**HTML**

1) HTML5 знаете (новые теги секционирования документа, Canvas, Geolocation, Web Storage)?

**Ответ должен быть положительным желательно, т.к. HTML5 сейчас применяется на очень большом количестве веб-проектов.**

*Новые теги секционирования:*

1. ***HEADER****: содержимое верхней (начальной) части страницы или ее секции.*
2. ***FOOTER****: содержимое нижней (завершающей) части страницы или ее секции.*
3. ***SECTION****: секция веб-страницы.*
4. ***ARTICLE****: содержимое внешней статьи.*
5. ***ASIDE****: связанное содержимое или выдержка.*
6. ***NAV****: навигационные ссылки.*

***Canvas****– элемент для создания растровых 2D-изображений при помощи Java Script.*

***Geolocation****– элемент для определения местоположения пользователя на карте с помощью навигационной спутниковой системы (GPS), привязки к сотам мобильного оператора (GSM или CDMA), MAC-адресам в сетях WI-FI (WI-FI или Bluetooth), IP-адресам или данным, определенным пользователем.*

***Web Storage****– программный интерфейс, упрощающий сохранение данных посредством веб-запросов. Этот интерфейс дает возможность сохранять данные непосредственно на стороне клиента, в браузере, где они остаются доступными даже после закрытия браузера, тем самым снижая сетевой трафик.*

*Что еще есть в HTML5: работа со звуком (****AUDIO****), работа с видео (****VIDEO****), коммуникационные средства (****XMLHttpRequest Level 2****), веб-сокеты (****WebSocket****), работа с формами (****HTML5 Forms****), технология****Web Workers****, создание автономных веб-приложений.*

2) Что такое **DOCTYPE** и зачем он нужен?

**DOCTYPE** – определение типа документа (Document Type Definition (DTD)), правила, в соответствии с которыми осуществляется проверка конкретного документа (веб-страницы) XML или (X)HTML.

3) Чем отличается **DIV** от **SPAN**?

**DIV** – блочный контейнер, а **SPAN** – линейный. **DIV** используется для разметки блоков, а **SPAN** – для текста.

4) Как обозначаются комментарии в HTML?

|  |  |
| --- | --- |
| **1** | **<!-- комментарий HTML -->** |

5) Что такое **OL**, **UL** и **LI**?

**OL** – нумерованный список, а **UL** – ненумерованный (маркированный) список. Каждый элемент списка должен начинаться с тега **LI**.

6) Как сделать так, чтобы все гиперссылки сайта открывались в новом окне?

|  |  |
| --- | --- |
| 1 | **<head> <base target="\_blank" /> </head>** |

**CSS**

1) В чем разница между **#MY** и **.MY**?

**#MY** – селектор ID, а **.MY** – селектор класса.

2) В чем разница между **MARGIN** и **PADDING**?

**MARGIN** – внешний отступ, а **PADDING** – внутренний отступ.

3) Какая разница между значениями **0** и **AUTO** в свойстве **MARGIN**?

В вертикальных полях – **AUTO** всегда означает **0**. В горизонтальных полях – **AUTO** означает **0**только тогда, если свойство **WIDTH** также **AUTO**.

4) Какое свойство задает цвет фона?

Свойство фона задает свойство **BACKGROUND-COLOR**.

5) Как убрать подчеркивание для всех ссылок на странице?

|  |  |
| --- | --- |
| **1**  **2**  **3** | **a {**  **text-decoration: none;**  **}** |

6) Что делает свойство **CLEAR**?

Свойство **CLEAR** устанавливает, с какой стороны элемента запрещено его обтекание другими элементами.

7) Что такое псевдоклассы?

Псевдоклассы обычно пишут после селектора через двоеточие, чтобы определить реакцию или состояние для данного селектора. Самые известные псевдоклассы: **:LINK**, **:HOVER**, **:VISITED** и**:ACTIVE**.

**По верстке спрашивали:**  
Как прижать футер к низу окна браузера?  
Как выровнять по центру блок с фиксированной шириной? (margin: 0 auto)  
Чему равна высота блока, если у него задано float: left? (0)  
Как сделать очищение потока при использовании элементов с float ? (через oveflow: hidden или через псевдоэлемент after [How to avoid empty clear divs?](http://stackoverflow.com/questions/874877/how-to-avoid-empty-clear-divs))  
Какие новые теги разметки появились в html5? (header, section, article, aside и тд), рассказать немного в каких случаях какой тег используется.

Задаю простенькие задачи:

* <http://jsfiddle.net/RubaXa/36Jve/>
* <http://jsfiddle.net/RubaXa/5jaHB/>
* <http://jsfiddle.net/RubaXa/YBSTX/>
* <http://jsfiddle.net/RubaXa/ZSd3w/>
* <http://jsfiddle.net/RubaXa/kQ8fX/>

по ходу решения которых спрашиваю, почему именно так сделал, какие ещё есть способы решения.

И ещё:

* Подходы/методологии написания CSS?
* GruntJS, Bower, npm — для чего используются?
* RequireJS (или подобные) — зачем нужен?
* Тестирование: какие библиотеки знаешь (+/-), что использовал?
* Как логировать JS ошибки с клиента, какие "подводные" камни?
* Какие фреймворки знает/использовал (если не использовал, то почему), как они устроены внутри.

P.S. Также до собеседования, можно посмотреть акаунт на github.

##### 1

Представьте, что у нас есть квадратное поле-сетка, по которому ходит бродяга.  
Напишите объект «Бродяга», содержащий его координаты и четыре метода, реализующих шаг бродяги вверх, вниз, вправо и влево.  
Как реализовать в таком объекте цепочку вида:

vagabond.goUp().goLeft().goDown().goDown().goRight();

##### 2

Имеется массив произвольной длины, заполненный случайными значениями:

var list = [34, 1, 529, -78, 89.7, 1243];

необходимо найти значение максимального элемента.

##### 3

Необходимо отправить на сервер набор данных без перезагрузки страницы. Перечислите способы сделать это, назовите самый кроссбраузерный.

##### 4

В чем разница между ~~уткой~~ == и ===?

##### 5

Что такое Local Storage, Cookies и Session Storage? В чем разница между ними?

##### 6

@media only screen and (max-width: 1200px) {

}

Что это за конструкция? Зачем она может понадобиться?

##### 7

Чем в Git отличаются merge и rebase?

##### 8

Дано:

var a = new F();

var b = new F();

alert(a == b);

function F() {}

Сделать так, чтобы a == b возвращало true.

##### 9

Напишите результаты выполнения операций:

"1" + 2;

1 + "2";

null == undefined;

null == 0;

null > 0;

null >= 0;

"a" > "A";

##### 10

Дано:

var list = [1, 2, 3, 4, 5, 6];

function sum() {

var sum = 0;

for (i=0; i<arguments.length; i++) {

sum += arguments[i];

}

}

Подсчитать с помощью функции sum сумму элементов list.

##### 11

Представьте большую и сложную страницу, которая очень медленно загружается. Допустим, на ней плохо всё. Ваша задача — оптимизировать время загрузки. Ваши действия?

#### Заключение

Практически все задания нужно было выполнить устно или в старой доброй IDE «Paper & Pencil». Безбажного кода никто не требовал. Тех, что поадекватнее, в первую очередь, интересовал ход мысли, а не знание тонкостей технологии. Но, естественно, уровень знаний HTML, JS и CSS тоже имеет значение (а вы как хотели). Собеседования в некоторых случаях больше напоминают экзамен с хорошими преподавателями (так было в ITMozg и в Topface), в некоторых — приятную дружескую беседу с чаепитием (Яндекс, Radario), а в некоторых — вообще чёрт знает что (не буду называть компании, к тому же, они малоизвестны).

Если кому-нибудь из читателей после прочтения придёт в голову мысль «Оказывается, на собеседованиях даже в самые крутые компании бояться нечего. Работа в Яндексе и Топфэйсе — это вполне досягаемо», значит, я писал это не зря.

## Вопрос 1: Возможности

Рассмотрим следующий код:

JavaScript

|  |  |
| --- | --- |
| 1  2  3  4  5 | (function() {     var a = b = 5;  })();    console.log(b); |

Что будет написано на консоли?

### Ответ

Код выше напишет 5.

Хитрость этого вопроса заключается в том, что в IIFE есть два задания, но переменная a объявляется ключевым словом var. Это означает, что a является переменной функции. b же присвоена глобальной области.

Другой хитростью этого вопроса является то, что он использует строгий режим  ('use strict';) в функции. Если был включен строгий режим, код покажет ошибку “Uncaught ReferenceError: b не определён”. Помните, что строгий режим требует, чтобы вы ссылались на глобальные области. Таким образом, вы должны написать:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | (function() {     'use strict';     var a = window.b = 5;  })();    console.log(b); |

## Вопрос 2: Создание “родных” методов

Определите повторяющуюся функцию в строке объекта. Функция подразумевает целое число, определяющее, сколько раз строка должна повториться. Функция повторяет указанную строку указанное количество раз. Например:

|  |  |
| --- | --- |
| 1 | console.log('hello'.repeatify(3)); |

Должно получиться "hellohellohello".

### Ответ

Возможная реализация показана ниже:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | String.prototype.repeatify = String.prototype.repeatify || function(times) {     var str = '';       for (var i = 0; i < times; i++) {        str += this;     }       return str;  }; |

Вопрос проверяет знания разработчиков о наследовании в JavaScript и свойстве прототипа. Так же проверяется и способность разработчика расширять функциональность собственных типов данных (хотя это не обязательно).

Ещё одним важным моментом является демонстрация знаний о том, как не отменить уже определенные функции. Это делается путём тестирования, подтверждающего, что функция не существует самостоятельно до ее определения.

JavaScript

|  |  |
| --- | --- |
| 1 | String.prototype.repeatify = String.prototype.repeatify || function(times) {/\* code here \*/}; |

Этот метод особенно полезен, когда вас попросили заклинить функцию JavaScript.

## Вопрос 3: Подъём

Каков результат выполнения и почему?

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | function test() {     console.log(a);     console.log(foo());       var a = 1;     function foo() {        return 2;     }  }    test(); |

### Ответ

Результаты кода – underfined и 2.

