Типы данных.

Продолжение

Чистяков Денис

В прошлой лекции мы обсудили:

- Строки, массивы, объекты и функции
- Отличия примитивных типов данных от сложных
- Основные методы для работы со строками и массивами

Методы объекта

```
var tweet = {
  likes: 16,
  getLikes: function() {
    return this.likes;
  },
  setLikes: function(value) {
    this.likes = parseInt(value) || 0;
    return this;
  },
  getAuthor: function() {
    return this.user.screenName;
  }
};
```

Методы объекта

```
tweet.getLikes(); // 16
```

```
tweet.setLikes(17) // { … }
     .getLikes();  // 17
```

Обработка исключений

```
var tweet = {
  likes: 16,
  getLikes() {
    return this.likes;
  },
  setLikes(value) {

    var likes = parseInt(value);

    if (isNaN(likes) || likes < 0) {
        throw new TypeError('Передано неверное значение');
    }

    this.likes = likes;
  }
};
```

Обработка исключений

```
try {
  tweet.setLikes('foo');
} catch (e) {
  if (e instanceof TypeError) {

    tweet.setLikes(0);
  }
  console.error(e);
}

tweet.getLikes(); // 0
```

TypeError

```
// Имя типа ошибки
e.name; // 'TypeError'
```

```
// Сообщение ошибки


e.message; // 'Передано неверное значение'
```

```
// Стек вызовов
e.stack;
// TypeError: Передано неверное значение
//     at Object.set likes [as likes] (<anonymous>:10:15)
//     at <anonymous>:18:15
```

✉️ == 'Ого сколько фронтендеров. #wstdays

```
var tweet = {
  toString: function() {
    return 'Ого сколько фронтендеров. #wstdays';
  }


}
```

```
var anotherTweetText = 'Я и IoT, пятый доклад на WSD в Питере
```

```
tweet == anotherTweetText; // ???
```

✉️ == 'Я и ИоТ, пятый доклад на WSD в Пите

```
isPrimitive(tweet); // false
```

```
isPrimitive(anotherTweetText); // true
```

```
typeof tweet.toString === 'function'; // true
```

```
tweet.toString() === anotherTweetText; // false
```

```
'Ого сколько фронтендеров. #wstdays' === 'Я и ИоТ, пятый докл
```

✉️ == 'Ого сколько фронтендеров. #wstdays

```
var tweet = {
  toString: function() {
    return 'Ого сколько фронтендеров. #wstdays';
  }


}
```

```
var tweetText = 'Ого сколько фронтендеров. #wstdays'
```

```
tweet == tweetText; // true
```

Операция сравнения двух сложных типов вернет истину только в том случае, если внутренние

ссылки обоих объектов ссылаются на один и тот же объект в памяти

```
{} == {}; // false
```

Нестрогое и строгое сравнения

```
var tweet = {
  toString: function() {
      return 'Ого сколько фронтендеров. #wstdays';
  }


}
```

```
tweet == 'Ого сколько фронтендеров. #wstdays'; // true
```

```
tweet === 'Ого сколько фронтендеров. #wstdays'; // false
```

Приведение объекта к строке

```
var tweet = {
  toString: function() {
      return 'Ого сколько фронтендеров. #wstdays';
  }


}
```

```
String(tweet); // 'Ого сколько фронтендеров. #wstdays'
```

```
'' + tweet; // 'Ого сколько фронтендеров. #wstdays'
```

Неперечисляемые методы

```
var tweet = {
  toString: function() {
      return 'Ого сколько фронтендеров. #wstdays';
  }
};


Object.keys(tweet); // ['toString']
```

```
var emptyObject = {};
Object.keys(emptyObject); // []
```

```
typeof tweet.toString === 'function'; // true
typeof emptyObject.toString === 'function'; // true
```

Объявление методов объекта

```
var tweet = {};
```

```
Object.defineProperty(tweet, 'toString', {

  value: function() {
    return 'Ого сколько фронтендеров. #wstdays',
  enumerable: false,
  configurable: true
});
```

Object.defineProperties

Значения параметров `writable`, `enumerable` и `configurable` по умолчанию — `false`.

