


Реквием по фронту

библиотека Korolev

Андрей Михеев

Jjoy @ Рязань 2018

Пример клиентского приложения

LAST_NAME
FIRST_NAME


ACCT_NBR
ADDRESS_1

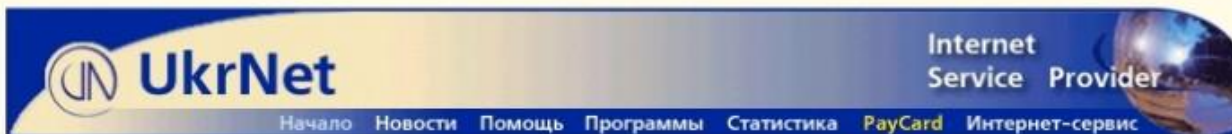
CITY
STATE
ZIP

TELEPHONE
DATE_OPEN

SS_NUMBER
BIRTH_DATE
RISK_LEVEL
OCCUPATION

OBJECTIVES
INTERESTS

ACCT_NBR	SYMBOL	SHARES	PUR_PRICE	PUR_DATE
1023495	VG	2000	20	02.04.93
1023495	HGT	1000	30	20.01.93
1023495	EMC	500	30	01.02.90
1023495	CIN	1500	38	09.09.93



Почта @ukr.net

Логин @ukr.net

Пароль Войти

ИНТЕРНЕТ-СЕРВИС

услуги и цены

Предоставление доступа к ресурсам сети Интернет по коммутируемым каналам.
Подключение в Киеве, Одессе, Днепропетровске. Скоро - Харьков.

Цены указаны С УЧЕТОМ НДС.

Регистрация и web-страница пользователя (до 2 Mb) - бесплатно.

ПАКЕТНЫЕ РЕЖИМЫ

Наименование пакета	Стоимость
New!!! Тестовая карточка 1 час доступа в течение двух суток	4.00 грн
New!!! "1 е-Q в сутки", без ограничения. Подробнее...	1.00 е-Q
Ночной, с 01-00 до 09-00	6.00 е-Q
Ночной+, с 23-00 до 09-00	11.90 е-Q
Домашний, с 19-30 до 09-00, выходные и праздники - круглосуточно	19.00 е-Q
Домашний+, с 15-00 до 09-00, выходные и праздники - круглосуточно	23.90 е-Q
Unlimited	27.00 е-Q

[Анкета Розыгрыш призов](#)

[Новости](#)

[О компании](#)

[История](#)

[Контакты](#)

[Вакансии](#)

Услуги и цены

[Выделенные линии](#)

[Dial-up](#)

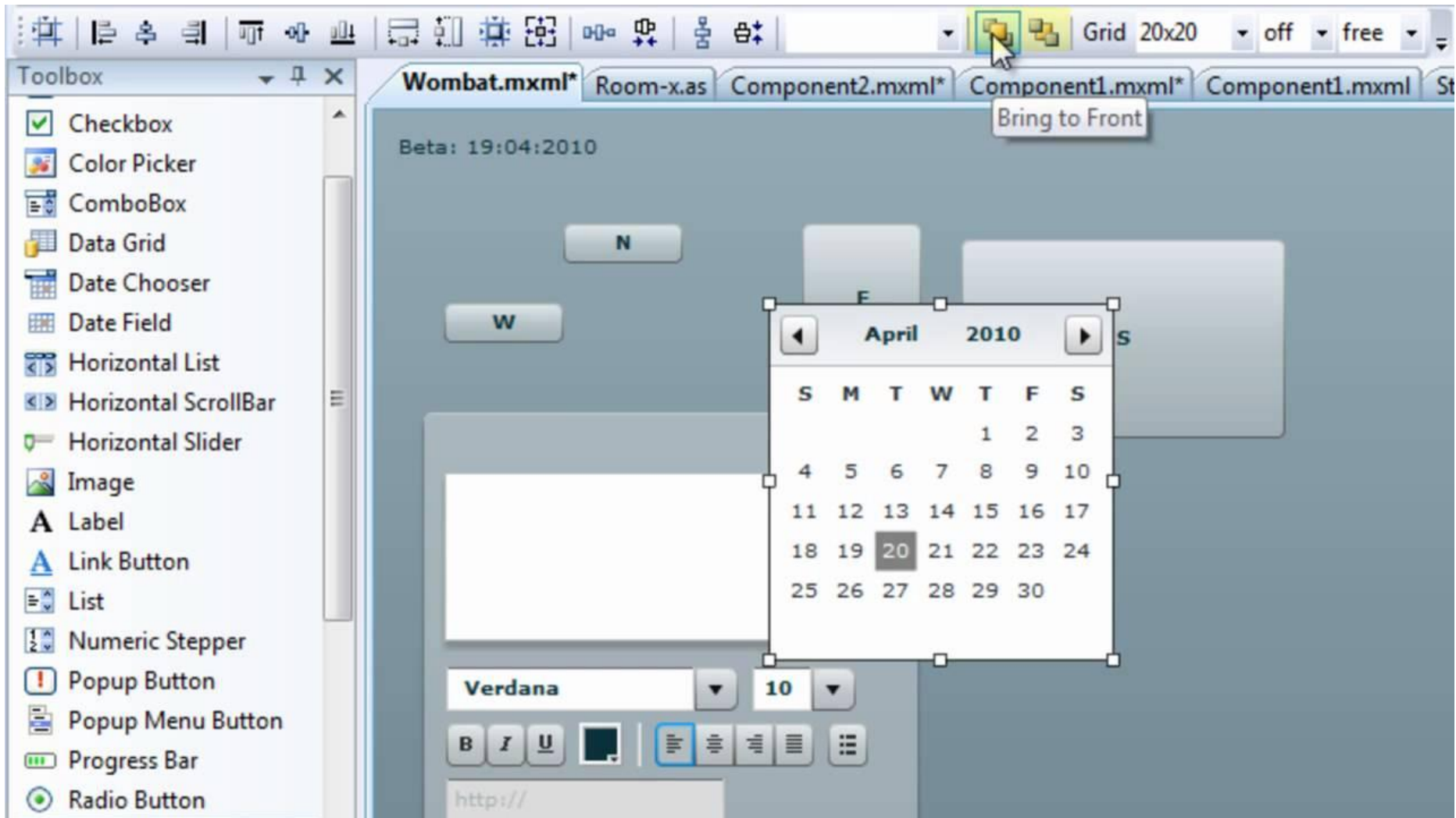
[UkrNet PayCard](#)

