

# *ISOMORPHIC JAVASCRIPT APPS WITH NASHORN AND REACT.JS*

*Niko Köbler*

*Software-Architect, Developer & Trainer*

*[niko@n-k.de](mailto:niko@n-k.de) | [www.n-k.de](http://www.n-k.de) | [@dasniko](https://twitter.com/dasniko)*

*JAVA*

*IS TO*

*JAVASCRIPT*

*AS HAM IS TO HAMSTER.*

# ATWOOD'S *LAW*

*Any application that can be written in  
JavaScript, will eventually be written in  
JavaScript*

Jeff Atwood

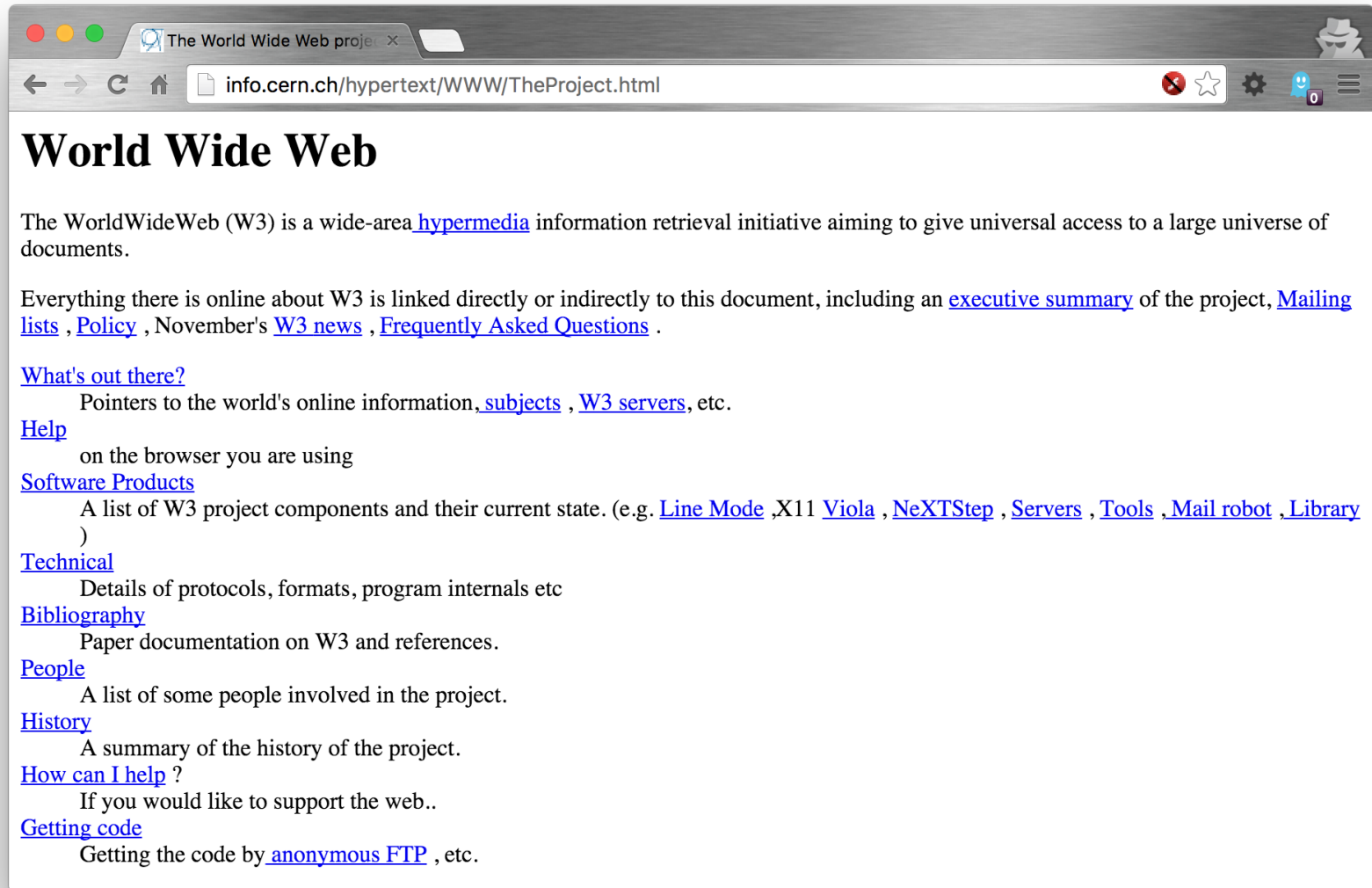
<http://blog.codinghorror.com/the-principle-of-least-power>