Причина в том, что обе переменные функции подняты (помещены в верхней части функции), но переменные не удерживают присвоены значения. В то время, когда переменная a печатается, она существует в функции (это декларируется), но она все ещё undefined. Другими словами, код выше эквивалентен следующему:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | function test() {  var a;  function foo() {  return 2;  }    console.log(a);  console.log(foo());    a = 1;  }    test(); |

## Вопрос 4: Как это работает в JavaScript

Каков результат следующего кода? Обоснуйте свой ответ.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | var fullname = 'John Doe';  var obj = {     fullname: 'Colin Ihrig',     prop: {        fullname: 'Aurelio De Rosa',        getFullname: function() {           return this.fullname;        }     }  };    console.log(obj.prop.getFullname());    var test = obj.prop.getFullname;    console.log(test()); |

### Ответ

Код выдаст Aurelio De Rosa и John Doe. Причина в том, что контекст функции, вызываемый ключевым thisсловом, в JavaScript зависит от того, как именно вызывается функция, а не от того, как она определена.  
Вызов первых console.log ( ), getFullname ( ) вызывается функцией объекта obj.prop. Таким образом, контекст относится к последнему и возвращает “fullname” как свойство объекта. В противном случае, когдаgetFullname ( ) присваивается переменной, контекст относится к глобальному объекту. Это происходит потому, что тест устанавливается как свойство глобального объекта (window). По этой причине функция возвращает значение свойства fullnameиз window, которая в данном случае является кодом, устанавливаемым в первой строке фрагмента.

## Вопрос 5: call ( ) и apply ( )

Исправьте предыдущий вопрос так, чтобы последний console.log ( ) стал Aurelio De Rosa.

### Ответ

Вопрос может быть изменен, повлияв на контекст функции, используя функции call ( ) или apply ( ). Если вы не знаете их и их отличия, вам стоит прочесть статью. В коде ниже я буду использовать call ( ), но в этом случае применяется apply ( ), результат будет тот же:

|  |  |
| --- | --- |
| 1 | console.log(test.call(obj.prop)); |

## Заключение

В этой статье мы обсудили пять типичных вопросов, которые задают на собеседовании, чтобы проверить разработчика JavaScript. Фактические вопросы могут отличаться от собеседования, но понятия и темы, как правило, схожи. Я надеюсь, что вы хорошо провели время, тестируя свои знания. В случае, если вы не смогли ответить на все вопросы, не волнуйтесь: нет ничего, что изучение и опыт не могут исправить.  
Если у вас были другие вопросы на собеседовании, не стесняйтесь поделиться ими с нами. Это может помочь многим разработчикам.

**Java Script**.

1) Как перенаправить страницу?

|  |  |
| --- | --- |
| 1  2  3 | <script type=”text/javascript” language="javascript">      window.location = "http://myway-blog.ru";  </script> |

2) Сколько параметров можно передать функции?

**Сколько угодно.**

3) Как реализовано наследование в JS?

На уровне языка реализованно наследование на прототипах. В Java Script каждый объект может иметь ассоциацию с другим объектом – так называемый “прототип” (prototype). В случае, если поиск некоторого свойства (или метода) в исходном объекте заканчивается неудачно, интерпретатор пытается найти одноименное свойство (метод) в его прототипе, затем – в прототипе прототипа и т.д.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <script type=”text/javascript” language="javascript">      var Obj = {          x: "1"      } // создаем объект Obj и записываем в него свойство: х = 1      var MyType = function() {} // создаем пустой объект MyType      var MyType.prototype = Obj; // MyType наследует свойства от Obj      var newObj = new MyType();      alert(newObj.x); // выводится "1"  </script> |

4) Пара слов об объектах в JS.

Объекты (они же – ассоциативные массивы, хэши) и работа с ними в Java Script реализована не так, как в большинстве языков. Объект в Java Script представляет собой обычный ассоциативный массив или, иначе говоря “хэш”. Он хранит любые соответствия “ключ => значение” и имеет несколько стандартных методов.

5) Что представляет из себя метод объекта?

Метод объкта в Java Script – это просто функция, добавленная в ассоциативный массив.

6) Зачем в JS писать **VAR** перед переменной?

Если создать переменную через обычное присваивание – будет создана глобальная переменная. А вот если создать переменную с использованием слова **VAR**, тогда будет создана локальная переменная, которая перестает существовать после завершения работы функции.

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4** | **<script type=”text/javascript” language="javascript">**  **variable = 100; // глобальная переменная**  **var variable = 100; // локальная переменная**  **</script>** |

7) Как создать массив в JS?

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <script type=”text/javascript” language="javascript">      var array = ["Value 1", "Value 2", "Value 3"]; // 1 способ (заполенный массив)      var array = []; // 2 способ (пустой массив)      var array = new Array("Value 1", "Value 2", "Value 3"); // 3 способ (заполненный массив)      var array = new Array(); // 4 способ (пустой массив)  </script> |

8) Можно ли в JS использовать функцию в качестве конструктора?

**Можно.**

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4** | **<script type="text/javascript" language="javascript">**  **var A = function() {}**  **var myA = new A();**  **</script>** |

9)Технически, замыкание – это особый вид функции, которая определена в теле другой функции и создается каждый раз во время ее выполнения. Синтаксически это выглядит как функция, находящаяся целиком в теле другой функции и при этом вложенная внутренняя функция содержит ссылки на локальные переменные внешней функции. Трюк с замыканием происходит потому, что каждый раз при выполнении внешней функции происходит создание нового экземпляра внутренней функции с новыми ссылками на переменные внешней функции.

Чтобы было более понятно, о чем идет речь, давайте посмотрим на пример замыкания:

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10** | **function function\_counter() {**  **var count\_calls = 0;**  **return function() {**  **return ++count\_calls;**  **}**  **}**  **var count = function\_counter();**  **console.log(count()); // в консоли видим "1"**  **console.log(count()); // в консоли видим "2"**  **console.log(count()); // в консоли видим "3"** |

Мы создали функцию **function\_counter**, которая возвращает количество собственных вызовов. Далее мы определили переменную **count**, которой присвоилась ссылка на возвращаемую внутреннюю функцию функции **function\_counter** (произошло создание нового экземпляра внутренней функции с новыми ссылками на переменные внешней функции). При каждом вызове функции **count** происходит увеличение значения переменной **count\_calls**. Т.е. здесь произошло замыкание и значение переменной **count\_calls** сохранилось в памяти.

Только после удаления переменной **count**, которая ссылается на возвращаемую функцию, переменная **count\_calls** будет удалена сборщиком мусора.

Надеюсь, что теперь вам стало понятно, что такое замыкание и как оно технически устроено.

Замыкания чаще всего используют для назначения функций-обработчиков событий:

HTML

|  |  |
| --- | --- |
| **1**  **2** | **<div> <button id="button"> Скрыть красный квадрат </button> </div>**  **<div id="block"> </div>** |

CSS

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5** | **#block {**  **background: red;**  **height: 100px;**  **width: 100px;**  **}** |

JS

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9** | **function add\_hide\_handler(source, target) {**  **var source\_id = document.getElementById(source);**  **var handler = function() {**  **var target\_id = document.getElementById(target);**  **target\_id.style.display = "none";**  **}**  **source\_id.onclick = handler;**  **}**  **add\_hide\_handler("button", "block");** |

Ну и напоследок я бы хотел привести пару примеров, где используется замыкание:

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10** | **function output\_word() {**  **var word = "Квадрат";**  **var output = function() {**  **alert(word);**  **}**  **word = "Круг";**  **return output;**  **}**  **var func = output\_word();**  **func();** |

Как вы думаете, какое слово будет выводиться во всплывающем окне? Надеюсь, что вы правильно ответили на этот вопрос. А правильный ответ: “Круг”.

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6** | **function multiplication(a) {**  **return function(b) {**  **return a\*b;**  **}**  **}**  **alert(multiplication(5)(5));** |

*Функция выше выполняет умножение одного числа на другое с использованием трюка с замыканием.*

10) Функция будет возвращать GET-параметры в виде объекта JavaScript.

**Предлагаю сразу ознакомиться с конечным вариантом функции, а потом я вам расскажу, как она работает:**

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16**  **17**  **18**  **19**  **20**  **21**  **22**  **23**  **24**  **25**  **26**  **27**  **28** | **<!doctype html>**  **<html>**  **<head>**  **<meta charset="utf-8">**  **</head>**  **<body>**  **<script>**    **'use strict';**    **/\*\***  **\* Распарсить GET-параметры из адресной строки.**  **\***  **\* @return {object} GET-параметры в виде объекта.**  **\*/**  **function parseGETParameters() {**  **var result = {};**  **var gets = window.location.search.replace(/&amp;/g, '&').substring(1).split('&');**  **for (var i = 0; i < gets.length; i++) {**  **var get = gets[i].split('=');**  **result[get[0]] = typeof(get[1]) == 'undefined' ? '' : get[1];**  **}**  **return result;**  **}**    **</script>**  **</body>**  **</html>** |

Предположим, что адресная строка браузера содержит: **http://myway-blog.ru/?par\_1=1&par\_2=2&par\_3=3&amp;par\_4=4**. Нам нужно получить GET-параметры из этой строки в виде объекта JavaScript.

**В самом начале создадим объект result, в который далее запишем GET-параметры.**

|  |  |
| --- | --- |
| **1** | **var gets = window.location.search.replace(/&amp;/g, '&').substring(1).split('&');** |

**В строке выше происходит следующее:**

1. **window.location.search** содержит все GET-параметры в виде 1 строки. Результат: “**?par\_1=1&par\_2=2&par\_3=3&amp;par\_4=4**“.
2. Далее с помощью метода **replace()** мы сделаем глобальную замену всех HTML-представлений амперсанда (&amp; =&) на тот же самый амперсанд, только в привычном виде (&). Результат: “**?par\_1=1&par\_2=2&par\_3=3&par\_4=4**“.
3. С помощью метода **substring()** мы обрежем вопросительный знак в начале строки. Результат: “**par\_1=1&par\_2=2&par\_3=3&par\_4=4**“.
4. Метод **split()** преобразует строку в массив. В качестве разделителя мы использует знак амперсанда (&). Результат: **[“par\_1=1”, “par\_2=2”, “par\_3=3”, “par\_4=4”]**.

**Потом мы пробежим в цикле по получившемуся массиву.**

|  |  |
| --- | --- |
| **1** | **var get = gets[i].split('=');** |

Строка выше преобразует каждый элемент массива в отдельный массив. Например, строка “**par\_1=1**” станет массивом **[“par\_1”, “1”]**.

|  |  |
| --- | --- |
| **1** | **result[get[0]] = typeof(get[1]) == 'undefined' ? '' : get[1];** |

Как мы помним, в объекте **result** в конечном итоге должны находиться GET-параметры. На предыдущем шаге мы преобразовали строку “**par\_1=1**” в массив **[“par\_1”, “1”]**, т.е.: **get[0] = par\_1**,**get[1] = 1**.

