Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет инженерно-экономический

Кафедра экономической информатики

КОНТРОЛЬНАЯ РАБОТА

по дисциплине

«САиПИС (с РВР)»

Выполнил: Проверила:

студент гр 572303 ст. преподаватель

Велюгин Е.А. Атрощенко Н. А.

Минск 2017

**ИНСТРУМЕНТАРИИ, ИСПОЛЬЗОВАННЫЕ В РАБОТЕ**

1.Блочная верстка

1.1 Использование блочной верстки с использованием 2-3 видов позиционирования

<div id="auth-section">  
 <% if(currentUser) { %>  
 <nav>  
 <a class="dropdown-toggle" href="#" title="Menu"><span id="user"><%= currentUser %></span></a>  
 <ul class="dropdown">  
 <li><a href="/schedule">TV Schedule</a></li>  
 <li><a href="/logout">Log Out</a></li>  
 </ul>  
 </nav>  
 <% } else { %>  
 <ul id="auth">  
 <li><a href="/signup" onclick="return makeSignUpSectionVisible()">Sign up </a></li>  
 <li><a href="/login" onclick="return makeLogInSectionVisible()">Log In</a></li>  
 </ul>  
 <% } %>  
</div>

header div#auth-section {  
 position: absolute;  
 left: 80%;  
}

<button id="history-button" onclick="return readJSON()">Show Boston Celtics title history</button>

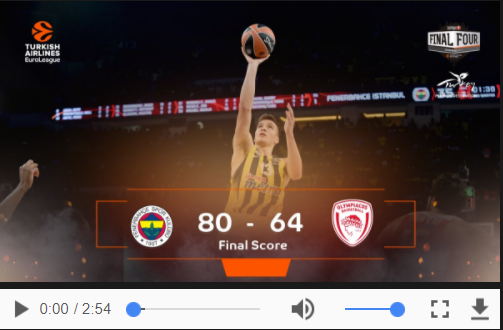
button#history-button{  
 position: relative;  
 left: 35%;  
 background-color: #58595B;  
 border-style: none;  
 margin: 2px 10px 0 10px;  
 border-radius: 5px;  
 font-size: 14pt;  
}

1.2 Гибкая блочная верстка

div#content {  
 display: flex;  
 align-content: center;  
}

2. CSS

2.1 API Видео



<video id="video" width="400" height="300" controls="controls"  
 poster="../resources/image/DAYAfPVW0AEeWQ2.jpg"  
 onclick="this.paused ? this.play() : this.pause();">  
 <source src="../resources/video/Championship.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>  
</video>

2.2 SVG



<svg height="10" width="200">  
 <line x1="0" y1="0" x2="200" y2="0" style="stroke:rgb(227, 80, 0);stroke-width:4"/>  
</svg>

3.DHTML

<body onload="checkCookies()">

function checkCookies() {  
 if (navigator.cookieEnabled == false) {  
 document.getElementById("error").innerHTML = "Cookies are enabled, please turn them on";  
 }  
}

3.2 Извлечение и обработка динамического содержания документа (с помощью DOM и InnerHTML)

function() {  
 var authSection = document.getElementById('auth-section');  
 authSection.innerHTML =  
 '<ul id="auth">' +  
 '<li><a href="/signup" onclick="return makeSignUpSectionVisible()">Sign up </a></li>' +  
 '<li><a href="/login" onclick="return makeLogInSectionVisible()">Log In</a></li>' +  
 '</ulid>';  
 $('.invalid:visible').fadeOut(3000);   
 return false;  
}

3.3 Вставка HTML-содержимого между узлами DOM имеющегося document-содержимого по событию экрана

