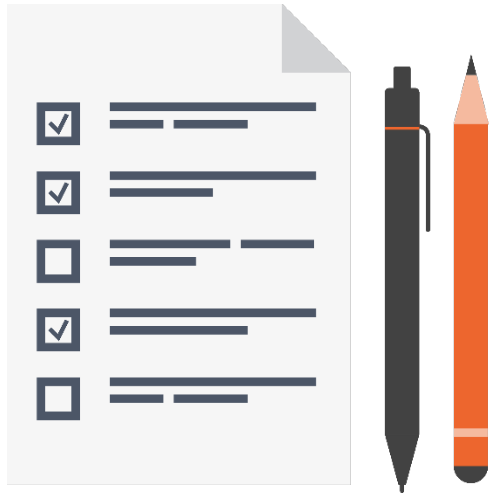
Stateful Reactive Concurrent SPAs with SignalR and Akka.NET

Introduction

Jason Roberts

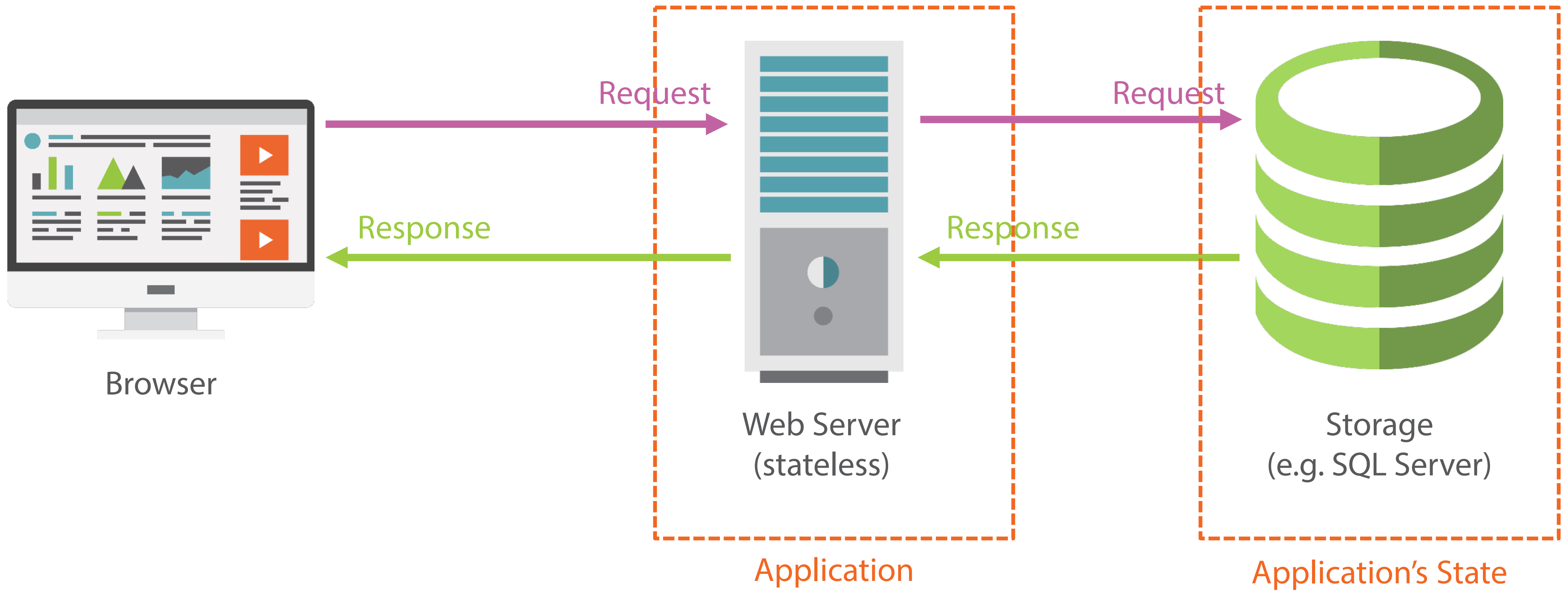
@robertsjason | dontcodetired.com

Overview



- The move to a stateful web
- Examples of state
- Why Stateful?
- Overview of reactive systems
- Architectural overview
- Getting started in Visual Studio