# *JAVASCRIPT IS THE LANGUAGE OF THE WEB!*

*And thus the platform for building  
rich and highly interactive web apps.*

*THE WEB HAS EVOLVED  
THROUGH THE YEARS*

*Where documents were linked statically together...*



<http://info.cern.ch/hypertext/WWW/TheProject.html>

...through weird Ajax sites...





...to highly functional Single-Page-Apps



*JAVASCRIPT HAS  
BECOME THE  
APP RUNTIME  
ENVIRONMENT*

*(in the browser)*

# *JAVASCRIPT IS EVERYWHERE!*

- *Browsers*
- *Servers*
- *IoT/Embedded devices*
- *Mobiles/Smartphones/Tablets*
- *TV's*
- *Cars*
- *you name it...*

# *ISOMORPHIC*

*/ˌaɪ.səʊ'mɔːf.ɪk/*

*greek: "isos" = equal, "morph" =  
shape*

# ISOMORPHIC

## MATHS

*Related by an isomorphism; having a structure-preserving one-to-one correspondence.*

## BIOLOGY

*Having a similar structure or function to something that is not related genetically or through evolution.*

<https://en.wiktionary.org/wiki/isomorphic>

# ISOMORPHIC *JAVASCRIPT*

*If you look at the same entity (*code*) in two different contexts (*client & server*), you should get the same thing (*result, html, DOM, ...*).*

*ISOMORPHIC JAVASCRIPT*

*aka*

*UNIVERSAL JAVASCRIPT*

# *USE CASES*

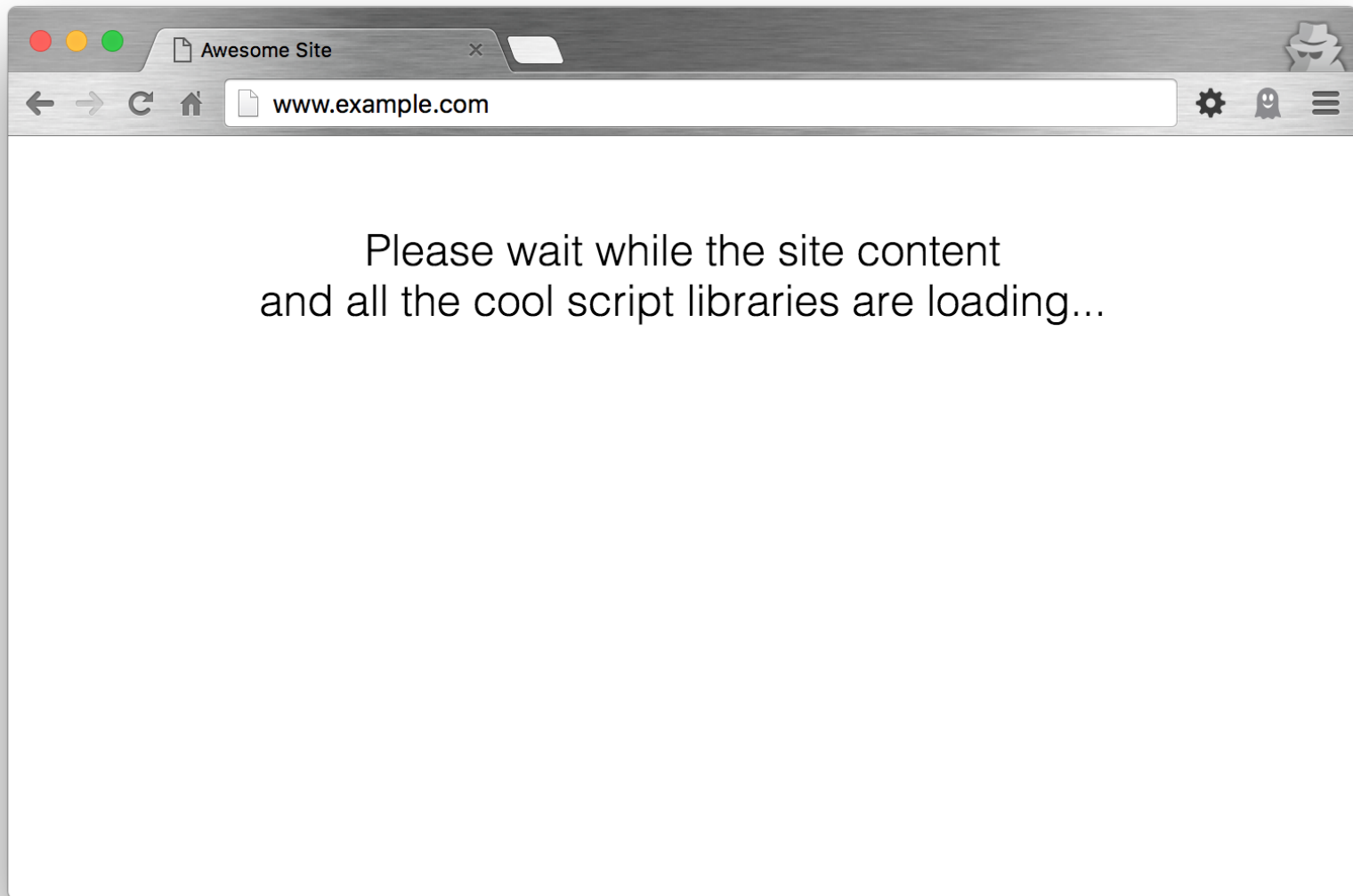
- *l18n*
- *Validation*
- *Templating / UI*
- *Routing*



