

Node.JS | localStorage

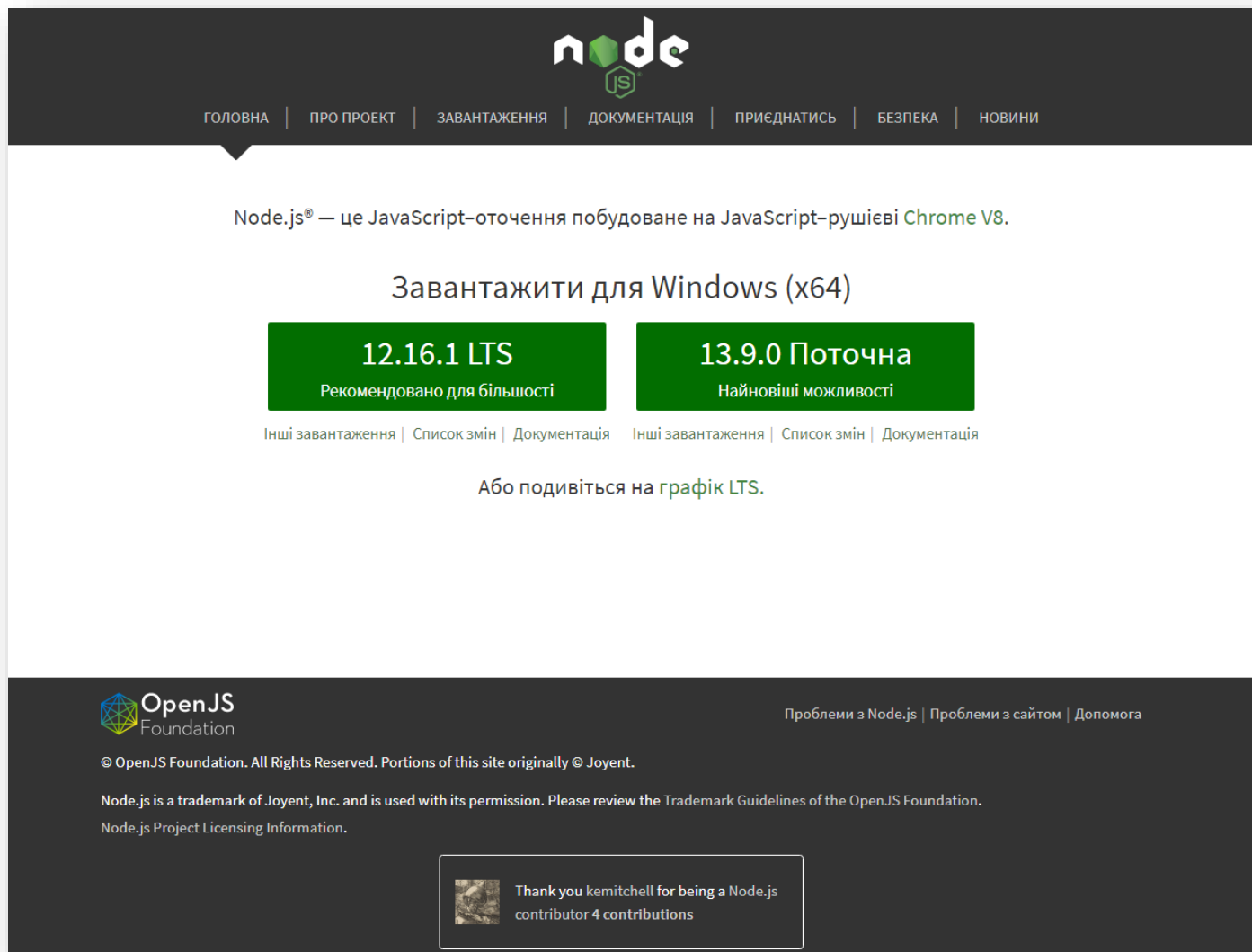


JavaScript
Courses

www.courses.dp.ua

Node.js

JavaScript вне браузера



Node.JS

Node.JS – «виртуальная машина» способная выполнять JavaScript-код, которую можно установить на компьютере, и которая построена на базе «куска» браузера «Chrome» отвечающего за обработку JavaScript. **Node.JS** позволил превратить JavaScript в язык общего пользования, поставив его в один ряд с Python, Ruby, Java, C# и другими.

