
Яндекс

Яндекс

JavaScript: основы языка

Занятие 4, 13 марта 2019

Зачем?

Позволяет выполнять алгоритмы внутри браузера, на компьютере клиента

- Дает нам возможность менять содержимое окна браузера без нового клиент-серверного взаимодействия (запроса новой страницы)
- Можно быстрее реагировать, например, на ввод данных пользователем и показывать ему ошибки
- Делает возможными всяческие необычные анимации внутри окна браузера, удобные элементы интерфейса такие, как выпадающие меню и так далее
- А еще, можно обращаться к разным устройствам на компьютере/ телефоне клиента - фотокамере, геолокации и т.п.

Зачем?

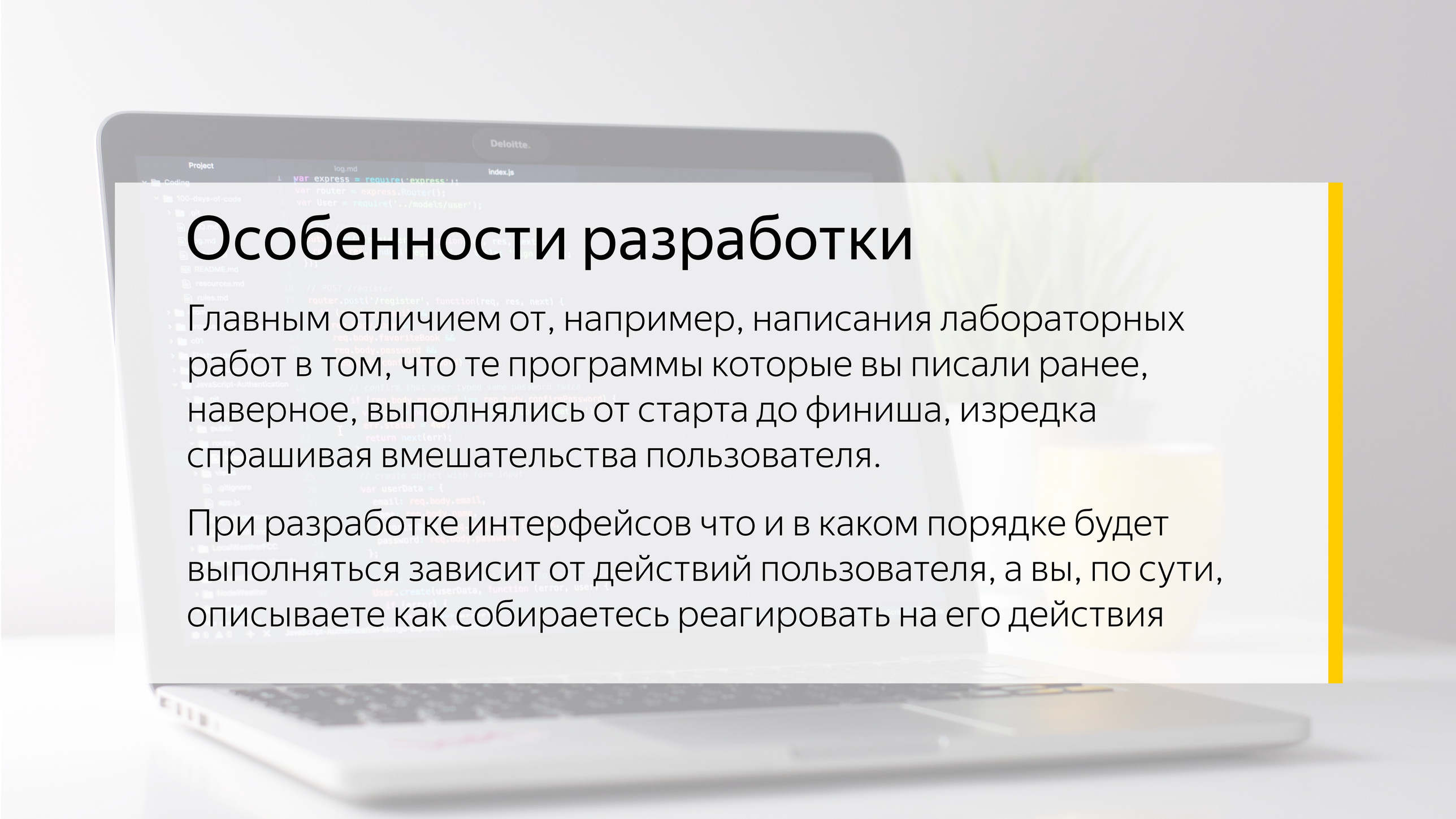
Позволяет реагировать на «события» - по сути, на действия пользователя