Получается, что **get[0]** – GET-параметр и в **result** он записывается в качестве ключа объекта. А**get[1]** – значение GET-параметра. С помощью функции **typeof()** проверим тип элемента **get[1]**, чтобы убедиться в том, что значение **get[1]** не является **undefined**, но в случае, если все-таки**undefined**, то мы в **result** запишем пустую строку.

На выходе получаем объект: **Object {par\_1: “1”, par\_2: “2”, par\_3: “3”, par\_4: “4”}**.

Функцию использовать очень легко, например:

JavaScript

|  |  |
| --- | --- |
| 1  2 | **var getParams = parseGETParameters(); // распарсить GET-параметры из адресной строки**  **alert(getParams.par\_1); // выводится GET-параметр par\_1, т.е. значение 1** |

### Вопросы, относящиеся к общим знаниям в области JavaScript

**ООП в JavaScript**. Один из самых популярных вопросов, на мой взгляд. Время от времени можно услышать вариации а-ля «Что такое прототипное наследование», «Инкапсуляция в JavaScript». Информацию можно найти [тут](http://habrahabr.ru/company/enterra/blog/153365/) или [тут](http://javascript.ru/tutorial/object/inheritance).  
  
**Утечки памяти в JS**. Огромная тема, можно минут 10 рассказывать, мне кажется. Самым лаконичным примером я считаю var a = {}; a.a = a;. Такой код создаст утечку в IE <= 7 версии, т.к. осёл до 8 версии не умеет чистить циклические зависимости. Должно хватить для простой демонстрации понимания механизма работы с утечками. Далее будут вопросы а-ля «Как создать утечку, используя фреймворк N». В деталях можно прочитать [тут](http://learn.javascript.ru/memory-leaks).  
  
**Замыкания в JS**. Нет, ну правда, какое собеседование без вопроса о [замыканиях](http://habrahabr.ru/post/38642/)? Вопрос отчасти относится к предыдущему («Инкапсуляция в JavaScript»), но чаще всего рассматривается отдельно.  
  
**Самовызывающиеся функции**. Тоже весьма частый вопрос на собеседованиях. Ещё иногда подходят к этому со стороны jQuery: «Почему все плагины обособлены в конструкцию (function() {})();?».  
  
**Область видимости переменных. Ключевое слово «this»**. Да и, честно говоря, всё, что связано с [баблингом переменных и функций](http://habrahabr.ru/post/127482/). Мол, чем отличается запись var a = function() {}; от записи function a() {};. Рассказать про [this](http://javascript.ru/tutorial/object/thiskeyword).  
  
**Как поменять контекст функции. Частичное применение функции**. Ну, тут можно оставить без комментариев, всё упирается в знание разницы между .call, .apply и .bind и некоторых стандартных решений на уровне языка. Обо всём этом можно прочитать в[одной замечательной статье](http://habrahabr.ru/post/149581/).  
  
**Разница между операторами == и ===**. За ответом [сюда](http://habrahabr.ru/post/138272/)  
  
**Напишите регулярное выражение для проверки строк соответствующих формату даты dd.mm.yyyy**.

**Каков будет результат выполнения следующего кода:**

for (var i = 0; i < 10; i++) {

setTimeout(function () {

console.log(i);

}, 0);

}

В дополнение к вопросу хочу сказать, что обычно вторым вопросом идет «Как сделать так, чтобы выводило цифры по порядку?»  
  
**Что вернет выражение +new Date()? Чем отличается от Date.now()**. Отличный вопрос. Ответ в том, что +new Date(); создаст экземпляр объекта Date и благодаря + переведет его в числовой формат. Во втором случае вызовется статический метод конструктора, который является более приоритетным, т.к. во-первых он не требует создания экземпляра, а во-вторых является более понятным.  
  
**Напишите функцию принимающую строку с именем файла и возвращающую расширение (фрагмент после последней точки)**. Я просто оставлю [ссылку на ответ](http://habrahabr.ru/post/231071/#comment_7810791).  
  
**Производительность циклов**. Иногда (но далеко не всегда) такой вопрос имеет место. Лично мне он нравится. По крайней мере, человек должен знать, почему for быстрее, чем Array.prototype.forEach. Но желательно ознакомиться с информацией из [этой статьи](http://habrahabr.ru/post/49052/).  
  
На этом, как правило, стандартные вопросы заканчиваются. Дальше идут вопросы, касательно специфики вакансии. Я имею ввиду, более «профильные» для компании. Например, если требуется JS разработчик со знанием Backbone/Marionette(Chaplin), то следующий блок вопросов будет про jQuery, underscore/lodash и прочие связанные технологии, как бы очевидно это не звучало. Вопросов по данным тематикам море, но парочку любимых я всё-таки приведу:

### Вопросы, относящиеся к jQuery

**Почему перед исходным кодом jQuery стоит восклицательный знак?**   
  
**Расскажите про $.Deferred**. Могу посоветовать одно: кроме знания [$.Deferred](http://habrahabr.ru/post/113073/), было бы неплохо упомянуть о [нативных Promise'ах](http://habrahabr.ru/post/209662/)  
  
**Как создать утечку памяти с помощью jQuery**.   
Я люблю простые примеры, поэтому обычно привожу следующее:

var jqSelector = $("#selector"),

nativeSelector = document.getElementById("selector");

// Удаляем элемент "нативным" способом

nativeSelector.parentNode.removeChild(nativeSelector);

// Выводим закешированное значение селектора jQuery

console.log(jqSelector); // Привет, jQuery.cache!

А вообще, в части вопросов о нативном JS уже было похожее, и там же приведена ссылка на примеры с использованием jQuery.  
  
**jQuery.extend**. Рассказать всё про функцию, возможно попросят написать аналог. Кстати, возможно будет полезно прочитать статью про [примеси в JS](http://habrahabr.ru/post/147901/).  
  
**Составьте AJAX-запрос на сервер, используя jQuery**. Не особо популярный вопрос, но если спрашивать нечего, спросят что-то вроде него. Разумеется, вы должны знать, как использовать функции $.ajax, $.get, $.post и устанавливать настройки в$.ajaxSettings. Всё это уже [давно есть на хабре](http://habrahabr.ru/post/14246/).  
  
Остальные вопросы скорее «для массовки», я считаю, что о jQuery больше нечего спрашивать. Конечно, будут ещё вопросы типа «Как добавить класс к объекту, используя jQuery», но я думаю, это не будет неожиданностью и не вызовет проблем.

### Вопросы, относящиеся к Underscore/Lodash

Такие встречаются, честно говоря, очень редко. Но всё же, приведу краткий список того, что я могу вспомнить:  
  
**Чем \_.throttle отличается от \_.debounce.** Опять же, ответ есть [на хабре](http://habrahabr.ru/post/60957/).  
  
**Как работает функция \_.extend**. Аналог вопроса про jQuery.extend, ссылка на доку выше.  
  
**Расскажите о шаблонизаторе Underscore**. Кстати, не исключено, что это только начало, и вас будут спрашивать про шаблонизатор jQuery и все остальные. Я бы советовал хоть одним глазом прочитать [эту статью](http://habrahabr.ru/post/219873/).  
  
**Lodash vs Underscore**. Вопрос для разведения холивара на собеседованиях

1. Напишите функцию inArray проверяющую вхождение элемента в массив и возвращающую значение типа Boolean.

2. Что вернет выражение +new Date()? Чем отличается от Date.now()?

3. Напишите функцию принимающую строку с именем файла и возвращающую расширение (фрагмент после последней точки).

4. Каков будет результат выполнения следующего кода:

for (var i = 0; i < 10; i++) {

setTimeout(function () {

console.log(i);

}, 0);

}

5. Напишите регулярное выражение для проверки строк соответствующих формату даты dd.mm.yyyy.

6. Опишите разницу между операторами == и ===. Когда имеет смысл использовать каждый из вариантов?

### Общие вопросы: [[↑]](http://ymatuhin.ru/front-end/front_end_developer_interview_quersions/#toc)

* Вы пользуетесь Твиттером?
  + Если да, то кто присутствует в вашей ленте?
* Кого из мировых front-end разработчиков вы знаете?
  + А российских front-end’щиков?
* У Вас есть аккаунт на GitHub?
  + Если да, то за какими интересными проектами вы следите?
* Какие блоги Вы читаете?
* Какие системы управления версиями вам приходилось использовать?
* Расскажите о своей среде разработки (ОС, редактор, браузер(ы), прочие инструменты)
* Опишите последовательность Ваших действий, когда вы создаете новую Web-страницу
* Можете ли пояснить разницу между progressive enhancement и graceful degradation?
  + Бонус, если также расскажете про feature detection (определение возможностей браузера)
* Объясните, что означает «Семантическая разметка»
* Какой Ваш основной браузер для разработки и какими инструментами Вы в нем пользуетесь?
* Как можно оптимизировать загрузку внешних ресурсов на странице?
  + Комбинация из потенциальных решений:
    - Конкатенация
    - Минификация
    - Использование CDN
    - Кэширования
    - и т.д.
* Каково преимущество в подгрузке внешних ресурсов с нескольких доменов?
  + Сколько ресурсов браузер может одновременно качать с одного домена?
* Назовите три способа уменьшения времени загрузки страницы (воспринимаемого или реального)
* Если Вы присоединились к проекту, где для форматирования используются табы, а Вы привыкли использовать пробелы, как Вы поступите?
  + Предложите использовать EditorConfig ([editorconfig.org](http://editorconfig.org/))
  + Останетесь верным своим привычкам
  + Выполните команду :retab!
* Используете ли приёмы для ускорения написания кода (сниппеты, Emmet)?
* Знакомы ли с шаблонизаторами (Smarty, HAML etc.)
* Реализуйте примитивное слайд-шоу
  + Бонус, если Вы это сделаете без использования JS
* Какие инструменты Вы используете для тестирования производительности кода?
  + JSPerf ([jsperf.com](http://jsperf.com/))
  + Dromaeo ([dromaeo.com](http://dromaeo.com/))
  + и т.д.
* Если бы у Вас была возможность освоить новую технологию в этом году, что бы это было?
* Объясните важность стандартов и комитетов по стандартам
* Что такое FOUC (Flash Of Unstyled Content)? Как его избежать?

