### 1. Что такое протокол HTTP? Какие он поддерживает методы, как передаёт параметры? Для чего предназначены заголовки?

HTTP - протокол передачи данных. Поддерживает методы OPTIONS, GET, HEAD, POST, PUT, PATCH, DELETE, TRACE, CONNECT.

Параметры передаются по-разному. Какие-то передаются в заголовках: часто это название браузера, информация о содержании запроса (длина, тип данных, кодировка), язык пользователя. Заголовки могут содержать пользовательские заголовки - например, токены авторизации.

Запрос может содержать или не содержать тело. Так, в GET запросе тело отсутствует и параметры передаются непосредственно в URL. POST запрос позволяет добавлять тело, в которое может быть добавлены различная информация: как содержимое файла, так и JSON строка (как вариант передачи параметров).

### 2. Для чего предназначен nginx? Ключевые особенности конфигурирования и использования в качестве прокси. В чём заключается настройка для работы с PHP?

Nginx - сервер, предназначенный для обработки запросов.

server {

listen 80 default;

server\_name localhost;

#information about server in headers

server\_tokens off;

location / {

root c:/nginx/public\_html;

index index.html index.htm index.php;

}

location ~ \.php$ {

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

include fastcgi\_params;

}

location ~ /\.ht {

deny all;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

}

Настройка PHP заключается в передачи запроса на FastCGI-сервер (в нашем случае PHP). В конфигурации выше он запущен по адресу 127.0.0.1:9000.

Пример настройки прокси:

location / {

proxy\_pass http://192.168.0.1:8080;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

### 3. Чем ECMAScript отличается от JavaScript? В чём ключевые отличия ECMAScript 5 от более новых версий? Расскажите особенности работы с массивами.

ES/JS: <https://habr.com/ru/company/nix/blog/342904/>

2016-2018: https://habr.com/ru/company/ruvds/blog/353174/

ECMAScript - это спецификация создания скриптового языка.

JavaScript - язык, соответствующий какому-то из стандартов ECMAScript.

Отличия:

* Работа со строками (перенос на новую строку, поиск регулярок, padEnd/padStart, )
* async/await
* import
* arrays - добавлена куча функций вроде reduce, find, some, all, forEach, map и т.д.
* \*\*
* Object.values(.entries, .keys)
* Завершающие запятые в параметрах функци
* Тегированные строки (`Hello, ${name}`)
* асинхронная итерация (for await (a in b) {…})

### 4. Опишите ключевые особенности промисов, генераторов, Map, WeakMap, прокси, async/await.

Promise

Promise.resolve()

.then(() => {

return api.getItem(1)

})

.then(item => {

item.amount++

return api.updateItem(1, item);

})

.then(update => {

return api.deleteItem(1);

})

.catch(e => {

console.log('error while working on item 1');

})

Генераторы

function\* generateSequence() {

yield "a";

yield "b";

return "c";

}

let generator = generateSequence();

let value;

while(!(value = generator.next()).done) {

console.log(value);

}

Map - как объект, но ключи любого типа (set, get, has, delete, clear, size)

WeakMap - ключи всегда объекты. WeakMap не препятствует удалению ключей из памяти. <https://learn.javascript.ru/weakmap-weakset>

Прокси

let proxy = new Proxy(user, {

get(target, prop) {

console.log(`Чтение ${prop}`);

return target[prop];

},

set(target, prop, value) {

console.log(`Запись ${prop} ${value}`);

target[prop] = value;

return true;

}

});

proxy.firstName = "Тест";

proxy.firstName;

async/await понятно

### 5. Для чего нужны npm и yarn? Перечислите основные команды ими поддерживаемые. Приведите примеры стандартных пакетов и особенностей их применения. Чем отличается Deno?

npm/yarn - менеджеры пакетов

npm:

* init
* publish
* adduser
* install
* login
* uninstall

yarn:

* init
* add
* upgrade
* remove
* install (загрузить все, что указано в pachage.json)