- Делает интерфейс интерактивным - отзывчивым на действия пользователя
- Дает возможность получить дополнительные данные от «сервера» и вывести эти данные на страницу, также, без перезагрузки
- Можно отслеживать «сложные»/«нестандартные» события и сверстать какой-то необычный элемент управления
- И данные о пользователе можно сохранить, для того например, чтобы восстановить содержимое вкладки пользователя, например

Почему?

Ничего другого нет

- В браузере клиента будет только интерпретатор JavaScript
- Любые другие «производные» типа CoffeeScript, TypeScript etc. транслируются в JS и уже JS выполняется на клиенте
- Сам язык - Си-подобный, относительно несложный
- Есть очень много готового кода, написанного другими разработчиками, который запросто можно и нужно использовать, ускоряя разработку
- Абсолютное большинство библиотек на JavaScript - OpenSource, то есть их можно использовать с чистой совестью, а можно и поучаствовать в разработке, прокачивая скиллы



Особенности разработки

Главным отличием от, например, написания лабораторных работ в том, что те программы которые вы писали ранее, наверное, выполнялись от старта до финиша, изредка спрашивая вмешательства пользователя.

При разработке интерфейсов что и в каком порядке будет выполняться зависит от действий пользователя, а вы, по сути, описываете как собираетесь реагировать на его действия

Стандарты

JavaScript - диалект ECMAScript

1. Метка проверенного партнёра
2. Есть разные стандарты языка
3. Для интерфейсов, пожалуй, пока еще рано использовать ES6 и выше
4. Будем делать время от времени отсылки к ES6

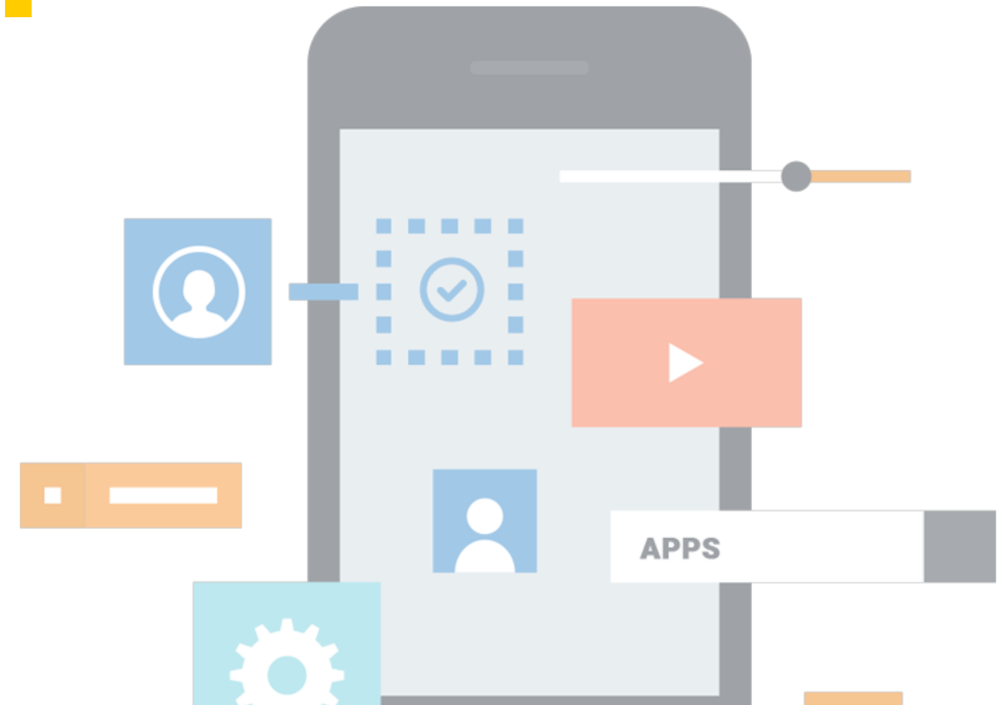
Ок, а как начать-то?