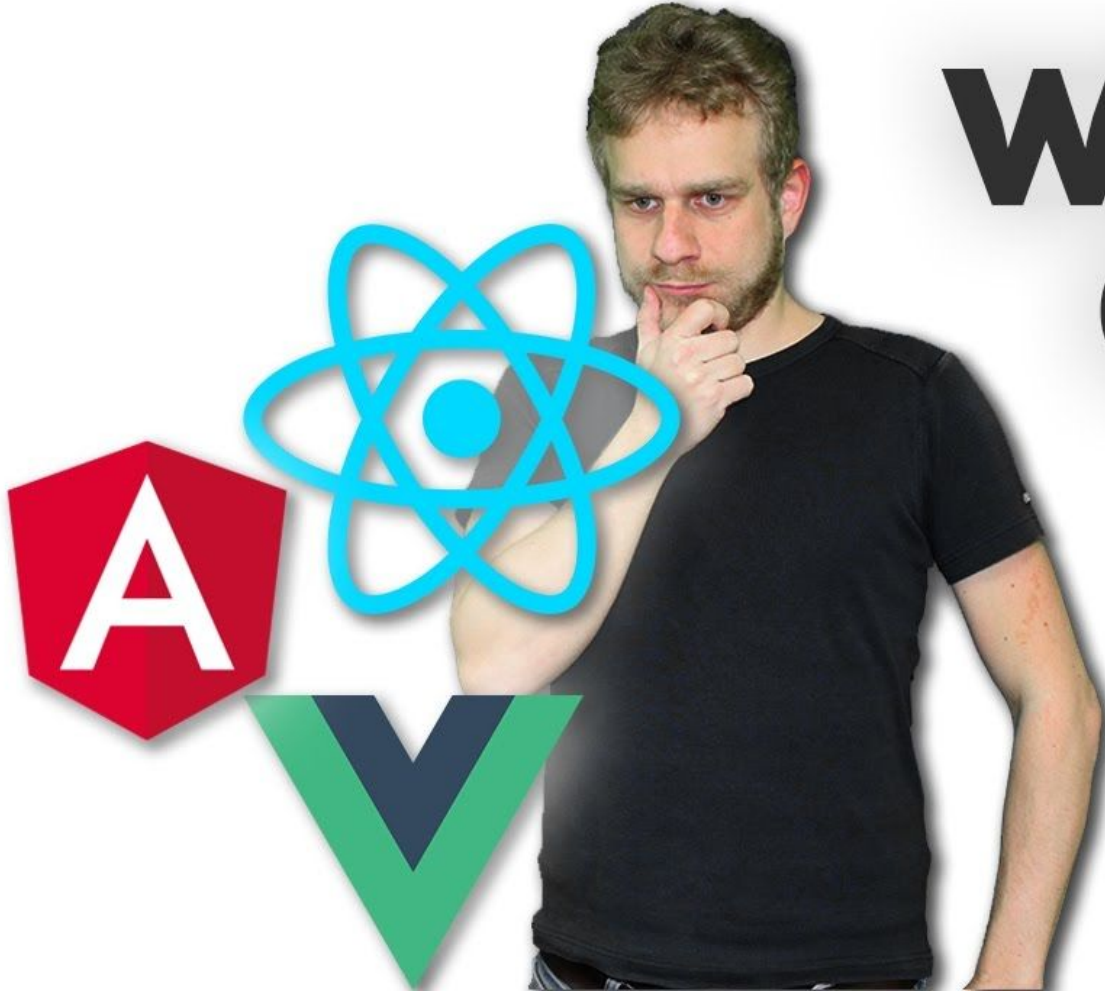
[Web-хостинг](#)

[Web-дизайн](#)

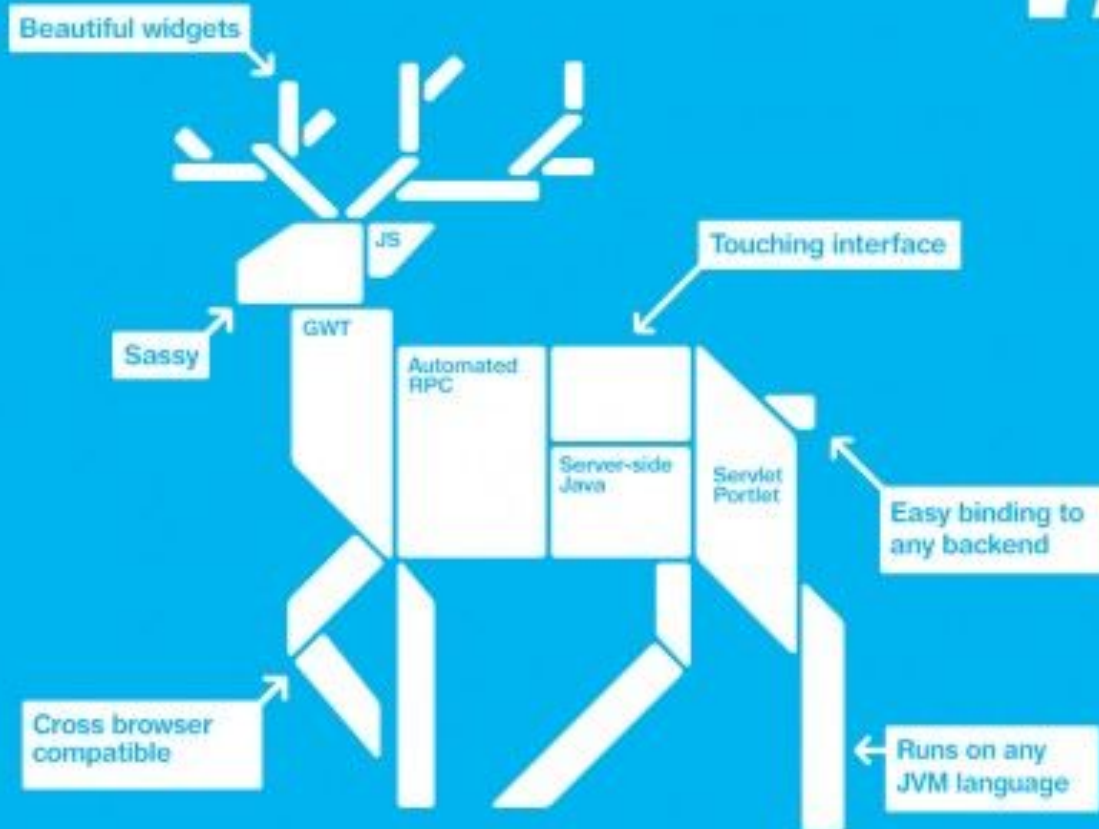
Тех.поддержка

[FAQ](#)