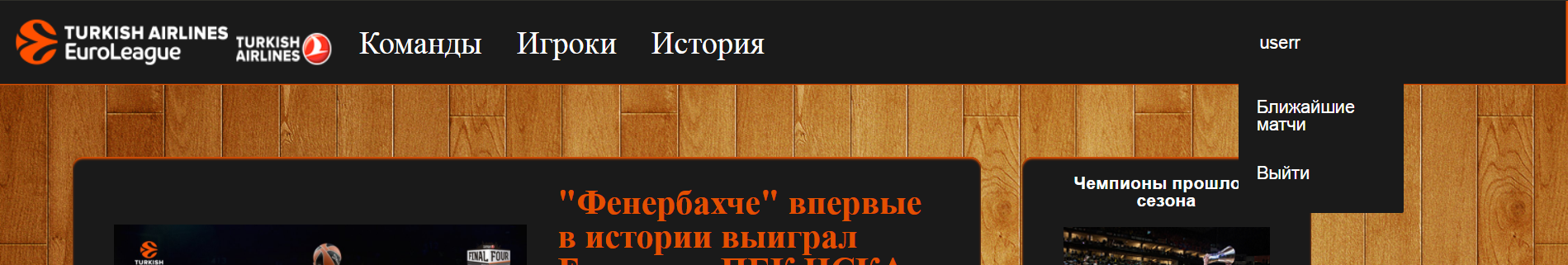
function makeSignUpSectionVisible() {  
 var authSection = document.getElementById('auth-section');  
 authSection.innerHTML = '';  
  
 var form = document.createElement('form');  
 form.id = 'signup';  
 form.action = '/signup';  
 form.method = 'post';  
 form.classList.add('visible-form');  
  
 var loginDiv = document.createElement('div');  
 loginDiv.classList.add('field');  
 var passwordDiv = document.createElement('div');  
 passwordDiv.classList.add('field');  
 var loginLabel = document.createElement('label');  
 loginLabel.for = 'username';  
 loginLabel.innerHTML = 'Имя пользователя: ';  
 var loginInput = document.createElement('input');  
 loginInput.type = 'text';  
 loginInput.id = 'username';  
 loginInput.name = 'username';  
 loginInput.required = true;  
 loginInput.onfocus = function () {  
 loginInput.style.borderColor = '#e35000';  
 };  
 loginInput.onblur = function () {  
 loginInput.style.borderColor = 'transparent';  
 };  
 loginDiv.appendChild(loginLabel);  
 loginDiv.appendChild(loginInput);  
 var passwordLabel = document.createElement('label');  
 passwordLabel.for = 'password';  
 passwordLabel.innerHTML = 'Пароль: ';  
 var passwordInput = document.createElement('input');  
 passwordInput.type = 'password';  
 passwordInput.id = 'password';  
 passwordInput.name = 'password';  
 passwordInput.required = true;  
 passwordInput.onfocus = function () {  
 passwordInput.style.borderColor = '#e35000';  
 };  
 passwordInput.onblur = function () {  
 passwordInput.style.borderColor = 'transparent';  
 };  
 passwordDiv.appendChild(passwordLabel);  
 passwordDiv.appendChild(passwordInput);  
 var cancelButton = document.createElement('button');  
 cancelButton.onclick = function () {  
 require(['makeAuthSectionClear', 'jquery'], function (makeAuthSectionClear, $) {  
 makeAuthSectionClear();  
 });  
 return false;  
 };  
 cancelButton.innerHTML = 'Отмена';  
 var signUpButton = document.createElement('input');  
 signUpButton.type = 'submit';  
 signUpButton.value = 'Зарегистрироваться';  
  
 form.appendChild(loginDiv);  
 form.appendChild(passwordDiv);  
 form.appendChild(cancelButton);  
 form.appendChild(signUpButton);  
  
 authSection.insertBefore(form, authSection.firstChild);  
  
 form.addEventListener('submit',  
 require(['checkEmptyFields', 'jquery'], function (checkEmptyFields, $) {  
 checkEmptyFields();  
 }));  
  
 return false;  
}

4. Использование библиотеки JQuery

4.1 Применение фильтров контента и контента

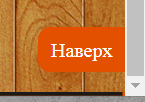
$('html').bind('mousedown', function () { //jquery  
 $('.error:visible').filter(':parent').fadeOut(4000);   
});