### Вопросы по HTML: [[↑]](http://ymatuhin.ru/front-end/front_end_developer_interview_quersions/#toc)

* Для чего нужен doctype и сколько разновидностей Вы можете назвать?
* В чем разница между standards mode и quirks mode?
* Какие ограничения накладывают стандарты XHTML?
  + Могут ли возникнуть проблемы при подаче страниц с типом application/xhtml+xml?
* Как следует оформлять страницу, в которой контент может быть на разных языках?
  + Что нужно иметь в виду при разработке многоязычных сайтов?
* Можно ли использовать синтаксис XHTML в HTML5?
* Как использовать XML в HTML5?
* Чем полезны data- атрибуты?
* Если рассматривать HTML5 как открытую web-платформу, на чем она строится, из каких компонентов состоит?
* Объясните разницу между cookies, sessionStorage и localStorage.
* Знакомы ли с подводными камнями вёрстки почтовых шаблонов?
* Какая разница между GET и POST?

### Вопросы по JavaScript: [[↑]](http://ymatuhin.ru/front-end/front_end_developer_interview_quersions/#toc)

* Какими js-библиотеками Вы пользовались?
* Вы когда-нибудь заглядывали в исходный код библиотек/фреймворков, которыми пользовались?
* Чем JavaScript отличается от Java?
* Что-такое хэш-таблица?
* Что такое неопределенные (undefined) и необъявленные (undeclared)переменные?
* Что такое замыкание и как/для чего его используют?
  + Как вы предпочитаете их использовать?
* Где обычно используются анонимные функции?
* Объясните «JavaScript module pattern» и где он (паттерн) применяется
  + Бонус, если упомянута чистота глобального пространства имен
  + Что, если Ваш модуль не заключен в пространство имен?
* Как Вы организуете свой код? (module pattern, наследование)
* В чем разница между host-объектами и нативными объектами?
* В чем разница между последними двумя строчками:

**function** Person(){}

**var** person **=** Person()

**var** person **=** **new** Person()

* В чем разница между .call и .apply?
* Что делает и для чего нужна функция Function.prototype.bind?
* Когда Вы оптимизируете свой код?
* Объясните, как работает наследование в JavaScript?
* Где до сих пор используется document.write()?
  + В большинстве генерируемых баннеров, хотя так делать не рекомендуется
* В чем разница между feature detection (определение возможностей браузера), feature inference (предположение возможностей) и анализом строки user-agent?
* Расскажите об AJAX как можно более подробно
* Объясните, как работает JSONP (и почему это не настоящий AJAX)
* Вы когда-нибудь использовали шаблонизацию на JavaScript?
  + Если да, какие библиотеки использовали? (Mustache.js, Handlebars etc.)
* Объясните, что такое «hoisting»
* Объясните event bubbling
* В чем разница между «атрибутом» (attribute) и «свойством» (property)?
* Почему не следует расширять нативные JavaScript объекты?
* Почему следует расширять нативные JavaScript объекты?
* В чем разница между событиями document load и document ready?
* В чем разница между == и ===?
* Как получить параметры из URL’а текущего окна?
* Объясните same-origin policy в контексте JavaScript
* Объясните event delegation
* Какие Вы знаете паттерны организации наследования в JavaScript?
* Сделайте так, чтобы этот код работал:

[1,2,3,4,5].duplicator(); *// [1,2,3,4,5,1,2,3,4,5]*

* Опишите принцип мемоизации (избежание повторных вычислений) в JavaScript
* Почему тернарный оператор так называется?
* Что такое арность функции?
* Что делает строчка "use strict";? Какие достоинства и недостатки от её использования?

### Примеры кода на JavaScript [[↑]](http://ymatuhin.ru/front-end/front_end_developer_interview_quersions/#toc)

**~~**3.14

Вопрос: Какое значение возвращает данное предложение? **Ответ: 3**

"i'm a lasagna hog".split("").reverse().join("");

Вопрос: Какое значение возвращает данное предложение? **Ответ: «goh angasal a m’i»**

( window.foo **||** ( window.foo **=** "bar" ) );

Вопрос: Чему равно window.foo? **Ответ: «bar»**, только если выражение window.foo было ложным, иначе переменная сохранит своё изначальное значение

**var** foo **=** "Hello";

(**function**() {

**var** bar **=** " World";

alert(foo **+** bar);

})();

alert(foo **+** bar);

Вопрос: Что покажут эти два alert? **Ответ: «Hello World» и ReferenceError: bar is not defined**

**var** foo **=** [];

foo.push(1);

foo.push(2);

Вопрос: Чему равно foo.length? **Ответ: 2**

**var** foo **=** {};

foo.bar **=** 'hello';

Вопрос: Чему равно foo.length? **Ответ: undefined**

### Вопросы по jQuery: [[↑]](http://ymatuhin.ru/front-end/front_end_developer_interview_quersions/#toc)

* Объясните «chaining».
* Объясните «deferreds».
* Какие Вы знаете приемы оптимизации кода, использующего jQuery?
* Что делает .end()?
* Как добавить пространство имён к обработчику событий? Для чего это нужно?
* Назовите 4 разных вида аргументов, которые принимает функция jQuery()?
  + Селектор (строка), HTML (строка), Callback (функция), HTMLElement, объект, массив, массив элементов, объект jQuery etc.
* Что такое очередь эффектов (fx queue)?
* В чем разница между .get(), [], и .eq()?
* В чем разница между .bind(), .live(), и .delegate()?
* В чем разница между $ и $.fn? Что вообще такое $.fn?
* Оптимизируйте данный селектор:

$(".foo div#bar:eq(0)")

### Вопросы по CSS: [[↑]](http://ymatuhin.ru/front-end/front_end_developer_interview_quersions/#toc)

* Что такое «reset» CSS и для чего он нужен?
* Объясните, что такое плавающие элементы (floats) и как они работают?
* Какие вы знаете методы запрета обтекания (clearing) и какие где применяются?
* Что такое CSS спрайты? Как они реализуются на странице или сайте?
* Какие Ваши любимые методы подмены текста картинкой (image replacement) и когда Вы их используете?
* CSS property hacks, conditionally included .css files…
* Как Вы обеспечиваете отображение страниц в старых/ограниченных браузерах?
  + Какие приёмы/процессы Вы при этом используете?
* Какими способами можно визуально скрыть элемент (оставив его доступным для экранного диктора, screen reader)?
* Вы когда-нибудь использовали сеточную верстку (grid system, grid design)? Если да, какая Ваша любимая?
* Приходилось ли вам использовать или реализовывать media queries или верстку под мобильные устройства?
* Приходилось ли вам стилизовать SVG?
* Как оптимизировать страницы для печати?
* Какие есть подводные камни в оптимизации производительности CSS?
* Вы используете CSS препроцессоры? (Sass, Compass, Bourbon, Stylus, LESS)
  + Если да, расскажите, что вам в них нравится и не нравится?
* Как Вы сверстаете макет, который использует нестандартные шрифты?
  + Webfonts (сервисы вроде Google Webfonts, Typekit, Fontsquirrel etc.)
* Объясните, как браузер определяет, на какие элементы накладывать CSS стили?
* Приходилось ли сталкиваться с Retina-дисплеями?
* Слышали ли о системе БЭМ (Блок-Элемент-Модификатор)?
  + Если да, то в чем её суть? Расскажите о системе нотификации селекторов.

### «Светская беседа»: [[↑]](http://ymatuhin.ru/front-end/front_end_developer_interview_quersions/#toc)

* Самое крутое, что Вы когда либо делали и чем гордитесь?
* Вы знаете секретный жест HTML5-банды?
* ([непереводимый юмор](https://vimeo.com/18848658)) Are you now, or have you ever been, on a boat.
* Что Вы больше всего любите в Ваших инструментах разработки?
* У Вас есть какие-нибудь личные проекты?
* Возьмите листок бумаги и напишите в столбик буквы A B C D E. Теперь отсортируйте столбик в алфавитном порядке по убыванию, не написав ни строчки кода.
  + Засеките, через сколько времени кандидат перевернет листок
* Пират или ниндзя?
  + Бонус за комбинацию. Аргументированную. +2 за зомби-пират-ниндзя-обезьяну
* Чем бы Вы занимались, если бы не Web-разработкой?
* Какая Ваша любимая «фишка» Internet Explorer?
* Закончите предложение: Brendan Eich и Doug Crockford являются**\_\_**\_\_ языка JavaScript.
* jQuery: хорошая библиотека или великая библиотека? Тема для дискуссии…

**Общие вопросы:**  
  
Что вы изучили вчера / на прошлой неделе?  
  
Что для вас наиболее интересно или захватывающе в программировании?  
  
Каким было последнее испытание ваших технических навыков и как вы нашли решение?  
  
На какие моменты, связанные с пользовательским интерфейсом, производительностью, SEO, лёгкостью сопровождения или технологичностью вы обращаете внимание при построении веб-приложения или сайта?  
  
Расскажите о самой удобной, на ваш взгляд, среде разработки (ОС, редактор или IDE, браузеры, инструменты, и т.д.).  
  
С какими версиями систем управления версиями вы знакомы?  
  
Опишите ваш рабочий процесс при создании веб-страницы?  
  
Если у вас 5 различных таблиц классов, как лучше всего интегрировать их в сайт?  
  
Опишите различие между прогрессивным улучшением и постепенным сокращением функциональности?  
  
Как бы вы оптимизировали цифровые объекты / ресурсы сайта?  
  
Почему традиционно считается, что лучше управлять цифровыми объектами с разных доменов?  
  
Как много ресурсов может загрузить браузер с указанного домена за один раз?  
  
Какие бывают исключения?  
  
Назовите 3 способа ускорить загрузку страницы (расчетное и действительное время загрузки).  
  
Если вы начинаете работать с существующим проектом, и в нем использовались табы, в то время как вы использовали пробелы, что вы будете делать?  
  
Напишите простую страницу с показом слайдов.  
  
Какие инструменты вы используете для проверки работоспособности вашего кода?  
  
Если бы вы могли освоить одну технологию в этом году, что бы вы выбрали?  
  
Какое различие между Long-Polling, Websockets и SSE?  
  
Объясните важность стандартов и органов по стандартизации.  
  
Что такое появление неоформленного контента (FOUC)? Как его избежать?  
  