**WHICH
ONE?**







Проблемы

- Многомегабайтные клиенты
- Тормоза, особенно на мобильных устройствах
- Время на разработку, согласование и поддержку Rest API
- Сложная кастомизация

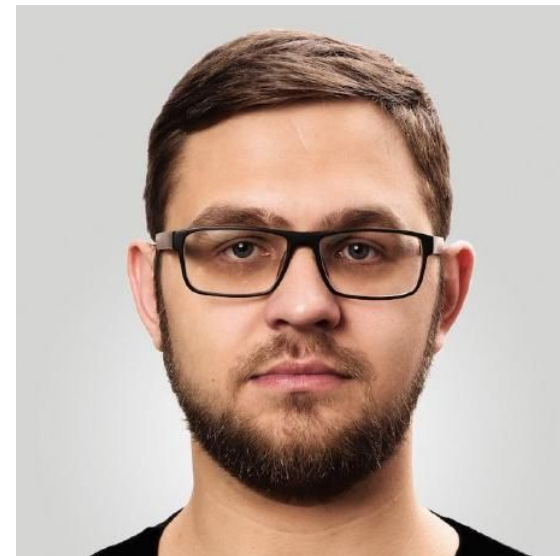
Korolev

- В честь Сергея Павловича Королёва
- Библиотека для построения SPA
- Полностью на сервере
- Написана в функциональном стиле

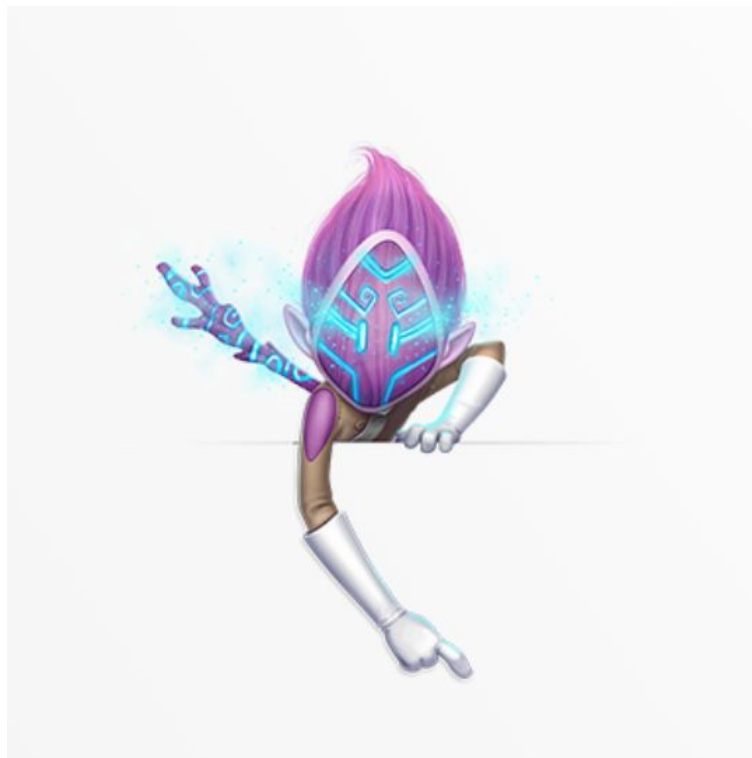


Korolev

- Алексей Фомкин
- <https://github.com/fomkin/korolev>
- <https://github.com/fomkin/levsha>
- https://t.me/korolev_io
- <https://gitter.im/fomkin/korolev>



Korolev



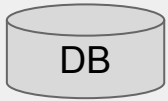
<https://mytc.io/>

Korolev

- <https://contractpen.com>
- Один серьезный банк
- ...

Korolev

Server

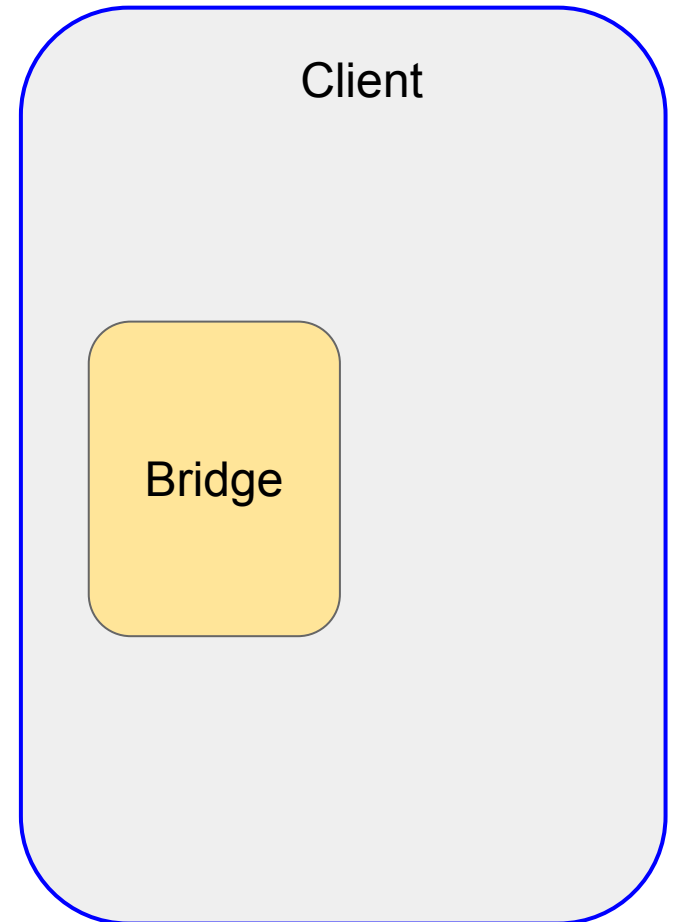
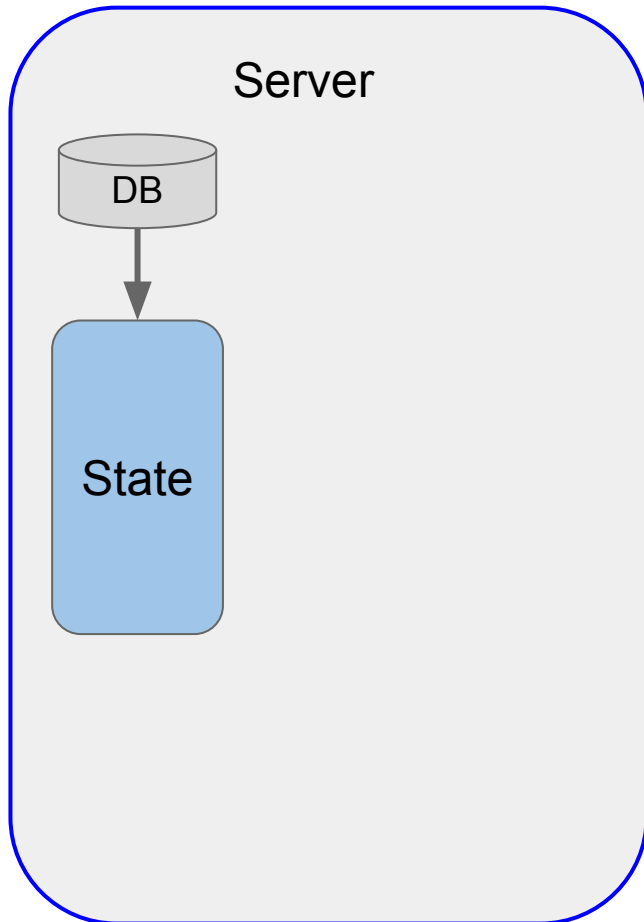