EventEmitter

let EventEmitter = require("events").EventEmitter;

let manager = new EventEmitter();

manager.on("request", (request)=> {

console.log(request);

});

manager.on("handler", (request)=> {

console.log(request);

});

manager.emit("request", {dataRequest:"Данные"});

manager.emit("handler", {dataHandler:"Ещё данные"});

manager.emit("error");

http

let http = require("http");

let server = new http.Server((req, res)=>{

console.log(req.method, req.url);

});

server.listen(3000, "127.0.0.1");

url

url.parse(req.url, true);

debug <https://www.npmjs.com/package/debug> - красивый вывод в лог (более удобный/функциональный)

var debug = require('debug')('http')

, http = require('http')

, name = 'My App';

debug('booting %o', name);

http.createServer(function(req, res){

debug(req.method + ' ' + req.url);

res.end('hello\n');

}).listen(3000, function(){

debug('listening');

});

require('./worker');

Winston https://www.npmjs.com/package/winston - тоже логгер. Имеется возможность разделения логов по уровням, вывода в разные файлы.

### 6. Что такое ORM? Какие ORM вы знаете для JavaScript? Приведите примеры использования.

ORM - прослойка между бд и программистом. Например, вместо написания SQL запросов мы будем работать с функциями/классами/полями классов в используемом языке.

TypeORM (TypeScript)

@Entity()

export class User {

@PrimaryGeneratedColumn()

id: number;

@Column()

firstName: string;

@Column()

lastName: string;

@Column()

age: number;

}

//##############

const repository = connection.getRepository(User);

const user = new User();

user.firstName = "Timber";

user.lastName = "Saw";

user.age = 25;

await repository.save(user);

const allUsers = await repository.find();

const firstUser = await repository.findOne(1); // find by id

const timber = await repository.findOne({ firstName: "Timber", lastName: "Saw" });

await repository.remove(timber);

prisma (<https://www.prisma.io/docs/getting-started/quickstart-typescript>)

### 7. Приведите пример создания эхо-сервера на Node.js. Приведите способы и примеры журналирования («логгирования») в Node.js?

var winston = require('winston')

, http = require('http')

, name = 'My App';

winston.error('booting %o', name);

http.createServer(function(req, res){

winston.info(req.method + ' ' + req.url);

res.end('hello\n');

}).listen(3000, function(){

winston.info('listening');

});

### 8. В чём отличие асинхронного выполнения кода от синхронного? Приведите соответствующие примеры при работе с файлами. Как выполнить «асинхронный try-catch» (обработку ошибок в асинхронном режиме)?

Выполнение JS программы можно представить как набор блоков, которые поочередно выполняются (вспомни dart и поехали).

Обработка ошибок:

1. Promise

var a = new Promise(function(res, rej) { rej(“AAAAAA”) });

a.catch(error => console.log(error));

// or a.then(res => console.log(res), error => console.log(error));

2. async/await

try {

var a = new Promise(function(res, rej) { rej(“AAAAAA”) });

await a;

catch (e) {

console.log(e);

}

### 9. Опишите структуру HTML, понятие DOM-модели. Как осуществляется обработка событий в HTML?

<!DOCTYPE HTML>

<html>

<head onclick=”console.log(‘CLICKED’)”>

<title>О лосях</title>

</head>

<body>

Правда о лосях.

</body>

</html>

### 10. Опишите, как использовать CSS. Что такое псевдоклассы, селекторы? Приведите примеры. Что такое динамическая вёрстка и как она выполняется?

<style type="text/css">

p {

text-align: center;

color: blue;

}

</style>

…

<link rel="stylesheet" href="simple.css">

Псевдоклассы - :link, :hover и т.д.

Селекторы - #id, .class, [атрибут=значение] и т.д.

