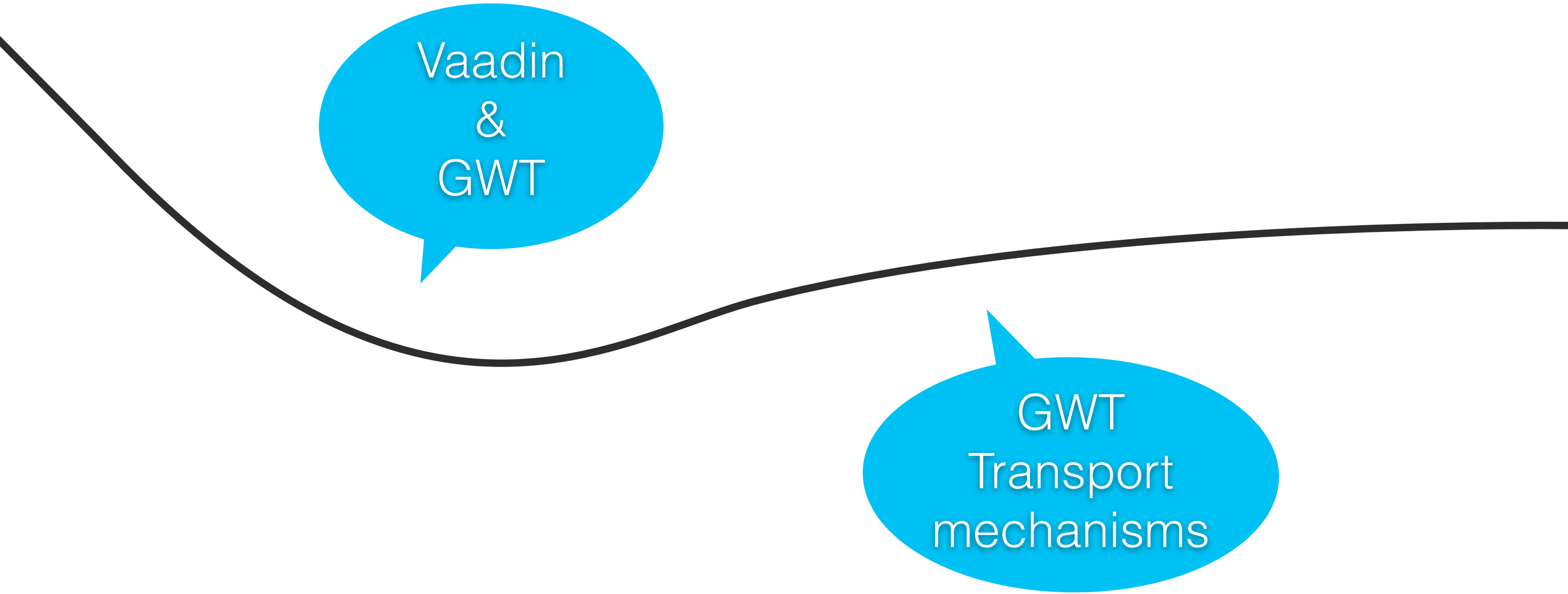
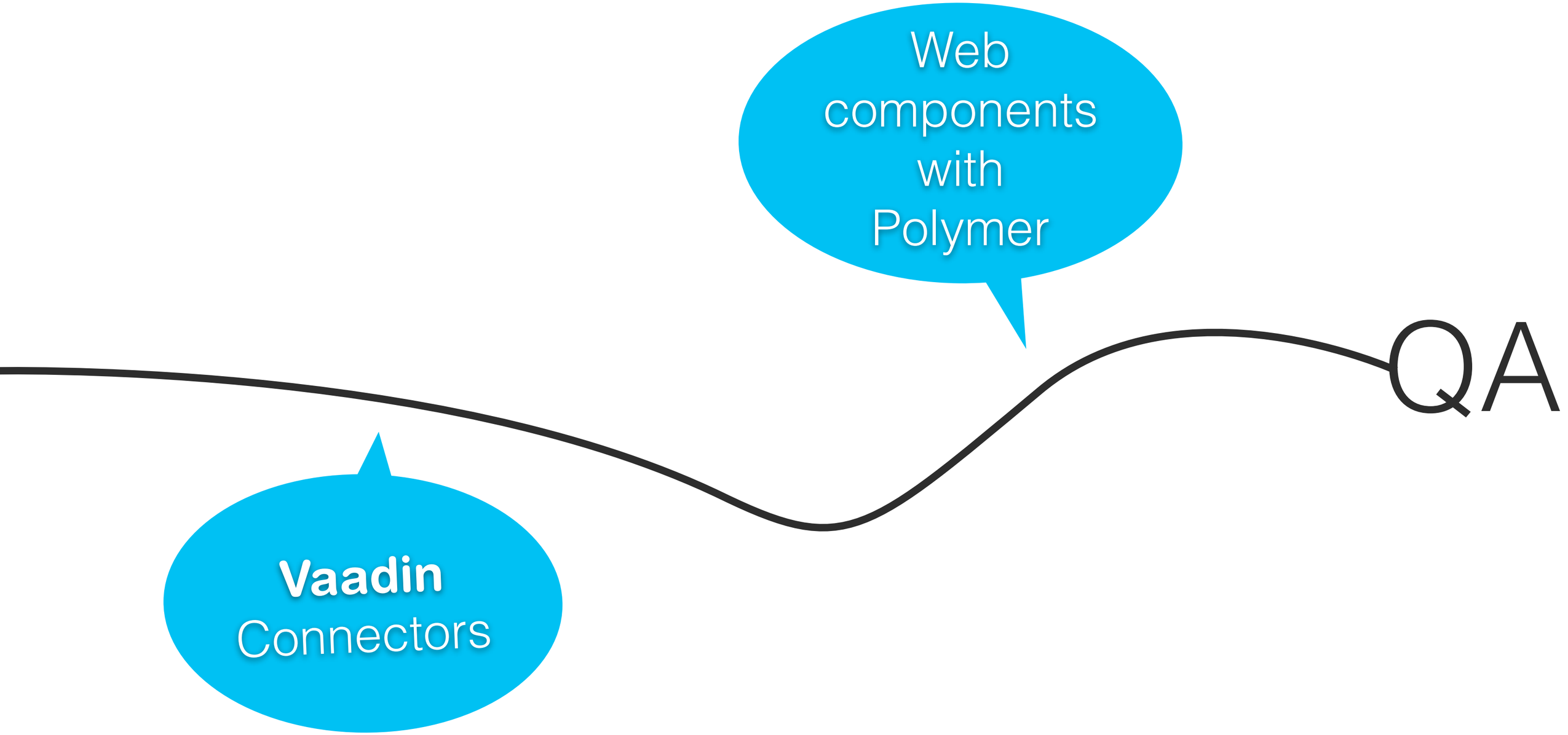
JavaCro