Постарайтесь описать процесс, происходящий от момента набора URL веб-сайта до окончания загрузки страницы на экран.  
  
Объясните, что такое ARIA и программы экранного доступа, и как сделать веб-сайт доступным.  
  
Расскажите о некоторых преимуществах и недостатках CSS-анимации относительно JavaScript-анимации.   
  
Опишите нижеперечисленные заголовки запросов и откликов:  
  
Различие между:  
— Expires, Date, Age и If-Modified-…  
— DNT  
— Cache-Control  
— Transfer-Encoding  
— ETag  
— X-Frame-Options  
  
**Вопросы по HTML:**  
  
Какую функцию выполняет doctype?  
  
В чем различие между режимом стандартов и режимом совместимости?  
  
В чем состоят ограничения при работе со страницами XHTML?  
  
Существуют ли трудности при работе с такими страницами как application/xhtml+xml?  
  
Как работать со страницей, на которой присутствует контент на нескольких языках?  
  
О чем следует помнить при оформлении или разработке многоязычных сайтов?  
  
Для чего рекомендуется использовать атрибуты data-?  
  
Представьте себе HTML5 как открытую веб-платформу. Что является строительными блоками HTML5?  
  
Опишите различие между cookies, sessionStorage и localStorage.  
  
Можете ли вы объяснить различие между GET и POST?  
  
Объясните различие между

<script>, <script async> и <script defer>

**Вопросы по CSS:**  
  
В чем различие между классами и идентификаторами (ID) в CSS?  
  
Расскажите о функции «reset»-файла в CSS и для чего его можно использовать.  
  
Опишите функции Float и принцип их работы.  
  
Расскажите о функции z-index и формировании контекста наложения.  
  
Расскажите о различных методиках очистки и о принципах выбора подходящей для конкретного контекста?  
  
Расскажите о CSS-спрайтах и порядке их применения на странице или сайте.   
  
Какой ваш любимый метод замены изображений, и в каких случаях вы используете другие?  
  
CSS-хаки, conditionally included или… что-нибудь еще?  
  
Как вы работаете со страницами в браузерах с ограниченной функциональностью?  
  
Какие методики / процессы вы используете?  
  
Назовите несколько различных способов визуально скрыть контент (и сделать его доступным только для программ экранного доступа)?  
  
Использовали ли вы сеточную верстку, и, если да, какую вы предпочитаете?  
  
Использовали или внедряли ли вы медиа-запросы или макеты / CSS для мобильных устройств?  
  
Знакомы ли вы с оформлением SVG?  
  
Как вы оптимизируете страницы для печати?  
  
В чем заключаются «подводные камни» при написании CSS?  
  
В чем заключаются преимущества / недостатки использования препроцессоров CSS? (Sass, Compass, Stylus, LESS)  
  
Расскажите, что вам нравится и не нравится в препроцессорах CSS, которые вам довелось использовать.  
  
Как бы вы внедряли макет веб-дизайна, использующий нестандартные шрифты?  
  
Расскажите, как браузер определяет то, какие элементы соответствуют селектору CSS.  
  
Расскажите, что в вашем понимании означает блочная модель, и как объяснить браузеру при помощи CSS, как отобразить макет в различных блочных моделях.  
  
Какую функцию выполняет \* { box-sizing: border-box; }? В чем преимущества данного правила?  
  
Перечислите как можно больше значений свойств отображения, которые можете вспомнить.  
  
В чем различие между inline и inline-block?   
  
В чем различие между относительным, закрепленным, абсолютным и статическим элементами?  
  
Буква 'C' в CSS означает «каскадные» (cascading). Как назначается приоритет в назначаемых стилях (приведите несколько примеров)? Как можно использовать данную систему для своего удобства?  
  
Какие существующие программные оболочки для написания CSS вы использовали самостоятельно или на работе? (Bootstrap, PureCSS, Foundation и т.д.)  
  
Какие именно на работе, а какие самостоятельно? Как бы вы улучшили / доработали их, если бы была такая возможность?  
Работали ли вы с новыми спецификациями CSS Flexbox или Grid?  
  
Чем отличается отзывчивый дизайн от адаптивного?  
  
Работали ли вы когда-нибудь с retina-графикой? Если да, когда и какие методики вы использовали?  
  
**Вопросы по JS:**  
  
Расскажите о делегировании событий.   
  
Расскажите, как работает функция this в JavaScript.  
  
Расскажите о принципах работы наследования через прототипы.  
  
Как вы выполняете проверку скриптов JavaScript?  
  
AMD или CommonJS?  
  
Объясните, почему следующий код не будет работать в IIFE: 

function foo(){ }();

Что необходимо изменить, чтобы код начал работать в IIFE?  
  
Какое различие между состояниями переменных: null, undefined и undeclared?  
  
Как производится проверка наличия одного из указанных состояний?  
  
Что такое замыкания, и как и где они используются?  
  
Где чаще всего используются анонимные функции?  
  
Как вы организовываете свой код? (модульная архитектура, классическое наследование?)  
  
В чем различие между серверными объектами и собственными объектами?  
  
В чем различие между: function Person(){}, var person = Person(), и var person = new Person()?  
  
В чем различие между .call и .apply?  
  
Объясните Function.prototype.bind.  
  
Когда вы оптимизируете свой код?  
  
Когда бы вы использовали document.write()?  
  
В чем различие между обнаружением признаков, выводом признаков и использованием строки UA?  
  
Как можно более подробно расскажите об AJAX.  
  
Расскажите о принципах работы JSONP (и почему, на самом деле, это не AJAX).  
  
Использовали ли вы шаблоны JavaScript?  
  
Если да, то какие библиотеки вы использовали? (Mustache.js, Handlebars и т.д.)  
  
Объясните понятие «поднятие».  
  
Опишите event bubbling.  
  
В чем различие между «атрибутом» и «свойством»?  
  
Почему расширение, созданное при помощи объектов JavaScript – не лучшее решение?  
  
В чем различие между событием загрузки документа и событием готовности документа?  
  
В чем различие между == и ===?  
  
Расскажите о принципе одинакового источника применительно к JavaScript.  
  
Выполните данную работу:

duplicate([1,2,3,4,5]); // [1,2,3,4,5,1,2,3,4,5]

Почему это выражение называется «тернарным», и что означает слово «тернарный»?  
  
Что такое «use strict»;? В чем преимущества и недостатки использования данного выражения?  
  
Создайте цикл с оператором for с количеством итераций 100 с выводом «fizz» в итерациях с номером, кратным 3, «buzz» в итерациях с номером, кратным 5 и «fizzbuzz» в итерациях с номером, кратным и 3, и 5.  
  
Почему в общем случае рекомендуется оставлять глобальную область видимости веб-сайта нетронутой и вообще не менять ее?  
  
**Вопросы по jQuery:**  
  
Объясните понятие «цепочка».  
  
Объясните понятие «deferreds».  
  
Какие особые методы оптимизации, связанные с jQuery, вы можете применять?  
  
Какую функцию выполняет .end()?  
  
Назовите 4 различных значения, которые вы можете передать в jQuery.  
  
Селектор (строка), HTML (строка), Callback (функция), элемент HTML, объект, массив, массив элементов, объект jQuery, и т.д.  
  
В чем различие между .get(), [], и .eq()?  
  
**Вопросы по написанию кода:**  
Вопрос: Как бы вы выполнили следующее?

add(2, 5); // 7

add(2)(5); // 7

Вопрос: Какое значение возвращает выражение?

"i'm a lasagna hog".split("").reverse().join("");

Вопрос: Каково значение window.foo?

( window.foo || ( window.foo = "bar" ) );

Вопрос: Каков результат двух сигнализаторов, приведенных ниже?

var foo = "Hello";

(function() {

var bar = " World";

alert(foo + bar);

})();

alert(foo + bar);

Вопрос: Каково значение foo.length?

var foo = [];

foo.push(1);

foo.push(2);

**Вопросы личного характера:**  
— Над каким наиболее интересным проектом вы работали в последнее время?  
— Что вам нравится в инструментах, которые вы используете?  
— Есть ли у вас собственный домашний проект? Какого рода?  
— Какая ваша любимая особенность Internet Explorer?  
— Какой кофе вы предпочитаете?

**PHP**

1) Что такое PHP?

**PHP** – скриптовый язык программирования общего назначения, применяемый для разработки веб-приложений. Это язык является одним из лидеров среди языков программирования, применяющихся для создания динамических веб-приложений и сайтов. Также, PHP поддерживается подавляющим большинством хостинг-провайдеров.

2) Что такое динамические переменные?

Это переменные, у которых **имя хранится в самой переменной**. Их часто еще называют “переменные переменных”.

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5** | **<?php**  **$var = "first";**  **$$var = "Second";**  **// $$var == $first == "Second"**  **?>** |

**$$var** – динамическая, ее имя может меняться вместе с изменением **$var**. Также, можно связать имя переменной с содержимым другой переменной неявно:

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6** | **<?php**  **$first = "second";**  **$second = "third";**  **print $first; // напечатается "second"**  **print $$first; // напечатается "third"**  **?>** |

3) Каким способом можно перенаправить страницу в PHP?

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4** | **<?php**  **$url = "http://myway-blog.ru";**  **header("Location: ".$url);**  **?>** |

 4) Что такое тип данных?

**Тип данных** – описание, определяющее свойства и порядок обработки данных.

5) Что можете сказать про типизацию данных в PHP?

**PHP** – язык программирования с динамической типизацией данных, не требующий указания типа при объявлении переменных. Преобразования между скалярными типами зачастую осуществляются неявно без дополнительных усилий. Однако, PHP обладает возможностями для явного преобразования типов данных.

6) Какие типы данных знаете в PHP?

**PHP поддерживает 8 базовых типов данных.**

***4 скалярных типа: BOOLEAN, INTEGER, FLOAT, STRING.***

1. **BOOLEAN**: логический тип данных. Принимаемые значения: **TRUE** или **FALSE**.
2. **INTEGER**: целочисленный тип данных. Переменные принимают значения в диапазоне от **-2^31**до **+2^31**. Если значение превышает данный порог – оно автоматически переводится в тип**FLOAT**.
3. **FLOAT**: числовой тип данных с плавающей точкой. Переменные могут содержать как целые, так и дробные величины.
4. **STRING**: строковый тип данных. Содержит нефиксированное количество различных символов. PHP не накладывает никаких ограничений на длину строки.