Динамическая верстка. Подразумевается адаптивная или использование scss, less и т.п.?

less

@color: #4D486A;

#header {

color: @color;

}

h2 {

color: @color;

}

a (@x: 20px) {

color: red;

width: @x;

}

.mixin-class {

.a();

}

.mixin-id {

.a(100px);

}

Адаптивная (медиа запросы):

@media all and (orientation: landscape), all and (min-width: 480px) { ... }

### 11. Зачем нужен фреймворк LESS? Опишите его нотацию и особенности применения по назначению.

@color: #4D486A;

#header {

color: @color;

}

h2 {

color: @color;

}

Миксины

a (@x: 20px) {

color: red;

width: @x;

}

.mixin-class {

.a();

}

.mixin-id {

.a(100px);

}

Вложенные правила

#header {

background: lightblue;

a {

color: blue;

&:hover {

color: green;

}

}

}

Функции

.average(@x, @y) {

@Average: ((@x + @y) / 2);

}

div {

.average(12px, 10px); // вызов миксина

padding: @Average; // использование результата

}

### 12. Какие элементы HTML используются в <FORM>; для взаимодействия с пользователем? Как осуществляется валидация введённых значений?

label(for=inputId) (или просто обернуть input в label)

input(type=text|number|date|… value=…)

select

fieldset

legend

button

textarea

Валидация: подписываемся на события (например input на поле) и выставляем input.setCustomValidity(“ERROR”) (ну или кастомное поле)

keygen - генерация ключей

### 13. С помощью чего можно создавать шаблоны web-страниц? Как использовать средства шаблонизации совместно с модулем Express?

С помощью IDE! Ну или pug, если речь об этом.

На примере pug (остальные работают +- также):

* + 1. Перевод pug->html при сборке (gulp, webpack)
    2. Перевод перед отправкой клиенту с подстановкой актуальных параметров

app.set("view engine", "pug");

…

res.render("add-book", {

user: req.user,

});

### 14. Как можно выполнить авторизацию и аутентификацию? Что такое Cookies, как они работают и используются? Опишите применение Ajax.

let session = require('express-session');

…

server.use(session({secret: "Секретный ключ"}));

Кастомные библиотеки (passport) работают примерно также.

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200)

callback(this.responseText);

};

xhttp.open("GET", `/group/${id}`, true);

xhttp.send();

* onreadystatechange
* readyState (номер состояния)
* responseText (responseXML)
* status
* statusText (“Not Found”, “OK”, …)

### 15. Опишите реализацию RESTful на Node.js.

let express = require('express');

let bodyParser = require('body-parser');

let cookieParser = require('cookie-parser');

let server = express();

server.use(cookieParser());

server.use(bodyParser.json());

server.use(bodyParser.urlencoded({ extended: true }));

let groups = require('./groups.js');

server.use('/groups', groups);

server.listen(3000);

groups.js

let express = require('express');

let router = express.Router();

let groups = [

{id: 1, name: "5381", students: 15, rating: 4.1},

{id: 2, name: "5303", students: 13, rating: 4.7},

{id: 3, name: "5304", students: 14, rating: 5},

{id: 4, name: "5382", students: 20, rating: 3.9},

{id: 5, name: "5383", students: 14, rating: 4.9}

];

router.get('/', (req, res)=>{

res.json(groups);

});

module.exports = router;

### 16. Как можно работать с историей в браузере? В чём отличия sessionStorage, localStorage и Cookies? Приведите примеры.

sessionStorage - на одной вкладке

localStorage - между всеми вкладками/окнами/перезапусками ОС

Cookies - отправляются с сервера. Бывают различных видов.

### 17. Опишите для чего нужны и как используются web-сообщения, web-worker, webсокеты.

web-сообщения - ?

web-worker - запуск скрипта в отдельном потоке.

webсокеты - обмен сообщениями с сервером в одном сокете.