GWT integration with Vaadin

Peter Lehto
@peter_lehto
expert & trainer
vaadin


Vaadin
&
GWT
GWT
Transport
mechanisms

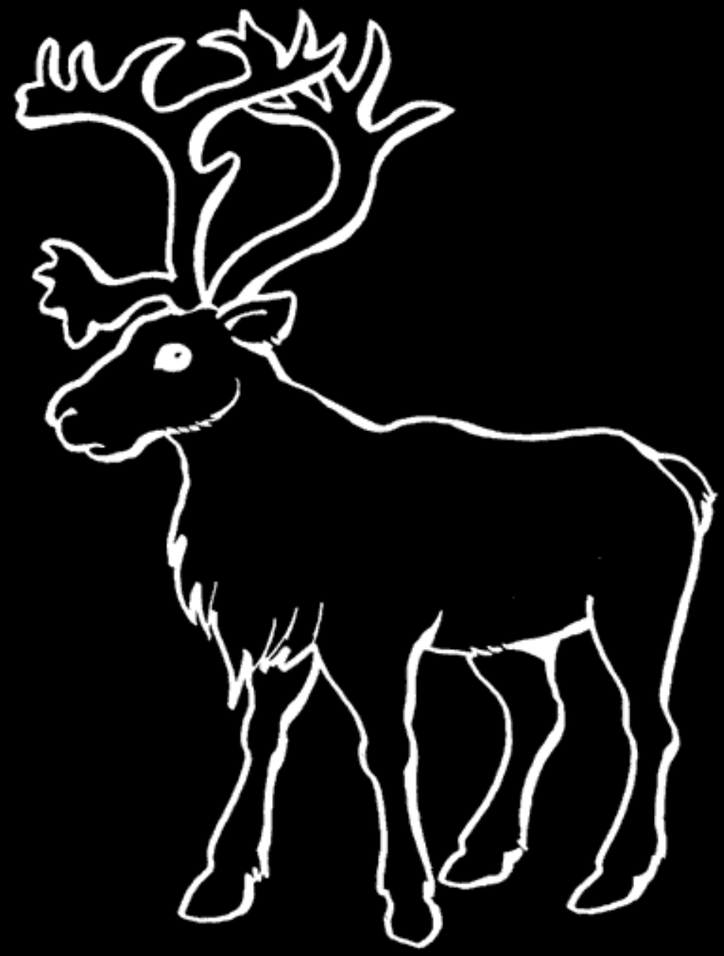


vaadin }>

Vaadin & GWT

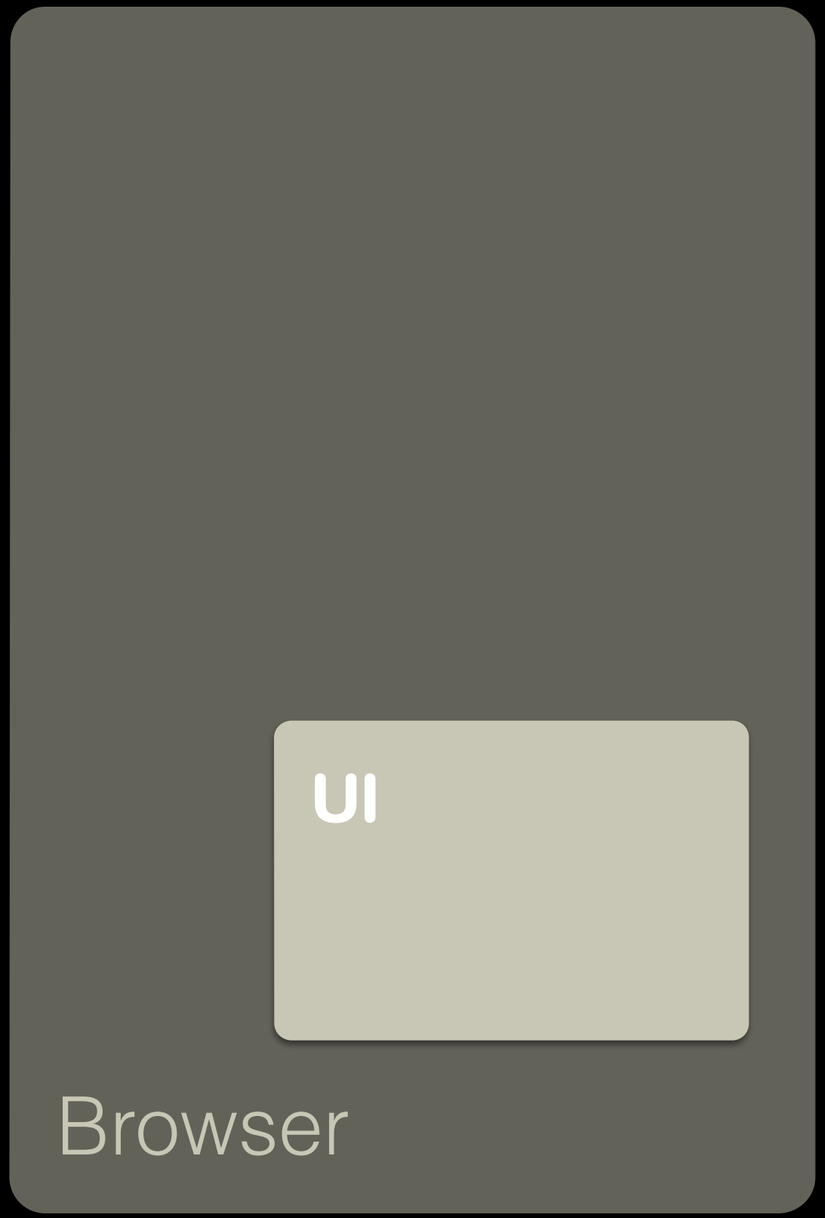
vaadin

vaadin }>

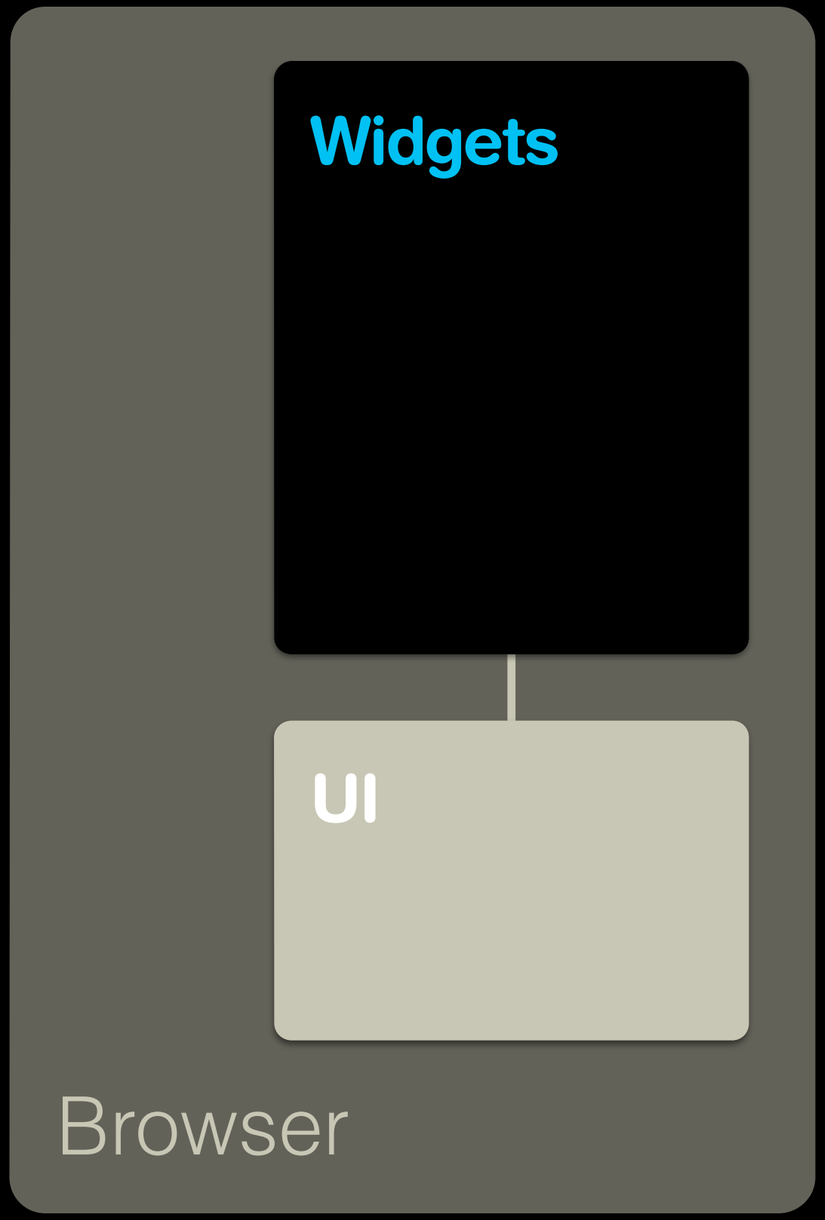


Server driven UI framework with GWT based thin client

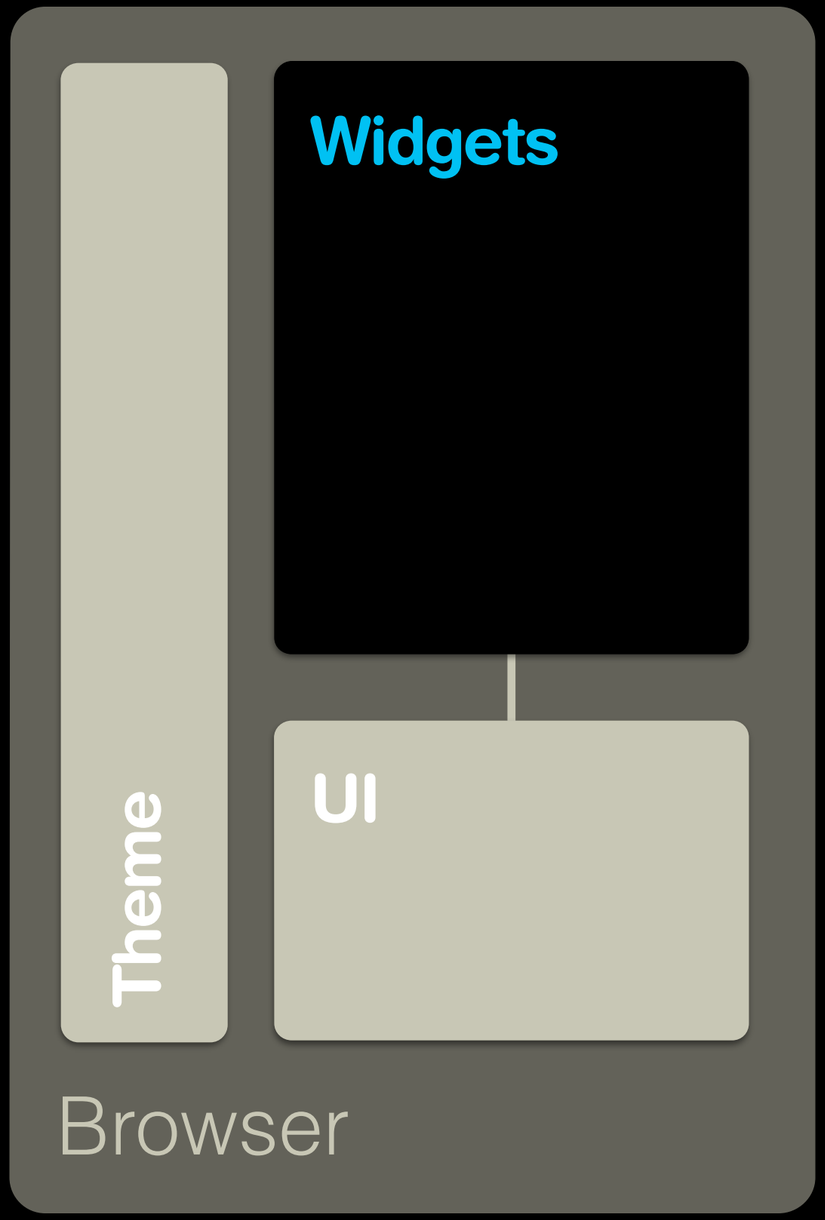
vaadin }>

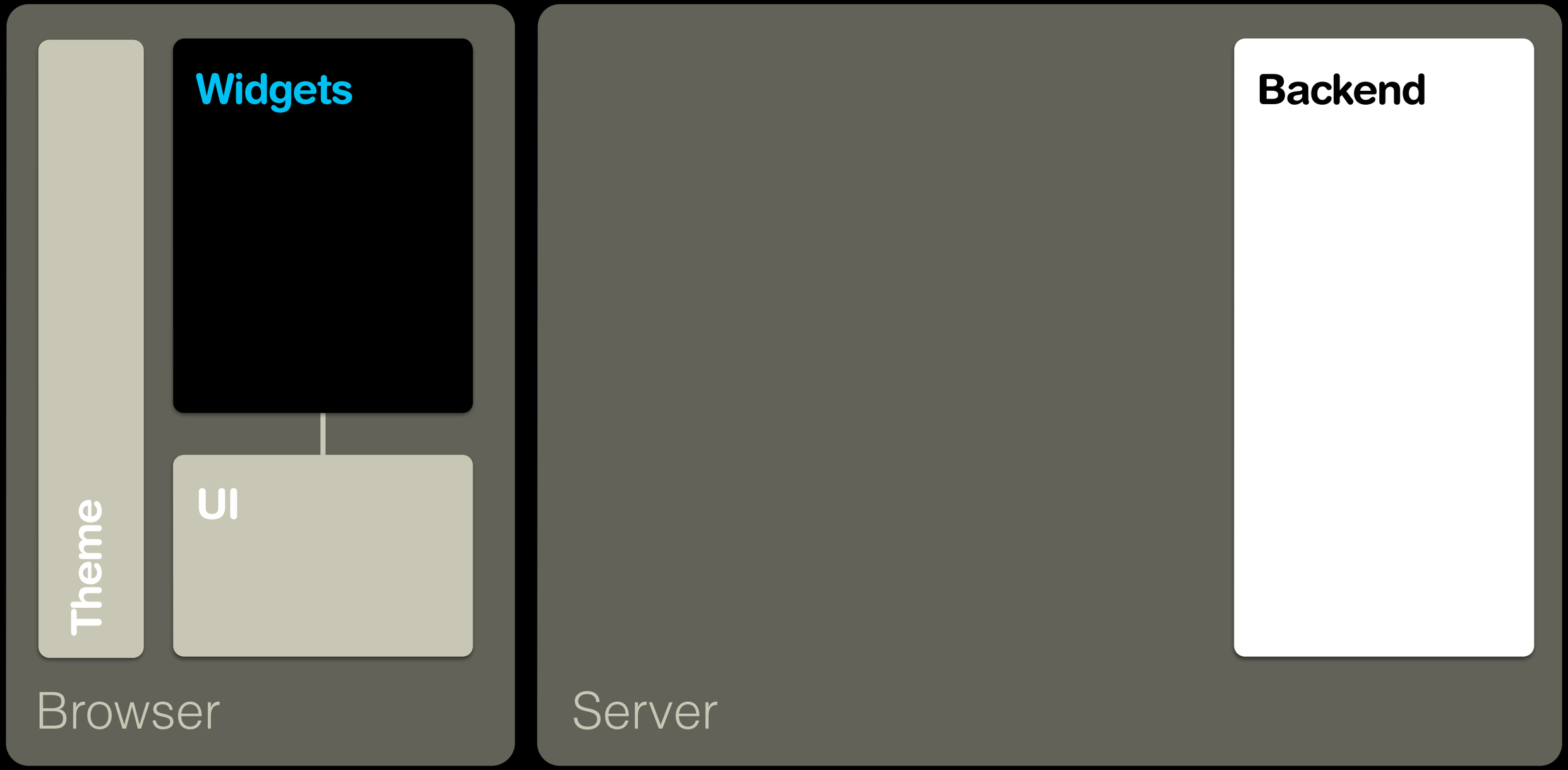


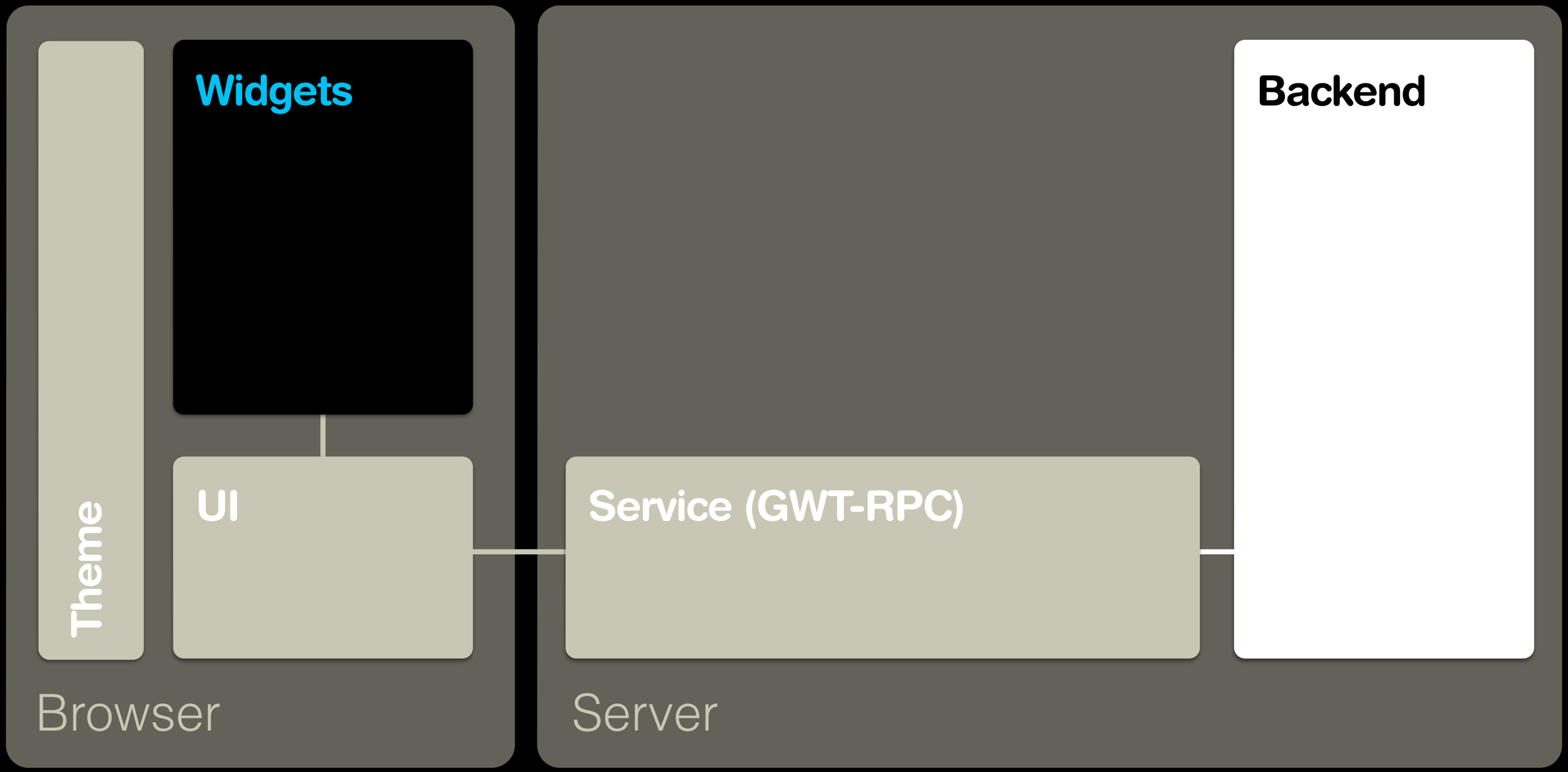
vaadin



vaadin }>


Widgets
UI
Theme
Browser

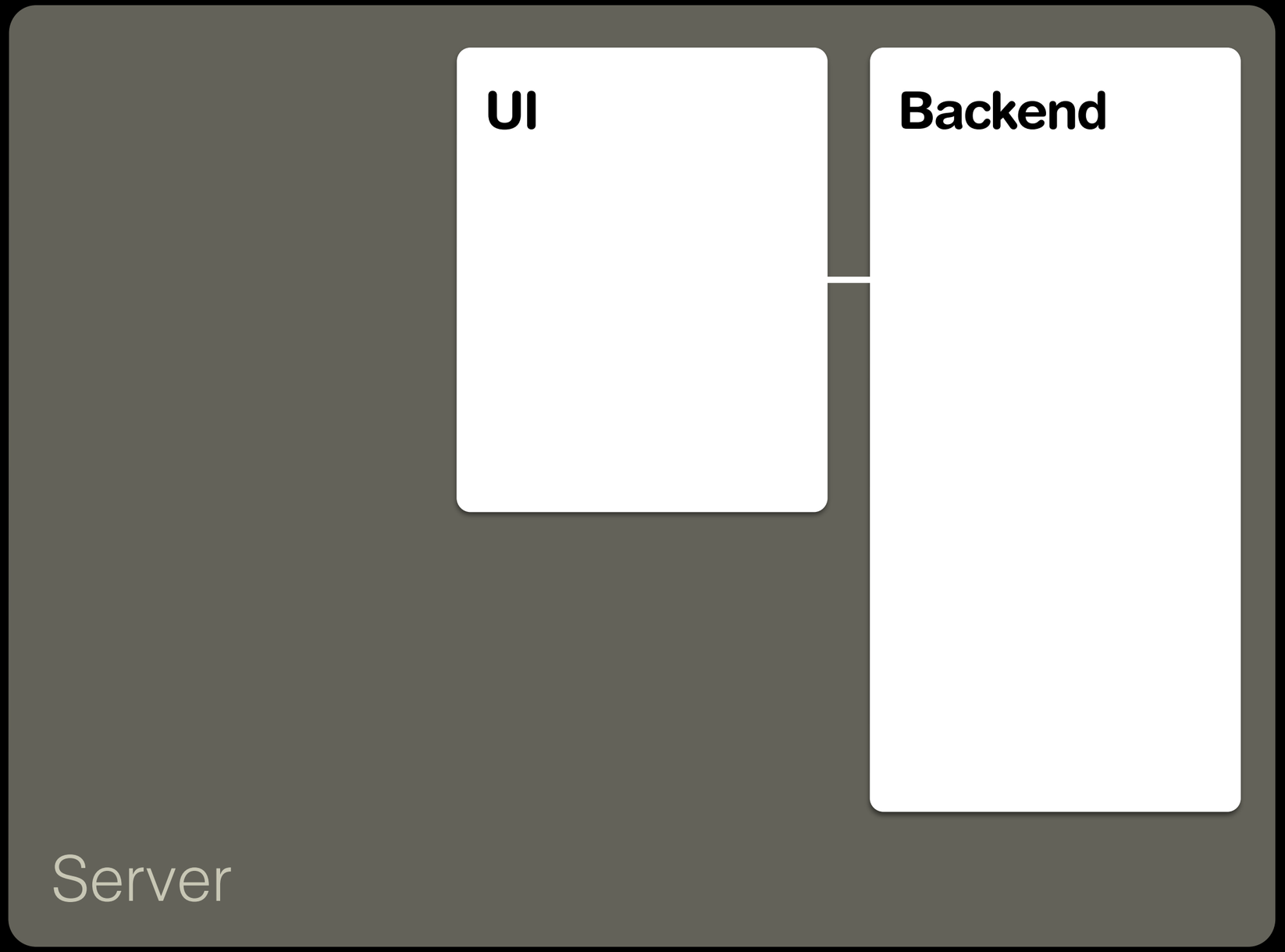

Widgets
UI
Theme
Backend
Browser
Server

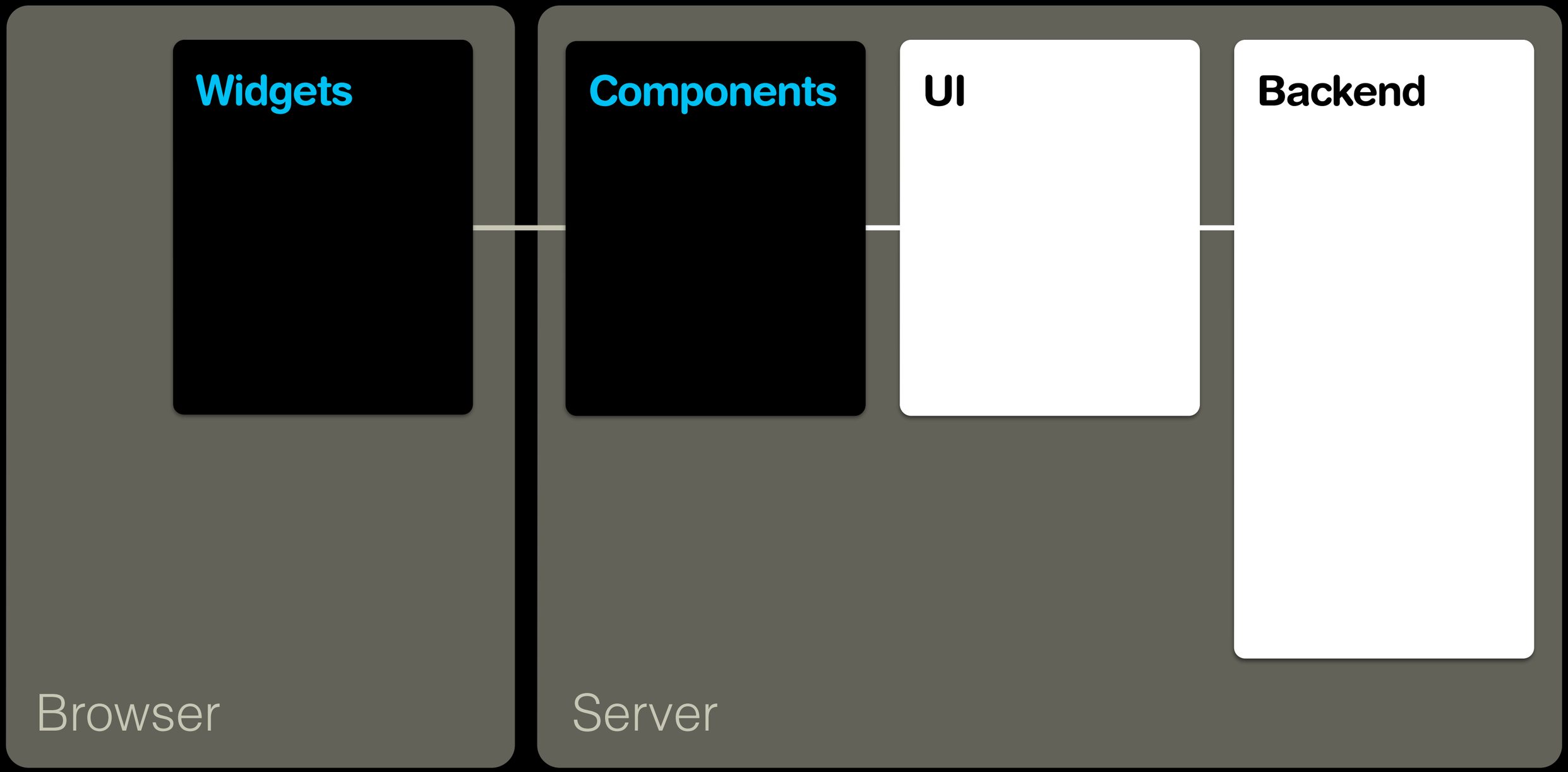

Widgets
Backend
Theme
UI
Service (GWT-RPC)
Browser
Server

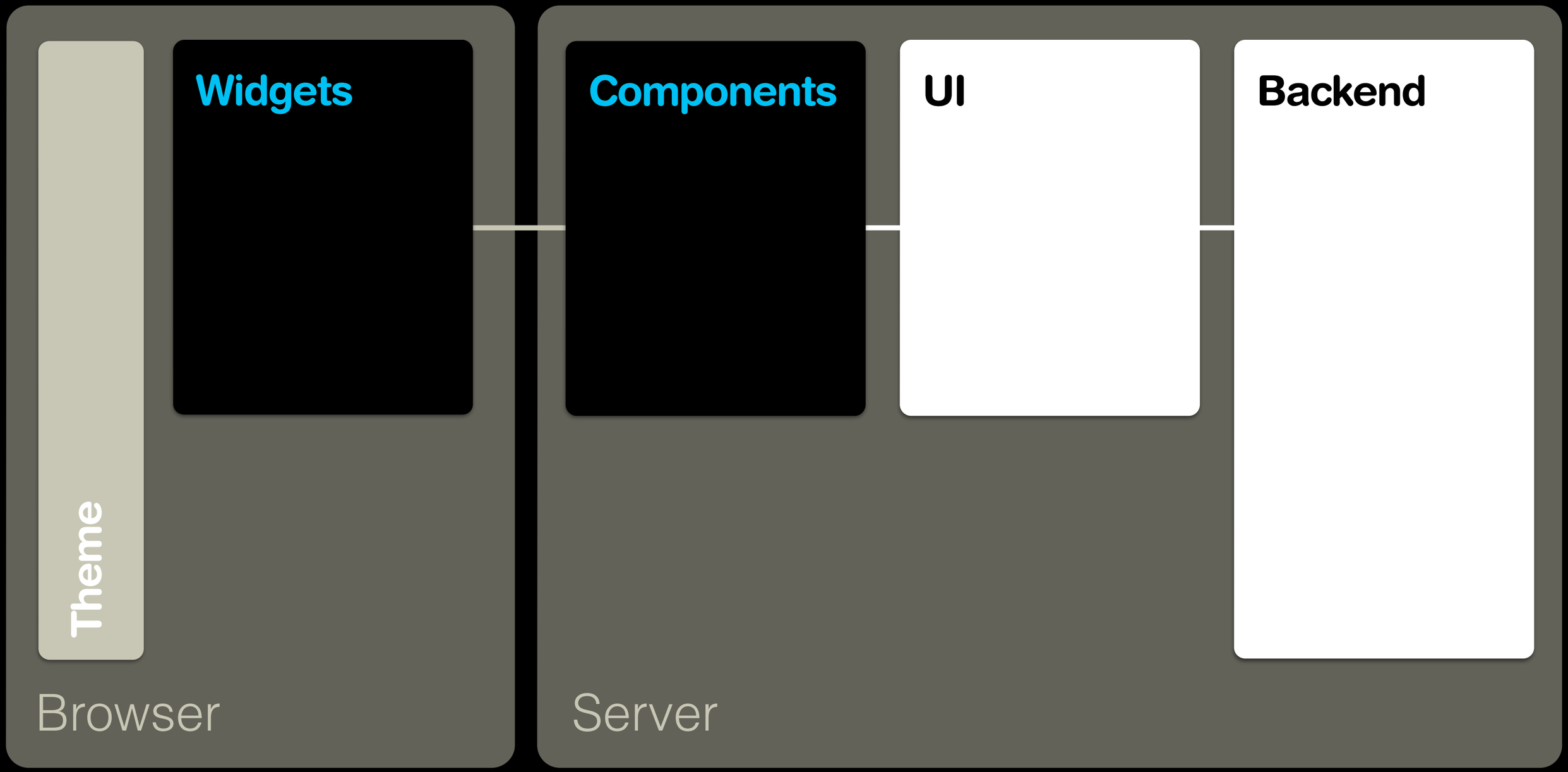
Backend

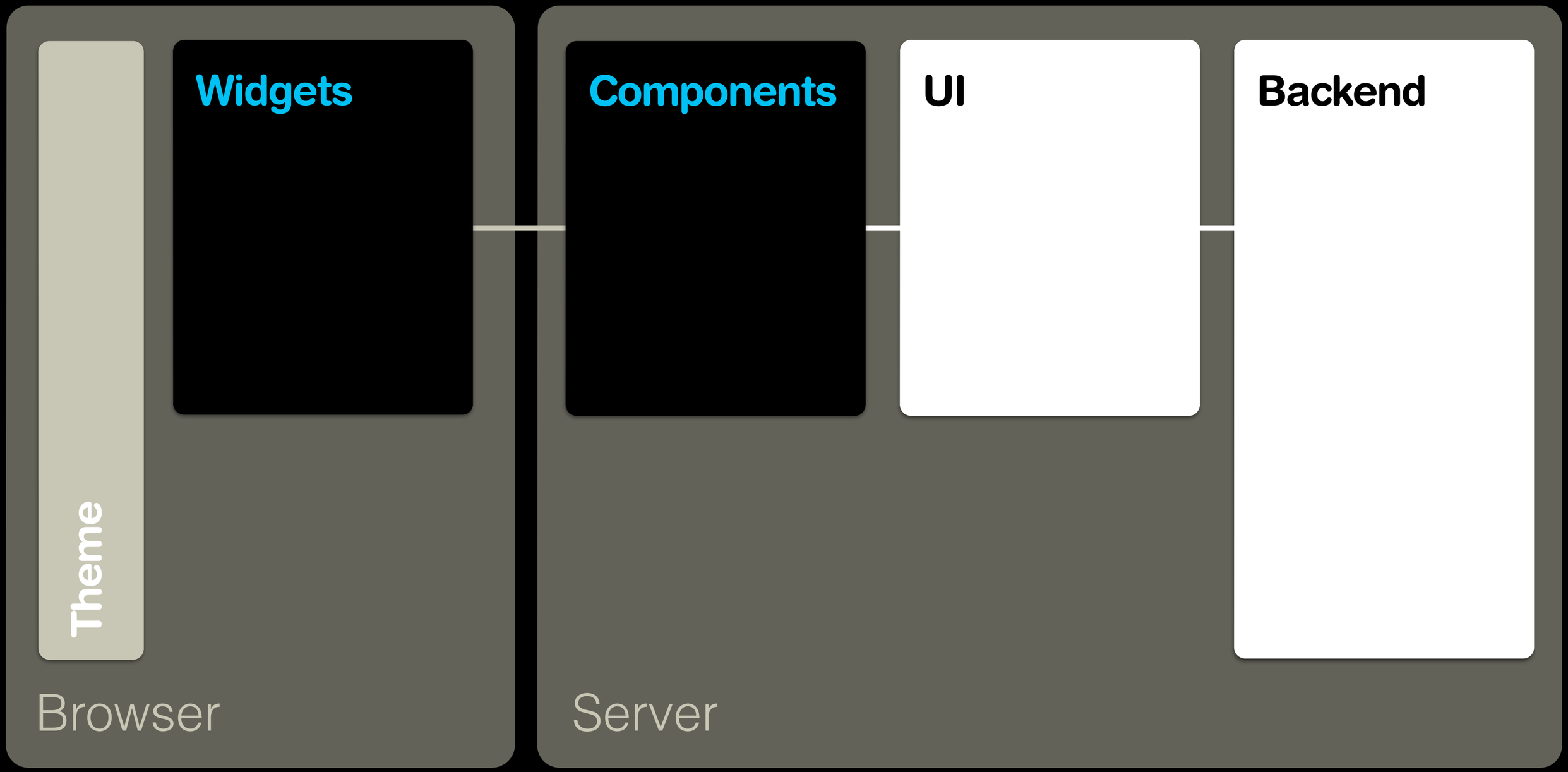
Server

vaadin

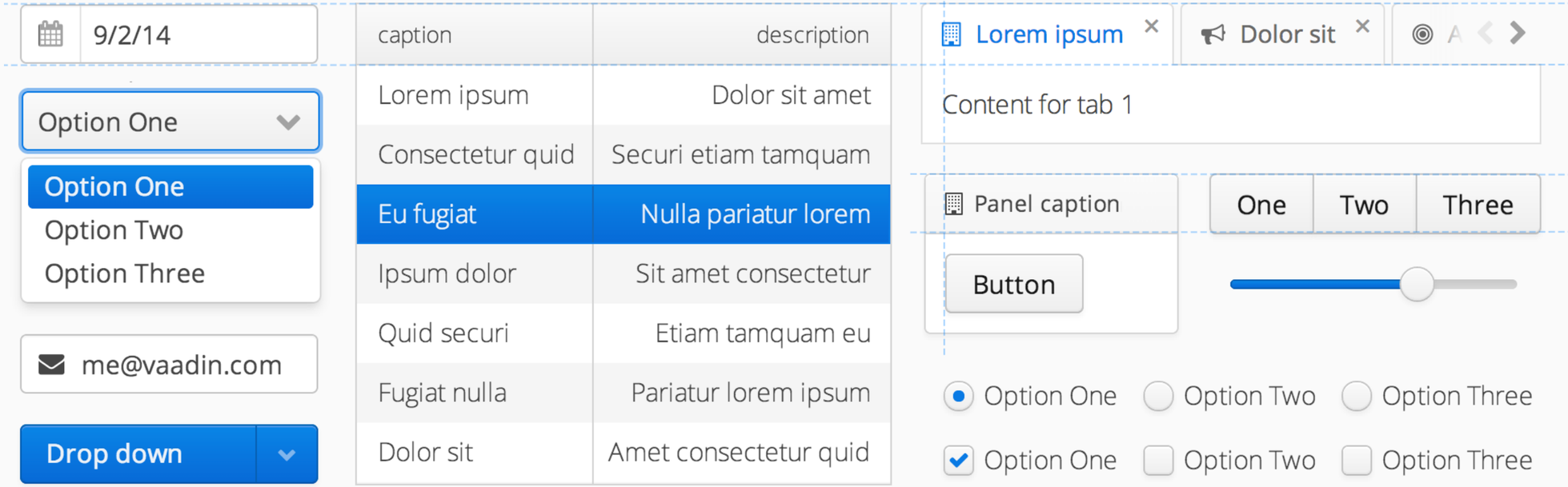

UI
Backend
Server

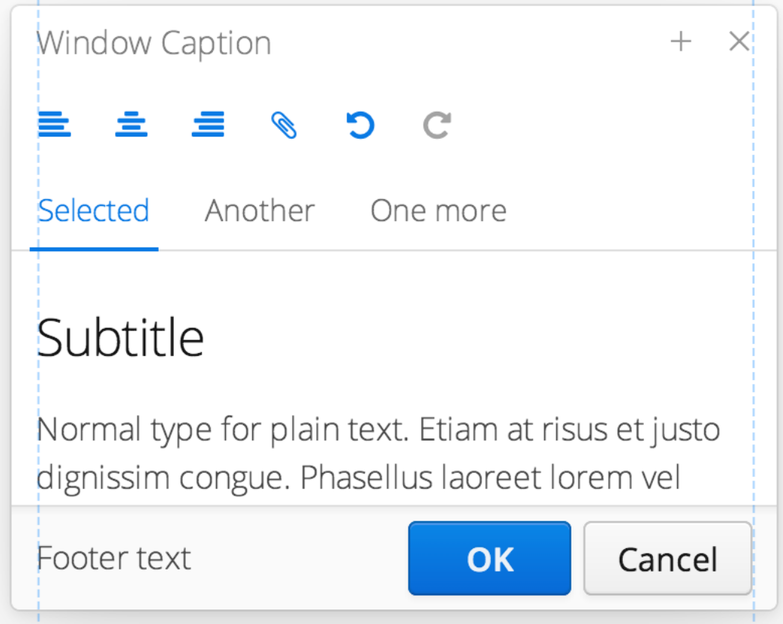

Widgets
Components
UI
Backend
Browser
Server


Widgets
Components
UI
Backend
Theme
Browser
Server


Widgets
Components
UI
Backend
Theme
Browser
Server

User Interface **Components**

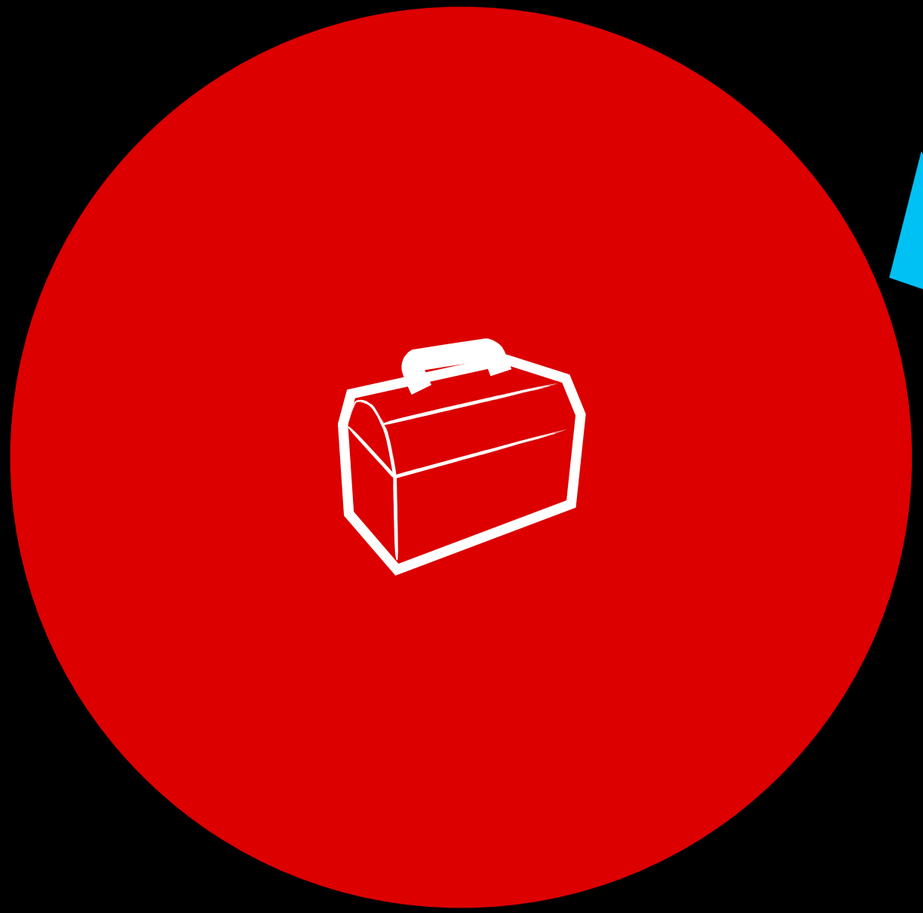




Developer Productivity + Rich UX

Vaadin += GWT

vaadin }>



vaadin }>

vaadin

RequestBuilder

```
RequestBuilder builder = new RequestBuilder(RequestBuilder.GET, url);
try {
    builder.sendRequest(requestDataString, new RequestCallback() {
        @Override
        public void onResponseReceived(Request request, Response response) {
            int statusCode = response.getStatusCode();
            String text = response.getText();
         }
         @Override
        public void onError(Request request, Throwable exception) {
            // TODO Handle asynchronous problems
         } });
} catch (RequestException e) {
