DESIGN FIRST. THEN CODE.

Create applications driven by your design

[designfirst.io]

HOW TO UNDERSTAND THE LOGIC OF A WEB APP

CLASSIC WAY

- read the documentation
- talk to the dev
- read the code
- launch some code analysis tools

BETTER WAY ?

- Learn from other languages
- Use the specificities of JavaScript:

« JavaScript is a **high level**, **dynamic**, **untyped**, and **interpreted** programming language »

-Wikipedia

DEMO

update the model of a web app and a server app

APPLICATIONS ARE SYSTEMS

CONCEPTS BEHIND SYSTEM

COMPOSITION

MODEL DRIVEN

METAMODEL

MDA

UML

STATE

BEHAVIOR

COMPONENT

MSON

ODM

DESIGN FIRST THEN CODE

DEFINITION

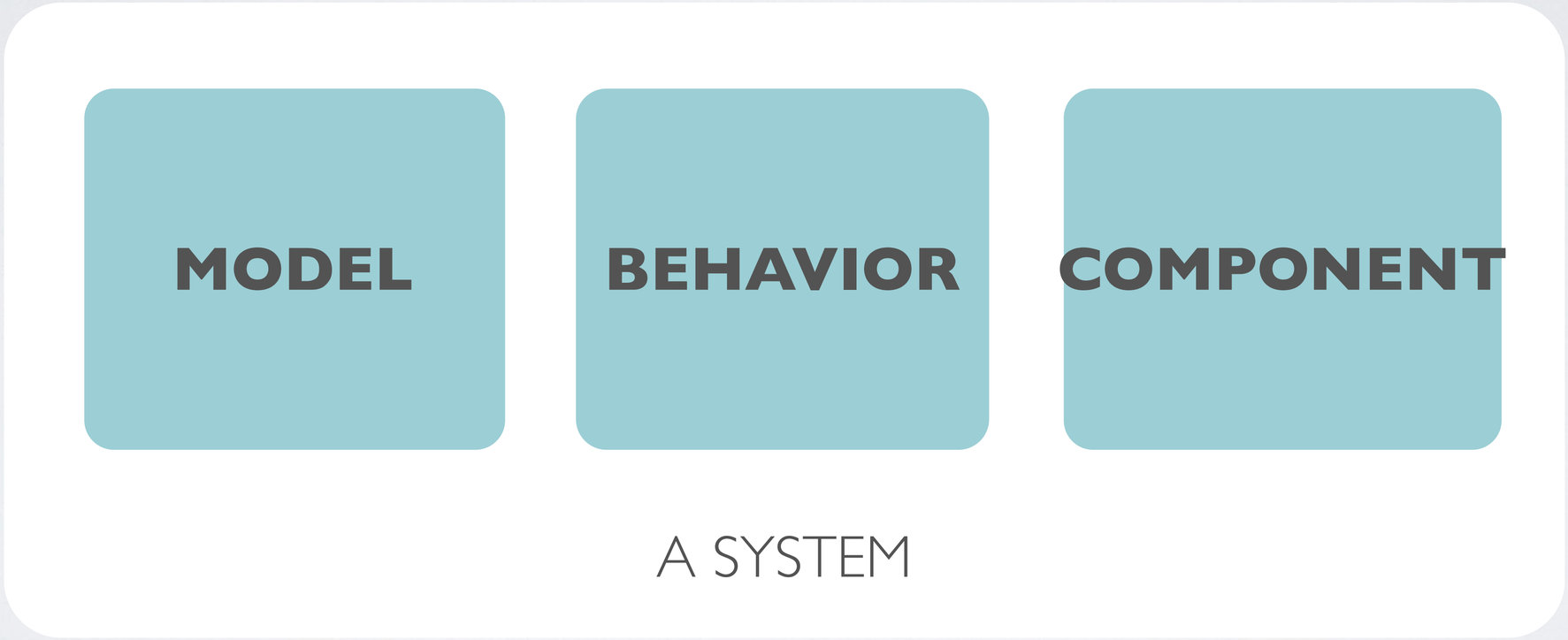
- « A system is a set of interacting or interdependent **components** forming an integrated whole. »
- « A system has **structure**, it contains parts (or components) that are directly or indirectly related to each other. »
- « A system has **behavior**, it exhibits processes that fulfill its function or purpose. »
- « A system has **interconnectivity**: the parts and processes are connected by structural and/or behavioral **relationships**. »
- « A system's structure and behavior may be decomposed via **subsystems** and sub-processes to elementary parts and process steps. »

-Wikipedia

APPLICATION AS A SYSTEM

- a system is a set of **models**, **behaviors** and **components**
- a component is an **immutable statefull object**
- a structure is defined by a **schema**
- a behavior is what a component does when we make an **action** on it
- interconnectivity is the **relationships** between components defined by the model
- every systems can be decomposed on **subsystems**