The Move to a Stateful Web



Stock levels, inventory, prices

Social media status updates

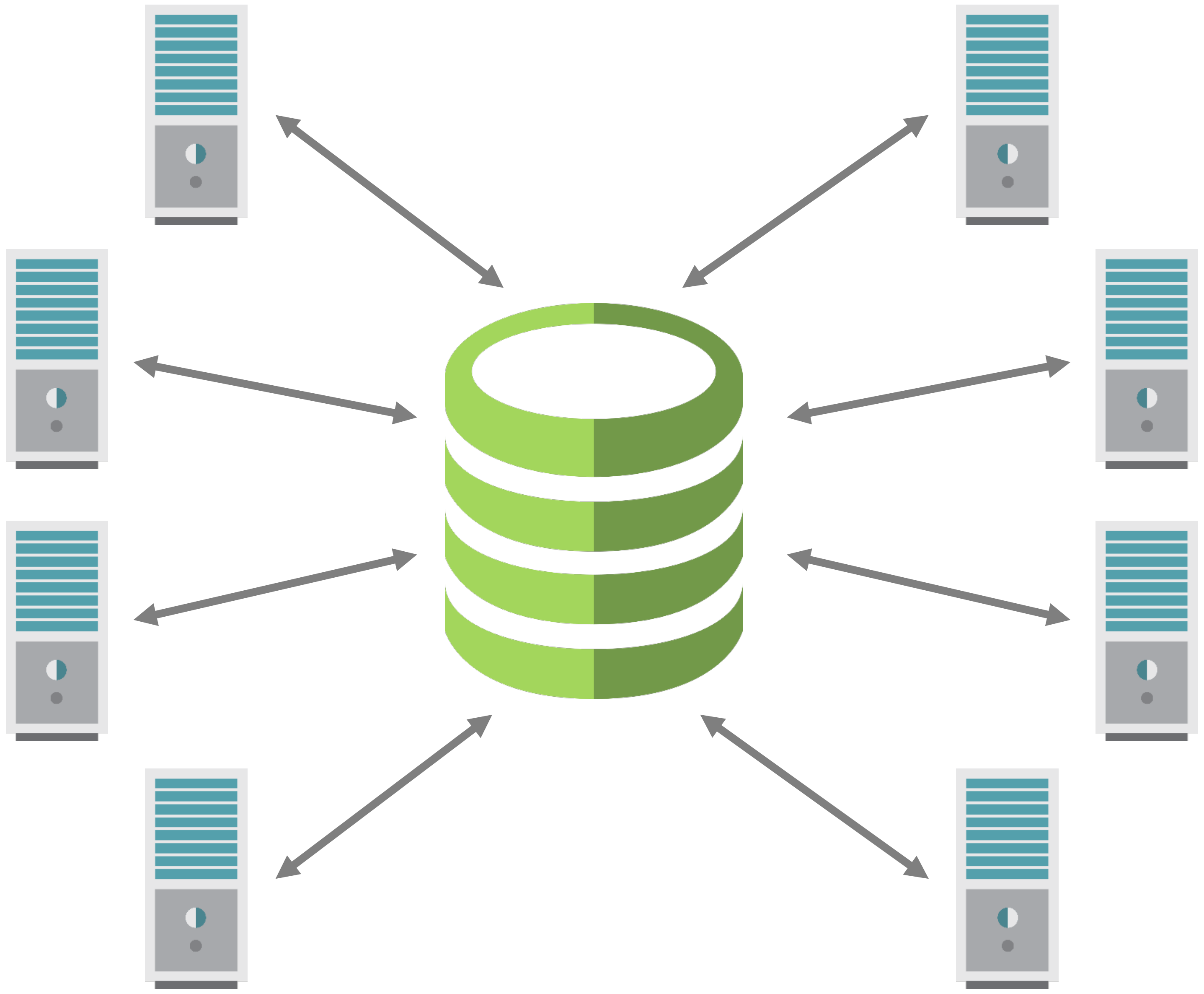
Marketing campaigns / rules

IOT device status / state

Multiplayer games / player state

Chat messages