### 18. Опишите конфигурирование проекта с использованием GULP.

import gulp from 'gulp';

import less from 'gulp-less';

import del from 'del';

import babel from 'gulp-babel';

import concat from 'gulp-concat';

import uglify from 'gulp-uglify';

import cleanCSS from 'gulp-clean-css';

import rename from 'gulp-rename';

import pugConverter from 'gulp-pug';

function stylesTasks() {

return confs.styles.map((conf) => {

return () => {

let task = gulp.src(conf.src);

if (conf.less)

task = task.pipe(less());

if (conf.clean)

task = task.pipe(cleanCSS())

.pipe(rename((path) => path.basename += ".min"));

return task.pipe(gulp.dest(conf.dest))

}

});

}

export const styles = gulp.parallel(...stylesTasks());

### 19. Опишите конфигурирование проекта с использованием Webpack.

const path = require('path');

module.exports = {

entry: path.join(\_\_dirname, 'src/www.js'),

output: {

filename: 'bundle.js',

path: path.join(\_\_dirname, "/lib/"),

},

devtool: "eval-source-map",

target: 'node',

module: {

rules: [

{

test: /\.m?js$/,

exclude: /node\_modules/,

use: {

loader: 'babel-loader',

options: {

"presets": [

["@babel/preset-env", {

"useBuiltIns": "usage",

"corejs": 3

}],

"@babel/preset-flow"

],

"plugins": [

"@babel/plugin-proposal-class-properties"

]

}

}

}

]

}

};

### 20. Опишите применение и приведите примеры использования jQuery (селекторы, манипуляция с DOM, обработка событий, CSS, эффекты)

$("h3").text("Проверка 1");

$("#list > div > div > div").text("Найденные потомки");

$("#smalllist").find("li").text("Использование traversing");

$("#mychild").parent() .css("color", "blue");

// события

$("button").on("click", ()=>{

$("#myP").text(`Счётчик нажатий = ${++counter}`);

});

// эффекты

.animate()

.fadeIn()

.fadeOut()

.fadeTo()

.fadeToggle()

.finish()

.hide()

.show()

…

### 21. Опишите применение и приведите примеры использования jQuery UI (взаимодействие, виджеты). Приведите примеры использования jQuery mobile.

$(()=>{

$(".draggable1").draggable({

start: ()=>{

$("#start").text(++events.start);

},

drag: ()=>{

$("#drag").text(++events.drag);

},

stop: ()=>{

$("#stop").text(++events.stop);

}

});

});

$("#d2").resizable({ animate: true });

$("#selectable").selectable();

$(".accordion").accordion({

collapsible: true

});