***2 комплексных (составных) типа: ARRAY, OBJECT.***

1. **ARRAY**: массив, содержащий упорядоченный список элементов.
2. **OBJECT**: объект, содержащий некий объект (экземпляр класса).

***2 специальных типа: RESOURCE, NULL.***

1. RESOURCE: ссылка на абстрактный элемент. Например, ссылка на файл или ссылка на результат выполнения запроса.
2. NULL: пустой тип данных, обозначающий отсутствие какого-либо значения.

7) Что такое STATIC-функция и чем она отличается от “обычной” (не STATIC)?

**STATIC** принадлежит классу, а не экземпляру класса. Следовательно, такая функция вызывается у класса, а не у объекта, т.е. напрямую.

8) Есть ли разница между SELF и THIS в PHP?

**SELF**  используется для статических фукнций и членов класса, а  **THIS**  для нестатических.

9) Что такое конструктор?

**Конструктор** используется для выполнения различных автоматических инициализаций, например, инициализация свойств. Конструктор – это метод **\_\_CONSTRUCT()**, который автоматически вызывается ключевым словом **NEW** после создания объекта. Конструктор также может принимать аргументы.

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8** | **<?php**  **class MyClass {**  **public function \_\_construct() {**  **echo "Вызов конструктора!";**  **}**  **}**  **$myObject = new MyClass();**  **?>** |

10) Поддерживает ли PHP множественное наследование?

**Не поддерживает.**

11) Что делает функция EVAL()?

**EVAL()** вычисляет строку как PHP-код.

12) Что такое рекурсия?

**Рекурсия** – вызов функции из нее же самой (простая рекурсия) или через другие функции (сложная или косвенная рекурсия). Например, функция **A** вызывает фукнцию **B**, а функция **B** вызывает функцию **A**. Количество вложенных вызовов функции или процедуры называется глубиной рекурсии.

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11** | **<?php**  **function fac($x) {**  **if ($x === 0) {**  **return 1;**  **}**  **else {**  **return $x\*fac($x-1);**  **}**  **}**  **echo fac(4); // вычисление факториала числа 4**  **?>** |

13) Есть ли разница между одинарными и двойными кавычками в PHP?

**В двойных** кавычках данные **“парсятся”,** а в одинарных – нет.

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5** | **<?php**  **$beer = 'Chimay';**  **echo "$beer"; // напечатается "Chimay"**  **echo '$beer'; // напечатается "$beer"**  **?>** |

14) Как вывести все элементы массива на экран?

Вывести с ключами можно с помощью функции **PRINT\_R()**.

А пройти и вывести массив:

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6** | **<?php**  **$cars = array("BMW", "Audi", "Mercedes", "Porsche");**  **foreach ($cars as $car) {**  **echo $car."<br />";**  **}**  **?>** |

15) Что такое ассоциативный массив?

Массивы, **индексами которых являются строки**, называются ассоциативными. Индексы ассоциативных массивов называются ключами.

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6** | **<?php**  **$people = array();**  **$people["Иванов"] = "Иван";**  **$people["Сидоров"] = "Николай";**  **$people["Петров"] = "Петр";**  **?>** |

16) Что такое магические методы?

Это методы, зарезервированные в PHP, которые начинаются с двойного подчеркивания “**\_\_**“.

Хочу поделиться своим опытом прохождения собеседований на должность PHP Backend Developer в Германии. Собеседований, в свое время, пришлось пройти немало. Но о двух наиболее запоминающихся хочется рассказать, возможно, кому-то и пригодится в будущем мой опыт.

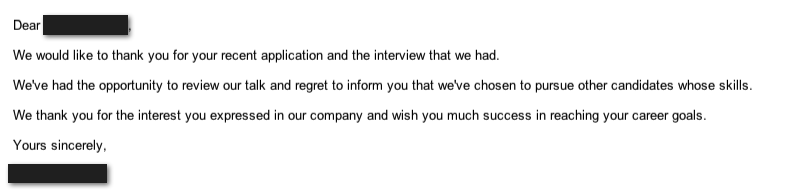
#### Компания 1.

Основное направление деятельности — разработка сайтов.   
Кол-во сотрудников: более 30.

##### **Описание должности:**

1. Хорошие знания PHP и многолетний опыт работы  
2. Прагматичное мышление, ориентация на результат  
3. SOLID, DRY, YAGNI vs. BUFD  
4. Понимание смысла TDD и BDD и умение их применять  
5. Понимание технологий HTTP, Frameworks, Patterns  
6. Умение объяснять принятые технологические решения  
7. Высокий уровень мотивации и желание работать именно в данной компании

##### **Собеседование**

Первым шагом было отправление резюме и мотивационного письма. В подробности вдаваться не буду, но если кому интересно, могу в отдельном посте написать о негласных правилах написания резюме в Германии.  
  
Через 3 дня пришел ответ с приглашением прийти на собеседование в офис. Назначили дату, предупредили, что вся процедура займет примерно полдня. Сначала фраза «полдня» меня смутила, потом, оказавшись непосредственно на месте все встало на свои места. Об этом подробнее:  
  
Началось все с небольшого интервью ( 30 мин.)  
  
**Вопросы, которые задавали**:  
  
Откуда вы узнали о нашей компании?  
Какие у вас ожидания от предстоящей работы?  
Есть ли у вас опыт работы в компаниях с плоской иерархией?  
Почему вы не хотите открыть свою IT-фирму? (в моем резюме были указаны 3 года фриланса)  
С какими языками программирования вы работали и почему именно остановились на PHP?  
Что знаете об Agile?  
Что такое SOLID и каковы основные принципы?  
Если бы вы придумали свой собственный язык программирования, какие фичи вы бы переняли из других языков?  
  
Еще было очень много технических вопросов, подробности, увы, не помню. Обсуждались конкретные ситуации, как бы вы написали код при… или в случае… Попросили написать алгоритм к заданию с фибоначи.   
  
Затем небольшая кофейная пауза и экскурсия по офису. (20 мин.)  
Далее посадили в пару к одному из программистов и дали задание, которое мы с ним вместе должны были сделать. (1,5 часа)  
Обеденный перерыв со всей командой разработчиков (1 час)  
Затем попросили дать отзыв, как, на мой взгляд, прошло собеседование, как мне понравилась фирма и не пропало ли желание в ней работать. (10 мин.) Попрощались, обещали связаться со мной в течение 2ух недель.  
  
Через неделю ровно мне позвонили и спросили, какую зарплату я себе представляю. После того, как цифра была названа, вежливо поблагодарили и обещали написать в ближайшее время.   
На следующий день пришел очень вежливый отказ, где было сказано, что решение было принято в пользу другого кандидата.   
Кстати, частенько приглашения на интервью и ответы приходят на английском, а само собеседование на немецком.  
  


#### Компания 2.

Основное направление деятельности — разработка онлайн-игр.   
Кол-во сотрудников: более 300.

##### **Описание должности:**

1. PHP 5.4  
2. Желателен опыт работы в game development  
3. MySQL  
4. Знание фрэймвокров аля ZF2  
5. Опыт в тестировании  
6. Знание REST и RPC  
7. Осознание, что Linux это не животное  
8. Особая любовь с HTML/CSS и JavaScript (jQuery)  
9. Понимание разницы между ‘agile’ и ‘хаос’  
10. Свободное владение английским и немецким языками

##### **Собеседование**

На этот раз нашли меня, нашли через немецкую сеть xing.com. Вкратце описали должность и требования и в случае заинтересованности попросили отправить резюме.  
Game Development – штука интересная, поэтому не долго думая резюме было отослано. Единственное, где оставались сомнения — это знания английского. Читаю и понимаю все, но не говорю.  
На следующий день звонок с приглашением на личную встречу.  
  
На самом собеседовании присутствовали 3 человека: HR-работник, который меня нашел в Xing, руководитель отдела разработки и рядовой PHP-Developer.  
  
Интервью. (30 мин.)  
  
**Вопросы, которые задавали**:  
  
Есть ли у вас опыт работы в Game Development?  
Почему больше нравится Backend, а не Frontend?  
Почему вы хотите уйти из фирмы, где проработали 9 лет?  
Как вы относитесь к внеурочной работе?  
Чем вы занимаетесь в свободное время?  
В какой области вы считаете себя наиболее успешным, а в какой области есть потенциал роста?  
Кем вы видите себя через 10 лет?  
  
В конце интервью руководитель отдела разработки сказал, что он заинтересован в совместной работе. Сказал, что в ближайшее время мне пришлют тестовое задание, с которым надо будет справится в течение максимум 2 часов.  
А вот и само задание:  
  
Имплементировать систему комментариев с PHP и MySQL. Рядом со статьей пользователь может писать комментарии, которые в дальнейшем также могут быть откомментированы им самим и другими пользователями. Регистрацию и Правила пользования можно не учитывать. Следующие пункты обязательны:  
  
· Акцент на функциональность, а не на дизайн  
  
· Таблицы в БД должны быть понятны и просто оформлены  
  
· Комментарий должен содержать Имя, Email и Текст  
  
· Внешний вид комментария на усмотрение  
  
· В случае добавления ссылки в комментарий, она должна быть рабочая и кликабельная автоматически  
  
· Для избежания спама имплементировать Капчу  
  
· Новые комментарии должны автоматически публиковаться на странице сайта (а не после обновления страницы)  
  
· Удобное и понятное описание CSS и JavaScript  
  
· Во всех браузерах форма должна выглядеть одинаково  
  
Для решения задания можно было использовать готовые фрэймворки, что сводило само решение задания к 10 минутам.  
  
Через неделю позвали на второе интервью, на котором присутствовал только HR-специалист. Обсуждалась зарплата и организационные вопросы. По всем этажам фирмы мне провели экскурсию. Затем через пару дней по почте пришел договор со всеми ранее обговоренными пунктами. 