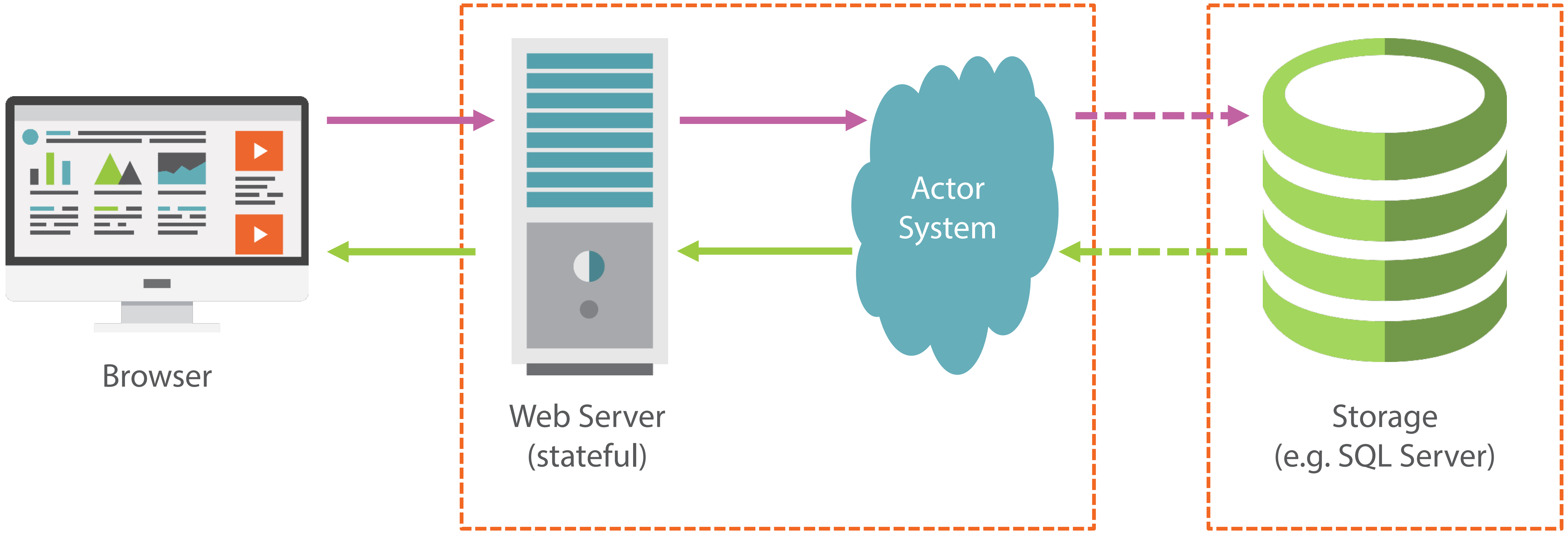
Current workflow state



Why Stateful?

- Highly responsive / reactive / real-time
- Increasingly larger workload / volume
- Concurrency concerns
- Fault tolerance
- Location transparency
- Common programming model

Why Stateful?