    // TODO Handle synchronous problems
}
```

vaadin

RequestBuilder

Good

- It just works

vaadin }>

RequestBuilder

Good

- It just works

Bad

- Very low level

vaadin

What to send?

Contact

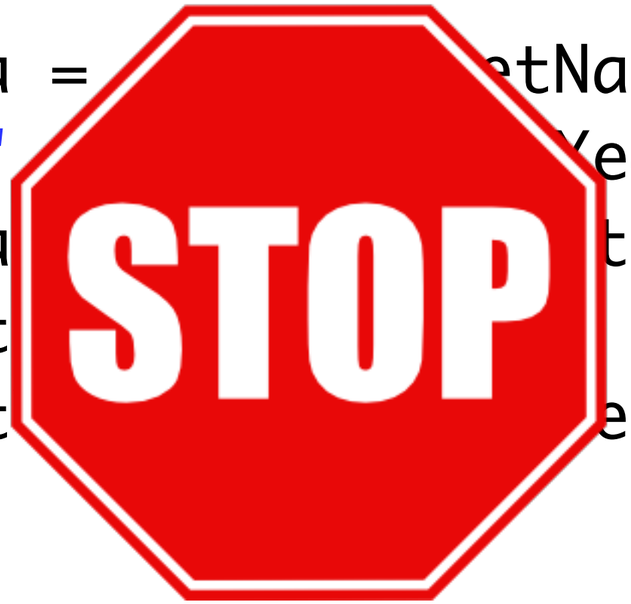
```
public class Contact {
    private String name;
    private int yearOfBirth;

    private List<String> emailAddresses;
    private Address address;

    public static class Address {
        private String street;
        private String city;
    }
    // + Getters and setters
}
```

String conversion

```
String data = contact.getName();
data += "," + contact.getYearOfBirth();
String[] parts = data.split(",");
contact.setName(parts[0]);
contact.setYearOfBirth(Integer.parseInt(parts[1]));
```

String conversion

```
String data =                    etName();
data += ","                      YearOfBirth();
String[] pa                      t(",");
contact.set
contact.set                      eger.parseInt(parts[1]));
```



JSON

```
{
  name: "John Doe",
  yearOfBirth: 1900,
  address: {
     street: "Happy Street 1",
     city: "Turku"
  },
  emailAddresses: ["john@doe.com", "johndoe@gmail.com"]
}
```

JSONValue parsing

```
JSONObject json = JSONParser.parseStrict(string).isObject();
contact.setName(json.get("name").isString().stringValue());
contact.setYearOfBirth(
     (int) json.get("yearOfBirth").isNumber().doubleValue());
contact.setAddress(
     parseAddress(json.get("address").isObject()));

JSONArray emailAddresses =
        json.get("emailAddresses").isArray();
     for (int i = 0; i < emailAddresses.size(); i++) {
        contact.getEmailAddresses().add(
     }
}
```

JSONValue

Good

- It’s standard
- Human readable
- Compact format

vaadin }>

JSONValue

Good

- It's standard
- Human readable
- Compact format

Bad