Client

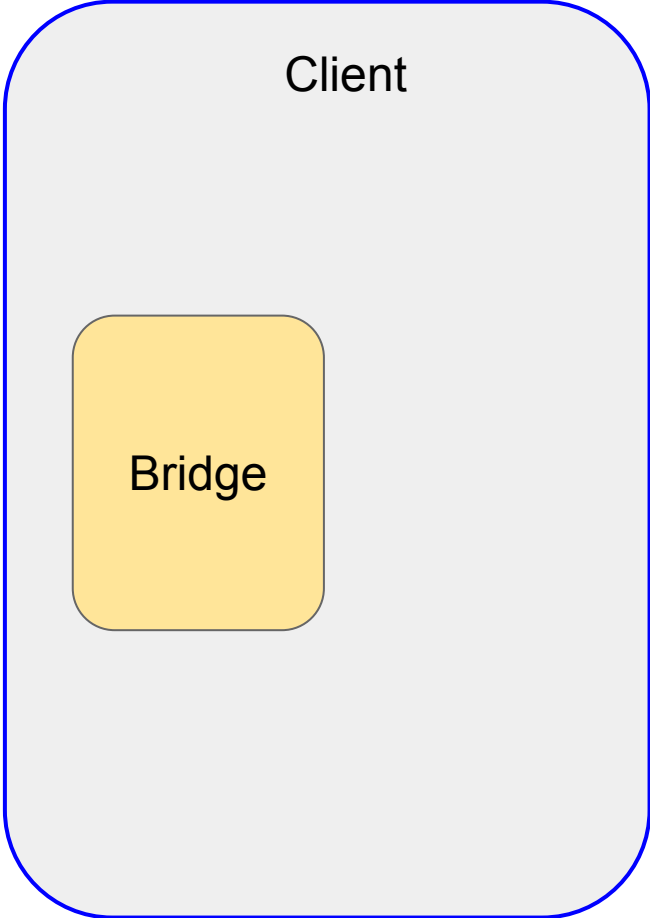
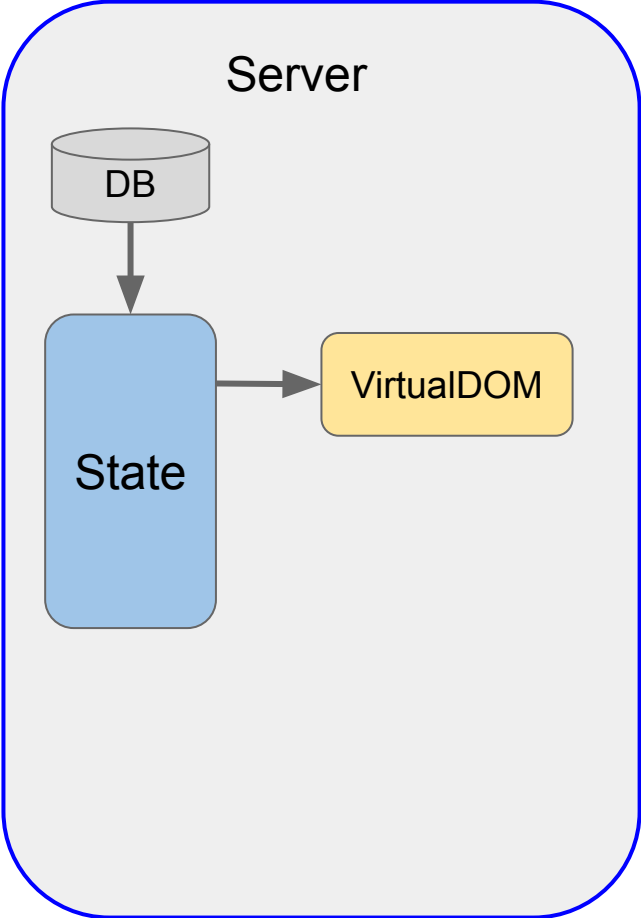
Bridge