Доставка JavaScript в браузер

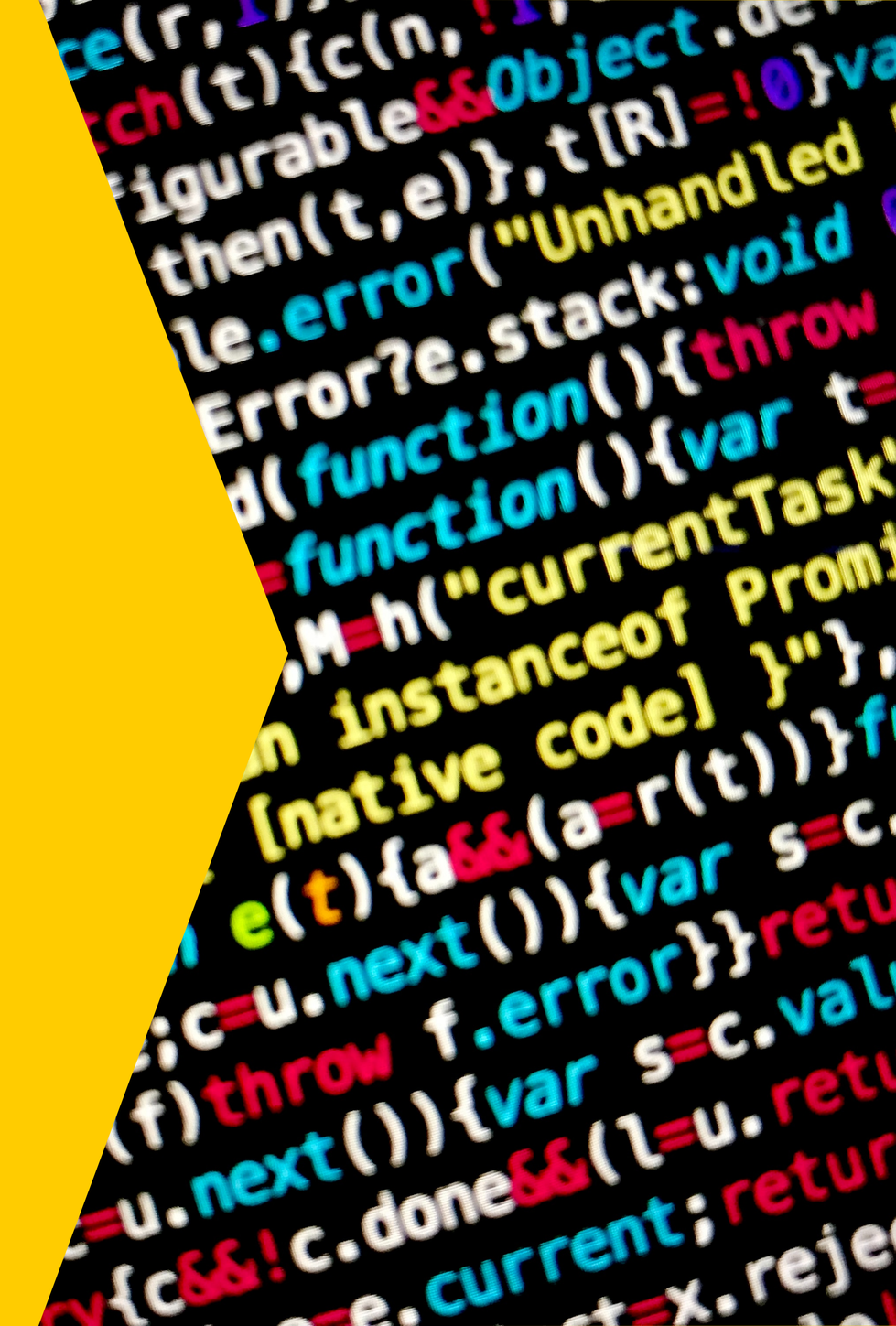
- **«Заинлайнить» - писать код в специальных атрибутах тегов типа onclick**
- **Тег <script>**
- **Тег <script> с атрибутом src**