###### Мои выводы

1. Если вы пишете резюме, то вписывайте туда все программы и фрэймворки, с которыми вы хоть 30 мин. в жизни, но работали.   
2. В мотивационном письме постарайтесь себя вкусно продать, при этом проштудируйте историю фирмы и узнайте все о корпоративной культуре. Во многих фирмах в Германии тебя могут не принять на работу из-за того, что ты не понравился команде. И никакой многолетний опыт работы и потрясающая квалификация тебя не спасут.  
3. Знания английского, как показывает практика, не всегда нужны в том объеме, о котором пишут в описании должности. Чаще всего речь идет об обычном понимании языка и умении читать технические вещи на английских сайтах, блогах и т.п. Т.е. не обязательно быть птицей-говоруном.  
4. Знайте себе цену, заранее посмотрите годовой уровень зарплат в вашей области и просите на 20% больше, чем вы хотели бы получать в итоге. В большинстве случаев будет торг.  
5. При личном собеседовании важна инициативность и коммуникабельность. Постарайтесь с ходу предложить пару идей для развития фирмы и идеи эти обосновать.  
6. Задавайте встречные вопросы и показывайте заинтересованность.  
7. Если вам предложат выполнить какое-нибудь задание, не пугайтесь и свободно пользуйтесь интернет-ресурсами, т.к. главное – показать результат, а не заученные в университете истины.  
8. Каждую строку кода будьте готовы объяснить.  
9. Не бойтесь давать фидбэк о собеседовании. Просто сказать, что вам все понравилось, недостаточно. Если понравилось, уточните, что именно, если есть вопросы, задавайте сразу. Если вопросы появились уже после собеседования, отправьте их по Email.  
10. Принципы SOLID спрашивают почти везде.  
11. В резюме указывайте ник в Github.   
12. Continuous integration тоже часто спрашивают.

Вполне может быть, что у задачи и нет правильного решения или однозначного ответа. Ни в коем случае нельзя говорить «я не знаю» или «затрудняюсь ответить», просто размышляйте вслух.

Чтобы вы знали к чему быть готовыми и не растерялись, [Work.ua](http://www.work.ua/) собрал самые популярные вопросы-головоломки и подготовил ответы и рекомендации к ним. Кто знает, может вам попадутся именно эти задачи.

Все эти головоломки могут вам задать при собеседовании на вакансию [программиста php](http://www.work.ua/jobs-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%81%D1%82+php/).

#### Канализационные люки

Это, пожалуй, самый распространенный вопрос среди всех головоломок.

**Вопрос:** Почему канализационные люки круглые?

**Ответ:** Есть несколько вариантов. Так как диаметр круга одинаков, как его не крути, то круглый люк не может провалиться в колодец. У квадрата же, например, диагональ больше его сторон, поэтому крышка могла бы упасть. Также можно ответить, что круглые люки проще транспортировать и передвигать.

#### Лампочки

**Вопрос:** В закрытой комнате есть 3 лампочки, а в коридоре 3 выключателя. За какое минимальное открытие дверей можно определить какой выключатель к какой лампочке относится?

**Ответ:** За одно открытие. Включаем одновременно 2 выключателя, через некоторое время выключаем второй. Заходим в комнату: одна из лампочек осталась гореть — это первый выключатель, далее щупаем оставшиеся лампочки — теплая и будет вторым выключателем, а холодная соответственно третьим.

#### Торт

**Вопрос:** Как разделить торт на 8 равных частей тремя разрезами?

**Ответ:** Сперва нужно сделать 2 разреза крест на крест, поделив торт на 4 равных части. А затем разрезать торт горизонтально пополам. Ну и что, что куски стали невысокие, зато у вас 8 равных частей. Еще можно после первых двух разрезов сложить кусочки друг на друга и переполовинить одним разрезом.

#### Мертвый человек и спичка

**Вопрос:** Среди поля найден мертвый человек со спичкой в руках, следов нет. От чего он умер и при каких обстоятельствах?

**Ответ:** Человек умер от падения с самолета, который начал терять высоту, и авария была неизбежна. На всех пассажиров не хватило одного парашюта и они тянули жребий. Ему досталась короткая спичка, и он был вынужден прыгать без парашюта.

#### Автобус и мячи

**Вопрос:** Сколько теннисных мячей поместится в автобус?

**Ответ:** Точного правильного ответа не знает и сам рекрутер, так как не уточняется, что за мячи и автобус — их размеры не известны, и никто досконально это не проверял. Поэтому тут важен ход ваших мыслей, вы можете только предположить. Назовите примерные длину, ширину и высоту автобуса, размеры одного мяча. Посчитайте объем автобуса и мяча — так вы узнаете сколько мячей поместится в пустой автобус. Уменьшите примерно это значение с учетом сидений и других деталей автобуса, сделайте поправку на то, что мячи не квадратные и дайте ответ. В этом случае процесс поиска ответа важнее самого ответа. Варианты вопроса могут быть разные: мячи — футбольные, вместо автобуса — комната и т.п.

#### Таблетки

**Вопрос:** Доктор выдал пациенту 4 таблетки двух видов — по 2 таблетки каждого, которые нельзя отличить по внешнему виду. Таблетки надо выпить за два приема: утром по одной таблетке каждого вида и так же вечером. Если нарушить дозировку или не принять таблетки, то пациент умрет. Так вышло, что таблетки перемешались. Как пройти курс лечения и выжить?

**Ответ:** Конечно, можно сказать, что лучше пойти к врачу и попросить еще, все таки вопрос жизни и смерти. Но это могут быть единственные таблетки на Земле, доктор может исчезнуть при загадочных обстоятельствах и т.д. Так что ответить все же придется. К тому же все достаточно просто: нужно разделить каждую таблетку на 2 части и выпить по половинке каждой таблетки утром и вечером.

#### В заключение

Помните, что прежде всего такие задачи и головоломки предназначены для того, чтобы проверить поведение кандидата в нестандартных ситуациях, оценить способность к размышлению, творческому и логическому подходу. Увы, нередки случаи, когда интервьюер не может правильно интерпретировать результаты таких вопросов, или вовсе не понимает их предназначения. Но даже в этой ситуации уверенное поведение и стремление прийти к ответу покажут вас с лучшей стороны и увеличат шансы получить должность.

**Общие принципы построения программ**.

1. Что такое MVC?

**Model-View-Controller** – схема использования шаблонов проектирования, с помощью которых модель данных приложения, пользовательский интерфейс (UI) и взаимодействие с пользователем раздалены на 3 отдельных компонента так, что модификация одного из компонентов оказывает минимальное воздействие на остальные.

2) Что за что отвечает в MVC?

В шаблоне MVC, как следует из названия, есть **3 основных компонента: модель, представление и контроллер.**

**ООП**

1) ООП знаете? Что это такое?

**Объектно-ориентированное программирование** – парадигма программирования, в которой основными концепциями являются понятия объектов и классов. В языках с прототипированием (Java Script, к примеру) **вместо классов используются объекты-прототипы.**

2) Расскажите основные принципы в ООП.

1. **Инкапсуляция** – механизм, который объединяет данные и методы, манипулирующие этими данными и защищает их от внешнего вмешательства или неправильного использования. Можно сказать, что инкапсуляция подразумевает под собой сокрытие данных, что позволяет эти данные защитить.
2. **Наследование** – процесс, с помощью которого, один объект может наследовать свойства другого объекта и добавлять к ним черты, характерные только для него.
3. **Полифорфизм** – свойство, позволяющее одно и то же имя использовать для решения несколькоих технически разных задач. Концепция полиморфизма: один интерфейс, множество реализаций. Это означает, что можно создать общий интерфейс для близких по смыслу действий.

3) Что такое класс? А объект?

**Класс** – модель еще не существующей сущности (объекта). Класс фактически описывает устройство объекта, являясь своего рода чертежом.

**Объект** – совокупность данных и методов для их обработки. Вообще, объектом является все то, что поддерживает инкапсуляцию.

**Класс** – это тип данных, а объект – это экземпляр типа класса.

4) Что такое область видимости переменной?

**Область видимости переменной** – место в программе, в которой доступно значение переменной. Каждая переменная имеет свою область видимости.

**Public**. Методы или переменные доступны из любого места в коде.

**Protected**. Защищенные методы или переменные доступные только внутри класса, где они были объявлены или из его произвольных классов.

**Private**. Закрытые методы или переменные доступны только внутри класса.

5) Чем отличается процедурный подход от объектно-ориентированного?

**Процедурный подход** предоставляет возможность программисту определять каждый шаг в процессе решения задачи. **Задачи разбиваются на шаги и решаются шаг за шагом.**

ООП предполагает заключение внутри одного класса, как данных, так и методов их обработки.

6) Что такое абстрактный класс?

**Абстрактный класс** – это базовый класс, который не предполагает создание экземпляров (обратите внимание, что некоторые языки позволяют это сделать, например Pascal (Delphi)). Абстрактные классы реализуют на практике один из принципов ООП – полиморфизм. Абстрактый класс содержит абстрактные методы и свойства. Абстрактный метод не реализуется для класса, в котором описан, однако должен быть реализован для его неабстрактных потомков.

7) Зачем нужен интерфейс, если есть абстрактный класс?

Затем, что можно унаследовать только от одного абстракторого класса, но реализовать множество интерфейсов. Все методы, описанные в интерфейсе, должны быть реализованны в классе, а в абстрактном классе их нужно для этой цели специально отмечать.

**Разное**

1) Какие JS-фреймворки знаете?

Есть множество JS-фреймворков, например: **Backbone.js,  AngularJS, Ember.js, Knockout** и другие.

2) Что такое сериализация? А десериализация?

**Сериализация** – процесс перевода какой-либо структуры данных в последовательность битов.

**Десериализация** – операция, обратная сериализации, т.е. восстановление исходного состояния структуры данных из последовательности битов.

3) Что такое система управления версиями?

Это программное обеспечение для облегчения работы с изменяющейся информацией. Системы контроля версий позволяют хранить несколько версий одного и того же документа, при необходимости, возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение и многое другое.

Примеры систем управления версиями: **SVN, Git, Mercurial, Bazaar** и другие.

4) Что такое сессии? Где хранятся сессии?

**Сессия (HTTP Session)** – логический объект, позволяющий передавать данные между последовательными HTTP-запросами от одного пользователя. Сессии создаются на стороне сервера и там же храняться.

Другими словами, сессия – механизм, позволяющий однозначно идентифицировать браузер и созданный для этого браузера файл на сервере, в котором храняться переменные сеанса.

5) Как долго “живет” сессия?

Время жизни куки, которая хранит идентификатор **сессии по умолчанию составляет 0 секунд**, т.е. кука храниться до того момента, как браузер будет закрыт. Это время можно изменить на стороне сервера.

6) Разница между HTTP-методами GET и POST?

**GET**  передает данные серверу с помощью URL, а **POST**  передает данные, используя тело HTTP-запроса.

7) Что такое Document Object Model (DOM)?

**DOM** – программный интерфейс, позволяющий программам получить доступ к содержимому HTML и XML-документов, а также изменять содержимое, структуру и оформление таких документов.