**// mobile**

<div data-role="fieldcontain"></div>

//####

<div data-role="page" id="main\_page" data-theme="b">

<div data-role="header" >

<h1 id="twi\_acc">Home page</h1>

<a href="#settings" data-icon="gear" class="ui-btn-right">Options</a>

</div>

<div data-role="content" >

<ul data-role="listview" data-inset="true" data-theme="a">

<li><a href="#twitter\_page">Twitter example</a></li>

<li><a href="#map\_page">Map example</a></li>

<li><a href="#search">Search example</a></li>

<li><a href="#about">About</a></li>

</ul>

</div>

<div data-role="footer">

</div>

</div>

крч в mobile очень много верстается просто через data-role

### 22. Опишите применение фреймворка Angular (структура приложения, компоненты, сервисы, директивы, задание маршрутов).

**src:**

**app/app.component.{ts,html,css,spec.ts}**

Корневой компонент приложения

**app/app.module.ts** Корневой модуль, который сообщает, как собирать приложение

**assets/\*** Для рисунков и других материалов

**environments/\*** Конфигурации целевой среды

**favicon.ico** Фавикон

**index.html** Главная страница

**main.ts** Точка входа для приложения (язык – TypeScript)

**polyfills.ts** Настройки для работы с различными браузерами

**styles.css** Глобальные стили

**test.ts** Точка входа для модульных тестов

**tsconfig.{app|spec}.json** Конфигурационные файлы компилятора

**корень:**

**e2e/** Тесты end-to-end

**node\_modules/** Папка для модулей node.js

**.angular-cli.json** Конфигурация Angular CLI

**.editorconfig** Конфигурация редактора

**.gitignore** Контроль коммитов в Git

**karma.conf.js** Модульные тесты на движке Karma

**package.json** Описание приложения

**protractor.conf.js** Конфигурация тестов end-to-end на основе Protractor

**README.md** Базовая документация

**tsconfig.json** Конфигурация компилятора TypeScript

**tslint.json** Конфигурация TSLint

**@NgModule** - описание метаданных модуля

* **declarations**: классы представлений (view classes), которые принадлежат модулю. Angular имеет три типа классов представлений: компоненты (components), директивы (directives), каналы (pipes)
* **exports**: набор классов представлений, которые должны использоваться в шаблонах компонентов из других модулей
* **imports**: другие модули, классы которых необходимы для шаблонов компонентов из текущего модуля
* **providers**: классы, создающие сервисы, используемые модулем
* **bootstrap**: корневой компонент, который вызывается по умолчанию при загрузке приложения

**@Component** - компонент приложения, что тут еще сказать?

* **selector**: A CSS selector that tells Angular to create and insert an instance of this component wherever it finds the corresponding tag in template HTML. For example, if an app's HTML contains <app-hero-list></app-hero-list>, then Angular inserts an instance of the HeroListComponent view between those tags.
* **templateUrl**: The module-relative address of this component's HTML template. Alternatively, you can provide the HTML template inline, as the value of the template property. This template defines the component's host view.
* **providers**: An array of providers for services that the component requires.

**@Directive**

@Directive({

selector: '[bold]'

})

export class BoldDirective{

constructor(private elementRef: ElementRef){

this.elementRef.nativeElement.style.fontWeight = "bold";

}

}

**@Pipe**

@Pipe ({

name: 'Power'

})

export class PowerPipe implements PipeTransform {

transform(value: number, power: string): number {

let myPower = parseFloat(power);

return value \*\* myPower;

}

}

///////

<div> 2\*\*{{myValue}} = {{2| Power:myValue}}</div>

**Параметризация вывода**

Пример lowercase: {{myname|lowercase}}<br>

Пример uppercase: {{myname|uppercase}}<br>

Пример slice:1:2: {{myname|slice:1:2}}<br>

Пример slice:2:4: {{myname|slice:2:4}}<br>

Пример date: {{current|date:"dd.MM.yyyy"}}<br>

Пример currency: {{index|currency}}<br>

Пример percent: {{float|percent}}<br>

**@Injectable**

@Injectable({providedIn: 'root'})

// тут provideIn говорит “Создай меня в NgModule!!!”

export class UsersHttpService{

constructor(private http: HttpClient){}

getUsers(): Observable<any>{

return this.http.get('/api/users');

}

}

**Роутинг** - пихаем <router-outlet> в template

<li><a [routerLink] = "['/myCompA']">myCompA</a></li>

<li><a [routerLink] = "['/myCompB']">myCompB</a></li>

/------------------------ или

export class myCompB {

constructor(private router: Router){}

myFunc(): void {

this.router.navigate(['/myCompA']);

}

}

/-------------------

const appRoutes: Routes = [

{ path: 'myCompA', component: myCompA },

{ path: 'myCompB', component: myCompB },

{ path: '\*\*', redirectTo: '/'}

];

@NgModule({

declarations: [AppComponent, myCompA, myCompB],

imports: [BrowserModule, RouterModule.forRoot(appRoutes)],

providers: [],

bootstrap: [AppComponent],

})

export class AppModule { }

### 23. Опишите применение и приведите примеры использования Backbone.js (модели, представления, события).

<div id="start" class="block">

<div class="userplace">

<label for="username">Имя пользователя: </label>

<input type="text" id="username" />

</div>

<div class="buttonplace">

<input type="button" value="Проверить" />

</div>

</div>

//----------------------

var Start = Backbone.View.extend({

el: $("#start"), // DOM элемент widget'а

events: {

"click input:button": "check" // Обработчик клика на кнопке "Проверить"

},

check: function () {

var username = this.el.find("input:text").val();

var find = (\_.detect(Family, function (elem) { return elem == username })); // Проверка имени пользователя

appState.set({

"state": find ? "success" : "error",

"username": username

});

}, });

var start = new Start();