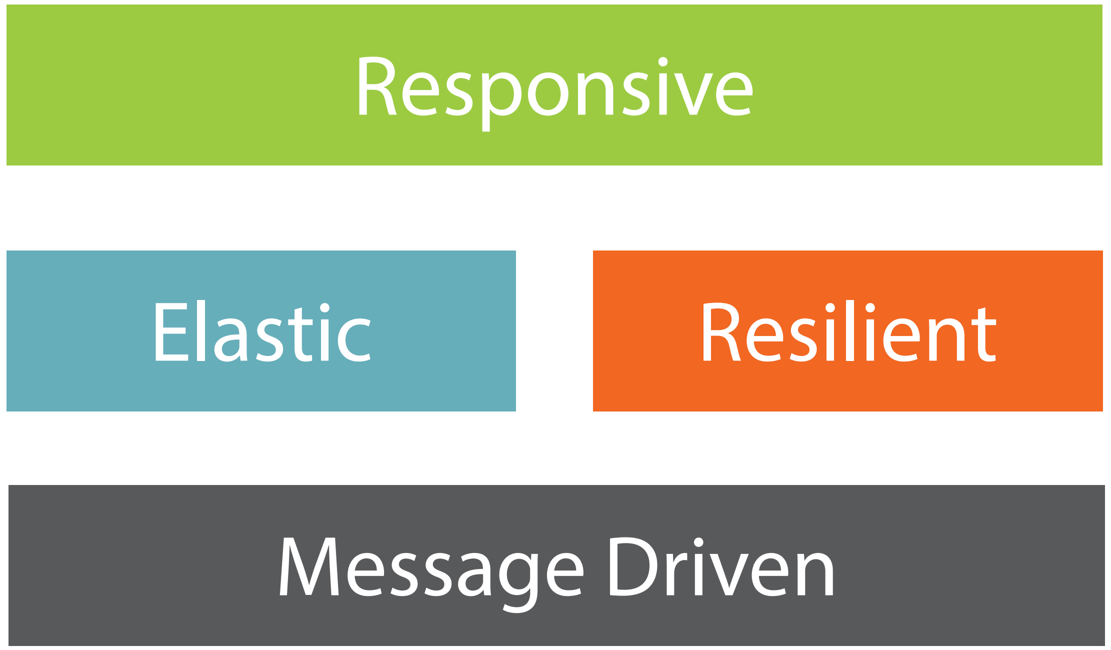


Application and State

State Backup /
“cold storage” /
reporting

Overview of Reactive Systems

“responds in a timely manner if at all possible”



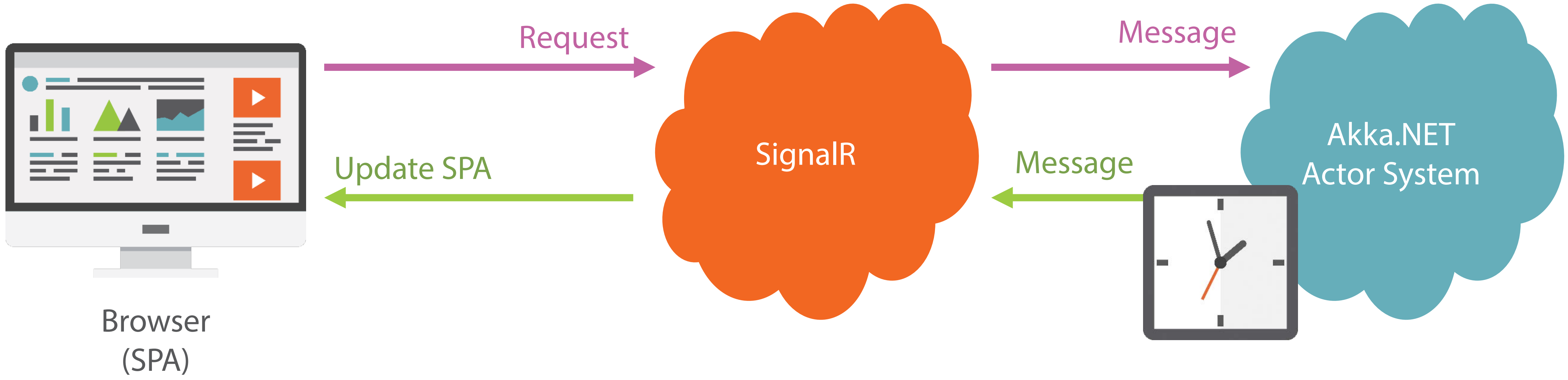
“stays responsive under varying workload”

“stays responsive in the face of failure”

