Павлов Дмитрий Гурьевич

Россия, Москва
тел.: +79151438165, e-mail: MitrickX@mail.ru

Резюме

Общая информация

Дата рождения: 8 января 1987 г.

Пол: мужской

Образование

2004 — 2009

Марийский государственный технический университет. Кафедра безопасности информации.

Специальность: Комплексное обеспечение информационной безопасности автоматизированных систем

Профессиональный опыт

2012 — наст.вр.

ООО "Артикус"

2010 — 2011

НИИ МКО

Профессиональные навыки

- Знание языков программирования: PHP5(ООП, CakePHP, Zend Framework, Smarty), C/C++(MFC, WinAPI), Javascript(jQuery)
- Знание СУБД: MySQL
- Верстка: HTML, CSS
- Иностранные языки: Английский на уровне чтения документации

Ответы на технические вопросы

1. alert, console, штатные отладчики в браузерах. Написание тестов
2. Поведение браузера по умолчанию - переход по ссылке, т.е. отсылка на сервер GET-запроса на получение страницы. После получения ответа от сервера происходит интерпретация html(xhtml, xml) кода. Поведение по умолчанию можно переопределить, повесив на событие onclick обработчик, который либо вызовет метод preventDefault() объекта события (для IE < 9 нужно установить свойство returnValue объекта события в false), либо возвратит false. Возвращение false не будет работать, если обработчик навешивается на событие через addEventListener/attachEvent.
3.
```
/**
 * Создает экземпляр Машины
 * @this {Car}
 * @param {string} manufacturer Производитель
 * @param {string} model Модель
 * @param {number} year Год производство
 */
function Car(manufacturer, model, year) {
    this.manufacturer = manufacturer;
    this.model = model;
    this.year = year || (new Date()).getFullYear();
}
Car.prototype.getCountry = function() {
    switch (this.manufacturer.toLowerCase()) {
        case 'bmw':
            case 'audi':
            return 'Germany';
        case 'toyota':
            return 'Japan';
        }
};
Car.prototype.getInfo = function() {
    return this.manufacturer + ' ' + this.model + ' ' + this.year;
};
Car.prototype.getDetailedInfo = function() {
    return 'Производитель: ' + this.manufacturer + '. Модель: ' + this.model + '. Год: ' + this.year;
};
Car.prototype.toString = Car.prototype.getInfo;

var bmw = new Car("BMW", "X5", 2010),
    audi = new Car("Audi", "Q5", 2012),
    toyota = new Car("Toyota", "Camry");

/**
 * Создает экземпляр Автосалона
 * @this {CarDealer}
 * @param {string} name Название автосалона
 */
function CarDealer(name) {
    this.name = name;
    this.cars = [];
    this.prices = {};
}
CarDealer.prototype.add = function() {
    for (var i = 0, n = arguments.length; i < n; i++) {
```

```
        if (arguments[i] instanceof Car) {
            this.cars.push(arguments[i]);
        }
    }
    return this;
};
/**
 * Установить цену на машину
 * @param {string} car идентификатор машины
 * @param {string} price стоимость
 */
CarDealer.prototype.setPrice = function(car_id, price) {
    this.prices[car_id] = price;
    return this;
};
CarDealer.prototype._list = function(filter, price_descriptor) {
    var list = [],
        cars = this.cars;
    if (typeof filter !== 'function') {
        filter = function() {
            return true;
        }
    }
    var formatCarInfo;
    if (price_descriptor) {
        var self = this,

            formatPrice = (typeof price_descriptor === 'object') ?
            function(price) {
                currency = price.charAt(0);
                price = price.substr(1);
                if (price_descriptor.exchange_rate[currency]) {
                    price = (price*price_descriptor.exchange_rate[currency]).toFixed(2);
                    currency = price_descriptor.currency;
                }
                return currency + price;
            } :
            function(price) {
                return price;
            };

        formatCarInfo = function(car) {
            var car_id = car.getInfo(),
                price = self.prices[car_id],
                output = car_id;
            if (price) {
                price = formatPrice(price);
                output += '. Price: ' + price;
            }
            return output;
        }
    } else {
        formatCarInfo = function(car) {
            return car.getInfo();
        }
    }
    for (var i = 0, n = cars.length; i < n; i++) {
        if (filter(cars[i])) {
            list.push(formatCarInfo(cars[i]));
        }
    }
    return list.join(', ');
}
CarDealer.prototype.list = function(price_descriptor) {
    return this._list(null, price_descriptor);
};
CarDealer.prototype.listByCountry = function(country, price_descriptor) {
    return this._list(function(car) {
        return car.getCountry() == country;
    }, price_descriptor);
}

var yandex = new CarDealer('Яндекс.Авто');
yandex
    .add(toyota)
    .add(bmw, audi);

yandex
    .setPrice('BMW X5 2010', '€2000')
    .setPrice('Audi Q5 2012', '€3000')
    .setPrice('Toyota Camry 2012', '¥3000');
```

```
yandex.list(); //Toyota Camry 2012, BMW X5 2010, Audi Q5 2012
yandex.list(true); //Toyota Camry 2012. Price: ¥3000, BMW X5 2010. Price: €2000, Audi Q5 2012. Price: €3000
yandex.listByCountry('Germany'); //BMW X5 2010, Audi Q5 2012
yandex.listByCountry('Germany', true); //BMW X5 2010. Price: €2000, Audi Q5 2012. Price: €3000

var var price_descriptor = {
    currency: 'R',
    exchange_rate: {
        '€': 39.19,
        '¥': 0.4
    }
};
yandex.list(price_descriptor); //Toyota Camry 2012. Price: R1200.00, BMW X5 2010. Price: R78380.00, Audi Q5 2012. Price: R117570.(
yandex.listById('Germany', price_descriptor); //BMW X5 2010. Price: R78380.00, Audi Q5 2012. Price: R117570.00
```