Объявление методов объекта. `writable`

```
var tweet = {};

Object.defineProperty(tweet, 'text', {
    value: 'Ого сколько фронтендеров. #wstdays',
    writable: false


});

Object.getOwnPropertyDescriptor(tweet, 'text');
// { value: 'Ого сколько фронтендеров. #wstdays',
//   writable: false,
//   enumerable: false,
//   configurable: false }

tweet.text; // 'Ого сколько фронтендеров. #wstdays'
tweet.text = 'Вёрстка писем. Развенчиваем мифы. … #wstdays';
tweet.text; // 'Ого сколько фронтендеров. #wstdays'
```

Объявление методов объекта. enumerable

```
var tweet = {};

Object.defineProperty(tweet, 'text', {
    value: 'Ого сколько фронтендеров. #wstdays',
    enumerable: false


});

Object.getOwnPropertyDescriptor(tweet, 'text');
// { value: 'Ого сколько фронтендеров. #wstdays',
//   writable: false,
//   enumerable: false,
//   configurable: false }

Object.keys(tweet); // []
```

Объявление методов объекта. configurable

```
var tweet = {};

Object.defineProperty(tweet, 'text', {
    value: 'Ого сколько фронтендеров. #wstdays',
    configurable: false
});
```

```
Object.getOwnPropertyDescriptor(tweet, 'text');
// { value: 'Ого сколько фронтендеров. #wstdays',
//   writable: false,
//   enumerable: false,
//   configurable: false }

tweet.text; // 'Ого сколько фронтендеров. #wstdays'
delete tweet.text; // false
tweet.text; // 'Ого сколько фронтендеров. #wstdays'
```

Геттеры и сеттеры

```
var tweet = {
  likes: 16,
  getLikes: function() {
    return this.likes;

  },
  setLikes: function(value) {
    this.likes = parseInt(value) || 0;
    return this;
  }
};
```

Геттеры и сеттеры

```
var tweet = {
    _likes: 16
};
```

```
Object.defineProperty(tweet, 'likes', {
  get: function() {
    return this._likes;
  },
  set: function(value) {
    this._likes = parseInt(value) || 0;
  }
});
```

Геттеры и сеттеры

```
// Сработал геттер
tweet.likes; // 16
```

```
// Сработал сеттер
tweet.likes = 17;
```

```
// Сработал геттер
tweet.likes; // 17
```

Заморозка

```
var tweet = {
  likes: 16,
  getLikes: function() {
    return this.likes;
  }


};
```

```
Object.getOwnPropertyDescriptor(tweet, 'likes')
// { value: 16,
//   writable: true,
//   enumerable: true,
//   configurable: true }
```

Заморозка

```
Object.freeze(tweet);
```

```
Object.getOwnPropertyDescriptor(tweet, 'likes')
// { value: 16,
//   writable: false,
//   enumerable: true,
//   configurable: false }
```

Заморозка

```
Object.isFrozen(tweet); // true
```

```
tweet.likes = 17;
```

```
tweet.likes; // 16
```

```
delete tweet.likes; // false
```

Объект Даты

```
// Создает объект с текущей датой в системном часовом поясе
new Date(); // Mon Oct 17 2016 09:37:20 GMT+0500 (YEKT)

tweet.createdAt; // 'Sat Oct 01 12:01:08 +0000 2016'
// Пытаемся сконвертировать строку в дату
new Date(tweet.createdAt); // Sat Oct 01 2016 17:01:08 GMT+0500 (YEKT)

// Создаем дату из UNIX Timestamp
new Date(1475323268000); // Sat Oct 01 2016 17:01:08 GMT+0500 (YEKT)

// Создаем дату из набора параметров
new Date(2016, 9, 1, 17, 1, 8); // Sat Oct 01 2016 17:01:08 GMT+0500 (YE

// Получаем UNIX Timestamp из даты
(new Date(2016, 9, 1, 17, 1, 8)).valueOf(); // 1475323268000

// Получаем текущее значение UNIX Timestamp
Date.now(); // 1476680054602
```