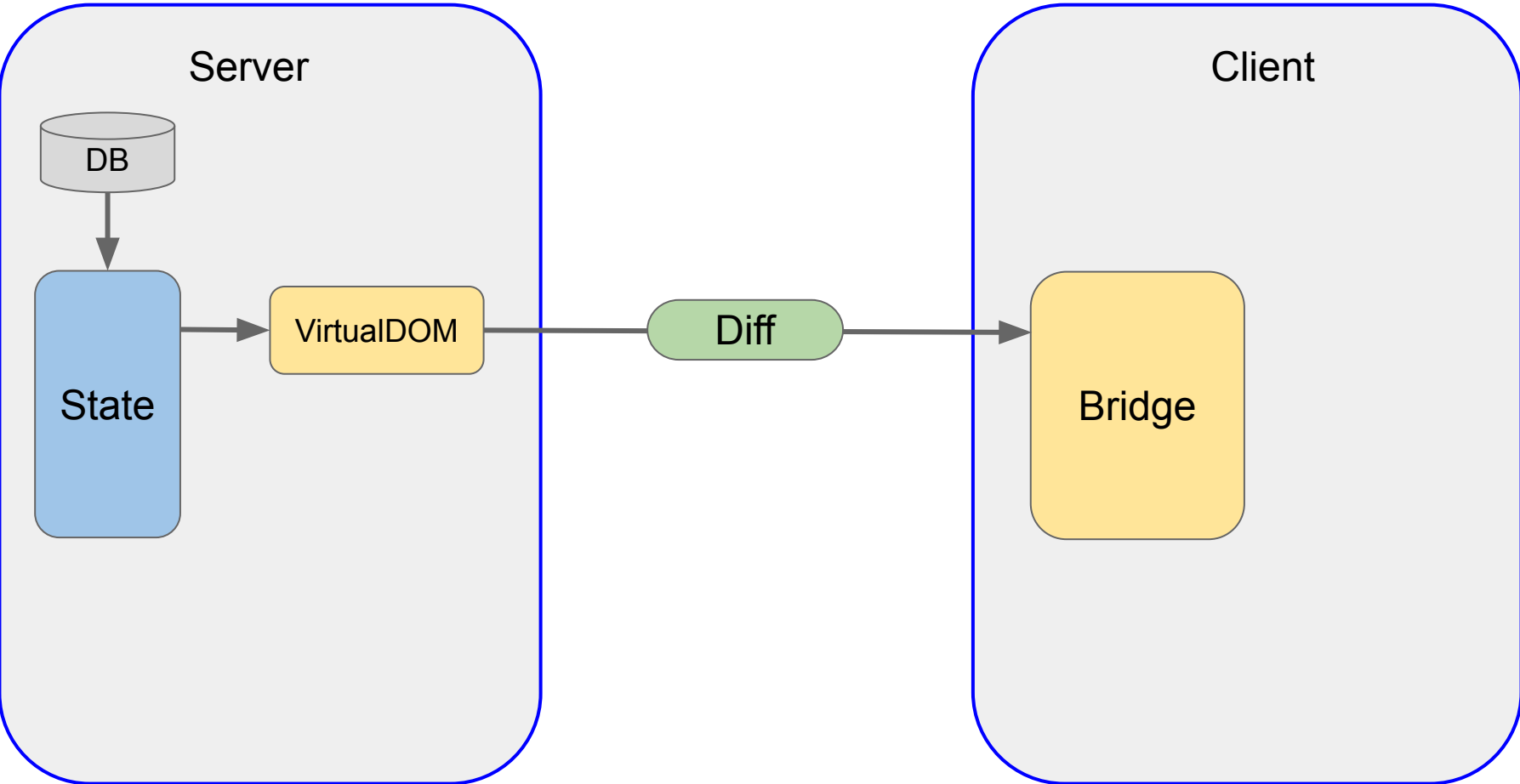
Korolev



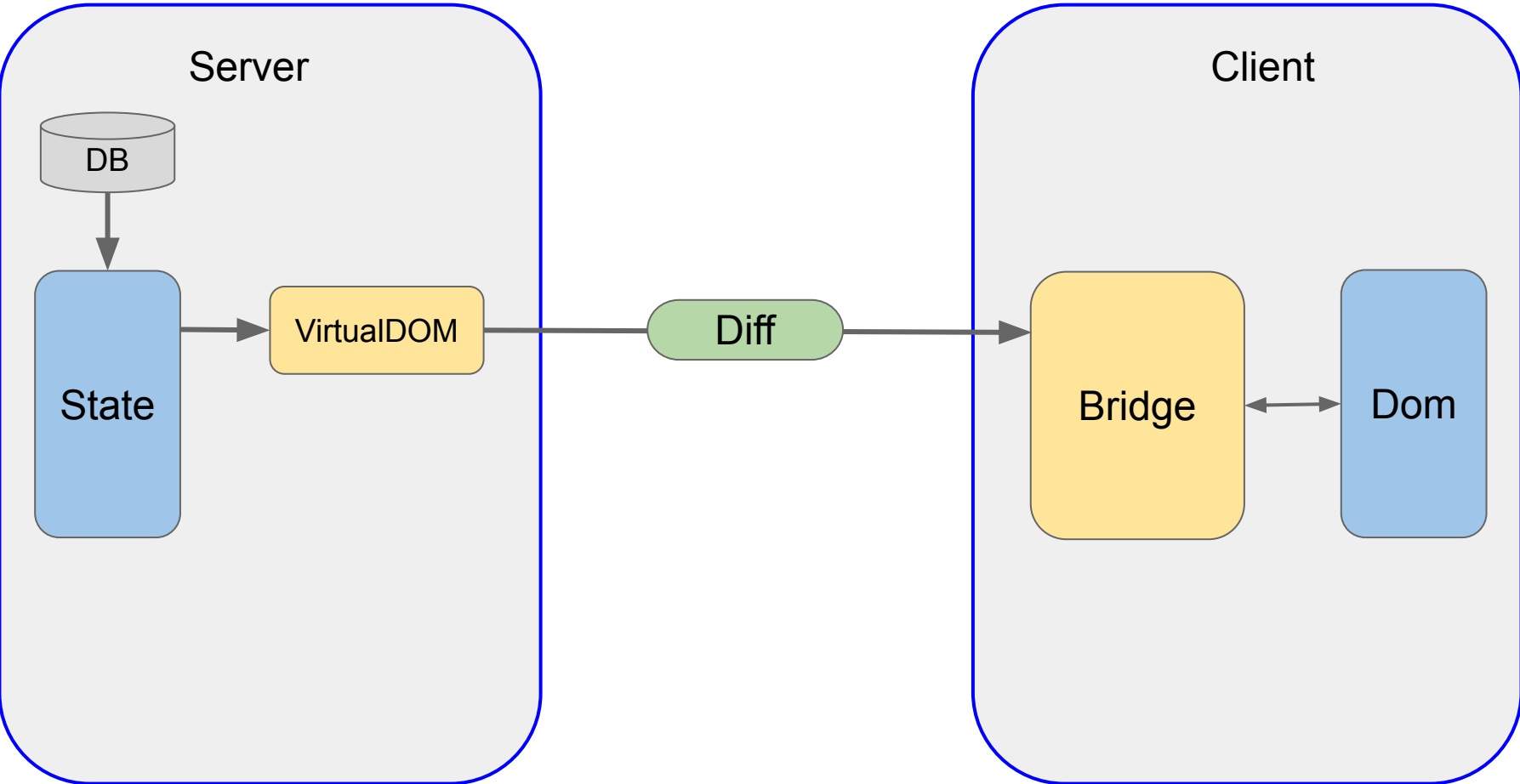
Korolev



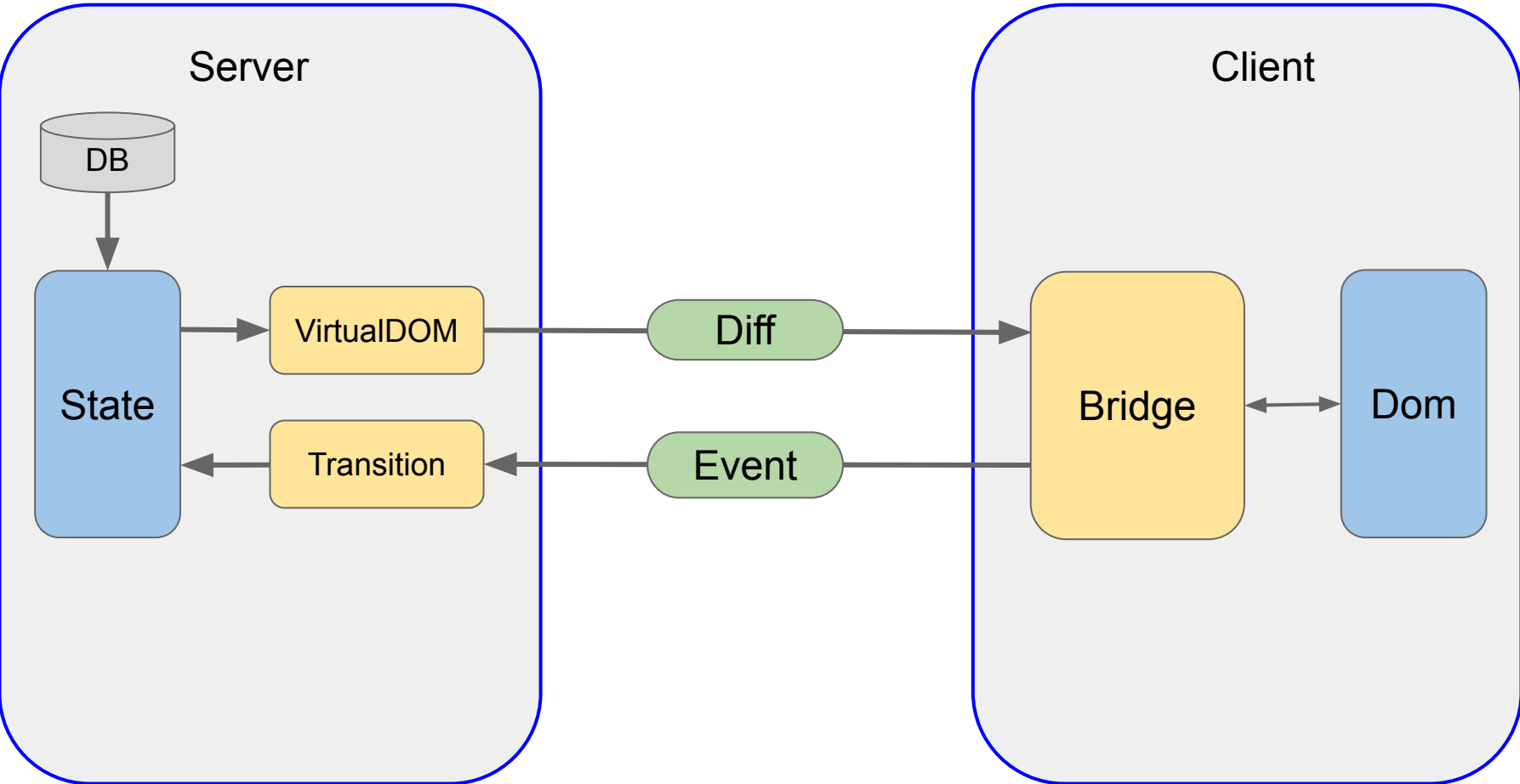
Korolev



Korolev



Korolev



Преимущества

- Очень маленький размер клиента
- Вся логика на сервере
- На клиент передается только то что видит пользователь
- Сверхнизкое потребление памяти на клиенте
- Не надо делать РЕСТ и все что с этим связано.
- Писать на Scala

Недостатки



Негостатки



Негосмакку



Недостатки

- Не работает без интернета
- Чувствительность к пингу
- Высокая нагрузка на GC

Рабочий пример

```
object AkkaHttpExample extends App {

  private implicit val actorSystem: ActorSystem = ActorSystem()
  private implicit val materializer: Materializer = ActorMaterializer()

  val applicationContext = Context[Future, Boolean, Any]

  import applicationContext._
  import symbolDsl._

  private val config = KorolevServiceConfig[Future, Boolean, Any](
    stateStorage = StateStorage.default(initialState = false),
    router = emptyRouter,
    render = { case _ => 'div("Hello akka-http") }
  )

  private val route = akkaHttpService(config).apply(AkkaHttpServerConfig())

  Http().bindAndHandle(route, interface = "0.0.0.0", port = 8080)
}
```

State

```
case class State(user: User)
```

```
case class User(name: String, role: String)
```

State

```
sealed trait MyState  
  
case class Anonymous(deviceId: DeviceId) extends MyState  
  