- Not completely typesafe
- Boilerplate
- Client only

vaadin

What about the server?

vaadin }>

Jackson on the server

```
ObjectMapper mapper = new ObjectMapper();
try {
    Contact contact = mapper.readValue(string, Contact.class);
} catch (VariousExceptions e) {
    // Do something sensible
}
```

Jackson and GWT?

- gwt-jackson

vaadin }>

gwt-jackson

```
public static interface ContactMapper extends
     ObjectMapper<Contact> {}

public Contact parseContact(String jsonString) {
    ContactMapper mapper = GWT.create(ContactMapper.class);
    Contact contact = mapper.read(jsonString);
    return contact;
}
```

Jackson

Good

- Minimal boiler plate
- Can share code between server and client

Bad

- Only creating and reading Strings

vaadin

Where to send?

vaadin }>

Remote procedure call

```
public interface ContactService extends RemoteService {
    public void saveContact(Contact contact);
    public List<Contact> getContacts();
}

public interface ContactServiceAsync {
    public void saveContact(Contact contact,
        AsyncCallback<Void> callback);
    public void getContacts(AsyncCallback<List<Contact>>
}
```

GWT-RCP

Good

- Simple but powerful concept
- Default solution
- Optimized

Bad

- Sending large object graph
- Polymorphism problems

vaadin

Prefer JSON + REST?

- me too :)

vaadin }>

REST

```
GET /contacts

DELETE /contacts/5

PUT /contacts { name: “John Doe”, ... }
```

vaadin

JAX-RS

```
@Path("/contacts")
public interface ContactsService {
    @GET
    public List<Contact> listContacts();

    @Path("/{id}")
    @DELETE
    public void deleteContact(@PathParam("id") int id);

    @PUT
    public void createContact(Contact contact);
}
```

vaadin

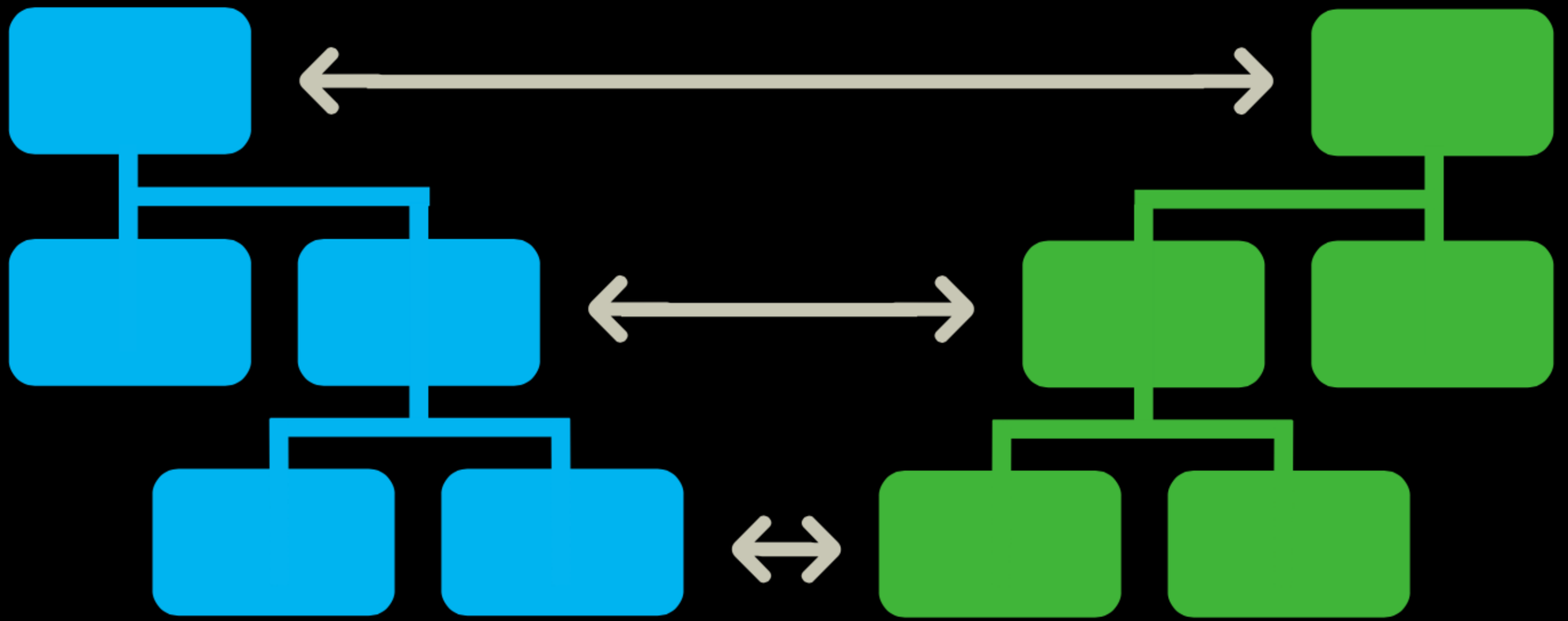
Vaadin Connectors

vaadin

So how about Vaadin?

