[DAN.IT] EDUCATION

Front-end overview.
JS Basics

Основы JavaScript

[DAN.IT] EDUCATION

JavaScript

– сайты

– мобильные приложения

– десктопные программы

например, Atom

– сервер и вспомогательные инструменты

node.js

Технологии веб-приложений

```
<!DOCTYPE html>
<html>
<head>
   <title>Page title!</title>
   <meta charset="utf-8">
</head>
<body>
    <div class="content">
    </div>
</body>
</html>
```

HTML
Разметка: структура элемента, Индексация в поисковиках, доступность, стандартное поведение
(ссылки, формы)

```
html,
body {
  margin: 0;
  padding: 0;
}
.content {
  box-sizing: border-box;
}
```

CSS
Внешний вид, простое поведение
(ховеры, фокусы)

```
'use strict';

var compose = function(fn1, fn2) {
  return function() {
    fn1(fn2);
  };
};
```

JavaScript
Любое сложное поведение. Переопределение стандартного поведения. Работа с сетью.

[DAN.IT] EDUCATION

Движки JavaScript

~~– MS Internet Explorer 11~~

– MS Edge

– Gecko (Moz;//a)

–Webkit (Safari)

– Blink (Chrome, Chromium, Opera, Yandex…)

JavaScript в браузере

– **DOM** — *Document Object Model*. Работа со страницей: разметка, события

– **BOM** — *Browser Object Model*. работа с браузером: история, хранилища, окно, cookies, таймеры, таймауты

– **HTTP** — работа с сетью

– **Графика <canvas "/>** — прослойка для создания графики

Синхронное подключение JS

```
<body>
  <header><img src="logo.jpg" /></header>
  <script src="example.js"></script>
  <main>
    <aside></aside>
    <section></section>
  </main>
</body>
```

Асинхронное подключение JS — 1

```
<body>
  <header><img src="logo.jpg" /></header>
  <script src="example.js"  async></script>
  <main>
    <aside></aside>
    <section></section>
  </main>
</body>
```

Асинхронное подключение JS — 2

```
<body>
  <header><img src="logo.jpg" /></header>
  <script src="example.js"  defer></script>
  <main>
    <aside></aside>
    <section></section>
  </main>
</body>
```

Переменные

Виды переменных:

1. **VAR**
2. **LET**
3. **CONST**

Создание переменных

– ключевое слово **var**

```
var speedOfFirstCyclist = 32;
var cyclistName = "Иван";
```

– в названии можно использовать буквы, цифры *(не в начале)*, специальные символы *(не операторы)*

```
var 1value = 32;
var value1 = 32;
var one+two = 3;
var onePlusTwo = 3;
var $price = 100;
```

– не стоит называть переменные так же, как существующие конструкции языка

```
var var = "var";
```

Название переменных

– английское существительное в именительном падеже для значений. Начинается с маленькой буквы

```
var userName = 'Ivan';
var age = 120;
var birthYear = 1988;
```

– английский глагол или фраза содержащая английский глагол для функций

```
var doSomethingGreat = function() {};
```

Разделение слов в переменных

– **camelCase** 
слова не отделяются пробелами, каждое последующее слово начинается с заглавной буквы

– **snake_case** 
вместо пробелов слова разделяются нижним подчеркиванием, каждое последующее слово
начинается
с прописной буквы

Типы данных

Примитивные

1. **строки (текст)**
2. **числа**
3. **логические (булевы значения)**
4. **служебные (null, undefined)**

Сложные

1. **Массив**
2. **Обьект**
3. **Symbol**

Числа *(number)*

формат данных для представления числовой
информации: целых, дробных чисел, бесконечности и неизвестного
числового значения

– записываются без дополнительных символов
– поддерживается работа с целыми и дробными числами, дробная часть
отделяется
точкой
– существует два дополнительных значения: Infinity и NaN
– целые числа могут быть записаны в десятеричном или в
шестнадцатеричном
формате 0xFF

Строки *(string)*

формат данных для представления текстовой

Информации

Строки *(string)*

– записываются внутри одинарных или двойных кавычек

'Hello, I am a string';

"Hello, I am also a string";

– называются строками, потому что переносы, пробелы и отступы являются обычными
символами, поэтому весь текст хранится в виде одной длинной последовательности
символов

– в случае, если в строке нужно использовать специальный символ, используется
экранирование

'\tHello, I\'m a string\n\n';

Булев *(логический)*

результат логических выражений. Имеет только два значения — истина (true) и ложь (false)

Служебные (null, undefined)

Значение «null»