case class Authorized(deviceId: DeviceId, user: User) extends MyState  
  
case class User(id: String, name: String, friends: Seq[String])
```

State

```
val stateStorage: StateStorage[Future, UiState] =  
  StateStorage.forDeviceId[Future, UiState] {  
    deviceId => StateService.forDevice(deviceId)  
  }
```

Transition

```
type Transition[State] = State => State
```

Transition

```
val newFriend = "Karl Heinrich Marx"
userService.addFriend(user.id, newFriend) flatMap { _ =>
  | access.transition { case
    | | state: Authorized(_, user) =>
    | | | state.copy(user = user.copy(friends :+ newFriend))
    | | }
  }
}
```

Transition

```
'a ("Login", 'class /= "button",
  event( name = 'click)(loginClick)
)

def loginClick: Access => EventResult = { access =>
  access.transition {
    case state: NotAuthorizedState =>
      StateService.registerDevice(state.deviceId)
  }
}
```

Render

```
PartialFunction[State, Node] = State => Node
```


Render

```
val render: Render = {  
  case Anonymous(_) =>  
    'body(  
      'form(  
        'input('placeholder /= "Login"),  
        'input('placeholder /= "Password"),  
        'button("Submit")  
      )  
    )  
  case Authorized(_, User(name, friends)) =>  
    'body(  
      'div(s"Your name is $name. Your friends:"),  
      'ul(  
        friends map { friend => |  
          'li(friend)  
        }  
      )  
    )  
}
```