- Put server in charge!

vaadin }>



vaadin

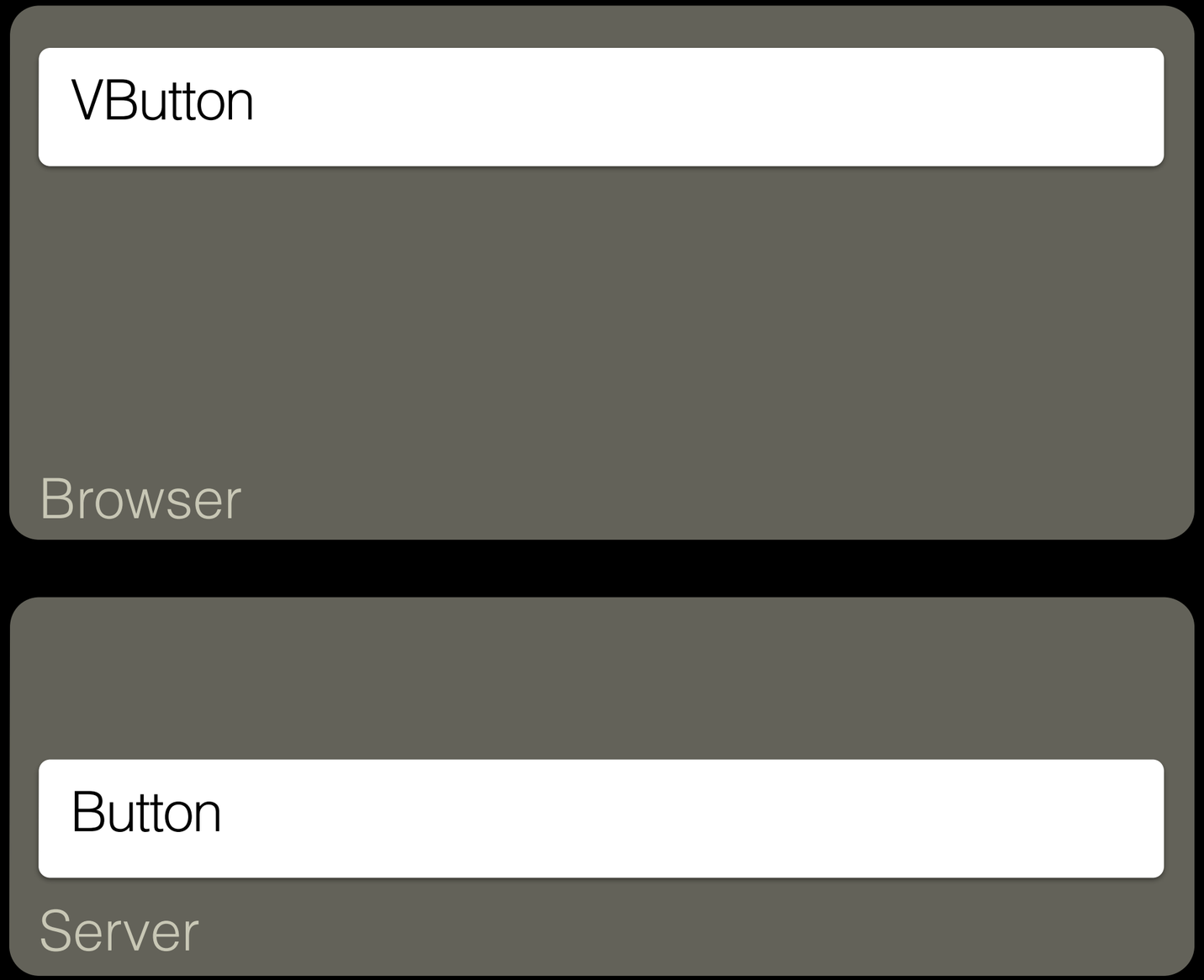
State synchronization

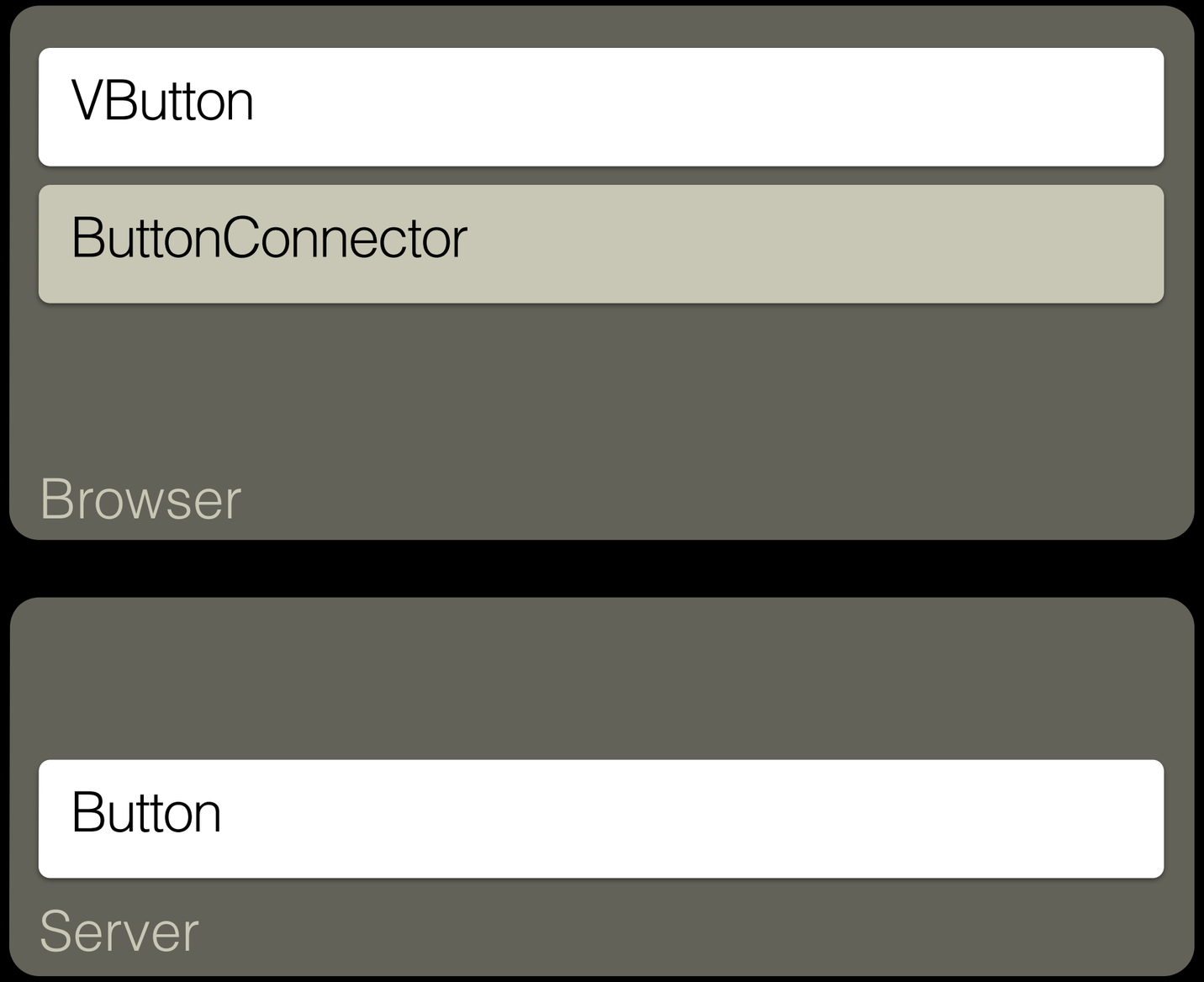
```
public class ContactState extends SharedState {
    public String name;

    @DelegateToWidget
    public int yearOfBirth;
}

@OnStateChange("name")
private void updateName() {
    doSomethingWithTheName(getState().name);
}
```