# WHY *SAME CODE* ON CLIENT AND SERVER!?

- *DRY principle*
- *share same logic*
- *one codebase / maintenance*
- *single point of truth (or failure)*
- *single technology*

*WHY RENDER ON THE SERVER?*

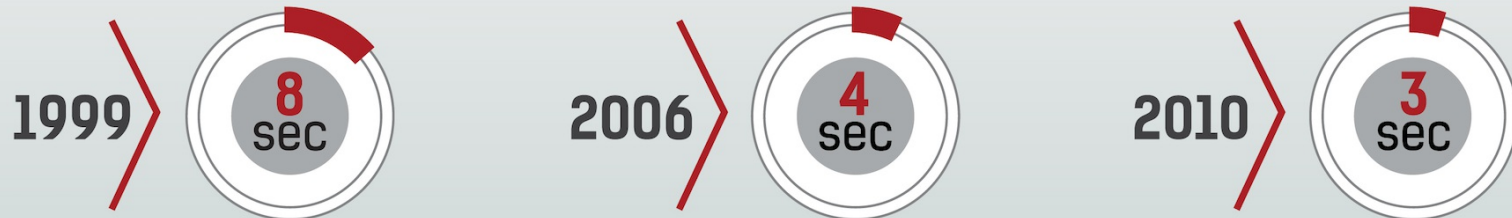


# WHY RENDER ON THE *SERVER*?

*b/c wait time matters!*

## INTERNET USERS ARE INCREASINGLY DEMANDING

In 1999, the average user was willing to wait 8 seconds for a page to load. By 2010, **57% of online shoppers said they would abandon a page after 3 seconds.**



<http://www.radware.com>

# WHY RENDER ON THE *SERVER*?

*this is how Google sees your site:*

```
<html>
<head>
  <title>Awesome Site</title>
  <script src="/app-bundle.js"></script>
</head>
<body></body>
</html>
```

# *WHY RENDER ON THE **SERVER**?*

- *UX*
- *SEO*
- *Performance*
- *Legacy Browser Support*

# *WHY RENDER ON THE CLIENT?*

- *UX*
- *Performance*

# *ISOMORPHIC FTW!*

- 1. User requests URL*
- 2. Server fetches content for that URL*
- 3. Server renders content to a response*
- 4. User enjoys the content*
- 5. Client is initialized*
- 6. User navigates to a different URL*
- 7. Client fetches content for that URL*
- 8. Client renders content to the DOM*