APPLICATION AS A SYSTEM



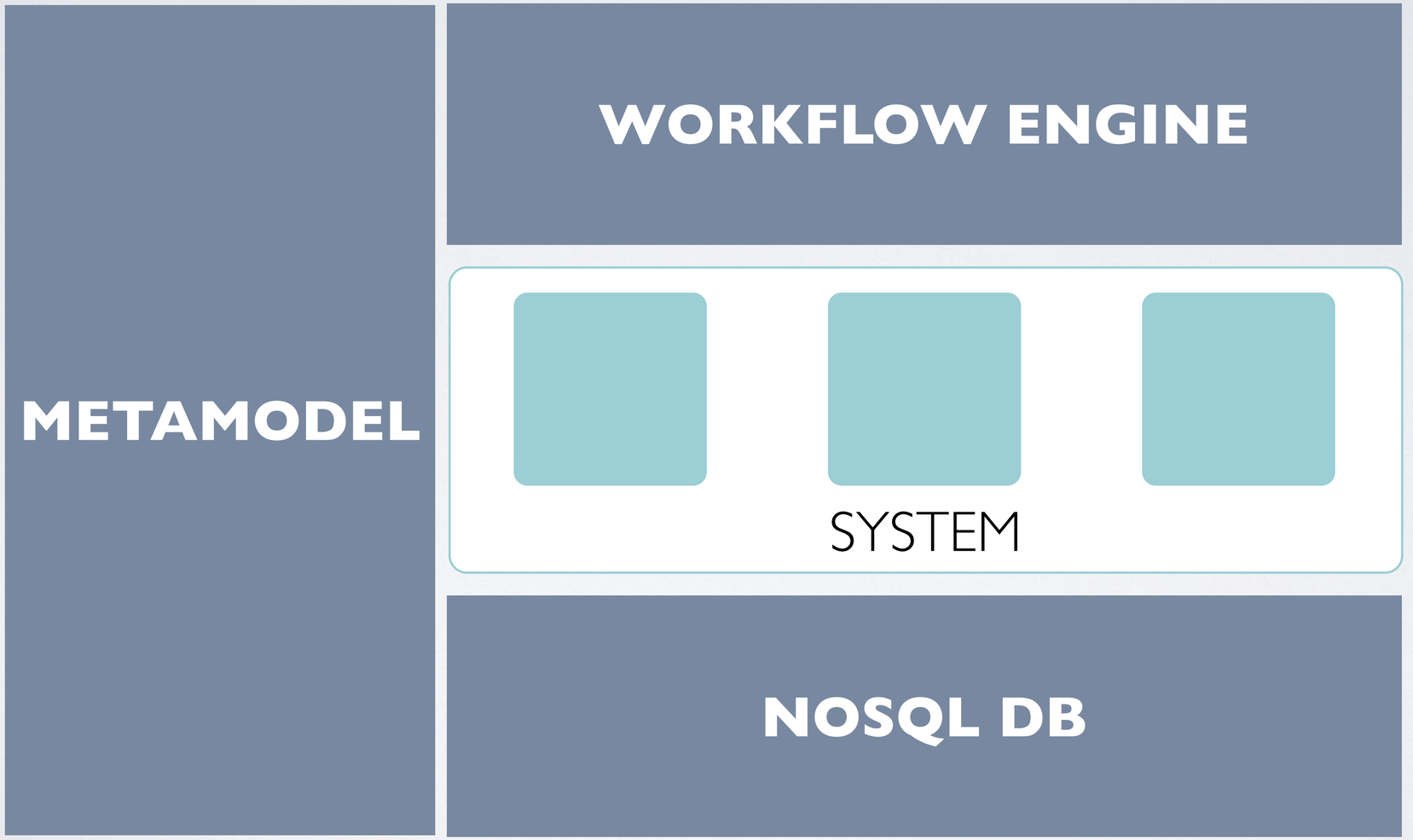
- Define the **structure of the components** and the **relationships** betweens them

- Define the behavior of the components, i.e what components do when **actions** are made (change of **state**, events, methods)

- **Statefull** objects that compose your application

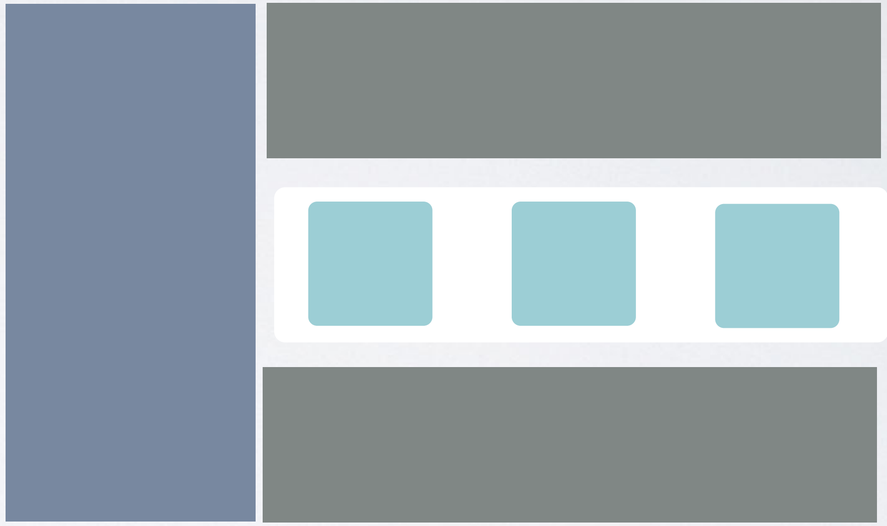
HOW TO RUN YOUR SYSTEM ?

