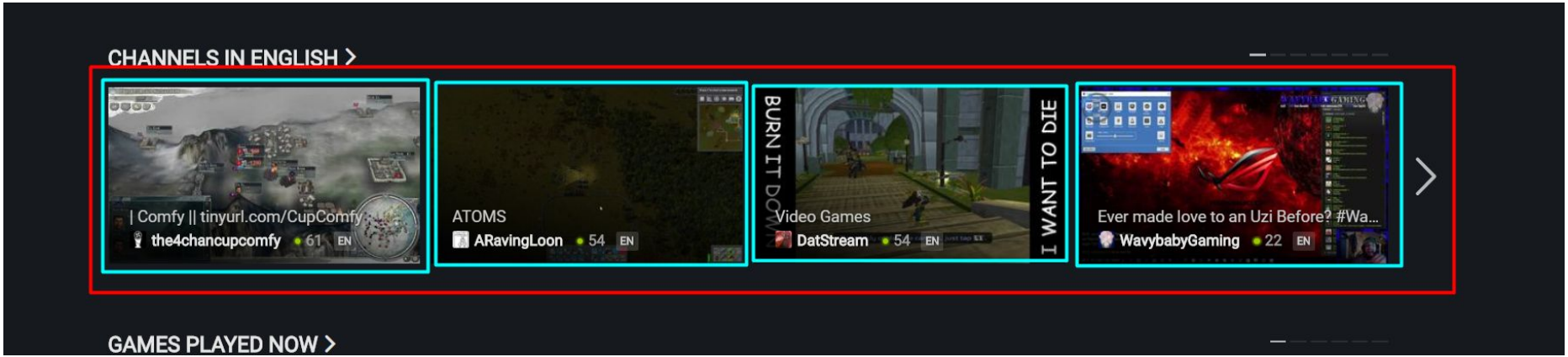


Case Study, Revisited: hitbox.tv (Form Inputs)

- Models and Views are used to display data, so how do we interact with our interface?
  - Answer: A Controller!
- Controllers will accept input and perform them as commands on the model or view.
- Animations, form validation, and view generation can be handled with controllers.



Putting It Back Together



- Hitbox's Channel Viewer is a **View** which has smaller Channel Preview **Views**, which are populated by (our hypothetical) channel_Info **Model.** The population of these Views with Model data is handled by a **Controller**.
- ……..but how do we program this?

The MVC and AngularJS

- What does AngularJS provide us with?
  - Two-Way Data Binding
    - Model Data is shared between View and Controller
  - Directives
    - In Angular’s implementation: they are functions that are executed when the Angular compiler sees them in the DOM.
    - Make DOM-manipulation and declarations of User Interaction functions easy to implement.
  - Designed for single-page applications
    - Define Angular App, and add Controllers.

AngularJS Two-Way Data Binding

- ng-model
  - Directive that binds the value of an input variable to both the view and controller
  - In other words… it creates a **Model**
- ng-repeat
  - Another directive that binds an array of values and allows repeated templating of them into the view
- ng-if/ng-show
  - Determine whether or not to render an HTML Element based on data bound values.
- ng-init
  - Initializes a data value
- ng-app/ng-controller
  - Bind external angularjs module and controller to your view

Directives

```
<body data-ng-app="myApp" data-ng-controller="myController">
  <h1>
  Instant Search Example
  </h1>
  Search: <input type="text" ng-model="searchTerm">
  <ul>
    <li ng-repeat="item in priceIndex | filter: searchTerm | orderBy: 'price'">
      {{item.name}} - ${{item.price}}
    </li>
  </ul>
</body>
```

- Directives are added directly to HTML tags, and the Angular compiler picks them up and executes them when the page loads.

JSFiddle Examples for Angular

- Form Validator
- Content Generation
- Image Carousel (not mine, uses bootstrap-ui + angular)
- Instant Search

Thank you!