Template DSL

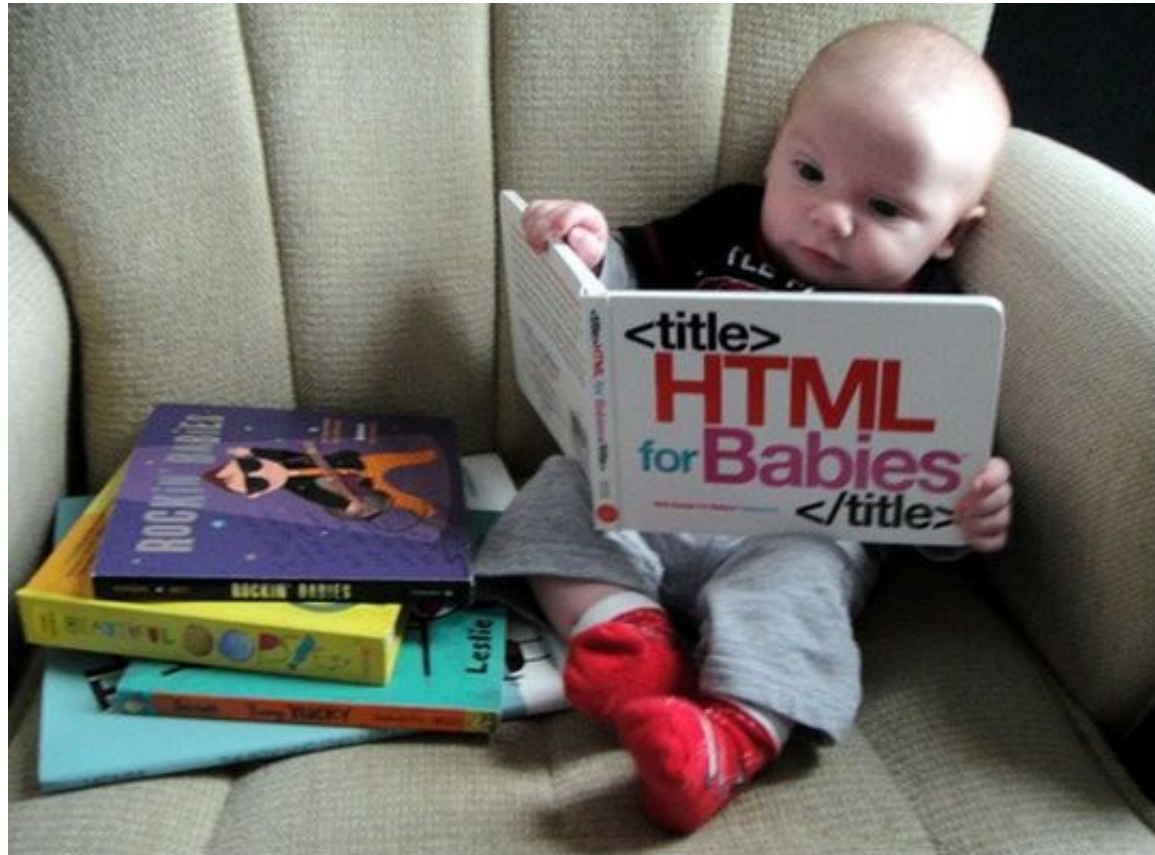
```
'div(  
|   'backgroundColor @= "yellow",  
|   'input('type /= "text")  
)
```

```
<div style="background-color: yellow">  
|   <input type="text"></input>  
</div>
```

Template DSL

```
'body (  
  'section ('class /= "hero is-success is-fullheight",  
    'div ('class /= "hero-body",  
      'div ('class /= "container has-text-centered",  
        'div ('class /= "column is-4 is-offset-4",  
          'h3 ("Login", 'class /= "title has-text-grey"),  
          'p ("Please login to proceed.", 'class /= "subtitle has-text-grey"),  
          'div ('class /= "box",  
            'figure ('class /= "avatar", 'img ('src /= "/img/avatar.png")),  
            'div (  
              'div ('class /= "field",  
                'div ('class /= "control",  
                  'input (  
                    'class /= "input is-large",  
                    'type /= "email",  
                    'placeholder /= "Your Email",  
                    'autofocus /= ""  
                  )  
                )  
              ),  
              'div ('class /= "field", 'label ('class /= "checkbox", 'input ("Remember me", 'type /= "checkbox"))),  
              'a ("Login", 'class /= "button is-block is-info is-large", event('click)(loginClick))  
            )  
          ),  
          'p ('class /= "has-text-grey",  
            'a ("Sign Up", 'href /= "../", "&nbsp;&nbsp;&nbsp;"),  
            'a ("Forgot Password", 'href /= "../", "&nbsp;&nbsp;&nbsp;"),  
            'a ("Need Help?", 'href /= "../")  
          )  
        )  
      )  
    )  
  )  
)
```

Template DSL



Template DSL converter

```
<body>
  <section class="hero is-success is-fullheight">
    <div class="hero-body">
      <div class="container has-text-centered">
        <div class="column is-4 is-offset-4">
          <h3 class="title has-text-
grey">Login</h3>
          <p class="subtitle has-text-
grey">Please login to proceed.</p>
          <div class="box">
            <figure class="avatar">
              
            </figure>
            <form>
              <div class="field">
                <div class="control">
                  <input class="input
is-large" type="email" placeholder="Your Email"
autofocus="">
                </div>
              </div>
              <div class="field">
                <div class="control">
                  <input class="input
is-large" type="password" placeholder="Your Password">
                </div>
              </div>
              <div class="field">
                <label
class="checkbox">
                  <input type="checkbox">
Remember me
                </label>
              </div>
            </form>
          </div>
        </div>
      </div>
    </div>
  </section>
</body>
```

```
,
'section(
  'class /= "hero is-success is-fullheight",
  'div(
    'class /= "hero-body",
    'div(
      'class /= "container has-text-centered",
      'div(
        'class /= "column is-4 is-offset-4",
        'h3(
          'class /= "title has-text-grey",
          "Login"
        ),
        'p(
          'class /= "subtitle has-text-grey",
          "Please login to proceed."
        ),
        'div(
          'class /= "box",
          'figure(
            'class /= "avatar",
            'img(
              'src /= "https://placeholder.it/128x128"
            )
          ),
          'form(
            'div(
              'class /= "field",
              'div(
```

Template DSL



BUTTON

```
def button(title: String, effect: Access ⇒ EventResult): Node =  
  'button ('class /≠ "button is-info is-large", title, event(name = 'click)(effect))
```

Template DSL



The image shows a screenshot of a web interface using the Bulma CSS framework. At the top left, there is a dark grey header with the Bulma logo (a teal diamond) and the text "BULMA" in white. Below the header, the main content area is light grey and contains a grid of elements. The first row consists of two columns: "Left Title" and "Right Title". The second row consists of six tabs: "Tab1", "Tab2", "Tab3", "Tab4", "Tab5", and "Tab6". The "Tab3" tab is highlighted with a white background and blue text. Below the tabs, the content area is divided into two columns: "We are the Tab1" and "We are the Tab3".