WITH A SYSTEM RUNTIME



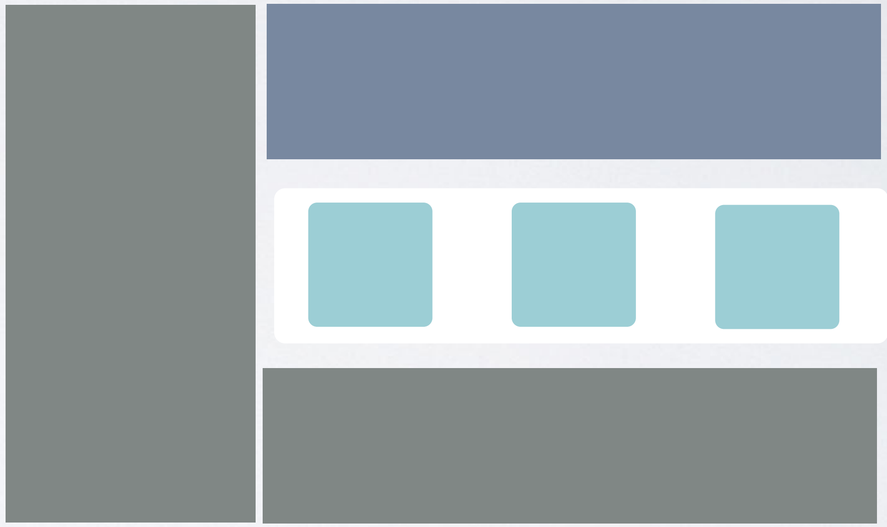
METAMODEL

- Abstraction of the model
- Manage the model
- Generate the classes of the component based on the model definition



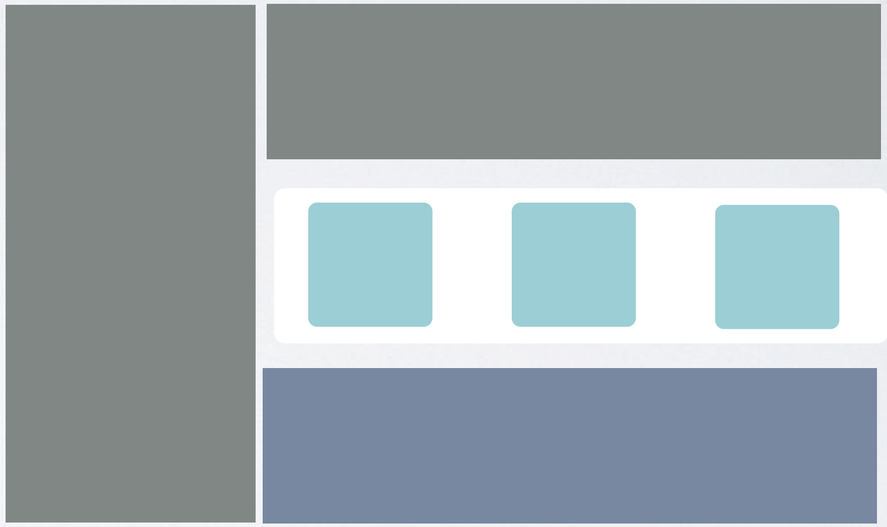
WORKFLOW ENGINE

- Automate components actions and insure that they are always compliant with the model