Специальное значение null не относится ни к одному из типов, описанных выше. Оно формирует отдельный тип, который содержит только значение null:

```
let age = null;
```

Служебные (null, undefined)

Значение «undefined»

Специальное значение undefined также стоит особняком. Оно формирует тип из самого себя так же, как и null. Оно означает, что «значение не было присвоено». Если переменная объявлена, но ей не присвоено никакого значения, то её значением будет undefined:

```
let age;
alert(age); // выведет "undefined"
```

Массив

упорядоченный набор элементов

Массив

Используется для описания группы значений, связанных между собой:

обращение

идёт сразу ко всем значениям массива

Элементы массива упорядочены

Элементы могут повторяться

Доступ к элементам производится по порядковому номеру

Нумерация начинается с 0 *(аналогия: года или количество элементов перед элементом)*

Массив

создается с помощью квадратных скобок *(литерал массива)*

```
var numbers = [];
```

элементы массива перечисляются внутри квадратных скобок через запятую

```
var numbers = [1, 2, 3];
var numbers = ['1', '2', '3'];
```

доступ к отдельным элементам массива производится через квадратные скобки

```
console.log(numbers[0]);
```

выражение в квадратных скобках может быть любым, главное, чтобы оно возвращало число `numbers[ Math.floor(Math.random( ) * numbers.length ) ]`

Объекты

сложный тип данных, где у каждой записи есть название

Объекты

Как описать кота?

- кличка
- пол
- порода
- окрас
- тип шерсти
- рост, вес

Объекты

Как описать кота?

один кот отличается от другого полным набором указанных данных, поэтому тип данных, описывающий кота, должен быть сложным, то есть *сложенным* из множества простых данных

Объекты

Объекты в JS

набор пар ключ-значение

Объекты

создается с помощью фигурных скобок *(литерал объекта)*

```
var cat = {};
```

элементы объекта представляют собой набор пар ключ-значение, которые отделяются

друг от друга двоеточием. Между собой пары разделяются запятыми

```
var cat = {
  name: 'Фёдор ',
  age: 4
};
```

доступ к элементам объекта происходит через точку или через квадратные скобки `console.log(cat.name, cat['age']);`

Symbol

«Символ» представляет собой уникальный идентификатор.
Создаются новые символы с помощью функции Symbol():

```
// Создаем новый символ - id
let id = Symbol();
```

При создании символу можно дать описание (также называемое имя),
в основном используется для отладки кода:

```
// Создаем символ id с описанием (именем) "id"
let id = Symbol("id");
```

Symbol

Символы гарантированно уникальны. Даже если мы создадим множество символов с одинаковым описанием, это всё равно будут разные символы. Описание – это просто метка, которая ни на что не влияет.
Например, вот два символа с одинаковым описанием – но они не равны:

```
let id1 = Symbol("id");
let id2 = Symbol("id");

alert(id1 == id2); // false
```

Операторы

– **унарные**

операторы, которые работают с одним значением

– **бинарные**

операторы, работающие с двумя значениями

Преобразование типов

Строковое преобразование

String()

Численное преобразование

Number()

Логическое преобразование

Boolean()

Операторы

- математические
- строковый оператор
- логические операторы
  - операторы сравнения
  - операторы булевой логики
- служебные операторы

Приведение типов

Приведение типов —

если операнды переданные оператору разного типа, они автоматически приводятся к типу, в котором операцию можно выполнить. Тип, в который будут приведены операнды зависит от оператора

Приведение типов

– '10' / '2' === 5
математический оператор — приведение производится к числу

– true + 'eee' === 'trueeee'
строковый оператор — приведение производится к строке

– '10' + 2 === '102'
у строкового сложения приоритет над математическим

Явное приведение

Приведение встроенными методами

– приведение к числу

```
parseInt('10', 10) === 10;
parseFloat('10.2') === 10.2;
```

– приведение к строке

```
10.toString(); Uncaught SyntaxError
10..toString() === '10';
10.0.toString() === '10';
```

Приведение операторами

– приведение к числу с помощью оператора знака (унарный плюс или унарный минус)

```
+'10' === 10;
-'3' === -3;
```

– приведение к строке сложением с пустой строкой

```
3 + '' === '3';
```

– приведение к булеву значению двойным отрицанием

```
!!0 === false;
!!1 === true;
```

Логические операторы

- операторы сравнения
  - операторы сравнения >, <, >=, <=
  - строгое и нестрогое сравнение ==, ===, !=, !==
- операторы математической *(булевой)* логики
  - или ||
  - и &&
  - не !