*ISOMORPHIC WEB APPS*  
*SOUND AWESOME,*  
*BUT...*

*I'M IN A JAVA  
ENVIRONMENT!*

## *Option 1*

# *DELEGATE EXECUTION OF JAVASCRIPT TO AN EXTERNAL **NODE.JS** PROCESS*

*more complicated deployment*

*performance overhead of interacting with an external process*

## *Option 2*

*HAVE **NODE.JS** RUNNING AS A **SMART-PROXY** IN FRONT OF JAVA*

*more complicated deployment*

*performance overhead of interacting with an external process*

*Option 3*

*RUN JAVASCRIPT ON  
THE JVM WITH NASHORN*

# NASHORN

- *JavaScript Engine on the JVM since Java 8*
- *ECMAScript 5.1 compatibility*
  - *plus language and api extensions*
- *ES6/ES2015 features come with Java 9*
  - *backports available*
- *compiles JavaScript to Java Bytecode*
  - *providing interoperability of Java and JavaScript*
- *Shell scripting mode*

# INVOKING *JAVASCRIPT* FROM JAVA

```
ScriptEngineManager manager = new ScriptEngineManager();
ScriptEngine engine = manager.getEngineByName("nashorn");
engine.eval("print('Hello Nashorn');");
> Hello Nashorn
```

```
...
engine.eval(new FileReader("script.js"));
Invocable invocable = (Invocable) engine;

Object result = invocable.invokeFunction("sayHello", "Nashorn");
System.out.println(result);
> Hello Nashorn
```

*script.js:*

```
var sayHello = function(name) {
    return 'Hello ' + name;
}
```

# INVOKING *JAVA* FROM JAVASCRIPT

```
package my.package;
public class MyJavaClass {
    public String sayHello(String name) {
        return String.format("Hello %s from Java!", name);
    }
}
```

```
var MyJavaClass = Java.type('my.package.MyJavaClass');
var myClass = new MyJavaClass();
var result = myClass.sayHello('Nashorn');
print(result); // Hello Nashorn from Java!
```



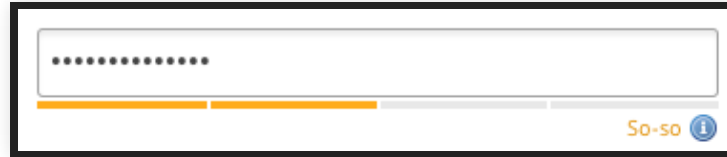
# NASHORN & CONCURRENCY

- *Browsers: no simultaneous execution of JavaScript code*
- *Nashorn itself is not thread-safe by design*
- *Thread-safety depends on your code!*
- *Use a `ThreadLocal<ScriptEngine>` when your JS code is not thread-safe (i.e. React.js, Handlebars, etc.)*

# NASHORN-POLYFILL.JS

```
var global = this;  
var window = this;  
var process = {env:{}};  
  
var console = {};  
console.debug = print;  
console.warn = print;  
console.log = print;
```

# EXAMPLE: PASSWORD *VALIDATION*



*JavaScript code:*

```
function isPasswordValid(password) {  
    var score = scorePasswordStrength(password);  
    return score >= 3;  
}
```

*Java code:*

```
public Boolean isPasswordValid(String password) {  
    try {  
        return (Boolean) nashorn.invokeFunction("isPasswordValid", pas  
    } catch (ScriptException | NoSuchMethodException e) {  
        throw new RuntimeException(e);  
    }  
}
```

# REACT.JS

- *invented by Facebook*
- *JavaScript library for building user interfaces*
- *component based*
- *not a full-stack framework*
- *virtual DOM (updates via diffs - no flickering)*
- *supports server-side rendering*

*"The new star on the web-dev  
heaven"*

<http://reactjs.org>

*JSX*

# JSX

```
var Book = React.createClass({
  render: function () {
    return (
      <div className="book">
        <h3>{this.props.author}</h3>
        <div className="lead">{this.props.children.toString()}</div>
      </div>
    );
  }
});
```

# JS

```
var Book = React.createClass({displayName: "Book",
  render: function () {
    return (
      React.createElement("div", {className: "book"},
        React.createElement("h3", null, this.props.author),
        React.createElement("div", {className: "lead"}, this.props
      )
    );
  }
});
```