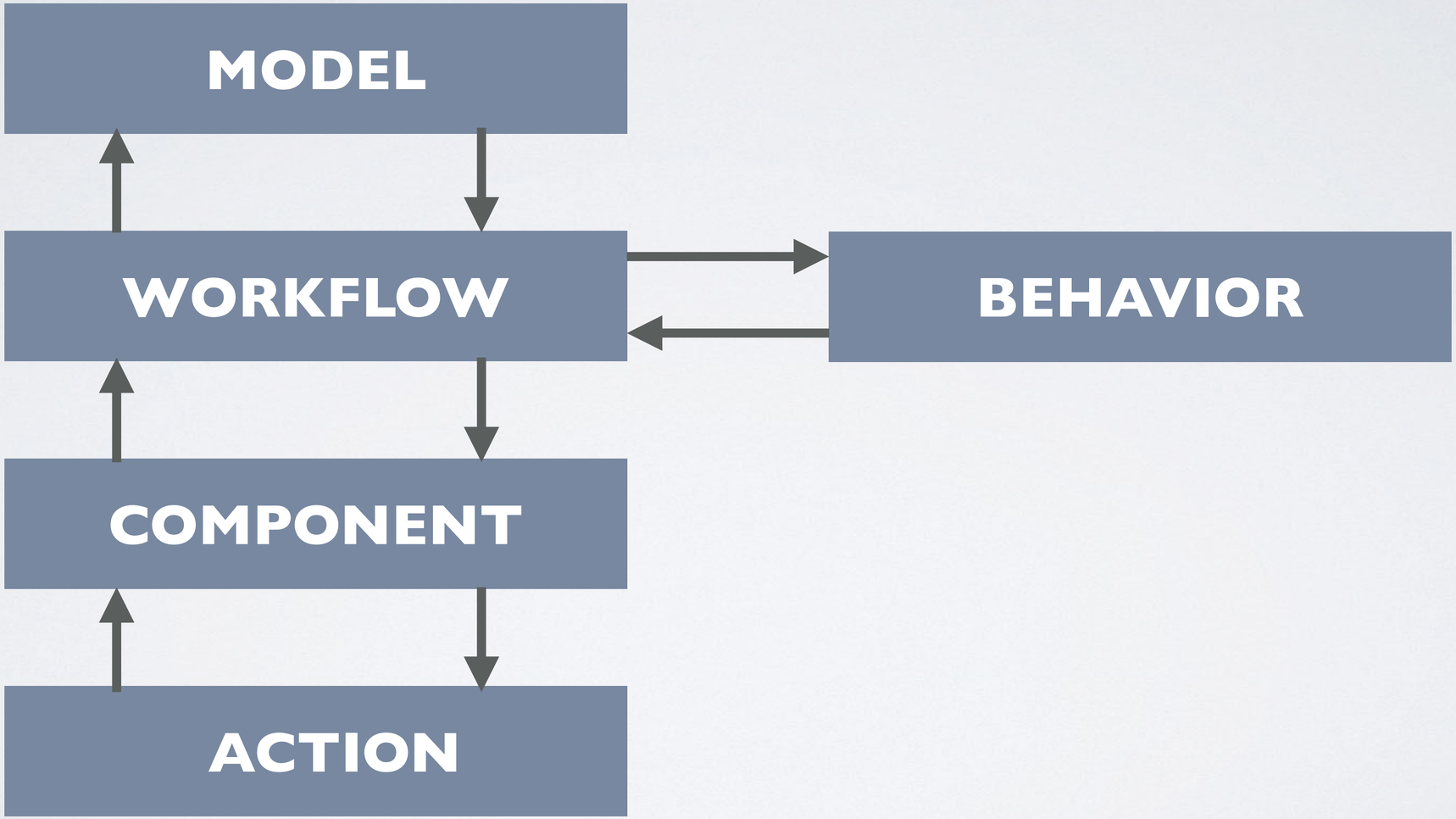


NOSQL DB

- Store your model, behaviors, components and states



ARCHITECTURE

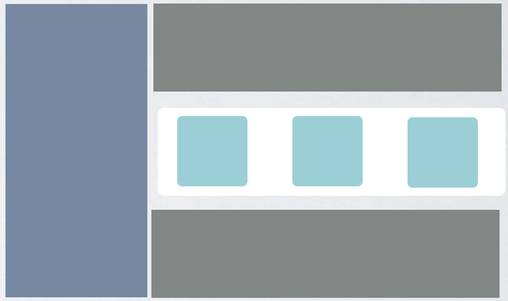


SYSTEM RUNTIME

A JavaScript runtime for systems

SYSTEM RUNTIME METAMODEL

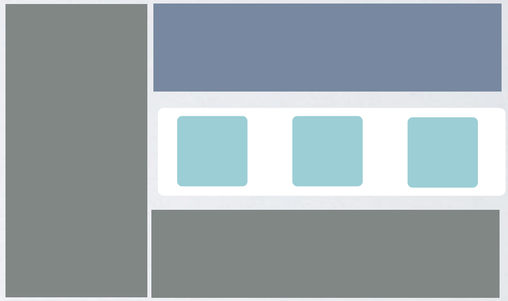
- The definition of the model is made on a JSON format called **MSON** (*Metamodel JavaScript Object Notation*), no code is needed
- With MSON you can define types, classes, one to one / one to many relationships and multi inheritance between classes
- MSON is based on UML
- System Runtime uses the **Model-Driven Architecture** approach to create classes based on your design. Use them to instantiate your components



SYSTEM RUNTIME WORKFLOW

- System Runtime checks **on runtime** that your components are compliant with your model
- System Runtime **can stop your system if a problem was found**

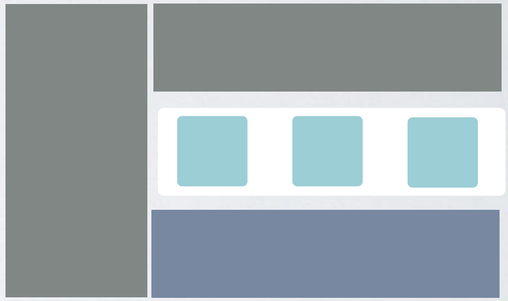
When you call a method, an event or update a property, System Runtime workflow will check if all is ok to valid the change of state of the component. It means:



- for a *property*: the type of the value is valid with the model
- for a *method*: the number of parameters, their types and the result are valid with the model
- for an *event*: the number of parameters and their numbers are valid with the model