//--------------------------

var Controller = Backbone.Router.extend({

routes: {

"": "start", // Пустой hash-тэг

"!/": "start", // Начальная страница

"!/success": "success", // Блок удачи

"!/error": "error" // Блок ошибки

},

start: function () {

$(".block").hide(); // Прячем все блоки

$("#start").show(); // Показываем нужный

},

success: function () {

$(".block").hide();

$("#success").show();

},

error: function () {

$(".block").hide();

$("#error").show();

}

});

//------------------------

var AppState = Backbone.Model.extend({

defaults: {

username: "",

state: "start"

}

});

var appState = new AppState();

appState.trigger("change");

//-------------------------------

appState.bind("change:state", function () { // подписка на смену состояния для контроллера

var state = this.get("state");

if (state == "start")

controller.navigate("!/", false);

else

controller.navigate("!/" + state, false);

});

### 24. Опишите особенности и приведите примеры использования TypeScript (типы переменных, интерфейсы, классы, функции, шаблоны, перечисления, импорт-экспорт, пространства имён, миксины).

interface Person {

firstName: string;

lastName: string;

}

function greeter(person: Person) {

return "Hello, " + person.firstName + " " + person.lastName;

}

class Student {

fullName: string;

constructor(public firstName: string, public middleInitial string, public lastName: string) {

this.fullName = firstName + " " + middleInitial + " " + lastName;

}

}

interface ClockInterface {

new (hour: number, minute: number);

currentTime: Date;

setTime(d: Date);

}

class Clock implements ClockInterface {

currentTime: Date;

constructor(h: number, m: number) { }

setTime(d: Date) {

this.currentTime = d;

}

}

interface Shape {

color: string;

}

interface Square extends Shape {

sideLength: number;

}

class Animal {

name: string;

constructor(theName: string) { this.name = theName; }

move(distanceInMeters: number = 0) {

console.log(`${this.name} moved ${distanceInMeters}m.`);

}

}

class Snake extends Animal {

constructor(name: string) { super(name); }

move(distanceInMeters = 5) {

console.log("Slithering...");

super.move(distanceInMeters);

}

}

class Employee {

private \_fullName: string;

get fullName(): string {

return this.\_fullName;

}

}

abstract class Department {

constructor(public name: string) {

}

printName(): void {

console.log("Department name: " + this.name);

}

abstract printMeeting(): void;

}

function identity<T>(arg: T): T {

return arg;

}

enum BooleanLikeHeterogeneousEnum {

No = 0,

Yes = "YES",

}

let list = [4, 5, 6];

for (let i in list) {

console.log(i);

}

export interface StringValidator {

isAcceptable(s: string): boolean;

}

export const numberRegexp = /^[0-9]+$/;

export class ZipCodeValidator implements StringValidator {

isAcceptable(s: string) {

return s.length === 5 && numberRegexp.test(s);

}

}

class ZipCodeValidator2 implements StringValidator {

isAcceptable(s: string) {

return s.length === 5 && numberRegexp.test(s);

}

}

export { ZipCodeValidator2 };

export { ZipCodeValidator2 as mainValidator };

import { ZipCodeValidator } from "./ZipCodeValidator";

let myValidator = new ZipCodeValidator();

import { ZipCodeValidator as ZCV } from "./ZipCodeValidator";

let myValidator = new ZCV();

import \* as validator from "./ZipCodeValidator";

let myValidator = new validator.ZipCodeValidator();

namespace Validation {

export interface StringValidator {

isAcceptable(s: string): boolean;

}

const numberRegexp = /^[0-9]+$/;

export class ZipCodeValidator implements StringValidator {

isAcceptable(s: string) {

return s.length === 5 && numberRegexp.test(s);

}

}

}

Миксины

class implements class1 {}

### 25. Опишите применение и приведите примеры использования Flow.

function concat(a: string, b: string) {

return a + b;