Date

Math — библиотека математических функций и констант

```
// Генерируем случайное число от 0 до 1
Math.random(); // 0.4468546273336771
```

```
// Определяем меньшее из чисел
Math.min(1, 5); // 1
```

```
// Определяем большее из чисел
Math.max(1, 5, 10); // 10
```

Math — библиотека математических функций и констант

```
// Округляем число до ближайшего целого
Math.round(2.7); // 3
Math.round(2.3); // 2
```

```
// Округляем число до целого в меньшую сторону
Math.floor(2.7); // 2
Math.floor(2.3); // 2
```

```
// Округляем число до целого в большую сторону
Math.ceil(2.7); // 3
Math.ceil(2.3); // 3
```

Math — библиотека математических функций и констант

```
// Возвращает натуральный (по основанию e) логарифм числа
Math.log(10); // 2.302585092994046
```

```
// Возвращает основание, возведённое в степень

Math.pow(2, 5); // 32
```

```
// Возвращает синус угла в радианах
Math.sin(1); // 0.8414709848078965
```

```
// Возвращает тангенс угла в радианах
Math.tan(1); // 1.5574077246549023
```

Регулярные выражения

Имеют стандартный PCRE-синтаксис

PCRE (Perl Compatible Regular Expressions)

Руководство по регулярным выражениям

Регулярные выражения

```
tweet.text; // 'Node.js, и модули, Джеймс о проблемах Node.j

// Проверяем содержится ли указанное регулярное выражение в
/#[a-z0-9]+/gi.test(tweet.text); // true
```

g — глобальное сопоставление

i — игнорирование регистра при сопоставлении

```
var tweetWithoutHashtag; // 'Я и IoT, пятый доклад на WSD в

/#[a-z0-9]+/gi.test(tweetWithoutHashtag); // false
```

Регулярные выражения

```
var tweet = {
  text: 'Node.js, и модули, Джеймс о проблемах Node.js #node
};
```

```
Object.defineProperty(tweet, 'linkify', {
  get: function() {
    return this.text.replace(
      /#[a-z0-9]+/gi,
      '<a href="$1">$1</a>'
    );
  }
});
```

Регулярные выражения

```
Object.getOwnPropertyDescriptor(tweet, 'linkify');
// { get: [Function: get],
//   set: undefined,
//   enumerable: false,

//   configurable: false }
```

```
tweet.linkify;
// 'Node.js, и модули, Джеймс о проблемах Node.js
// <a href="$1">$1</a> <a href="$1">$1</a> #модули'
```

Регулярные выражения

```
return this.text.replace(
  /#[a-z0-9а-я]+/gi,
  '<a href="$1">$1</a>'


);
```

```
tweet.linkify;
// 'Node.js, и модули, Джеймс о проблемах Node.js
// <a href="$1">$1</a> <a href="$1">$1</a> <a href="$1">$1</a
```

Регулярные выражения

```
return this.text.replace(
  /(#[a-z0-9а-я]+)/gi,
  '<a href="$1">$1</a>'


);
```

```
tweet.linkify;
// 'Node.js, и модули, Джеймс о проблемах Node.js
// <a href="#nodejs">#nodejs</a> <a href="#modules">#modules
```

Регулярные выражения

```
return this.text.replace(
  /(#([a-z0-9а-я]+))/gi,
  '<a href="$2">$1</a>'


);
```

```
tweet.linkify;
// 'Node.js, и модули, Джеймс о проблемах Node.js
// <a href="nodejs">#nodejs</a> <a href="modules">#modules</a
```

Регулярные выражения

```
return this.text.replace(
  /(#([\w]+))/gi,
  '<a href="$2">$1</a>'
);
```

```
tweet.linkify;
// 'Node.js, и модули, Джеймс о проблемах Node.js
// <a href="nodejs">#nodejs</a> <a href="modules">#modules</a
```

`\w` — соотвествует любому цифробуквенному символу, включая нижнее подчеркивание.

Эквивалентен `[A-Za-z0-9_]`

Судя по всему, буквами они называют только латинский алфавит :)

Домашнее задание

`0b11` друзей Оушена