<https://nodejs.org/en/>

Node.JS

Node.JS – чистый **ECMAScript**. И самый «свежий» **ECMAScript**, т.к. разрабатывая на node.JS не нужно думать о совместимости с браузерами.

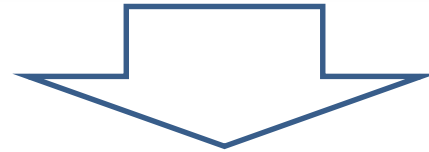
Если в браузере весь инструментарий обеспечивался нам объектом **window**, то в **node.JS** его нет, в нём «из коробки» нет ничего кроме чистого синтаксиса **JavaScript/ECMAScript-2015/2016/2017/2018/2019/....**

Но, в **node.JS** есть модули (пакеты) для решения тех или иных задач. (по аналогии с подключаемыми файлами в языке C/C++), и модулей для него существует большое количество.

<https://nodejs.org/en/>

Как работает Node.JS

```
var a = "Hello";  
var b = "world!!!";  
var c = a + " " + b;  
console.log(c);
```



```
Windows PowerShell  
(C) Корпорация Майкрософт (Microsoft Corporation), 2015. Все права защищены.  
  
PS F:\> cd node_dir  
PS F:\node_dir> node code.js  
Hello world!!!  
PS F:\node_dir> _
```

Консольная команда **node code.js** позволяет запустить на выполнение файл с JS-кодом.

Node.JS

Node работает на стороне сервера (непосредственно на компьютере, без песочниц типа браузера), для **Node** не существует **DOM** и прочей инфраструктуры браузера, т.е. никаких функций **prompt()**, **alert()** не существует (к сожалению).

Нет и объекта **XMLHttpRequest**, но это не мешает **Node** загружать данные из сети.

В **Node.JS** есть модули и есть функция **require()**.

Пакеты (модули) в Node.JS

```
1 var request_function = require('request');
2
3 var url = "http://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?json";
4
5 request_function(url, function(err, response, data){
6     if(!err){
7         console.log("Data recieved: " + data);
8     }else{
9         console.log("Request fail, error: " + err);
10    }
11 });
```

Однако не все пакеты идут в комплекте с **Node.JS**....

```
> Windows PowerShell
PS G:\nodejs> node code.js
module.js:329
    throw err;
    ^

Error: Cannot find module 'request'
    at Function.Module._resolveFilename (module.js:327:15)
    at Function.Module._load (module.js:278:25)
    at Module.require (module.js:355:17)
    at require (internal/module.js:13:17)
```

NPM – Node Package Manager

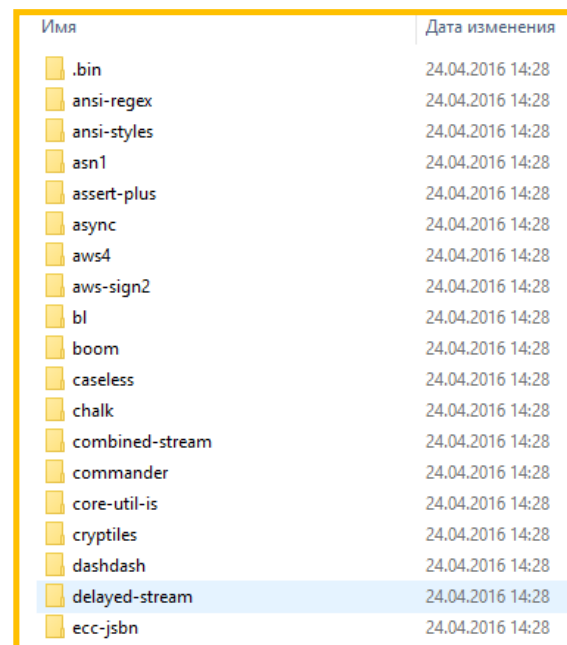
Всемирная библиотека пакетов (модулей) для Node.JS



<https://www.npmjs.com/>

Когда пакета не хватает, то пишем:

И система управления пакетами установит в текущую папку требуемый модуль и все зависимые модули тоже.