SYSTEM RUNTIME NOSQL DB

- System Runtime acts as an **ODM** (*Object-Document Mapper*) to manage your components as NoSQL Documents
- Update a document will update directly the related component
- System Runtime NoSQL DB stores your components and you can export/import them into another System Runtime NoSQL Database



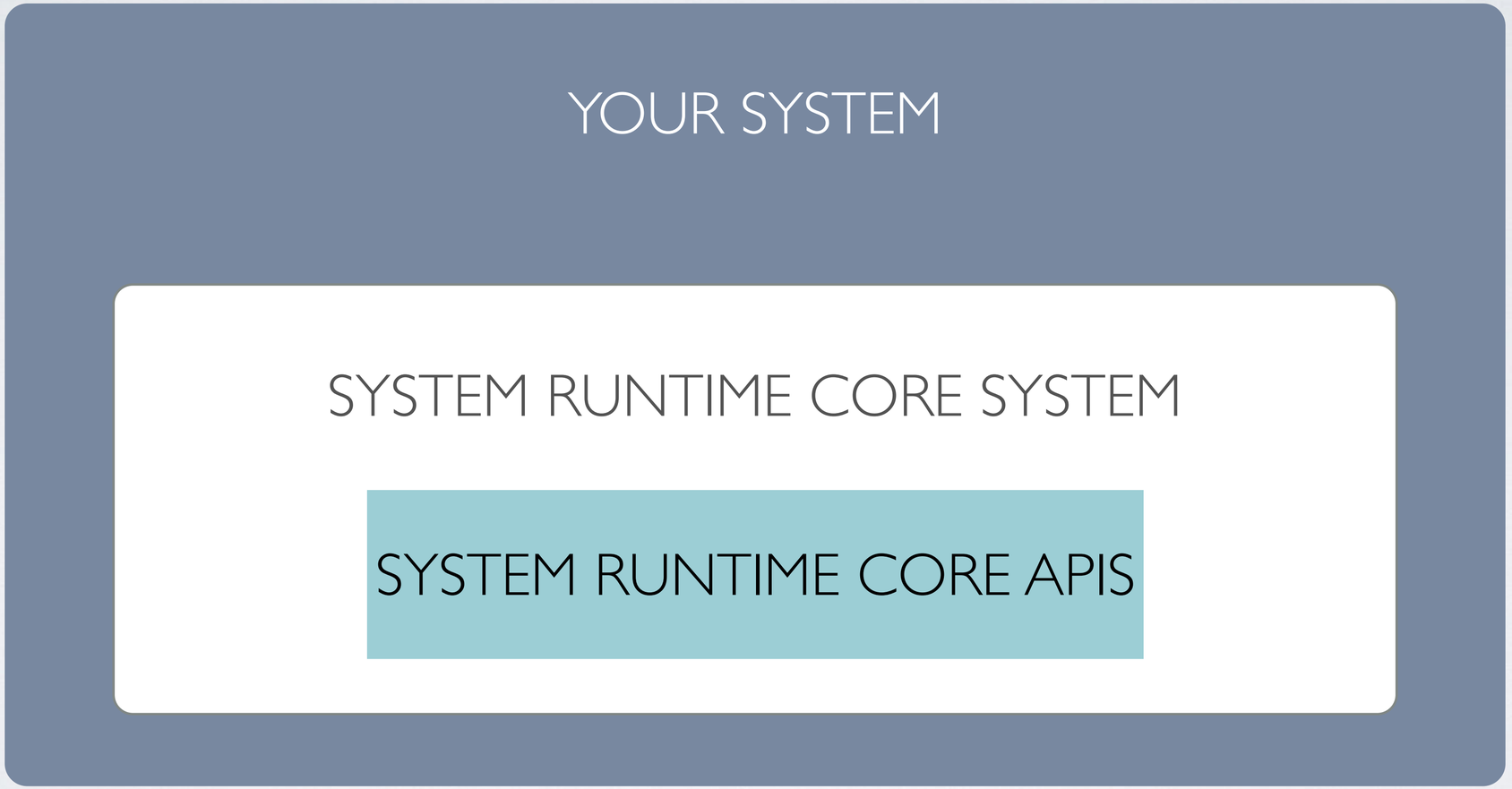
EXAMPLES

OTHER FEATURES