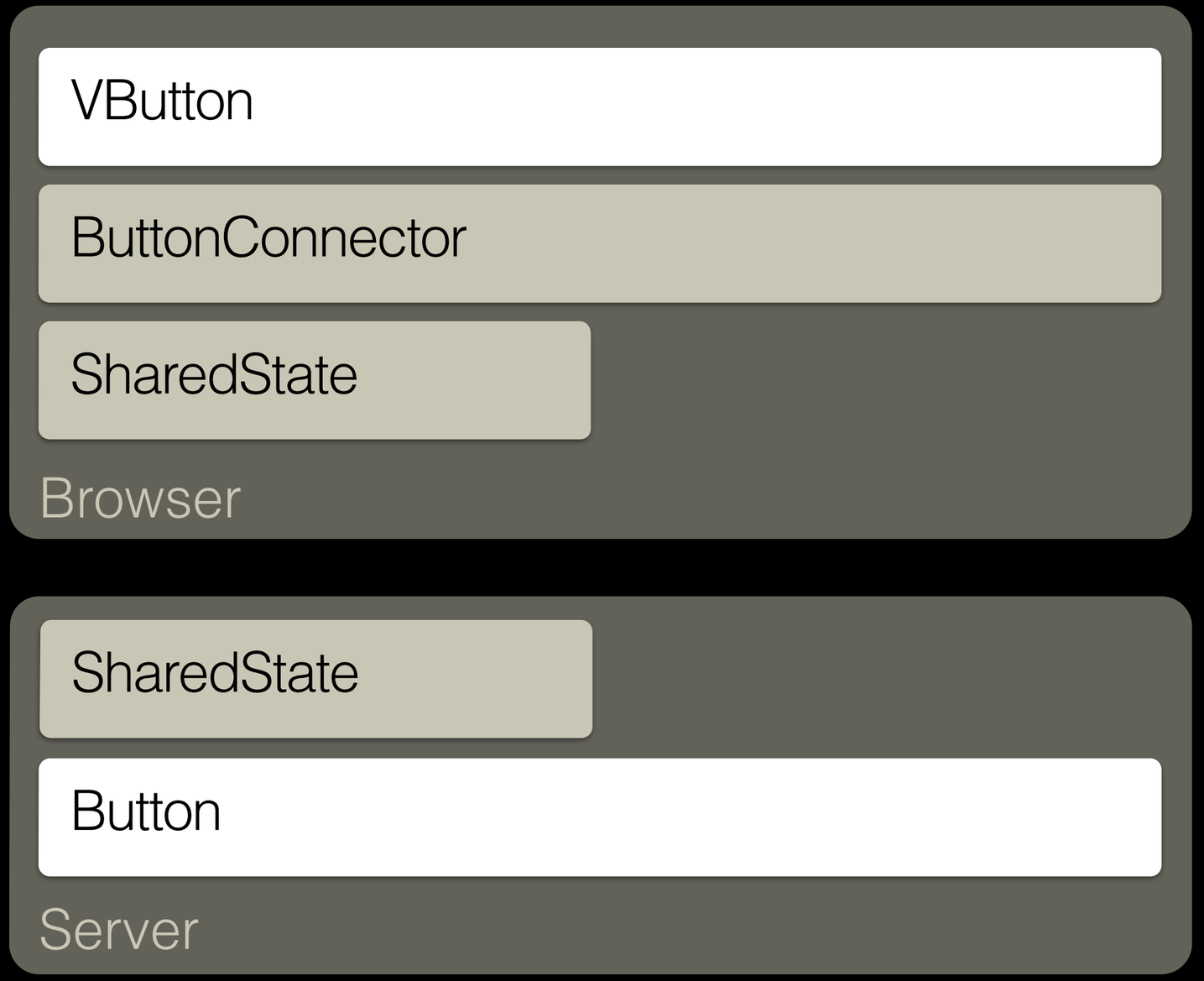
RPC

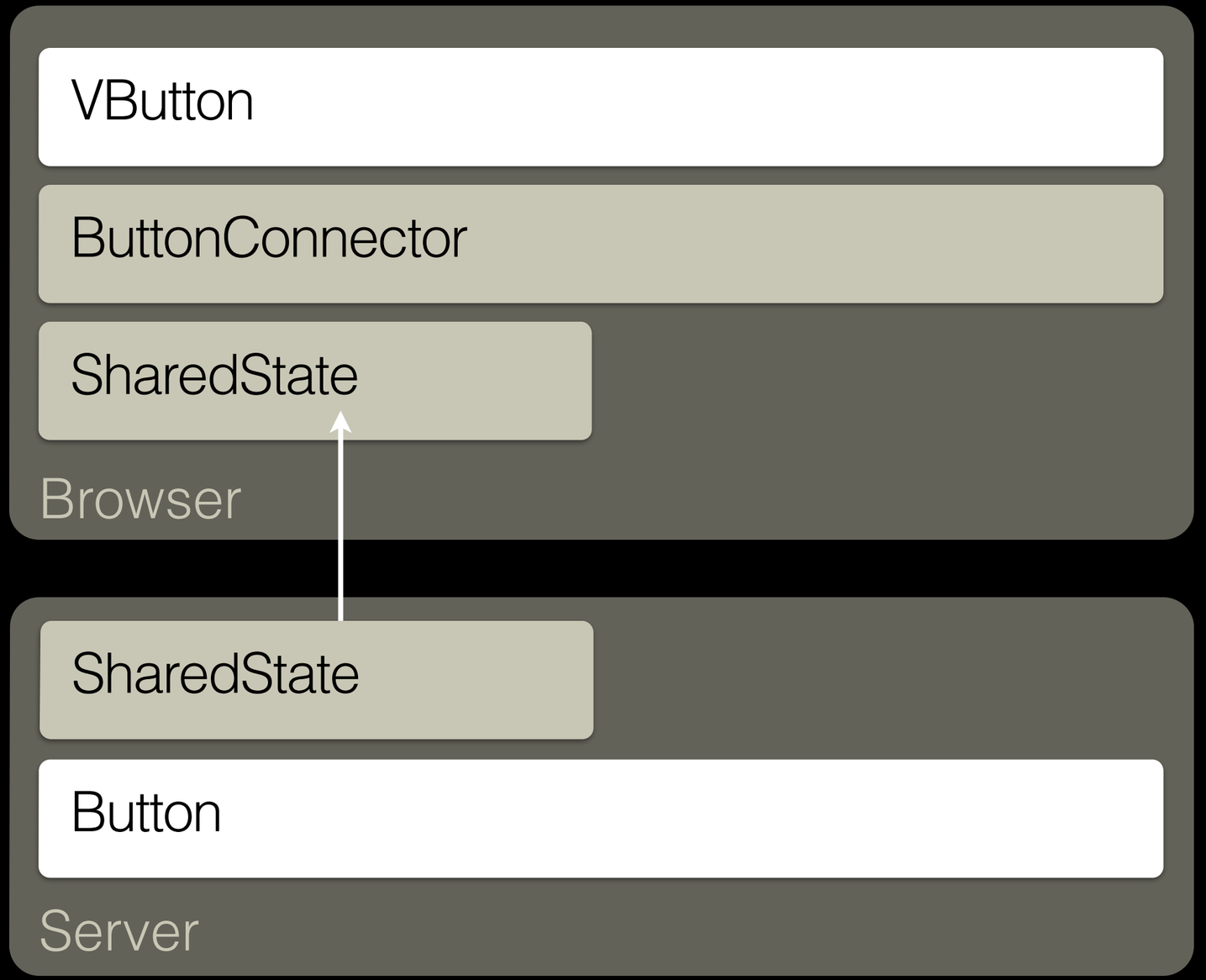
```
public interface ContactRpc extends ServerRpc {
    public void deleteContact(int id);
}
// Register RPC handler on the server
registerRpc(new ContactRpc() {
    @Override
    public void deleteContact(int id) {
        ContactDAO.deleteById(id);
} });
// Send RPC from the client
public void sendDelete(int contactId) {
    getRpcProxy(ContactRpc.class).deleteContact(contactId);
}
```