Пакеты в Node.JS

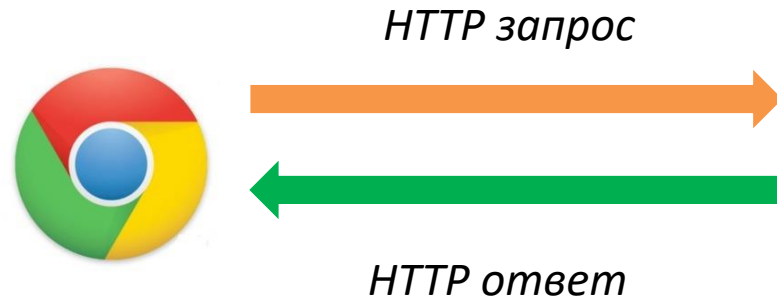
```
1 var request_function = require('request');
2
3 var url = "http://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?json";
4
5 request_function(url, function(err, response, data){
6   if(!err){
7     console.log("Data recieved: " + data);
8   }else{
9     console.log("Request fail, error: " + err);
10  }
11 });
```

```
PS F:\node_dir> node code.js
Data recieved: [{"r030":974,"txt":"Білоруський рубль","rate":0.00124,"cc":
,"txt":"Євро","rate":27.459078,"cc":"EUR","exchangedate":"12.07.2016"},{
"exchangedate":"12.07.2016"},{"r030":985,"txt":"Злотий","rate":6.211056,
986,"txt":"Бразильський реал","rate":7.707097,"cc":"BRL","exchangedate":
3.155619,"cc":"TJS","exchangedate":"12.07.2016"},{"r030":643,"txt":"Росій
edate":"12.07.2016"},{"r030":933,"txt":"Білоруський рубль","rate":12.3519
0":944,"txt":"Азербайджанський манат","rate":16.047067,"cc":"AZN","excha
ійський долар","rate":18.785714,"cc":"AUD","exchangedate":"12.07.2016"},
08612,"cc":"AMD","exchangedate":"12.07.2016"},{"r030":975,"txt":"Болгарск
ate":"12.07.2016"},{"r030":124,"txt":"Канадський долар","rate":19.019933
:152,"txt":"Чилійське песо","rate":0.0375579,"cc":"CLP","exchangedate":}
```

Теперь всё в порядке, модуль находится рядом с нашим проектом, и может быть задействован.

