



Javascript. Уровень 1

Урок 1

ОСНОВЫ ЯЗЫКА Javascript

Общее знакомство с
JavaScript, создание первого
кода и его запуск.

План курса

- Урок 1. Основы языка JavaScript.
- Урок 2. Основные операторы Javascript.
- Урок 3. Циклы, массивы, структуры данных.
- Урок 4. Объекты в JavaScript.
- Урок 5. Введение в DOM.
- Урок 6. Обработка событий в JavaScript.
- Урок 7. Практикум.
- Урок 8. Анонимные функции. Замыкания.



План урока

- Способы исполнения программ
- Как исполняется JavaScript-программа в браузере
- Принципы написания кода на JavaScript
- Структура кода
- Типы данных

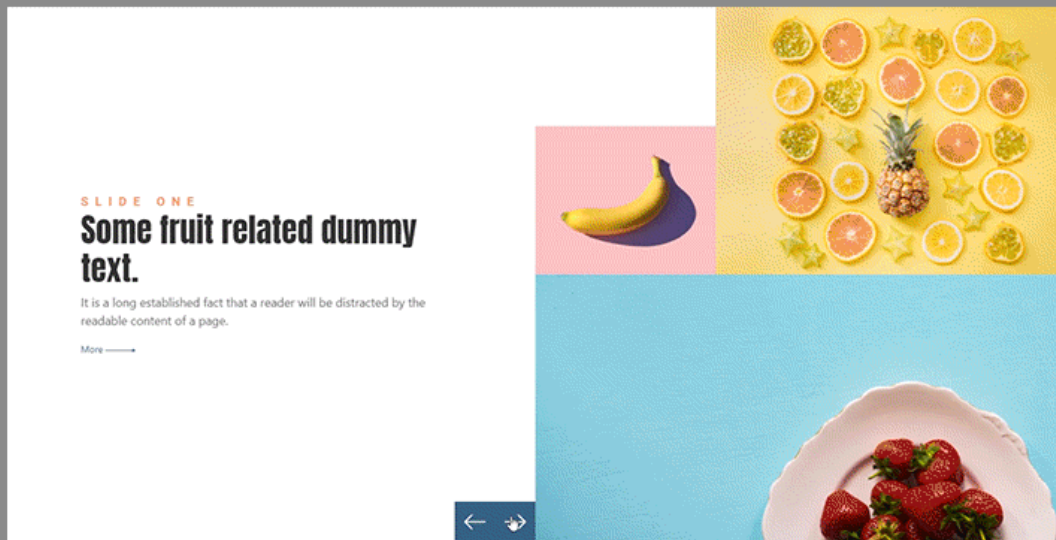


Где используется Javascript

- На веб-страницах в браузере
- Для серверного программирования с помощью node.js
- Мобильные и настольные приложения
- Где угодно, если есть интерпретатор или компилятор Javascript кода



Для чего нужен Javascript на веб-странице



Для чего нужен Javascript на веб-странице



6 May 2017 - 22 May 2017 ▾

OF

6 May 2017 - 22 May 2017

Today	May 17	June 17	◀ ▶
Yesterday	S M T W T F S	S M T W T F S	
Last 7 days	27 28 29 30 1 2 3	1 2 3 4 5 6 7	
Last 15 days	4 5 6 7 8 9 10	8 9 10 11 12 13 14	
Last Month	11 12 13 14 15 16 17	15 16 17 18 19 20 21	
	18 19 20 21 22 23 24	22 23 24 25 26 27 28	
Custom	25 26 27 28 29 30 31	29 30 1 2 3 4 5	
Apply	00:00	23:59	



Для чего нужен Javascript на веб-странице



 Anatoliy K. ▾



Add your first location

Lorem ipsum dolor sit amet, ea wisi latine duo, nec tractatos
deterruisset ne. Duo cu eius interesset adversarium, at eam quem.

+ Add Access Point



Исполнение программы. Компиляция в машинный код.

```
package main

import "fmt"

func main() {
    fmt.Println("Hello, world")
}
```

Исходный код(здесь язык Go),
который пишет человек.



Процесс компиляции



Исполняемый файл, который может выполнить процессор
компьютера.



Компилируемые языки

- C
- C++
- C#
- Go
- Fortran
- Delphi
- ...



Исполнение программы. Компиляция в байт-код.

```
class Hello {  
    public static void main(String[] args) {  
        System.out.println("Hello Java!");  
    }  
}
```

Исходный код(здесь язык Java).



Процесс компиляции в байт-код



Исполняемый файл, который может выполнить процессор **виртуальной** машины.



Виртуальная машина превращает байт-код в машинный код, который выполняет **физический** процессор компьютера.



Языки компилируемые в байт-код

- Java
- C#
- Python
- Erlang
- ...