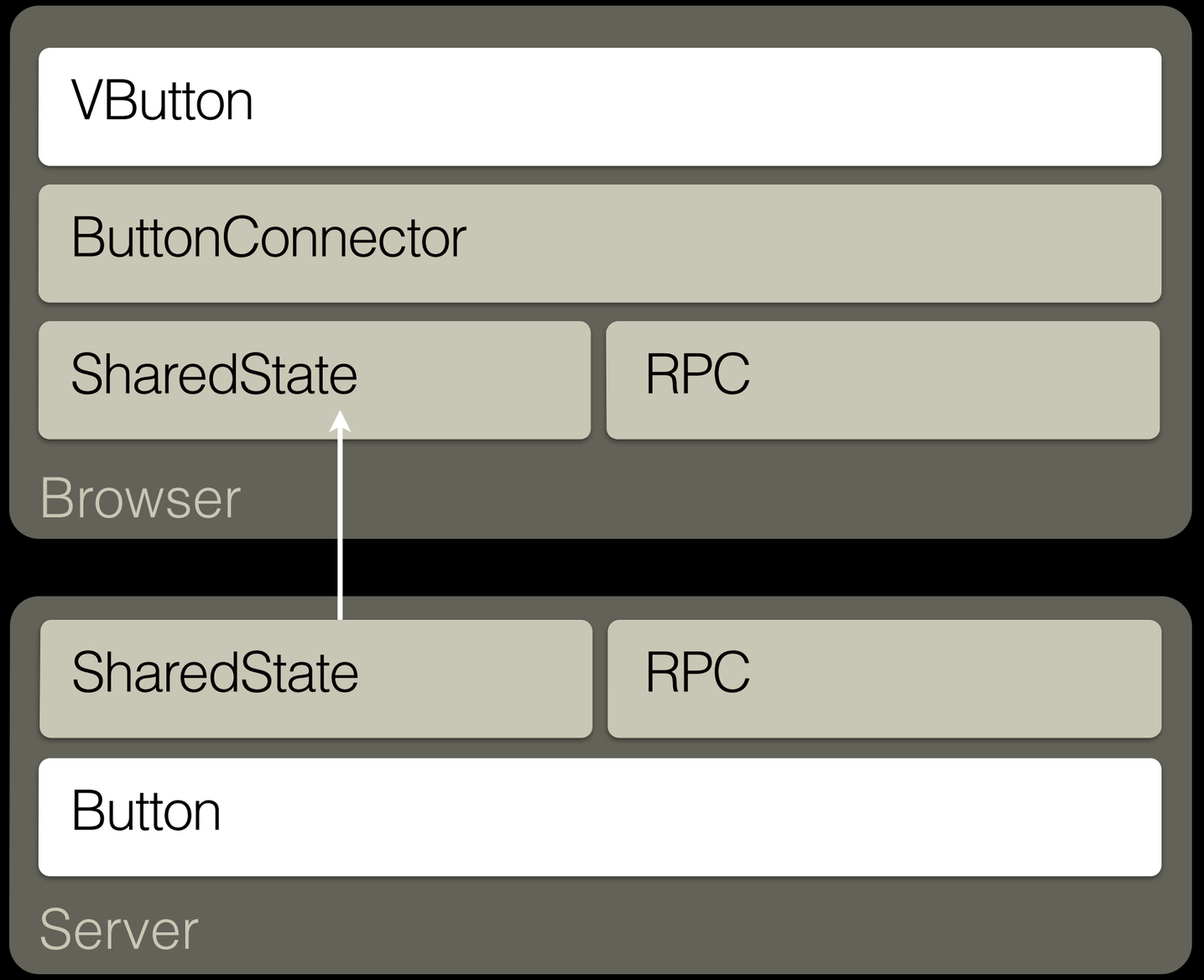
vaadin

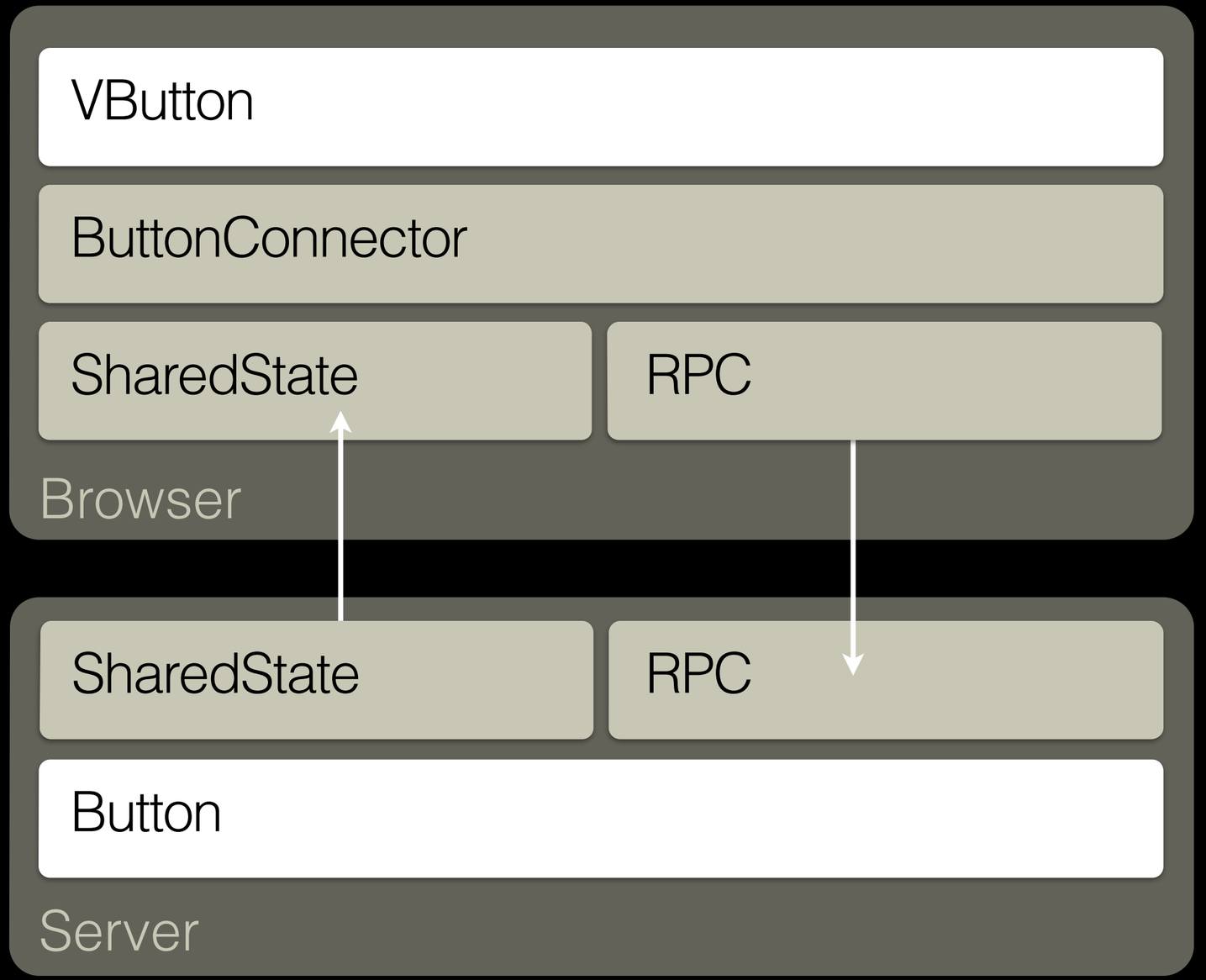

VButton
Browser
Button
Server

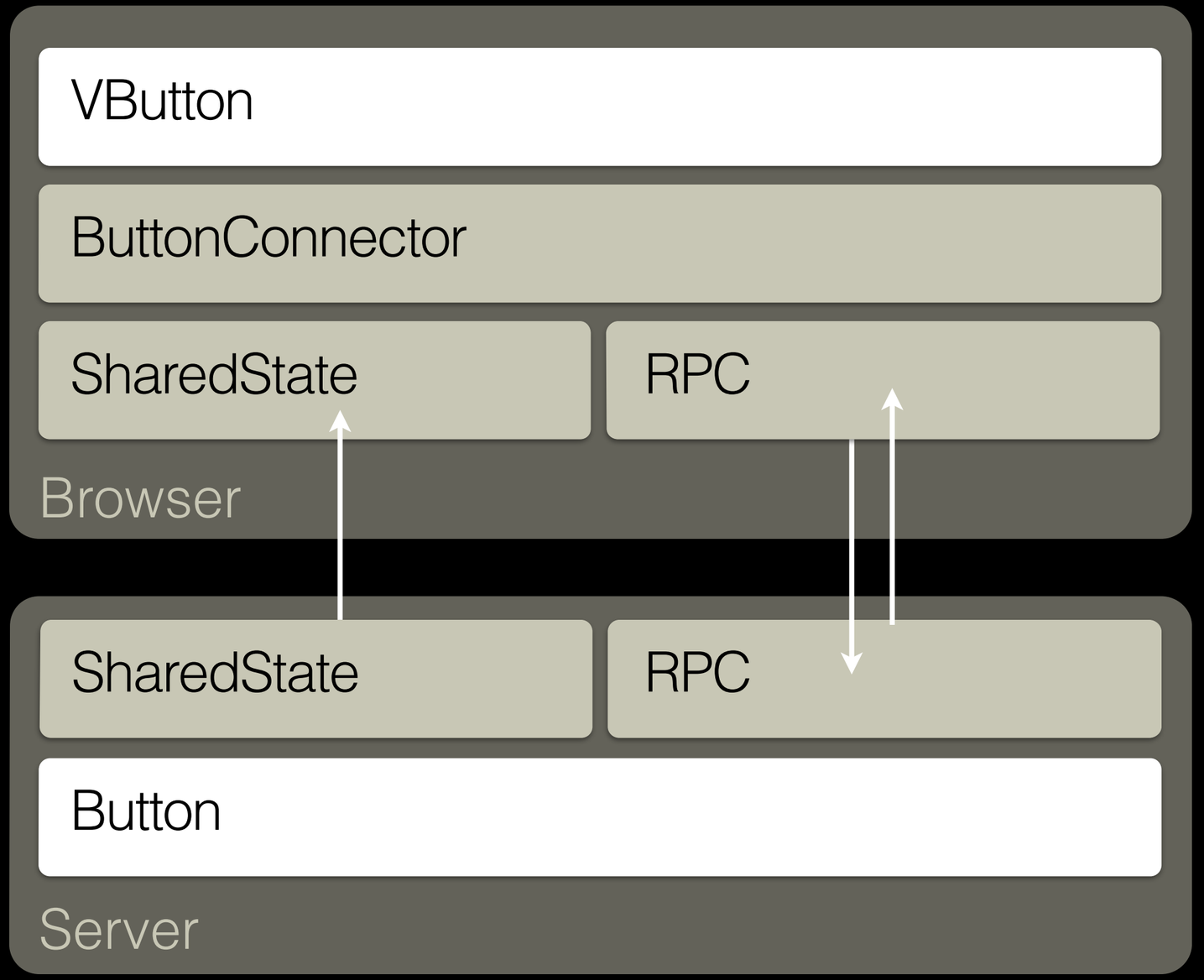

VButton
ButtonConnector
Browser
Button
Server

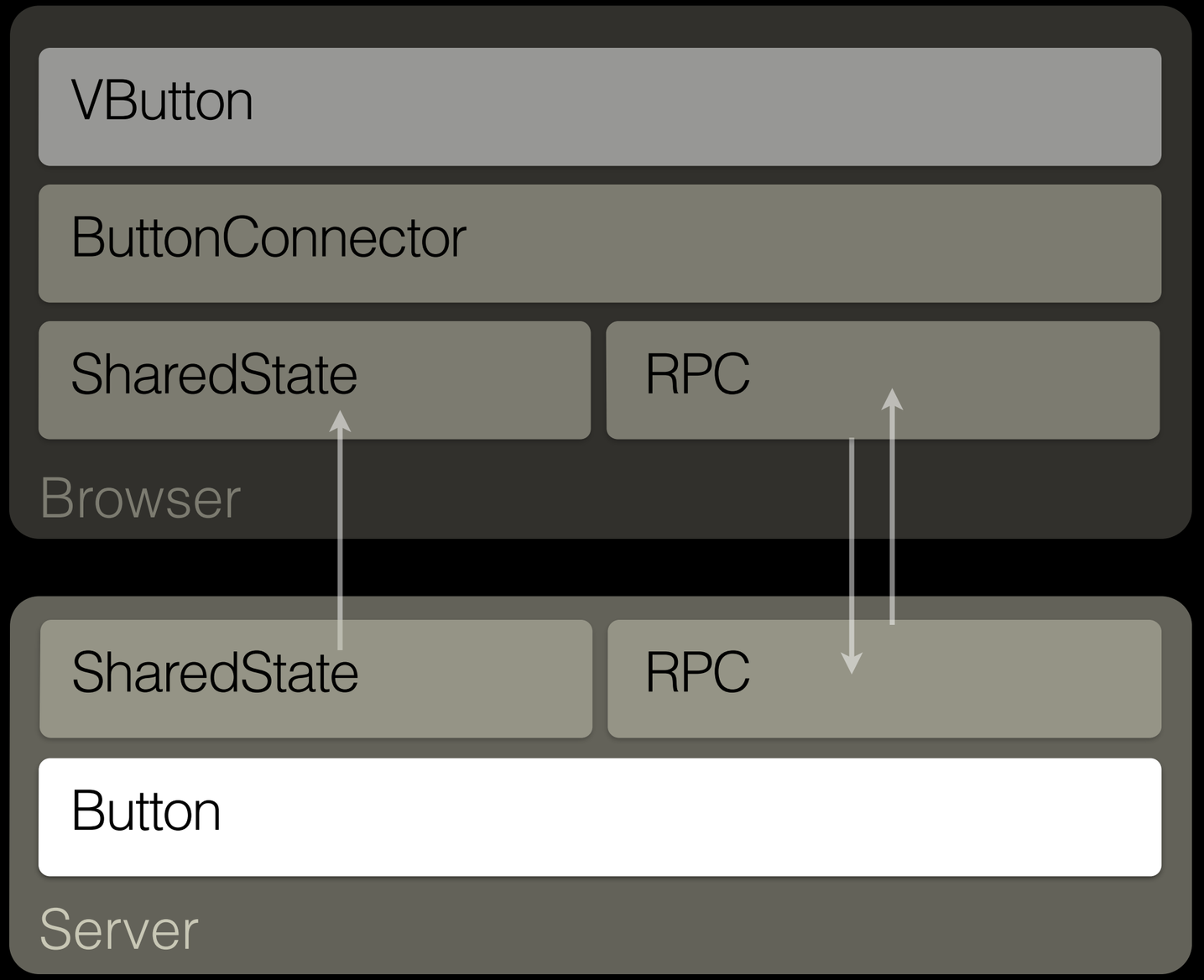

VButton
ButtonConnector
SharedState
Browser
SharedState
Button
Server

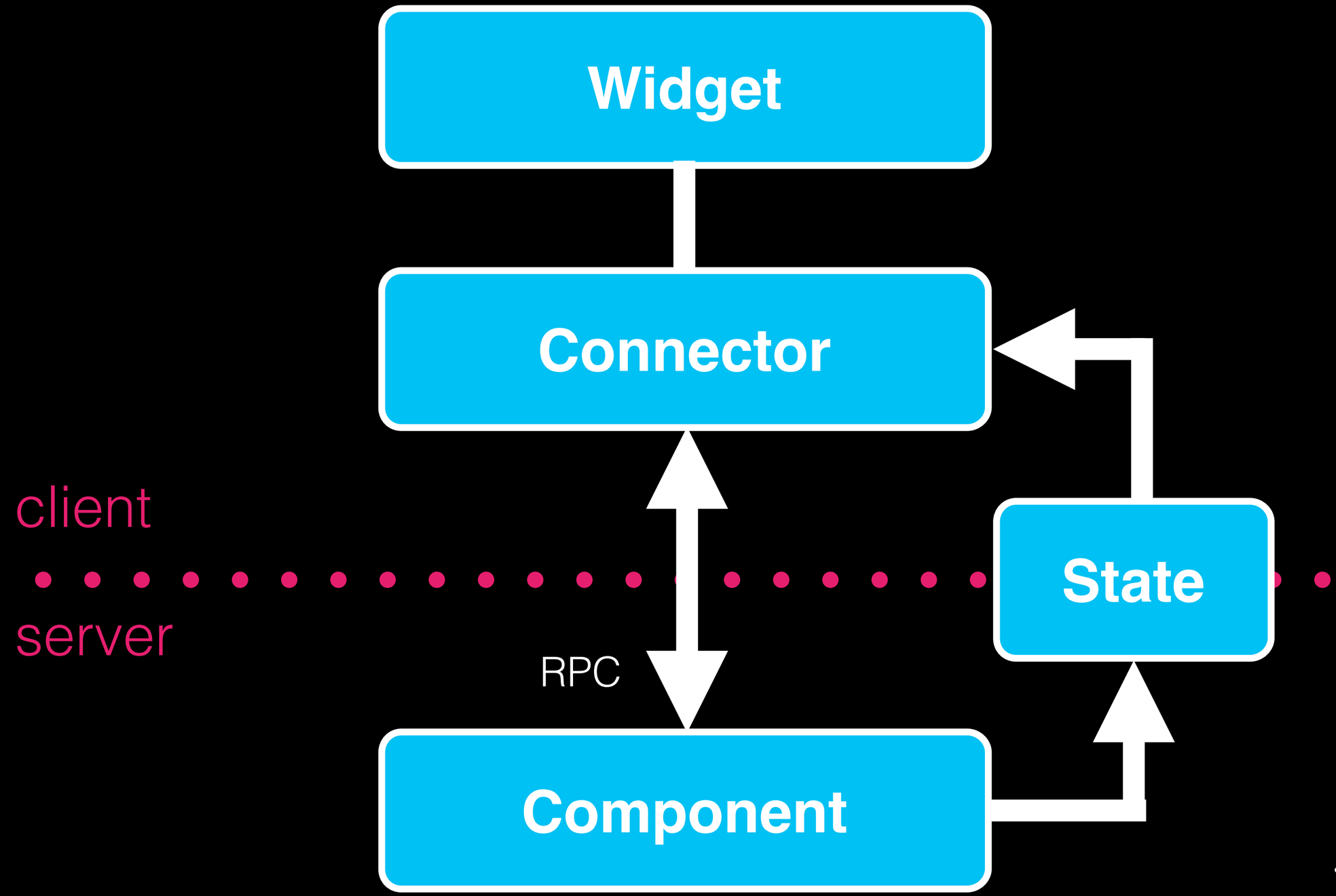

VButton
ButtonConnector
SharedState
Browser
SharedState
Button
Server


VButton
ButtonConnector
SharedState
RPC
Browser
SharedState
RPC
Button
Server


VButton
ButtonConnector
SharedState
RPC
Browser
SharedState
RPC
Button
Server


VButton
ButtonConnector
SharedState
RPC
Browser
SharedState
RPC
Button
Server


VButton
ButtonConnector
SharedState
RPC
Browser
SharedState
RPC
Button
Server


Widget
Connector
client
server
RPC
State
Component

ButtonConnector

```
@Connect(Button.class)
public class ButtonConnector extends AbstractComponentConnector
  implements ClickHandler, FocusHandler {

   public void init() {
      getWidget().addClickHandler(this);
   }

   @OnStateChange({ "caption", "captionAsHtml" })
   void setCaption() {
      getWidget().setCaptionText(getState());
   }
```

vaadin

```
    public void onClick(ClickEvent event) {
        getRpcProxy(ButtonServerRpc.class).
            click(buildDetails(event));
    }
}
```

ButtonState

```
public class ButtonState extends AbstractComponentState {
    …
    public String caption;
    public boolean disableOnClick;
    public int tabIndex;
    …
}
```

ButtonServerRPC

```
public interface ButtonServerRpc extends ServerRpc {

    public void click(MouseEventDetails mouseEventDetails);

}
```

Server RPC implementation

```
public class Button extends AbstractComponent {
    …
    private ButtonServerRpc rpc = new ButtonServerRpc() {
      public void click(MouseEventDetails details) {
         fireClick(details);
      }
    …
 }
```

Vaadin Connectors

Good

- Stateful server
- Websocket support
- Integrated JSON

Bad

- Stateful server
- Tied to framework

vaadin

Web components with Polymer

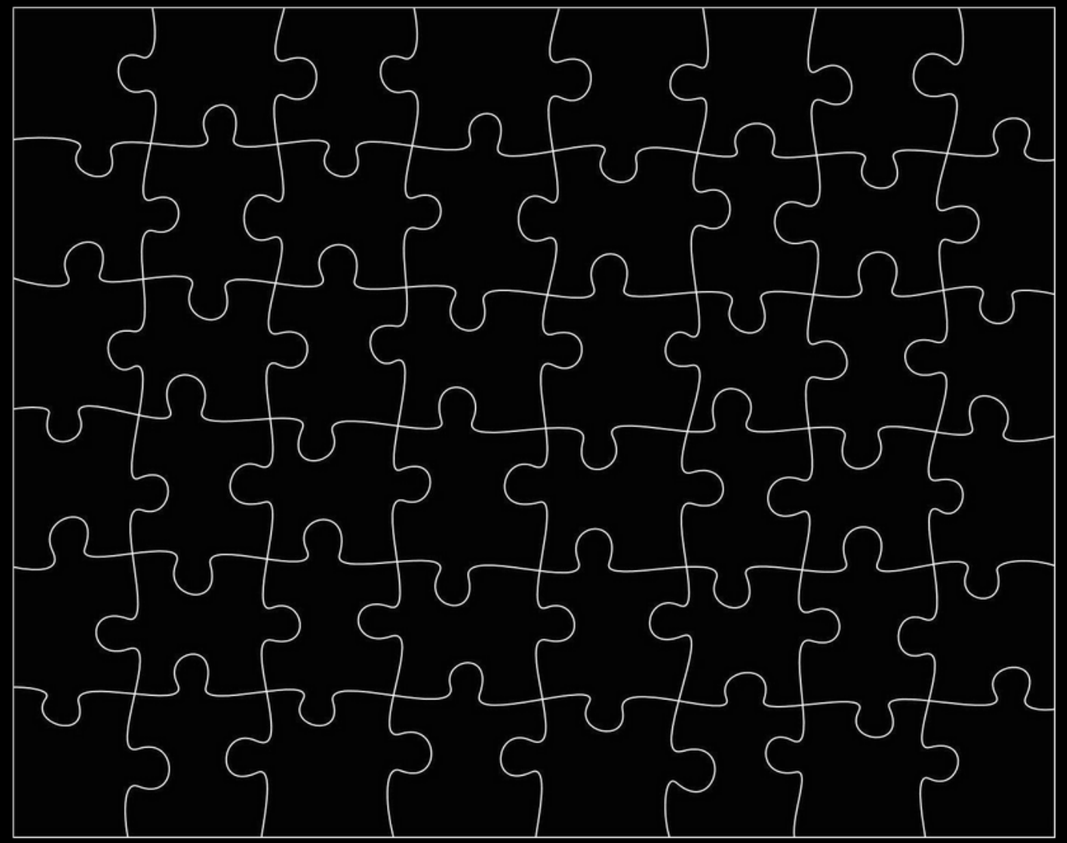
vaadin

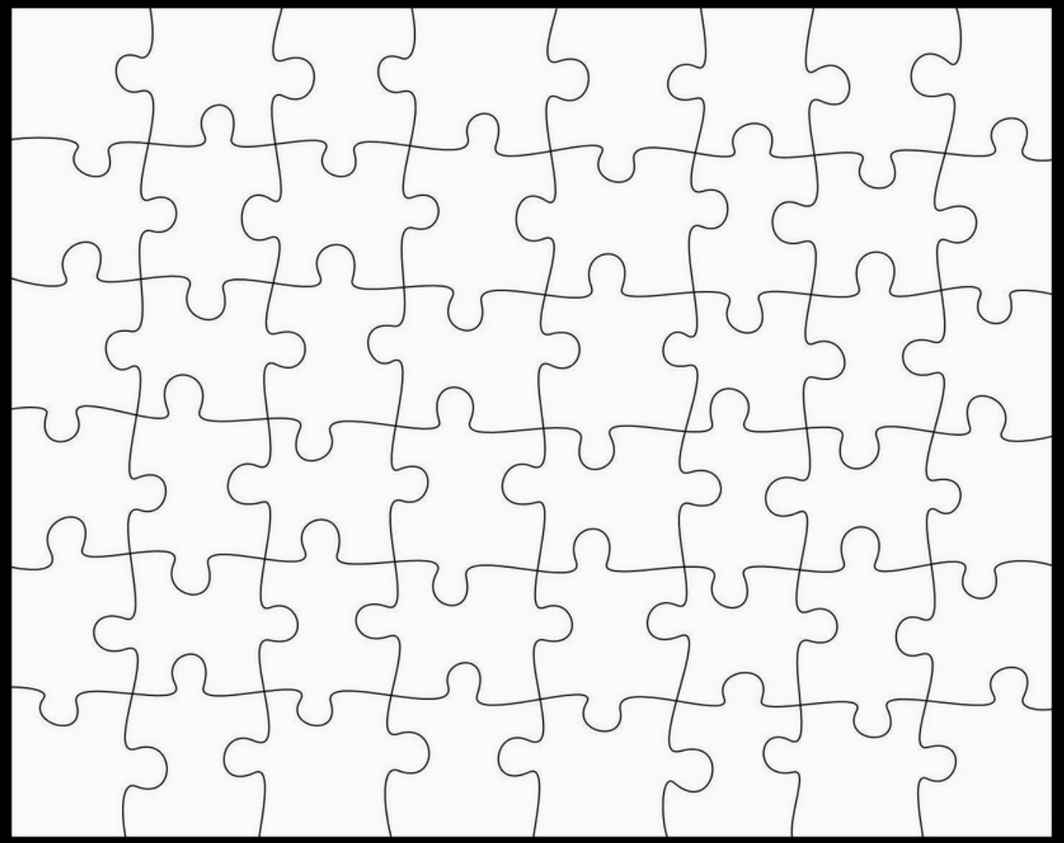
Reuseable HTML components!