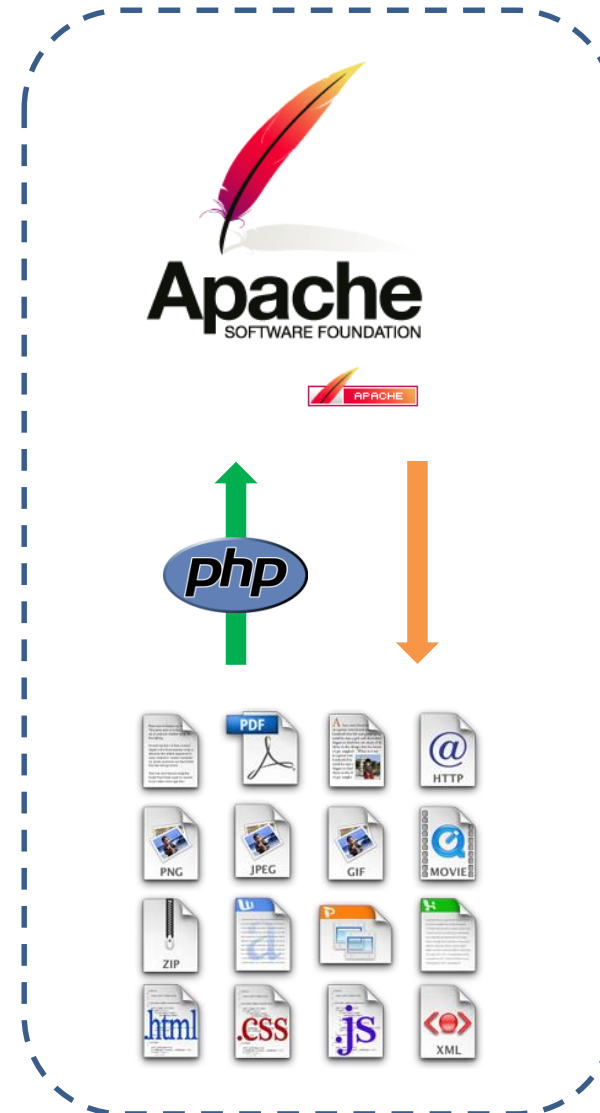
JavaScript на стороне сервера

HTTP-сервер (Web-сервер)