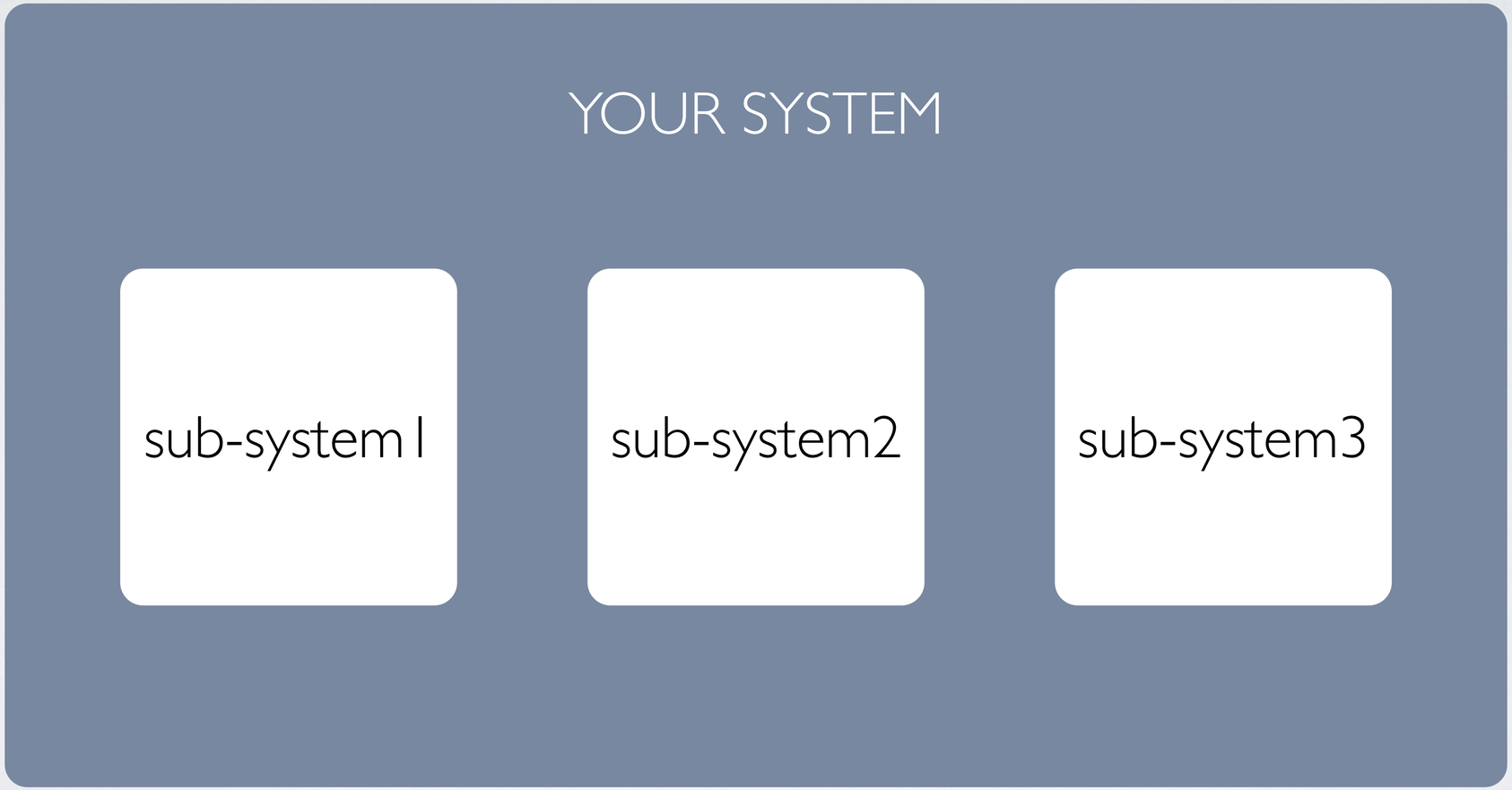
- **Everything is a component**
- **Everything is JSON**: all your code can be serialized in JSON
- **Behaviors are dynamically evaluated**: they can be replaced at runtime and they have a real scope
- **Universal**: you can create client and server systems
- **Unit tests are native**
- **Package management is native**
- **Few apis and light** (21 ko gzip)
- **VanillaJS** (ES5)