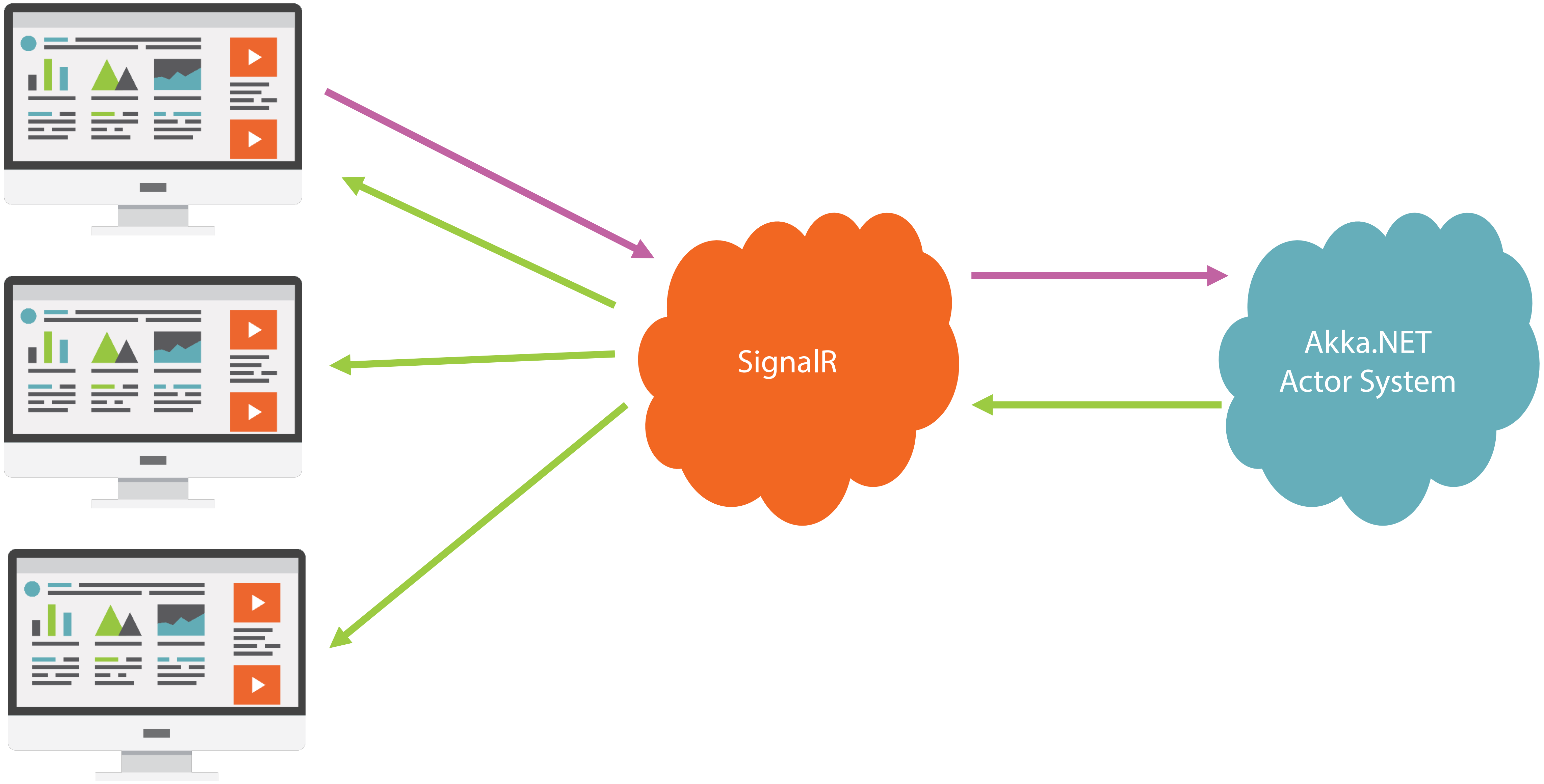
“asynchronous message-passing to establish a boundary between components that ensures loose coupling”