# HTML

```
<div class="book" data-reactid=".1c3dv7jhtco.1.$0">  
  <h3 data-reactid=".1c3dv7jhtco.1.$0.0">George Orwell</h3>  
  <div class="lead" data-reactid=".1c3dv7jhtco.1.$0.1">1984</div>  
</div>
```

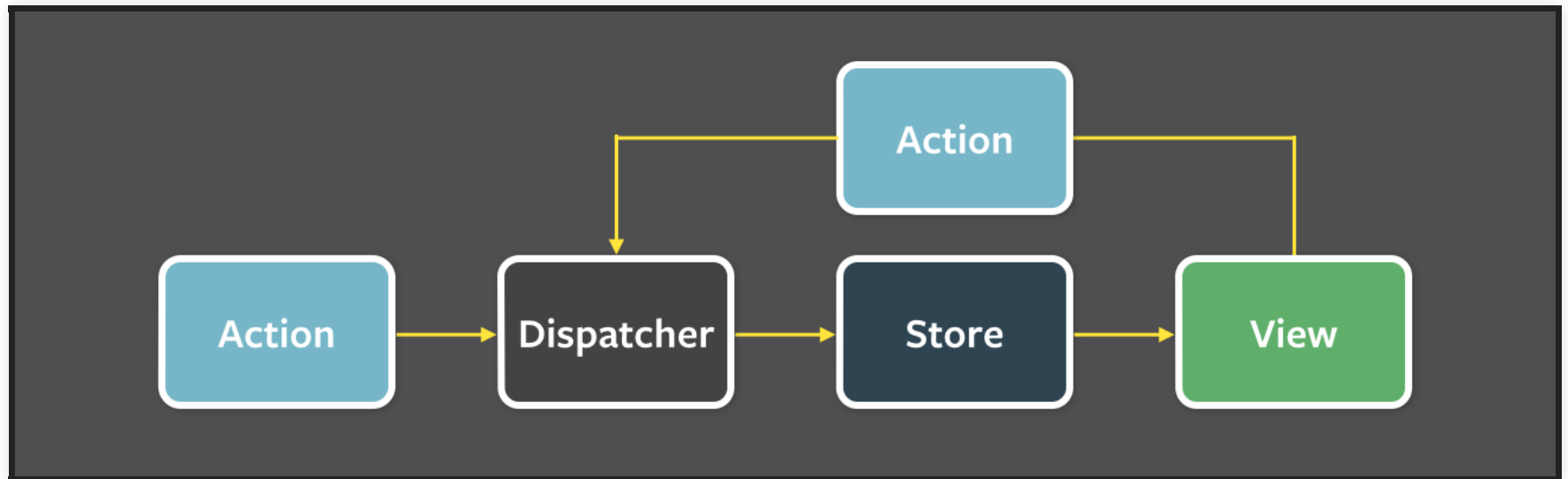
---

*Transpile JSX to JS with Babel*

[\*https://babeljs.io/\*](https://babeljs.io/)

*formerly: JSXTransformer*

# *FLUX*



[\*https://facebook.github.io/flux\*](https://facebook.github.io/flux)

## *Frameworks:*

*Facebook Flux, Fluxible by Yahoo, Reflux, Alt, Flummox, Marty.js, McFly, Lux, Material Flux, Redux, Nuclear.js, Fluxette, Fluxxor, Freezer, etc...*



*DEMO*

# *SPRING **BOOT** MVC*

*<https://github.com/sdeleuze/spring-react-isomorphic>*

*<https://github.com/winterbe/spring-react-example>*

# *JAVA **EE** 8 - MVC 1.0*

*Action-based Web-Framework*

JSR-371

[\*https://jcp.org/en/jsr/detail?id=371\*](https://jcp.org/en/jsr/detail?id=371)

[\*https://mvc-spec.java.net/\*](https://mvc-spec.java.net/)

OZARK (RI)

[\*https://ozark.java.net/\*](https://ozark.java.net/)