4.2 Использование динамических элементов на странице



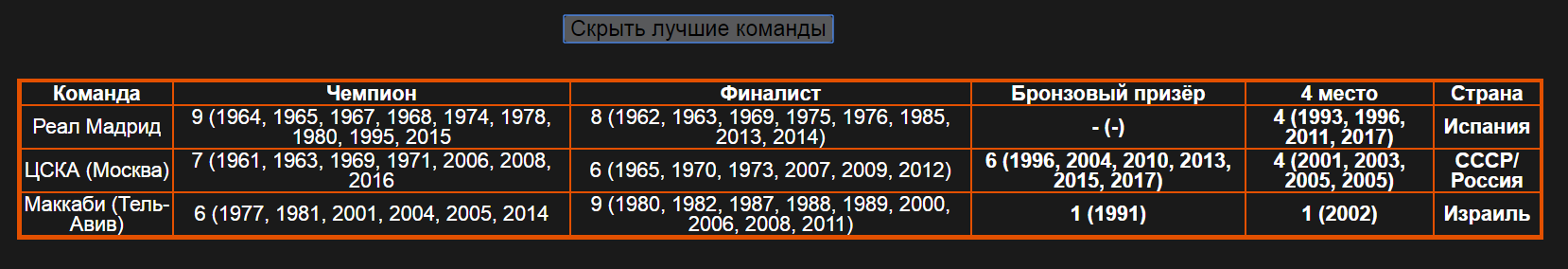
<nav>  
 <a class="dropdown-toggle" href="#" title="Menu"><span id="user"><%= currentUser %></span></a>  
 <ul class="dropdown">  
 <li><a href="/schedule">Ближайшие матчи</a></li>  
 <li><a href="/logout">Выйти</a></li>  
 </ul>  
</nav>

4.3 Интерактивная анимация



$(document).ready(function(){  
 $('body').append('<div id="toTop">Наверх</div>');  
 $(window).bind('scroll', function () {  
 if ($(this).scrollTop() != 0) {  
 $('#toTop').fadeIn();  
 } else {  
 $('#toTop').fadeOut();  
 }  
 });  
 $('#toTop').bind('click', function(){  
 $("html, body").animate({ scrollTop: 0 }, 600);  
 return false;  
 });  
});

4.4 Интерактивное добавление и удаление



var data = {  
 team: json.team[i].name,  
 win: json.team[i].win + ' (' + json.team[i].yearsWon.join(", "),  
 final: json.team[i].final + ' (' + json.team[i].yearsFinal.join(", ") + ')',  
 third: json.team[i].third + ' (' + json.team[i].yearsThird.join(", ") + ')',  
 place: json.team[i].place + ' (' + json.team[i].yearsPlace.join(", ") + ')',  
 country: json.team[i].country  
};  
var h = mustache.to\_html(templ, data);  
$("#table-cont").append(h);  
$("#history-table").css({display: "block"});  
$("#history-button").text(' Скрыть лучшие команды ');

5. Использование Ajax

5.1 Загрузка информации с JSON

$.getJSON("../resources/data/data.json")  
 .done(function (json) {  
 require(['mustache', 'jquery'], function (mustache, $) {  
 var templ = '<tr><td>{{team}}</td> <td>{{win}}</td> <td>{{final}}</td>' +  
 '<th>{{third}}</th> <th>{{place}}</th> <th>{{country}}</th></tr>';  
 for (var i = 0; i < json.team.length; i++) {  
 var data = {  
 team: json.team[i].name,  
 win: json.team[i].win + ' (' + json.team[i].yearsWon.join(", "),  
 final: json.team[i].final + ' (' + json.team[i].yearsFinal.join(", ") + ')',  
 third: json.team[i].third + ' (' + json.team[i].yearsThird.join(", ") + ')',  
 place: json.team[i].place + ' (' + json.team[i].yearsPlace.join(", ") + ')',  
 country: json.team[i].country  
 };  
 var h = mustache.to\_html(templ, data);  
 $("#table-cont").append(h);  
 $("#history-table").css({display: "block"});  
 $("#history-button").text(' Скрыть лучшие команды ');  
 }  
 });  
 });

5.2 Привязка (bind) 3 любых событий

$('html').bind('mousedown', function () {   
 $('.error:visible').filter(':parent').fadeOut(4000);   
});

$('html').bind('keyup', function (e) {   
 if (e.keyCode == 27) {  
 require(['makeAuthSectionClear', 'jquery'], function (makeAuthSectionClear, $) {  
 makeAuthSectionClear();  
 });  
 return false;  
 }  
});

$('#toTop').bind('click', function(){   
 $("html, body").animate({ scrollTop: 0 }, 600);  
 return false;  
});

6 Реализация модульной концепции AMD

requirejs.config({  
 paths: {  
 "jquery": "jquery-1.10.2",  
 "makeAuthSectionClear": "makeAuthSectionClear",  
 "checkEmptyFields": "checkEmptyFields",  
 "mustache": "mustache"  
 }  
});

define(function(){  
 return function (event) {  
 if (!$(':input[required]:text').val() || !$(':input[required]:password').val()) {  
 event.preventDefault();  
 $('.invalid').text('Не заполнены все поля');  
 return false;  
 }  
 }  
});

form.addEventListener('submit',  
 require(['checkEmptyFields', 'jquery'], function (checkEmptyFields, $) {  
 checkEmptyFields();  
}));