vaadin }>

Problem: Unique DOM tree

```
body { background-color: white; }
```

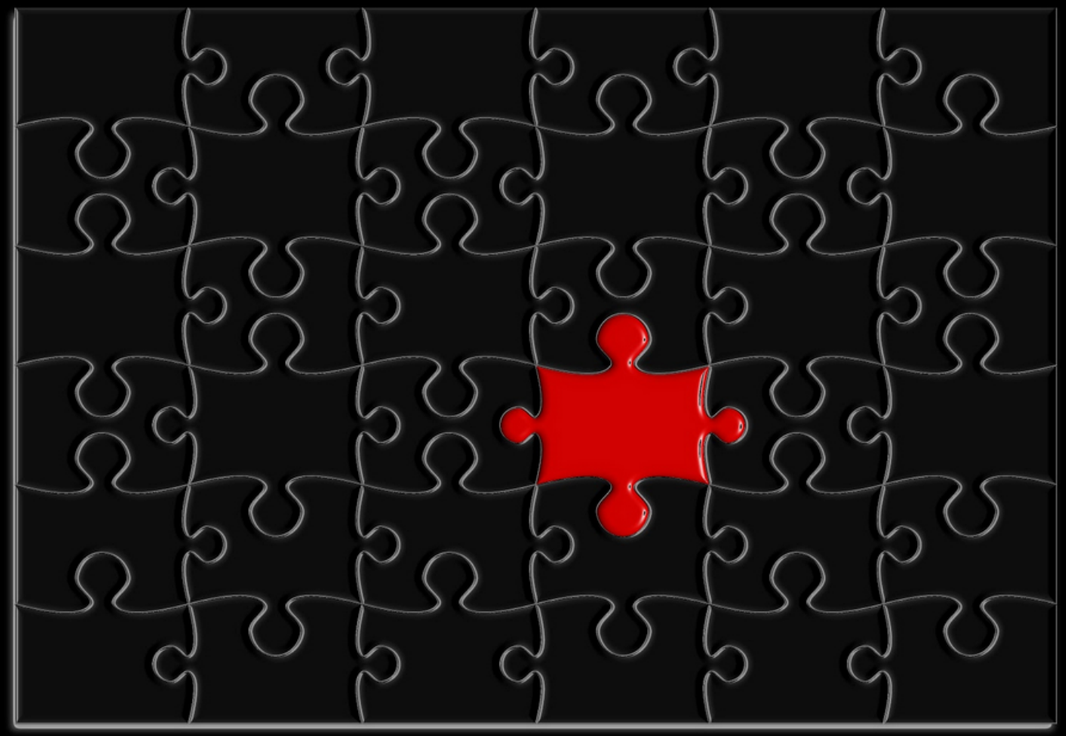


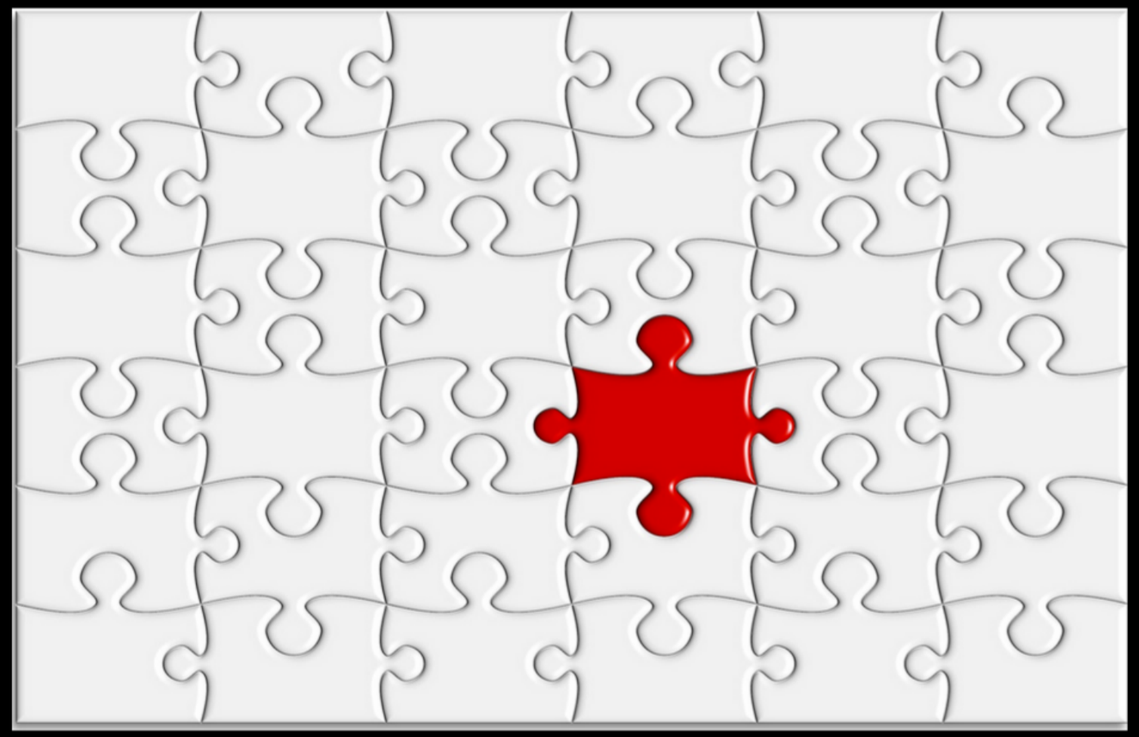


vaadin

Solution: Shadow DOM

```
body { background-color: white; }
```





vaadin }>

Encapsulation

vaadin

What shadow DOM looks like?


```
<v-slider value="33">
  #shadow-root
    <style>@import url('VAADIN/themes/valo/styles.css')</style>
    <div class="valo">
      <div tabindex="0" class="v-slider">
        <div class="v-slider-bigger" style="display: none;"></div>
        <div class="v-slider-smaller" style="display: none;"></div>
        <div class="v-slider-base">…</div>
      </div>
    </div>
</v-slider>
```

vaadin

How? With Polymer

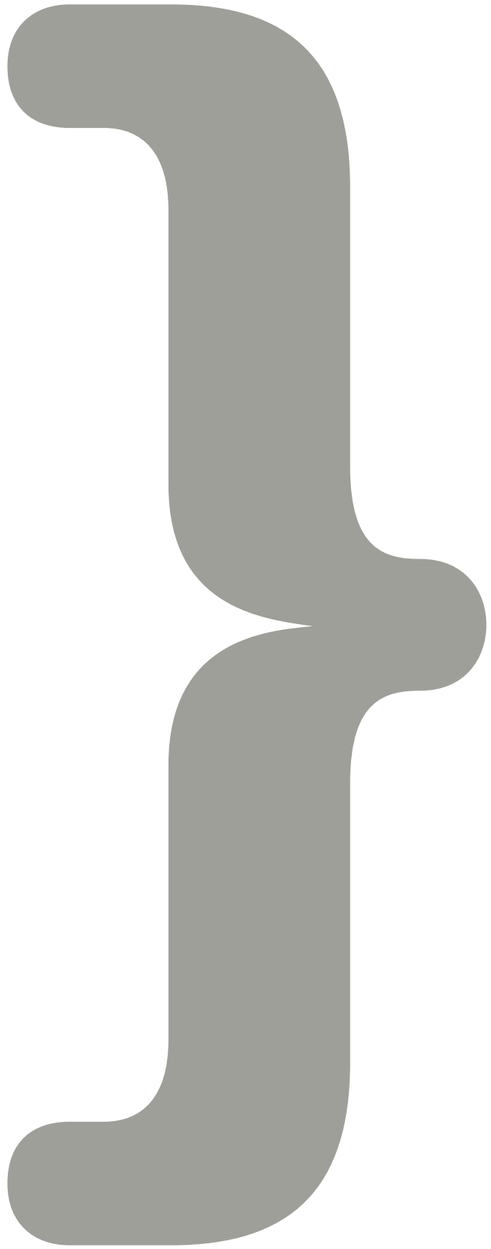
polymer-project.org/

vaadin }>

<v-grid>

vaadin }>

- Vaadin Grid Client Widget
- Expose JavaScript API via JsInterop from GWT 2.8
- Use Shadow DOM and HTML imports from Polymer



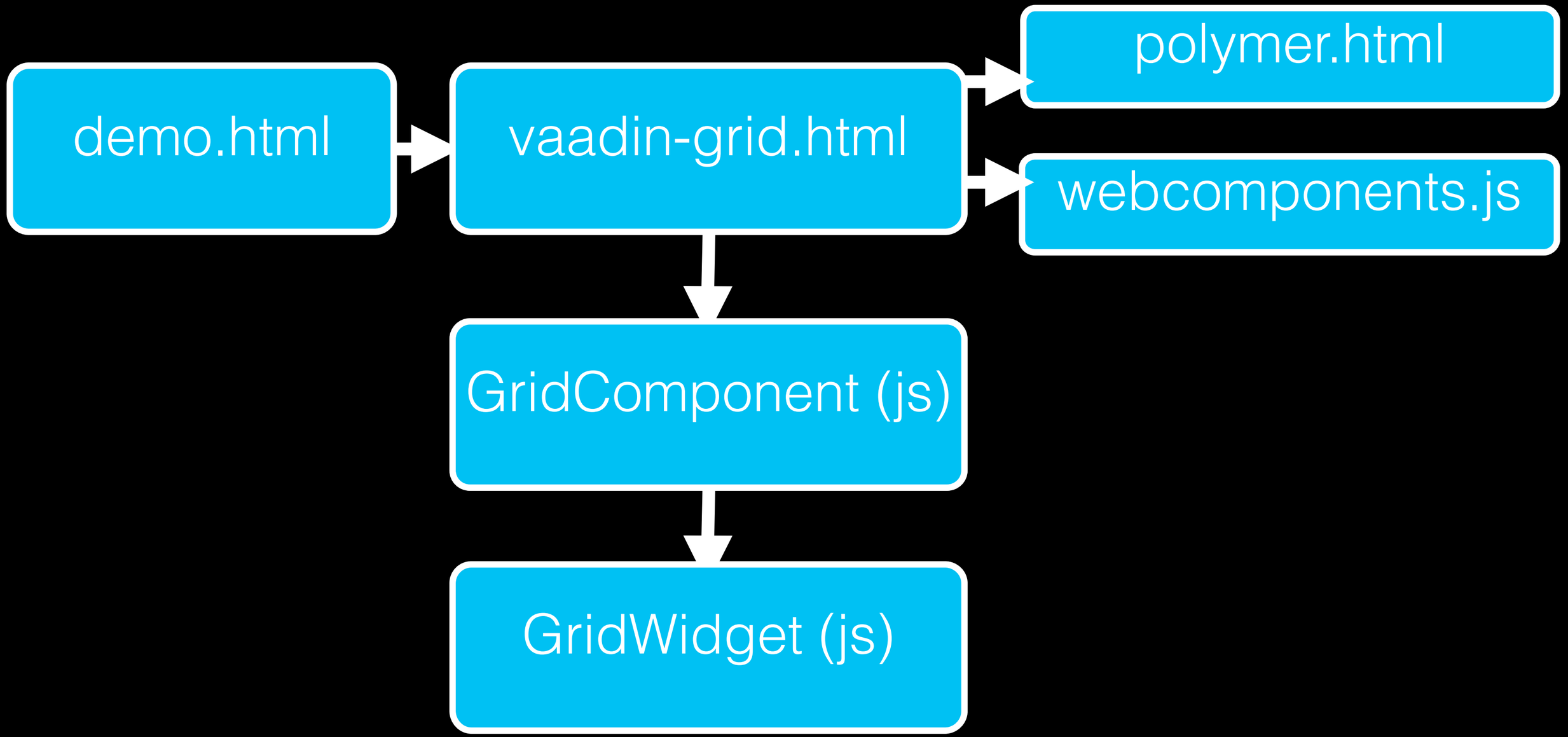
vaadin }>

```
@JsNamespace(JS.VAADIN_JS_NAMESPACE)
@JsExport
@JsType
public class GridComponent…
…
  private final Grid grid;  // Grid widget

  public JSColumn addColumn(JSColumn jsColumn, Object beforeColumnId) {
     grid.addColumn
  }
…
```

```java
@JsType
public interface JSColumn {

  @JsProperty
  String getName();

  @JsProperty
  void setName(String s);
```

```
<link rel='import' href='vaadin-grid-import.html'>
<link rel='import' href='../bower_components/polymer/polymer.html'>
<link rel="stylesheet" href="vaadin-grid.css" shim-shadowdom>

<dom-module id="v-grid">
  <template>
  </template>
</dom-module>

<script>
  var prototype = {
    is: “v-grid",
    properties: {
          …
    },
    created: function() {
       this._grid = new vaadin.GridComponent();
    },
    attached: function() {
       this._grid.attached(this, Polymer.dom(this).querySelector(“table”),
       Polymer.dom(this.root));

    },

     …
  }
</script>
```

vaadin


demo.html
vaadin-grid.html
polymer.html
webcomponents.js
GridComponent (js)
GridWidget (js)

Lessons learned today

1. There are several ways of communicating between GWT client and server
2. Vaadin Connectors use custom hybrid of JSON and RequestBuilder wrapped to higher level SharedState and RPC
3. gwt-jackson with REST is very interesting option
4. Polymer allows browsers to support features needed for for web components

JavaCro

Thank you!

Peter Lehto
@peter_lehto
expert & trainer
vaadin