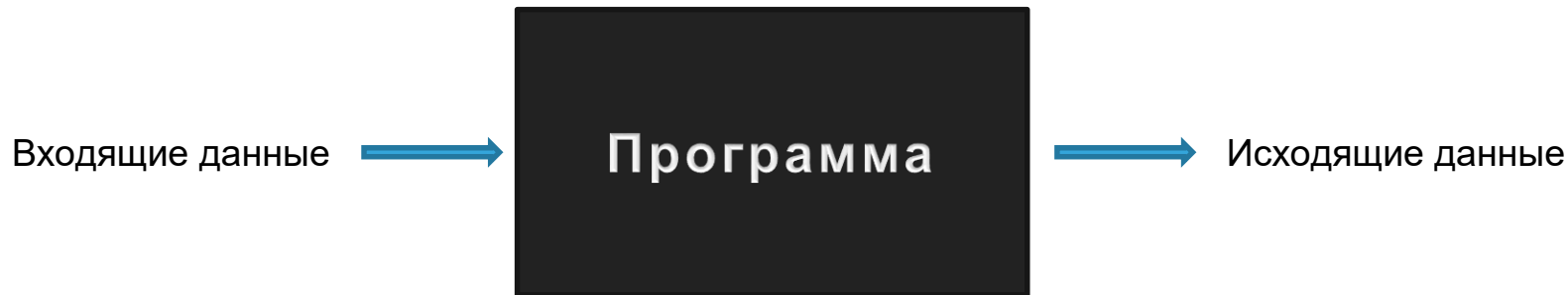


Интерпретируемые языки

- Javascript
- PHP
- Python
- ...



Схема работы программы



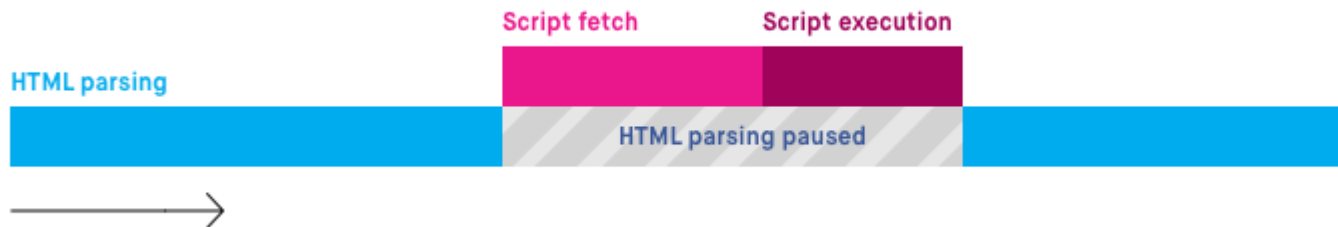
Примеры обработки данных

Вид программы	Входящие данные	Исходящие данные
3d игра	Нажатия на клавиши, перемещение мыши	Выстрелы из оружия, перемещение персонажа
Веб-страница	Заполненная форма регистрации на сайте	Письмо на почте пользователя о подтверждении email-адреса
Сервис онлайн-трансляций youtube	Видеопоток на сервер youtube	Видеопоток у пользователей на веб-странице в браузере



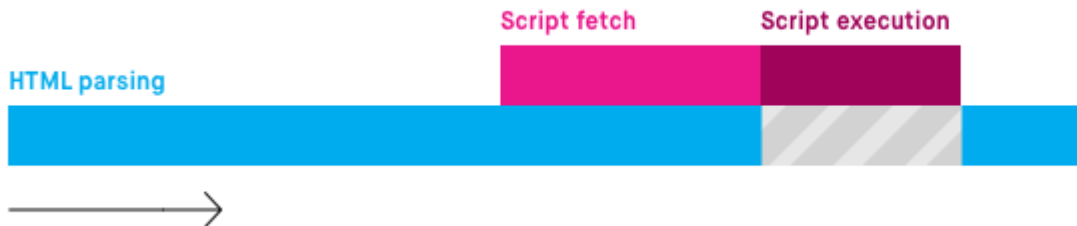
Обычный порядок выполнения скрипта

```
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body>
    ...
    <script src="script.js"></script>
    ...
  </body>
</html>
```



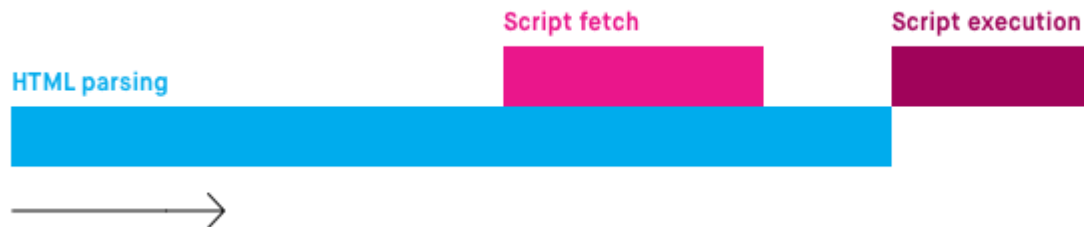
Порядок выполнения скрипта с атрибутом **async**

```
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body>
    ...
    <script async src="script.js"></script>
    ....
  </body>
</html>
```



Порядок выполнения скрипта с атрибутом **defer**

```
<!DOCTYPE html>
<html>
  <head> ... </head>
  <body>
    ...
    <script defer src="script.js"></script>
    ....
  </body>
</html>
```



Функция – блок кода, содержащий в себе некоторую логику и действия.