4. find ./ -name "*yandex*.txt" -type f -exec grep -l "школа разработки интерфейсов" {} \;
5. **Bash**

```
#!/bin/bash
usage() {
cat << EOF;
Usage: printargs.sh [OPTIONS] [ARGUMENTS]
  Print the number of arguments

OPTIONS:
  -h      print help message
  -m MSG  custom message
  -v      print arguments

Examples:
  printargs.sh a b c
  printargs.sh -m 'Arguments count: ' a b c
  printargs.sh -h
  printargs.sh -v -m 'Arguments count: ' a b c
EOF
}

VERBOSE=0

while getopts "hvm:" OPTION
do
    case $OPTION in
        h)
           usage
           exit 1
           ;;
        m)
           MESSAGE=$OPTARG
           ;;
        v)
           VERBOSE=1
           ;;
    esac
done

shift $((OPTIND-1))

if [[ "$#" != "0" ]]; then
    if [[ "$MESSAGE" != "" ]]; then
        echo -n $MESSAGE
    fi

    echo $#

    if [[ "$VERBOSE" == "1" ]]; then
        echo "Arguments:" $@;
    fi
else
    usage
fi
```

Python

```
#!/usr/bin/python

import sys
import argparse

parser = argparse.ArgumentParser(description='Print the number of arguments.')
parser.add_argument('arguments', metavar='ARG', type=str, nargs='*', help='some arguments')
parser.add_argument('-m', dest='message', default='', help='custom message')
parser.add_argument('-v', dest='verbose', action='store_true', help='print arguments')
```

```
args = parser.parse_args()

count = len(args.arguments)
if (count != 0):
    if args.message != '':
        sys.stdout.write(args.message)
    print(count)
    if args.verbose:
       print ("Arguments: " + ' '.join(args.arguments));
else:
    parser.print_help();
```

Perl

```
#!/usr/bin/perl

use strict;
use Getopt::Long;
use Pod::Usage;

my $help;
my $message = '';
my $verbose;

@ARGV and GetOptions(
    "h" => \$help,
    "m:s" => \$message,
    "v" => \$verbose
) or pod2usage(1);
pod2usage(-verbose => 2, -exitval => 2) if $help;

my $count = $#ARGV + 1;

if ($message ne "") {
    print $message;
}

print ($count . "\n");

if ($verbose) {
    print "Arguments: " . (join ' ', @ARGV) . "\n";
}

__END__

=head1 NAME

    printargs.pl - Print the number of arguments.

=head1 SYNOPSIS

    printargs.pl [options] [arguments]

=head1 OPTIONS

    -h      Show help message
    -m MSG  Specify a custom message
    -v      print arguments

=head1 EXAMPLE

    printargs.sh a b c
    printargs.sh -m 'Arguments count: ' a b c
    printargs.sh -h
    printargs.sh -v -m 'Arguments count: ' a b c
```

6. Языки программирования. Специальный предмет, в котором уделялось бы внимание классификации языков программирования, системам типо
   Структуры. В вузе был предмет "Структуры обработки данных" длительностью в 1 семестр. Изучали: стеки, очереди, деревья, графы.
   Парадигмы: Есть представлния, знания, практический опыт в ООП, события, параллельность/асинхронность, кеширование.
   Алгоритмы: сортировка, двоичный поиск проходили в рамках предмета "Структуры обработки данных".
   Паттерны: какие-то из паттернов использовал на практике, имею представление.