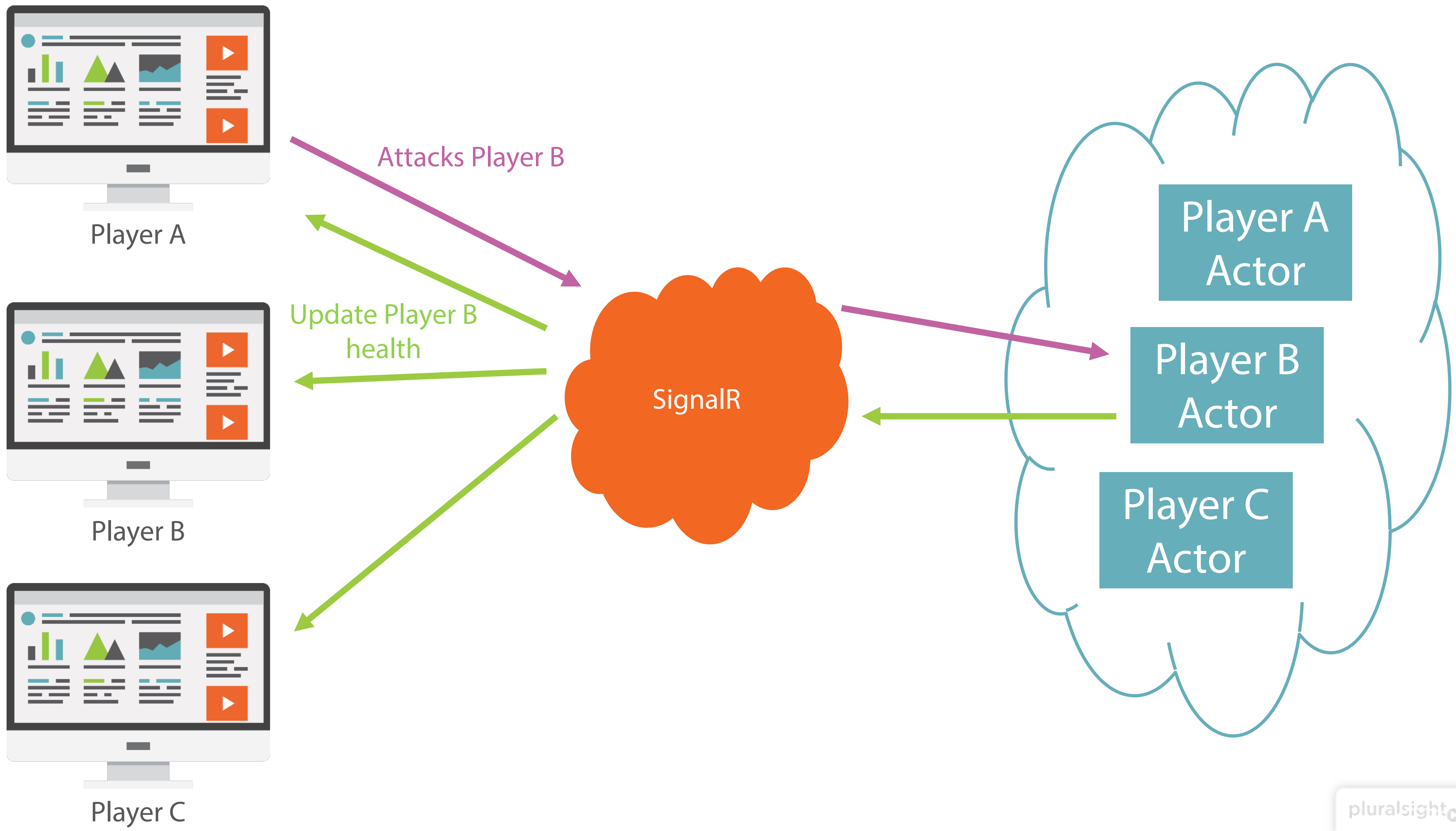
www.reactivemanifesto.org

Architectural Overview



SignalR is the glue that allows the client SPA to **react** to changes that happen in the actor model on the server.


SignalR
Akka.NET
Actor System


Player A
Player B
Player C
Attacks Player B
Update Player B health
SignalR
Player A Actor
Player B Actor
Player C Actor

Course Outline

Building the Akka.NET actors

Integrating Akka.NET with SignalR

Creating the SPA web user interface

Hosting game state in a Windows Service

Suggested Akka.NET Course Prerequisites

- Defining actors
- Defining messages
- Sending/receiving messages between actors
- Supervision hierarchies / child actors
- Akka.NET remote actors
- “Building Concurrent Applications with the Actor Model in Akka.NET” course

Getting Started in Visual Studio

Create a class library project to hold our actors/messages

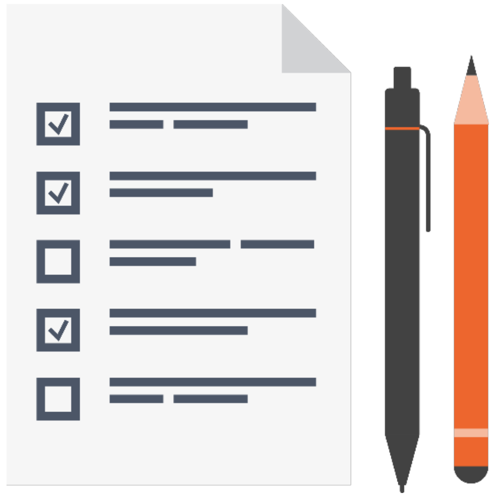
Create an ASP.NET MVC application to serve our HTML and host SignalR

Install Akka.NET NuGet packages

Creating the Starting HTML Skeleton

Create HTML outline to hold current player and list of other players who can be “attacked”

Summary



The move to a stateful web

State: stock levels, status updates, game state, etc.

Highly responsive / reactive / real-time

Increasingly larger workload / volume

Concurrency concerns

Overview of reactive systems

Architectural overview

Getting started in Visual Studio

Next:

Building the Player and Game Controller Actors