}

function method(x: number, y?: ?string = "def.value") {

}function getColor(name: "success" | "warning") {}

function stringifyBasicValue(value: string | number) {}

mixed - валидатор подставит тип от всех вызовов и проверит

function stringify(value: mixed) {}

function add(one: any, two: any): number {}

function method1(callback: (str: string, bool?: boolean, ...nums: Array<number>) => void) {}

let obj1: { foo: boolean } = { foo: true };

var o: { [string]: number } = {}; o["foo"] = 0;

var obj: { [user\_id: number]: string } = {}; obj[1] = "Julia";

let tuple: [number, boolean, string] = [1, true, "three"];

class MyClass1<A, B> {

property: A;

method(val: B): void {}

}

function logFoo3<T: { foo: string }>(obj: T): T {}

type Item<T: number = 1> = {

prop: T,

};

type MyObject = {

foo: number,

bar: string

}

let val: MyObject = {foo:12, bar:""};

interface Covariant { +readOnly: number | string }

class I implements Invariant {

property: number | string;

constructor(prop: number | string) {

this.property = prop;

}

}

приведение типов: (newValue: string)

перечисления: type Suit = "Diamonds" | "Clubs" | "Hearts" | "Spades";

### 26. Приведите реализацию шаблонов проектирования на JavaScript.

<https://habr.com/ru/company/ruvds/blog/427293/>

### 27. Опишите особенности и приведите примеры использования Immutable.js и Redux.

Immutable.js - неизменяемые типы данных

<https://habr.com/ru/company/devexpress/blog/302118/>

<https://vec.etu.ru/moodle/pluginfile.php/143500/mod_resource/content/1/Web-tech11%20-%20react.pdf>

reducer:

function reduce (state = initialState, action) {

switch (action.type) {

case Actions.AddError: {

let error = {id: errorId++, message: action.message};

return {

...state,

errors: [...state.errors, error]

}

}

case Actions.RemoveError: {

return {

...state,

errors: state.errors.filter(error => error.id !== action.id)

}

}

default: return state;

}

}

connect(

state => ({member: state.members.member, market: state.market.market})

)(MarketScreen)

### 28. Опишите применение React. Что такое JSX? Как осуществляется рендеринг в React? Приведите примеры.

class MarketWaiter extends React.Component {

\_intervalId;

constructor(props) {

super(props);

this.state = {

timerText: this.\_makeTimerText()

};

}

componentDidMount() {

…

}

componentWillUnmount() {

…

}

render() {

return (

<div>

…

</div>

);

}

}

JSX является синтаксическим сахаром для функции React.createElement(component, props, ...children).

<MyButton color="blue" shadowSize={2}>

Click Me

</MyButton>

компилируется в

React.createElement(

MyButton,

{color: 'blue', shadowSize: 2},

'Click Me'

)

29. Опишите применение React. Опишите компоненты и свойства в React, обработку событий. Приведите примеры.

<button onClick={activateLasers}>

// Bind - чтобы в callback-е работало

constructor(props) {

super(props);

this.state = {isToggleOn: true};

// This binding is necessary to make `this` work in the callback

this.handleClick = this.handleClick.bind(this);

}

handleClick() {

this.setState(prevState => ({

isToggleOn: !prevState.isToggleOn

}));

}

### 30. Опишите применение React. Опишите работу с формами в React. Приведите примеры.

<input type="text"

value={this.state.value}

onChange={this.handleChange} />

### 31. Опишите область применения PHP, основные элементы языка, особенности обработки запросов. В чём заключается критика языка.

array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43")

**$GLOBALS** Ссылки на все переменные глобальной области видимости

**$\_SERVER** Информация о сервере и среде исполнения

**$\_GET** GET-переменные HTTP, переданные скрипту через URL

**$\_POST** POST-переменные HTTP, переданные скрипту через HTTP метод POST

**$\_FILES** Переменные файлов, загруженных по HTTP

**$\_COOKIE** HTTP Куки

**$\_SESSION** Переменные сессии

**$\_REQUEST** Переменные HTTP-запроса

**$\_ENV**

include "demo01.php" (в отличие от require не упадет, если файла нет)

setcookie(name, value, expire, path, domain, secure, httponly);

simplexml\_load\_string($str) // чтение xml

class BooksFixedPrice {

var $price = 34;

var $title;

function setTitle($par){

$this->title = $par;