Функции предназначены для многократного использования.



```
alert(2 + 3);  
alert("hello" + " " + "world");
```

Выражение – это фраза языка, которую может вычислить интерпретатор для получения значения.



Переменные

			size (25)	

```
let size = 25;
```



Объявление переменных с помощью var и let

```
let myVar = "hello";  
var num = 45;
```

Предпочитайте let.



Имена переменных

Неправильно

```
let 10years = 10;  
let @myMail = "hello@example.com";  
let всеМПривет = "как дела?";  
let function = "do something";
```

Корректно

```
let years10 = 10;  
let $myMail = "hello@example.com";  
let hi_all = "как дела?";  
let _doSome = "do something";
```



Константа – область в памяти, в которую нельзя что-то присвоить повторно, но можно изменить содержимое, если в памяти находится сложный тип, например объект или массив.



Операторы

- Арифметические
- Присваивание
- Конкатенация
- Побитовые
- Доступ к значениям
- Сравнения
- Логические

Оператор – наименьшая единица языка, которая что-то делает.



Приоритеты операторов

Приоритет	Тип оператора	Ассоциативность	Конкретные операторы
20	Группировка	не определено	(...)
19	Вызов функции	слева направо	... (...)
17	Постфиксный инкремент	не определено	... ++
16	Логическое отрицание	справа налево	! ...
	Унарный плюс		+ ...
	typeof		typeof ...
14	Умножение	слева направо	... * ...
	Деление		... / ...
13	Сложение	слева направо	... + ...
	Вычитание		... - ...
11	Меньше	слева направо	... < ...
	Меньше или равно		... <= ...
	Больше		... > ...
	Больше или равно		... >= ...
10	Равно	слева направо	... == ...
	Не равно		... != ...
	Строго равно		... === ...
	Строго не равно		... !== ...
6	Логическое «И»	слева направо	... && ...
5	Логическое «ИЛИ»	слева направо
3	Присваивание	справа налево	... = ...

Приоритет операторов определяет порядок, в котором операторы выполняются. Операторы с более высоким приоритетом выполняются первыми.



Ассоциативность – определяет порядок, в котором обрабатываются операторы с одинаковым приоритетом.



Литерал – значение, указанное непосредственно в тексте программы.

```
//строковый литерал
```

```
alert("hello");
```

```
//числовой литерал
```

```
alert(10);
```

```
//логический(boolean) литерал
```

```
alert(true);
```



Тип string

```
let size = "middle";  
let greetings = 'hello';
```



Тип number

```
let integerValue = 125;  
let floatingPoint = 10.367;  
let negative = -34;
```



Тип number

```
let val1 = 10 / 0;      //Infinity  
let val2 = 3 * "hello"; //NaN
```



Тип boolean

```
let money = 100;  
let price = 20;  
  
//100 > 20 = true  
if (money > price) {  
    console.log("покупаем");  
} else {  
    console.log("отказываемся");  
}
```



Тип null

```
let message = null;
```

null – это **явное** указание на отсутствие значения.
Может быть написано программистом.



Тип undefined

```
let a;  
console.log(a); //undefined
```

undefined – означает «значение не присвоено».
Явно писать не надо, хотя это возможно.



Оператор typeof



Типизация

В Javascript	В другом языке (для сравнения)
<pre>let a = 44; //типизация не явная, т.к. тип переменной не указан явно</pre>	<pre>int x = 10; //тип указан явно</pre> <p>Здесь язык Java</p>
<pre>let value = prompt("введите значение"); value = "i'am string"; value = 50;</pre> <p>Код корректно отработает. Тип может меняться на лету, когда приходят данные. Типизация динамическая.</p>	<pre>value := 10 value = "hi"</pre> <p>Ошибка: cannot use "hi" (type string) as type int in assignment. Типизация статическая, мы не можем поменять тип переменной. Здесь язык Go.</p>
<pre>console.log(3 + "3");</pre> <p>Этот код выведет "33". Произойдет приведение к одному типу, а затем конкатенация. Типизация слабая (не строгая). Значения разных типов можно смешивать.</p>	<pre>fmt.Println(3 + "3")</pre> <p>Ошибка: invalid operation: 3 + "3" (mismatched types untyped int and untyped string) Типизация сильная(строгая). Язык Go. Значения разных типов смешивать нельзя.</p>



Домашнее задание

Актуальное домашнее задание находится в материалах урока в отдельном pdf-файле.



Вопросы участников . . .