Template DSL

```
trait PanelState {
  def tabs: TabState
}

trait Panel {
  def render(clazz: String, state: PanelState, filter: Boolean): Node =
    'nav (
      'class ≙ "panel ",
      'div ('class ≙ "tab-content",
        'div ('class ≙ "tabs is-boxed",
          'ul (
            tabHeader(state)
          )
        ),
        tabContent(state)
      )
    )

  def tabHeader(state: PanelState): Node

  def tabContent(state: PanelState): Node

  def tabClick(tab: TabPane): Access ⇒ EventResult
}
```


Events

```
case class MyState(i: String)

val renderAnonymous: Render = {
  case MyState(i) =>
    'body(
      i.toString,
      'button("Increment",
        event('click) { access =>
          access.transition {
            case MyState(i) =>
              | state.copy(i = i + 1)
          }
        }
      )
    )
}
```

Events

```
<- [6]
-> [9]
<- [0,"7:1_1_2_2_2_1_1_2_1:click"]
-> [4,3,"1_1_2_2_2_1_1_1",0,"class","",false,3,"1_1_2_2_2_1_1_2",0,"class","is-active",false,1,"1_1_2_2_2_2","1_1_2_2_2_2_1","We are the Tab4" ]
-> [0,8 ]
<- [0,"8:1_1_2_2_2_1_1_3_1:click"]
-> [4,3,"1_1_2_2_2_1_1_2",0,"class","",false,3,"1_1_2_2_2_1_1_3",0,"class","is-active",false,1,"1_1_2_2_2_2","1_1_2_2_2_2_1","We are the Tab5" ]
-> [0,9 ]
<- [6]
-> [9]
<- [6]
-> [9]
<- [0,"9:1_1_2_2_2_1_1_2_1:click"]
-> [4,3,"1_1_2_2_2_1_1_2",0,"class","is-active",false,3,"1_1_2_2_2_1_1_3",0,"class","",false,1,"1_1_2_2_2_2","1_1_2_2_2_2_1","We are the Tab4" ]
-> [0,10 ]
<- [6]
-> [9]
2 <- [0,"10:1:resize"]
<- [6]
12 <- [0,"10:1:resize"]
---
```

Events



The image shows a Bulma component with a dark header containing the Bulma logo and the text "BULMA". Below the header is a grid layout. The first row contains two columns: "Left Title" and "Right Title". The second row contains six tabs: "Tab1", "Tab2", "Tab3", "Tab4", "Tab5", and "Tab6". "Tab3" is the active tab, highlighted with a white background and blue text. Below the tabs, the content area shows "We are the Tab1" under the first column and "We are the Tab3" under the "Tab3" column.

Events

```
<- [0,"6:1_1_2_1_2_1_1_1_1:click"]
```

```
-> [4,3,"1_1_2_1_2_1_1_1",0,"class","is-active",false,3,"1_1_2_1_2_1_1_2",0
```

```
1 0 1 1
```

Events

BULMA

Left Title Right Title

Tab1 **Tab2** **Tab3** Tab4 Tab5 Tab6

Super TODO tracker We are the Tab3

- This is ~~TODO~~ #0
- This is TODO #1
- This is TODO #2
- This is ~~TODO~~ #3
- write code

What should be done?

Add todo

Events

```
<- [0,"2:1_1_2_1_2_1_1_2_1:click"]
```

```
->
[4,3,"1_1_2_1_2_1_1_1",0,"class","",false,3,"1_1_2_1_2_1_1_2",0,"class"
active",false,0,"1_1_2_1_2_2","1_1_2_1_2_2_1",0,"div",1,"1_1_2_1_2_2_1"
_1","Super TODO
tracker",0,"1_1_2_1_2_2_1","1_1_2_1_2_2_1_2",0,"div",5,"1_1_2_1_2_2_1_2"
scroll",5,"1_1_2_1_2_2_1_2","height","250px",0,"1_1_2_1_2_2_1_2","1_1_2
,"div",0,"1_1_2_1_2_2_1_2_1","1_1_2_1_2_2_1_2_1_1",0,"input",3,"1_1_2_1
,"type","checkbox",false,0,"1_1_2_1_2_2_1_2_1","1_1_2_1_2_2_1_2_1_2",0,'
_1_2_2_1_2_1_2","1_1_2_1_2_2_1_2_1_2_1","This is TODO
#0",0,"1_1_2_1_2_2_1_2","1_1_2_1_2_2_1_2_2",0,"div",0,"1_1_2_1_2_2_1_2:
_1_2_2_1",0,"input",3,"1_1_2_1_2_2_1_2_2_1",0,"type","checkbox",false,0
_2_2","1_1_2_1_2_2_1_2_2_2",0,"span",1,"1_1_2_1_2_2_1_2_2_2","1_1_2_1_2
This is TODO
```

Events

```
val loginField = elementId()
val passwordField = elementId()

val renderAnonymous: Render = {
  case Anonymous(_) =>
    'body(
      'form(
        'input('placeholder /= "Login", loginField),
        'input('placeholder /= "Password", passwordField),
        'button("Submit"),
        event('submit) { access =>
          for {
            login <- access.property(loginField, 'value)
            password <- access.property(passwordField, 'value)
            user <- authService.authorize(login, password)
            <- access.transition {
              case Anonymous(deviceId) =>
                Authorized(deviceId, user)
            }
          } yield ()
        }
      )
    )
}
```

Web Components

Username

demo

Password

....



Login

```
'div('class /= "login", 'theme /= "dark",  
  'vaadinTextField('label /= "Username", 'name /= "username", 'value /= "demo"),  
  'vaadinPasswordField('label /= "Password", 'name /= "password", 'value /= "demo"),  
  'span('class /= "error-message", ""),  
  'vaadinButton("Login", 'id /= "login-button", 'theme /= "primary", event('click))  
)
```


Спасибо за внимание!

Вопросы ?



solver.it@gmail.com

<https://github.com/solverit/korolev-demo>