SYSTEM COMPOSITION

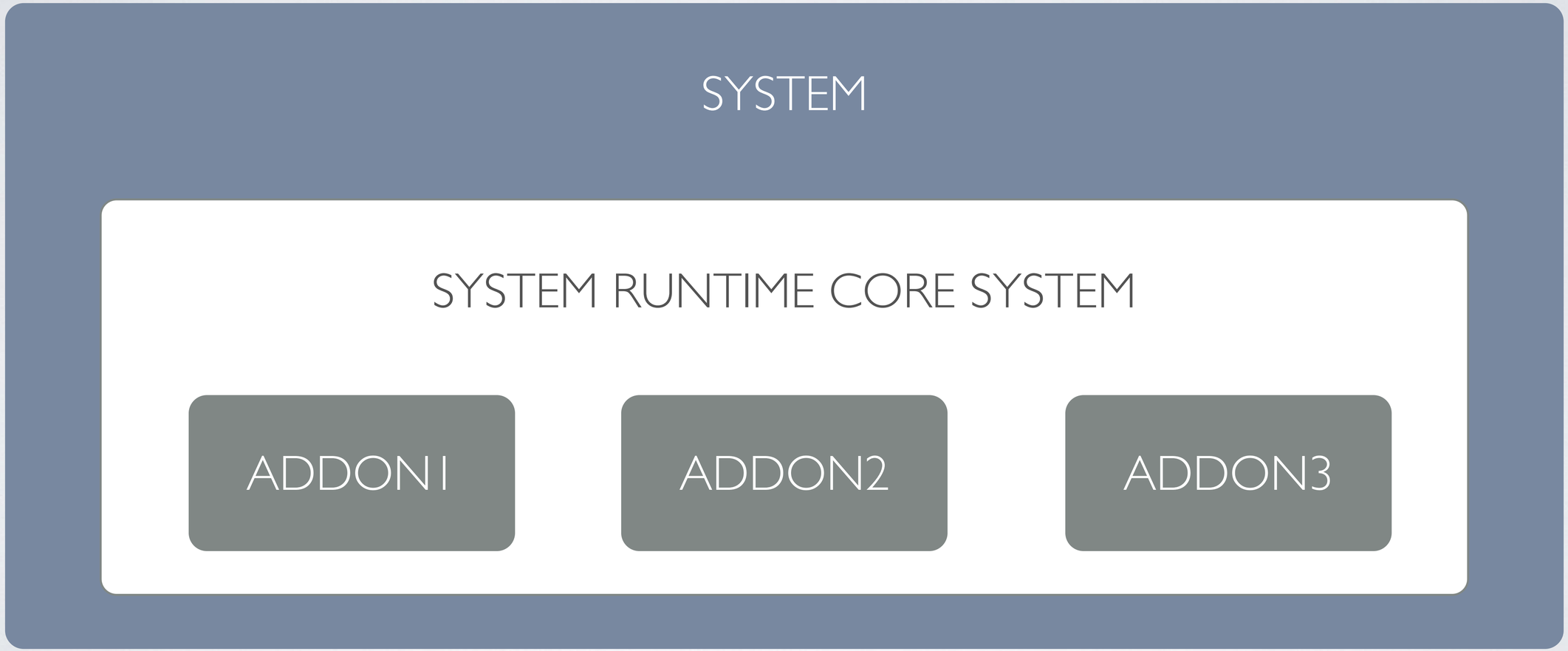
SYSTEM RUNTIME CORE SYSTEM



SUB-SYSTEM

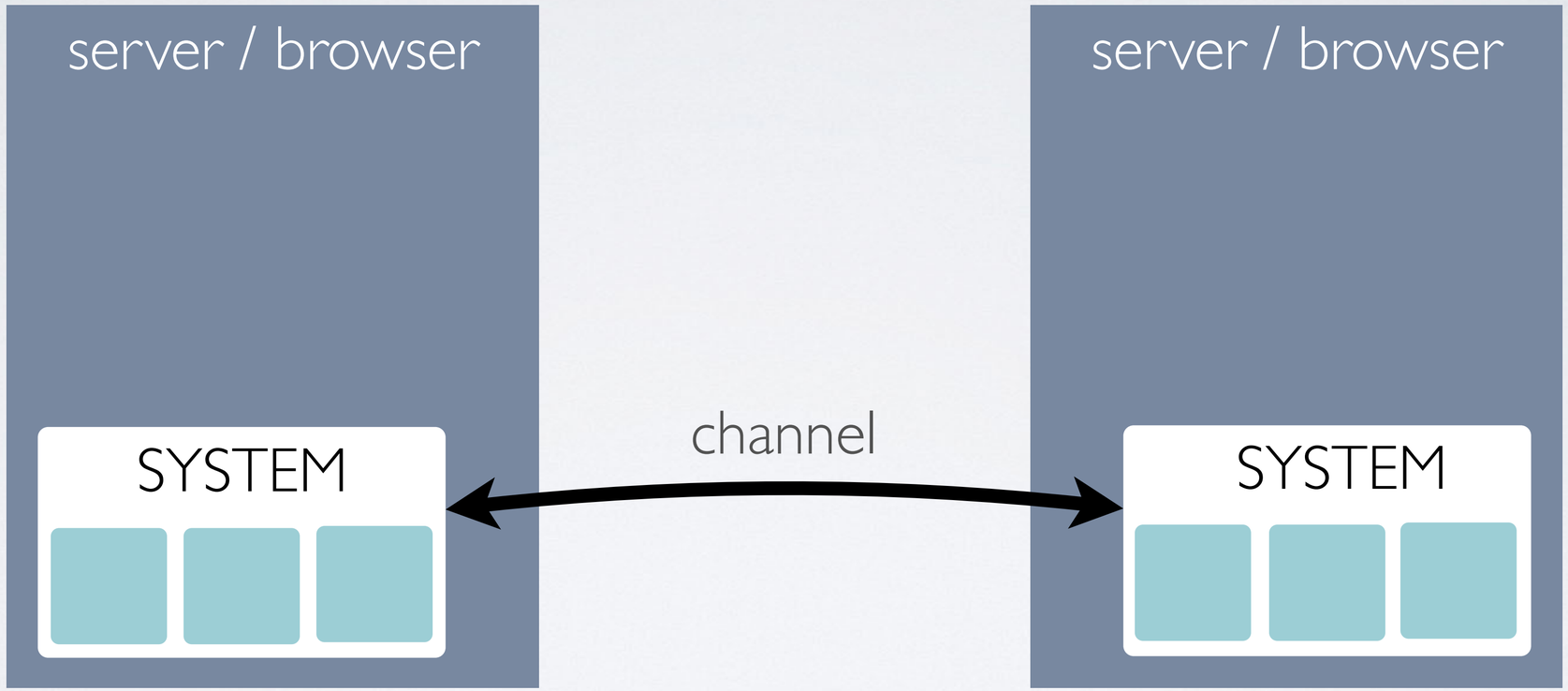


ADDONS



COMMUNICATION BETWEEN SYSTEMS

CHANNEL



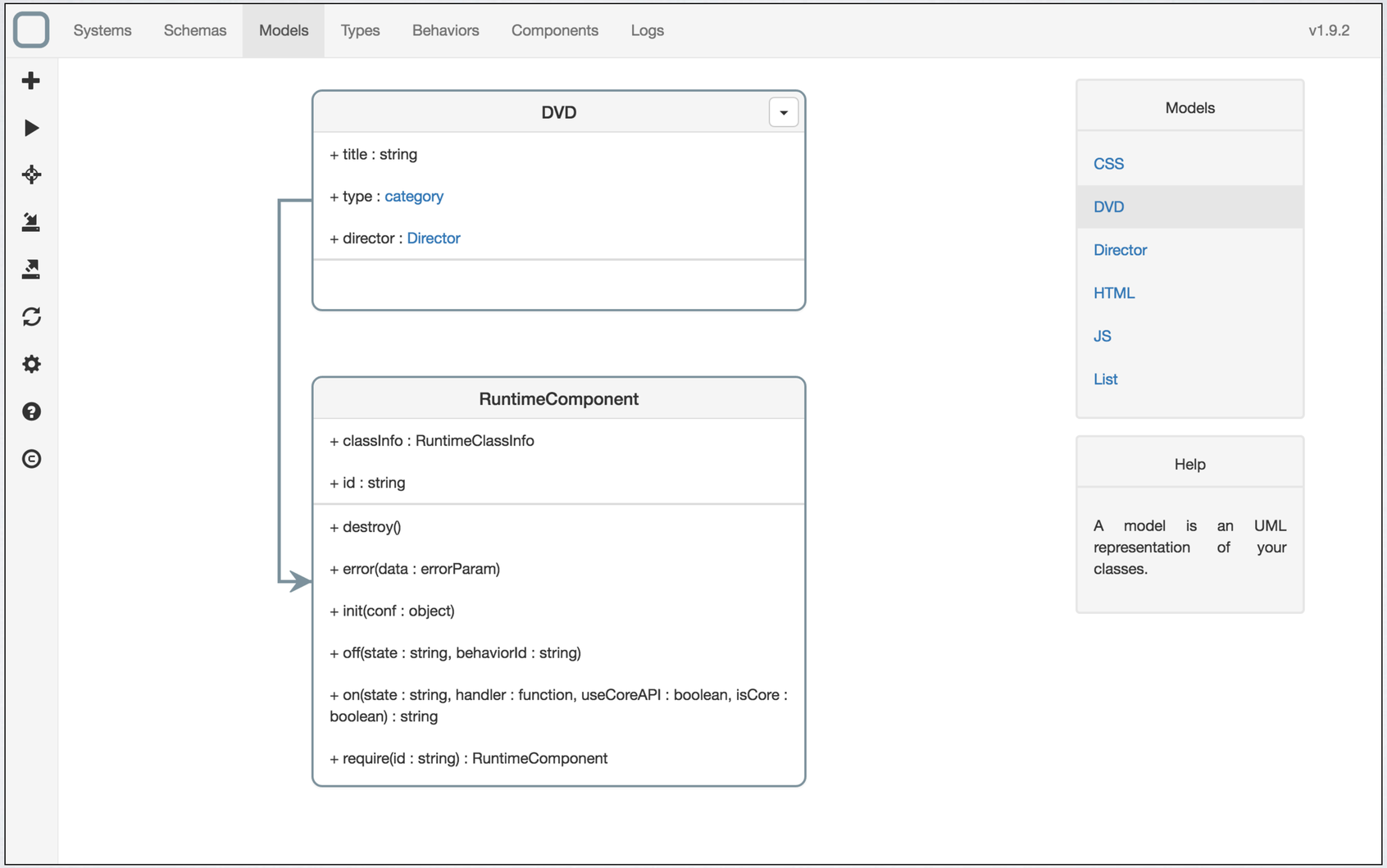
SYSTEM RUNTIME BUILD SYSTEM

- **continuus integration:** travis
- code analysis: jshint + code coverage
- test: karma + jasmine (server / client)
- doc generation: yuidoc

SYSTEM DESIGNER

An IDE for designing applications driven by the model

SYSTEM DESIGNER



FEATURES

- Create system
- Snapshot of system: **remote designing** of running systems
- **SDK**: the designer is made with System Runtime, so its system is configurable
- **Full front app:** no server need

EXAMPLES

Design System Designer with System Designer

QUESTIONS ?

[designfirst.io]