# *JAVA **EE** 8 - MVC 1.0*

## *OZARK-**REACT***

*ReactJS-based ViewEngine*

*<https://github.com/dasniko/ozark-react>*

*<https://github.com/dasniko/ozark-react-example>*

# MVC REACT CONTROLLER

```
@Controller
@Path("/react")
public class ReactController {
    @Inject
    private Models models;

    @Inject
    private BookService service;

    @GET
    public String index() throws Exception {
        List<Book> books = service.getBooks();
        models.put("data", books);
        return "react:react.jsp?function=renderServer";
    }
}
```

<https://github.com/dasniko/ozark-react-example/blob/master/src/main/java/dasniko/ozark/react/ReactController.java>

# REACTVIEWENGINE

```
public class ReactViewEngine extends ServletViewEngine {
    @Override
    public void processView(ViewEngineContext context) throws ViewEn
        // parse view and extract the actual template and the react.js
        String view = context.getView(); // react:react.jsp?function=r
        String template = view.substring("react:".length(), view.index
        String function = view.substring(view.indexOf("function=") + 9

        // get "data" from model
        Models models = context.getModels();
        Object data = models.get("data");

        // call given js function on data
        String content = react.render(function, data);

        // and put results as string in model
```

<https://github.com/dasniko/ozark-react/blob/master/src/main/java/dasniko/ozark/react/ReactViewEngine.java>

# REACT RENDERER

```
private ThreadLocal<ScriptEngine> engineHolder = ThreadLocal.withI
    ScriptEngine engine = new ScriptEngineManager().getEngineByName(
        engine.eval(read("nashorn-polyfill.js"));
        engine.eval(read("META-INF/resources/webjars/react/0.14.2/react.
        engine.eval(read("bookBox.js"));
        return engine;
    });
```

```
public String render(String function, Object object) {
    Object html = engineHolder.get().invokeFunction(function, object
    return String.valueOf(html);
}
```

<https://github.com/dasniko/ozark-react/blob/master/src/main/java/dasniko/ozark/react/React.java>

# JSX CODE

```
...  
var renderClient = function (books) {  
  var data = books || [];  
  React.render(  
    <BookBox data={data} url='books.json' pollInterval={5000} />,  
    document.getElementById("content")  
  );  
};  
  
var renderServer = function (books) {  
  var data = Java.from(books);  
  return React.renderToString(  
    <BookBox data={data} url='books.json' pollInterval={5000} />  
  );  
};
```

<https://github.com/dasniko/ozark-react-example/blob/master/src/main/resources/jsx/bookBox.js>



# JSP

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>ReactJS Bookstore with Ozark</title>
  <script src="/webjars/react/0.14.2/react.min.js"></script>
</head>
<body>
  <div id="content" class="container">${content}</div>
  <script type="text/javascript" src="/js/bookBox.js"></script>
  <script type="text/javascript">
    $(function () {
      renderClient(${data});
    });
  </script>
</body>
</html>
```

<https://github.com/dasniko/ozark-react-example/blob/master/src/main/webapp/WEB-INF/views/react.jsp>

# #ISSUES

- *slow creation/initialization of new Nashorn engines for each request (due to non-thread-safety of React.JS)*
    - *perhaps handle via an engine-pool in production*
  - *duplication of code (library loading) in client & server*
- 

*Contributions, PRs and Bug-Reports/-Fixes are  
explicitly welcome and highly appreciated!*

[\*https://github.com/dasniko/ozark-react\*](https://github.com/dasniko/ozark-react)

# *WHAT'S NEXT?*

- *ES6 (Java 9)*
- *TypeScript*
- *Angular 2*
- *Routing*

# CONCLUSION

- JavaScript *evolves*
- JavaScript frameworks becoming more mature  
(React, Angular 2, Ember 2)
- Isomorphic Applications help you to combine the best of both worlds: *server-side rendering and client-side RIA*
- Solutions for Java ecosystems are available  
(Spring Boot MVC, Java EE MVC)

*THANK YOU!*

*ANY QUESTIONS?*

*Niko Köbler*

*Software-Architect, Developer & Trainer*

*[niko@n-k.de](mailto:niko@n-k.de) | [www.n-k.de](http://www.n-k.de) | [@dasniko](https://twitter.com/dasniko)*

*<https://github.com/dasniko/ozark-react>*