7 Сервер

var express = require('express'),  
 http = require('http'),  
 path = require('path'),  
 mongoose = require('mongoose');  
  
var app = express();  
  
mongoose.connect("mongodb://localhost/myapp");  
  
var UserSchema = new mongoose.Schema({  
 username: String,  
 password: String  
});  
var User = mongoose.model('users', UserSchema);  
  
app.configure(function () {  
 app.use(express.bodyParser());  
 app.use(express.cookieParser('Authentication Tutorial '));  
 app.use(express.session());  
 app.use(express.static(path.join(\_\_dirname, 'public')));  
 app.use("/styles", express.static(\_\_dirname + '/styles'));  
 app.use('/script', express.static(\_\_dirname + '/script'));  
 app.use("/resources/image", express.static(\_\_dirname + '/resources/image'));  
 app.use("/resources/fonts", express.static(\_\_dirname + '/resources/fonts'));  
 app.use("/resources/video", express.static(\_\_dirname + '/resources/video'));  
 app.use("/resources/data", express.static(\_\_dirname + '/resources/data'));  
 app.set('/views', express.static(\_\_dirname + '/views'));  
 app.set('view engine', 'ejs');  
});  
  
app.use(function (req, res, next) {  
 var err = req.session.error,  
 msg = req.session.success,  
 usr = req.session.user;  
 delete req.session.error;  
 delete req.session.success;  
 res.locals.message = '';  
 res.locals.currentUser = '';  
 if (err) res.locals.message = err;  
 if (msg) res.locals.message = msg;  
 if (usr) res.locals.currentUser = usr.username;  
 next();  
});  
  
function authenticate(name, pass, fn) {  
 User.findOne({  
 username: name  
 },  
 function (err, user) {  
 if (user) {  
 if (err) return fn(new Error('Пользователь не найден'));  
 if (pass == user.password) return fn(null, user);  
 fn(new Error('Некорректный пароль'));  
 } else {  
 return fn(new Error('Пользователь не найден'));  
 }  
 });  
}  
  
function userExist(req, res, next) {  
 User.count({  
 username: req.body.username  
 }, function (err, count) {  
 if (count === 0) {  
 next();  
 } else {  
 req.session.error = "Пользователь с таким именем уже существует";  
 res.redirect("/");  
 }  
 });  
}  
  
app.post("/signup", userExist, function (req, res) {  
 var password = req.body.password;  
 var username = req.body.username;  
  
 var user = new User({  
 username: username,  
 password: password  
 }).save(function (err, newUser) {  
 if (err) throw err;  
 authenticate(newUser.username, password, function (err, user) {  
 if (user) {  
 req.session.regenerate(function () {  
 req.session.user = user;  
 res.redirect('/');  
 });  
 }  
 });  
 });  
});  
  
app.post("/login", function (req, res) {  
 authenticate(req.body.username, req.body.password, function (err, user) {  
 if (user) {  
 req.session.regenerate(function () {  
 req.session.user = user;  
 res.redirect('/');  
 });  
 } else {  
 req.session.error = 'Неправильный логин или пароль';  
 res.redirect('/');  
 }  
 });  
});  
  
app.get('/logout', function (req, res) {  
 req.session.destroy(function () {  
 res.redirect('/');  
 });  
});  
  
app.get("/", function (req, res) {  
 if (req.session.user)  
 res.locals.user = req.session.user;  
 res.render("index");  
});  
  
app.get("/teams", function (req, res) {  
 if (req.session.user)  
 res.locals.user = req.session.user;  
 res.render("teams");  
});  
  
app.get("/players", function (req, res) {  
 if (req.session.user)  
 res.locals.user = req.session.user;  
 res.render("players");  
});  
  
app.get("/history", function (req, res) {  
 if (req.session.user)  
 res.locals.user = req.session.user;  
 res.render("history");  
});  
  
app.get("/schedule", function (req, res) {  
 if (req.session.user)  
 res.locals.user = req.session.user;  
 res.render("schedule");  
});  
  
http.createServer(app).listen(3000);