}

function getTitle(){

echo $this->title;

}

}

Проблемы языка

* Нет поддержики юникода. Надёжно работает только ASCII
* Не транзитивен "foo" == TRUE, и "foo" == 0, но TRUE != 0

<https://vec.etu.ru/moodle/pluginfile.php/143504/mod_resource/content/1/Web-tech14%20-%20php.pdf>

### 32. Приведите описание понятия TDD-разработки. Опишите применение и приведите примеры Assert, Should и Chai.

TDD - test-driven development

**Assert**

assert(value[, message])

assert.deepEqual(actual, expected[, message])

assert.deepStrictEqual(actual, expected[, message])

assert.doesNotThrow(block[, error][, message])

assert.equal(actual, expected[, message])

assert.fail(message)

assert.fail(actual, expected[, message[, operator[,

stackStartFunction]]])

assert.ifError(value)

assert.notDeepEqual(actual, expected[, message])

assert.notDeepStrictEqual(actual, expected[, message])

assert.notEqual(actual, expected[, message])

assert.notStrictEqual(actual, expected[, message])

assert.ok(value[, message])

assert.strictEqual(actual, expected[, message])

assert.throws(block[, error][, message])

**Should**

const should = require("should")

const tested = require("../src/test\_should2")

tested.group5381.should.be.an.instanceOf(Object)

tested.group5381.should.have.property("number", "5381")

tested.group5381.should.not.have.property("members").with.lengthOf(15)

**Chai**

assert

const assert = require('chai').assert

const foo = 'бар'

const beverages = { tea: [ 'chai', 'matcha', 'oolong' ] }

assert.typeOf(foo, 'string') // без сообщения

assert.typeOf(foo, 'string', 'foo is a string') // с сообщением

assert.equal(foo, 'bar', 'foo равно `bar`')

assert.lengthOf(foo, 3, 'foo`s имеет длину 3')

assert.lengthOf(beverages.tea, 3, 'beverages предлагает 3 типа чая')

expect

const expect = require('chai').expect

const foo = 'бар'

const beverages = { tea: [ 'chai', 'matcha', 'oolong' ] };

expect(foo).to.be.a('string');

expect(foo).to.equal('bar');

expect(foo).to.have.lengthOf(3);

expect(beverages).to.have.property('tea').with.lengthOf(3);

should

const should = require('chai').should()

const foo = 'бар'

const beverages = { tea: [ 'chai', 'matcha', 'oolong' ] };

foo.should.be.a('string');

foo.should.equal('bar');

foo.should.have.lengthOf(3);

beverages.should.have.property('tea').with.lengthOf(3);

### 33. Приведите описание понятия TDD-разработки. Опишите фреймворк Mocha.

describe

before

after

it

describe("Асинхронный тест", function(){

describe("Чтение из файла", function () {

it("Текущий файл", function (done) {

fs.readFile(\_\_filename, "utf-8", done)

}) // Записи идентичны

it("Ошибочный файл", function (done) {

fs.readFile("err", "utf-8", function(err){

if (err)

return done(err);

else

return done();

})

})

})

})

TDD (test-driven development)

• suite()=decsribe

• test()=it

• suiteSetup()

• suiteTeardown()

• setup()=before

• teardown()=after

describe('Набор тестов (до 500ms)', function() {

this.timeout(500);

…

}

### 34. Опишите как обеспечивается безопасность web-приложений от взлома. Для чего и как используется журналирование («логгирование»)?

<https://vec.etu.ru/moodle/pluginfile.php/143505/mod_resource/content/1/Web-tech15%20-%20injection.pdf>

### 35. Что такое WebAssembly? Как это работает? Приведите примеры использования.

Байт-код. Исполняется виртуальной машиной. Может взаимодействовать с JS (вызов функций, переменных и т.д.). Возможно исполнение через Nodejs.

<https://habr.com/ru/post/342180/>