*Когда выполняется код внутри тега скрипт, все в браузере «замирает»

Первое «приложение»



Основы JavaScript



Переменные

Переменные

Позволяют что-то записать в памяти компьютера и дать этой области памяти имя, с помощью которого потом можно будет считать/изменить записанные данные

Объявляем переменные с помощью ключевого слова var

В JavaScript принято использовать camelCase для имен переменных

› например: var myFirstName;

Типы данных

Типы данных

boolean	true, false
number	1, 3.14, NaN, Infinity
string	'привет'
array	[], [1, 2, 3]
object	{ id: 1, name: ‘admin'}
null	null
undefined	undefined

Boolean

Отвечает на вопрос да/нет

Будет, например являться результатом оператора сравнения

```
> 5 > 3 // true, так как действительно, 5 больше
```

Number

- Служит для хранения чисел
- Нет отдельного типа для дробных чисел - это все number
- Есть специальные значения типа NaN, Infinity

String

- Строковый тип данных - про последовательности символов
- Нет отдельного типа для одного символа
- Пустая строка ''
- Строки принято стараться задавать одинарными кавычками
- В JavaScript строковый литерал нельзя задавать многострочным
- В ES6 есть символ `` позволяющий задать шаблонную строку’

Array

Позволяет задать несколько значений, например, в одну переменную

```
var books = ['колобок', 'Красная Шапочка’];
```

Доступ к конкретному значению - через его номер в массиве

```
books[1] // Красная Шапочка, так как колобок имеет номер 0
```

Пустой массив []

Подробнее скоро изучим

Object

Своего рода ассоциативный массив, несколько значений в одной переменной, но элементы имеют не номера, а ключи

```
> var user = { id: 1, name: 'dusty'};
```

Доступ к конкретному значению - через ключ

```
> user.name // dusty
```

Пустой объект {}

Подробнее скоро изучим

Null и Undefined

- Два типа данных для обозначения того, что значения нет
- Семантически разные
- null значит, что чего-то явно нет, undefined - значит, что информация неизвестна

Операторы

Операторы

- Присваивание =
- Математика + - / * %
- Сравнение > < >= <= == ===
- Булевские ! && ||

Напишем первую строку кода

```
var dusty = 5 + 3;
```

Выполнение кода

- Код выполняется сверху вниз, строка за строкой
- На строке интерпретатор разбивает код на операторы и операнды, выставляет порядок выполнения по приоритетам операторов и выполняет операции
- Управлять порядком выполнения можно с помощью скобок ()
- Поначалу, не экономьте на скобках

Как вывести значение переменной хоть куда-нибудь?

Вывод (и ввод, чего уж там) информации

- alert - вывести предупреждение в окно
- confirm - задать альтернативный вопрос и получить boolean ответ
- prompt - задать обычный вопрос и получить строку/null в ответ
- console.log - вывести информацию в консоль

Как начать взаимодействовать с документом?

Меняем документ

- document.getElementById('test') - получить элемент с атрибутом id, равным test
- Свойство innerText
- Свойство value
- Свойство onclick

Яндекс

Спасибо, вопросы?