8) Что такое AJAX и как он работает?

**AJAX** – подход к построение интерактивных пользовательских интерфейсов веб-приложений, заключающийся в “фоновом” обмене данными браузера с веб-сервером.

9) Что такое JSON? А XML?

**JSON** – текстовый формат обмена данными, основанный на Java Script и обычно используемый именно с этим языком. JSON легко читается людьми.

**XML** – расширяемый язык разметки, фактически представляющий собой свод общих синтаксических правил.

**Cookies (куки).**

По одной из версий термин “куки” (печенье) происходит от “волшебного печенья” – **набор данных, которые программа получает и затем отправляет обратно неизменными.** В июне 1994 года Лу Монтулли пришла идея использовать их при веб-соединении.

***Основные минусы использования Cookies:***

1. При каждом HTTP-запросе на сервер, все Cookies **отправляются на сервер и обратно**.
2. Общее количество Cookies на домен **ограничено 30-50**, в зависимости от браузера.
3. Cookies не должны превышать **размера 4 килобайта**, в зависимости от браузера.

***Cookies на практике обычно используется для следующих целей:***

1. Аутентификация пользователя.
2. Хранение персональных предпочтений и настроек пользователя.
3. Отслеживание состояния сеанса доступа пользователя.
4. Ведение статистики о пользователях.

**Cookies поддерживаются всеми основными браузерами**, но пользователь их может отключить. Также, если включить в браузере приватный режим **“Private Browsing”, то Cookies будут автоматически отключены**.

Теперь немного практики.

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16**  **17**  **18**  **19**  **20**  **21**  **22**  **23**  **24**  **25**  **26**  **27**  **28**  **29**  **30**  **31**  **32**  **33**  **34**  **35**  **36**  **37**  **38**  **39**  **40**  **41** | **/\*\***  **\* Сохранение cookie.**  **\*/**  **function setCookie(name, value, options) {**  **// options - объект с свойствами cookie (expires, path, domain, secure)**  **options = options || {};**  **var expires = options.expires;**  **if (typeof expires == "number" && expires) {**  **var d = new Date();**  **d.setTime(d.getTime()+expires\*1000);**  **expires = options.expires = d;**  **}**  **if (expires && expires.toUTCString) {**  **options.expires = expires.toUTCString();**  **}**  **value = encodeURIComponent(value);**  **var updatedCookie = name+"="+value;**  **for (var propName in options) {**  **updatedCookie += "; "+propName;**  **var propValue = options[propName];**  **if (propValue !== true) {**  **updatedCookie += "="+propValue;**  **}**  **}**  **document.cookie = updatedCookie;**  **}**  **/\*\***  **\* Получение cookie.**  **\*/**  **function getCookie(name) {**  **var matches = document.cookie.match(new RegExp("(?:^|; )"+name.replace(/([\.$?\*|{}\(\)\[\]\\\/\+^])/g, "\\$1")+"=([^;]\*)"));**  **return matches ? decodeURIComponent(matches[1]) : undefined;**  **}**  **/\*\***  **\* Удаление cookie.**  **\*/**  **function deleteCookie(name) {**  **setCookie(name, "", {**  **expires: -1**  **}); // сохранение cookie**  **}** |

Функции работают для сохранения и получения и удаления Cookies.

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16** | **<div id="cookie"> </div>**  **<script type="text/javascript" language="javascript">**  **var divCookie = document.getElementById("cookie");**  **var outputCookie = '';**  **if (document.cookie) {**  **outputCookie += '<p> <b> Cookies включены в браузере. </b> </p>';**  **setCookie("testName", "Test Value"); // сохранение cookie**  **var output = getCookie("testName"); // получение cookie**  **outputCookie += '<p> Cookie "testName" = "'+output+'". </p>';**  **deleteCookie("testName"); // удаление cookie**  **}**  **else {**  **outputCookie += '<p class="error"> <b> Cookies отключены в браузере. </b> </p>';**  **}**  **divCookie.innerHTML = outputCookie;**  **</script>** |

Код выше демонстрирует использование написанных ранее функций для доступа к Cookies.

**Хранилище sessionStorage.**

Данное хранилище появилось с появлением **HTML5**.

Сеансовое хранилище (sessionStorage) существует до тех пор, пока не закроется окно (или вкладка). Как только пользователь закроет окно (или браузер), значения, сохраненные в объекте sessionStorage, пропадают. Такое хранилище прекрасно подходит для “короткоживущих” процессов, с которыми обычно приходится сталкиваться в программах-мастерах и диалоговых окнах.

Теперь немного практики.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | **<div id="session\_storage"> </div>**  **<script type="text/javascript" language="javascript">**  **var divSessionStorage = document.getElementById("session\_storage");**  **var outputSessionStorage = '';**  **if (window['sessionStorage']) {**  **outputSessionStorage += '<p> <b> Хранилище sessionStorage поддерживается в браузере. </b> </p>';**  **sessionStorage.setItem("testName", "Test Value");**  **var output = sessionStorage['testName'];**  **outputSessionStorage += '<p> sessionStorage "testName" = "'+output+'". </p>';**  **sessionStorage.removeItem("testName");**  **}**  **else {**  **outputSessionStorage += '<p class="error"> <b> Хранилище sessionStorage не поддерживается в браузере. </b> </p>';**  **}**  **divSessionStorage.innerHTML = outputSessionStorage;**  **</script>** |

**Хранилище localStorage.**

Данное хранилище появилось с появлением **HTML5.**

Иногда приложению могут потребоваться значения, которые продолжают существовать после закрытия отдельной вкладки или окна, или значения, которые должны совместно использоваться несколькими различными представлениями. Для этих целей и существует localStorage.

Теперь немного практики.

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16** | **<div id="local\_storage"> </div>**  **var divLocalStorage = document.getElementById("local\_storage");**  **<script type="text/javascript" language="javascript">**  **var outputLocalStorage = '';**  **if (window['localStorage']) {**  **outputLocalStorage += '<p> <b> Хранилище localStorage поддерживается в браузере. </b> </p>';**  **localStorage.setItem("testName", "Test Value");**  **var output = localStorage['testName'];**  **outputLocalStorage += '<p> localStorage "testName" = "'+output+'". </p>';**  **localStorage.removeItem("testName");**  **}**  **else {**  **outputLocalStorage += '<p class="error"> <b> Хранилище localStorage не поддерживается в браузере. </b> </p>';**  **}**  **divLocalStorage.innerHTML = outputLocalStorage;**  **</script>** |

**Web SQL Database.**

Данное хранилище появилось с появлением **HTML5.**

Web SQL Database представляет доступ к индексируемым базам данных. Используя Web SQL Database, приложения могут получить доступ к СУБД SQLite посредством асинхронного JavaScript интерфейса. К сожалению, не все браузеры поддерживают данную спецификацию.

Теперь немного практики.

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16**  **17**  **18**  **19**  **20**  **21**  **22**  **23**  **24**  **25**  **26**  **27**  **28**  **29**  **30**  **31**  **32**  **33**  **34**  **35**  **36**  **37**  **38**  **39**  **40**  **41**  **42**  **43**  **44**  **45**  **46**  **47**  **48**  **49**  **50**  **51**  **52**  **53**  **54**  **55**  **56** | **<div id="web\_sql\_database"> </div>**  **<script type="text/javascript" language="javascript">**  **var divWebSQLDatabase = document.getElementById("web\_sql\_database");**  **var outputWebSQLDatabase = '';**  **var event = new CustomEvent("transactionsIsDone");**  **document.addEventListener("transactionsIsDone", function(e) {**  **divWebSQLDatabase.innerHTML = outputWebSQLDatabase;**  **});**  **if (typeof(openDatabase) !== "undefined") {**  **outputWebSQLDatabase += '<p> <b> Web SQL Database поддерживается в браузере. </b> </p>';**  **var db = openDatabase("testDB", "1.0", "Test DB", 1024\*1024\*5);**  **db.transaction(function(tx) {**  **tx.executeSql("CREATE TABLE IF NOT EXISTS test\_table (id INTEGER PRIMARY KEY ASC, text TEXT, created\_date VARCHAR)", []);**  **var text = "Test Value";**  **var date = new Date();**  **tx.executeSql("INSERT INTO test\_table (text, created\_date) VALUES (?, ?)", [text, date.toUTCString()]);**  **tx.executeSql("SELECT \* FROM test\_table", [], function(tx, result) {**  **outputWebSQLDatabase += ''+**  **'<p>'+**  **'<table width="100%" height="100%" border="1" cellpadding="3">'+**  **'<thead>'+**  **'<tr>'+**  **'<th> ID </th>'+**  **'<th> Текст </th>'+**  **'<th> Дата создания </th>'+**  **'</tr>'+**  **'</thead>'+**  **'<tbody>'+**  **'';**  **for (var i = 0; i < result.rows.length; i++) {**  **var id = result.rows.item(i).id;**  **var text = result.rows.item(i).text;**  **var created\_date = result.rows.item(i).created\_date;**  **outputWebSQLDatabase += ''+**  **'<tr>'+**  **'<td>'+id+'</td>'+**  **'<td>'+text+'</td>'+**  **'<td>'+created\_date+'</td>'+**  **'</tr>'+**  **'';**  **}**  **outputWebSQLDatabase += ''+**  **'</tbody>'+**  **'</table>'+**  **'</p>'+**  **'';**  **document.dispatchEvent(event);**  **});**  **tx.executeSql("DROP TABLE test\_table");**  **});**  **}**  **else {**  **outputWebSQLDatabase += '<p class="error"> <b> Web SQL Database не поддерживается в браузере. </b> </p>';**  **document.dispatchEvent(event);**  **}**  **</script>** |

**Local SharedObject.**

Подробно этот метод разбирать не будем, т.к. для его применения требуются сторонние инструменты. Попытаюсь рассказать кратко, чтобы вы были в курсе.

Начиная с версии Flash Player 6.0.40.0 флэш поддерживает Local SharedObject. С его помощью можно хранить самые разные объекты средствами Adobe Flash. Чтобы работать с этим хранилищем, нужно создать средство коммуникации JS <-> FLASH.

Минусы этого подхода:

1. Требуется установленный Flash-плеер у пользователя.
2. Со стороны веб-программиста требуется знание ActionScript (не существенный минус).
3. Ограничение по умолчанию на размер данных в 100 килобайт, пользователь может увеличить или уменьшить размер в специальном Flash-интерфейсе.