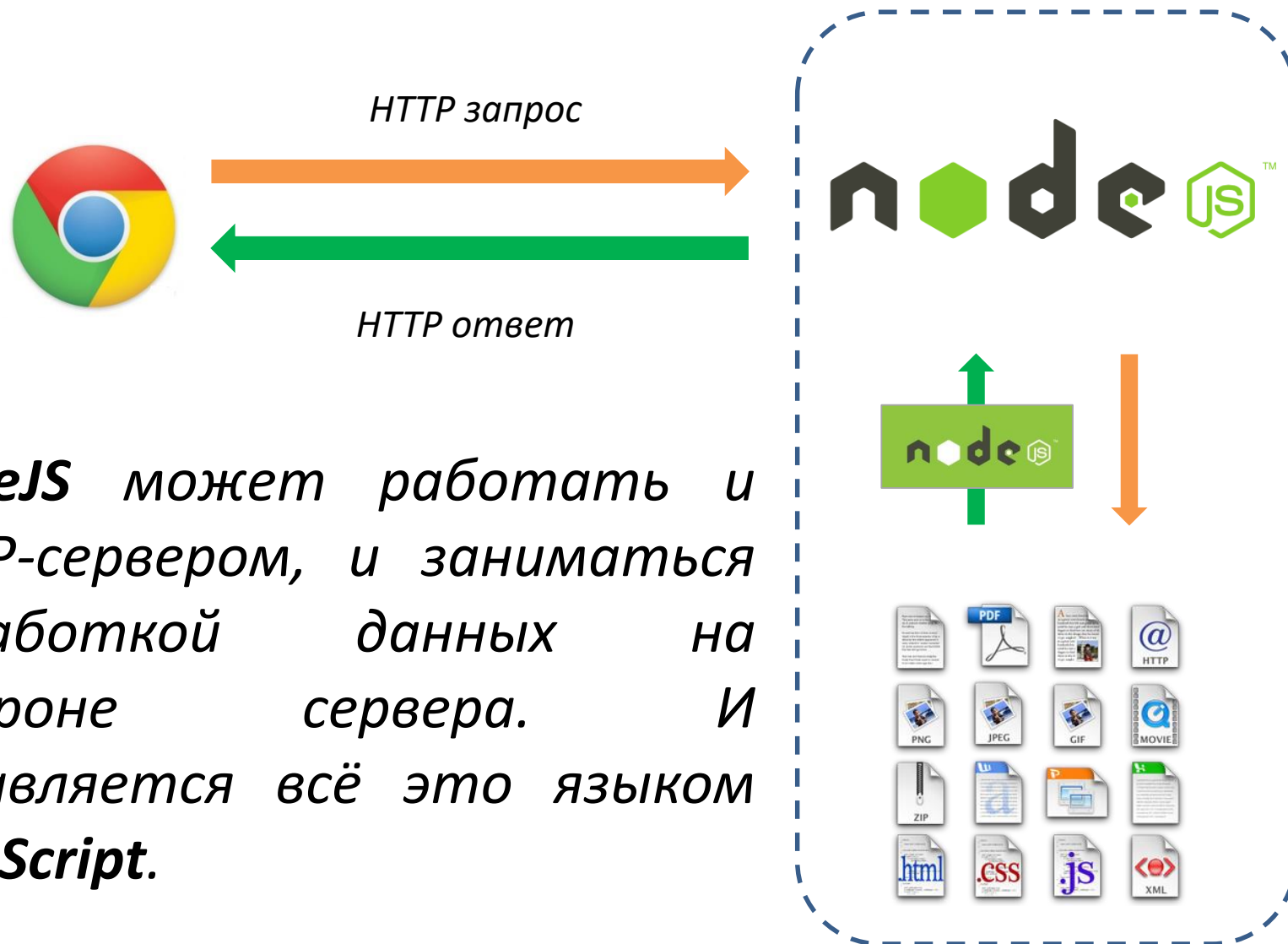


Основная задача **веб-сервера** ждать запросы от браузера и отправлять браузеру **веб-страницу** (и сопутствующие файлы) когда браузер их запросит.

Перед отправкой файлов, сервер может их модифицировать, при помощи сценариев написанных на одном из языков программирования, например **PHP**.



NodeJS может быть HTTP-сервером (Web-сервером)



NodeJS может работать и HTTP-сервером, и заниматься обработкой данных на стороне сервера. И управляется всё это языком **JavaScript**.

HTTP сервер на базе NodeJS

```
1  var http = require("http");
2
3  function when_request(request_data, response_data){
4      response_data.setHeader("Content-type", "text/html;");
5      response_data.write("<h1>Hello from my Node.JS HTTP server!");
6      response_data.end();
7  }
8
9  var server = http.createServer(when_request);
10 server.listen(80); //Ждём соединения на 80-м порту.
11 //Если 80-й порт занят другим приложением используйте например 8080
12 //И открывайте в браузере адрес http://localhost:8080
```

После запуска скрипта, и до закрытия окна консоли сервер ждём запросы и в ответ отсылает заданный фрагмент HTML-разметки. Сервер обрабатывает запросы ко всем IP-адресам которые есть на компьютере.

HTTP сервер на базе NodeJS

```
1  var http = require("http");
2
3  function when_request(request_data, response_data){
4      response_data.setHeader("Content-type", "text/html;");
5      response_data.write("<h1>Hello from my Node.JS HTTP server!");
6      console.log(request_data.method + " >> " + request_data.url);
7      response_data.end();
8  }
9
10 var server = http.createServer(when_request);
11 server.listen(80); //Ждём соединения на 80-м порту.
12 //Если 80-й порт занят другим приложением используйте например 8080
13 //И открывайте в браузере адрес http://localhost:8080
```

Модернизируем скрипт,
чтобы видеть в консоли детали запроса.

Свой статический веб-сервер

Свой статический веб-сервер на базе Node.JS

Статический веб-сервер – сервер который принимает запрос, ищет подходящий файл и отправляет его клиенту, другой обработки поступающих запросов не проводится.

Статический веб-сервер на базе Node.JS

```
1  var http      = require('http');
2  var ns        = require('node-static');
3
4  static_server = new ns.Server(".");
5
6  http.createServer((request, response) => {
7      static_server.serve(request, response);
8  }).listen(80);
9
```

Статический веб-сервер – сервер который принимает запрос, ищет подходящий файл и отправляет его клиенту, другой обработки поступающих запросов не проводится.

package.json

package.json – перечень зависимостей

```
npm init
```

Файл *package.json* – хранит информацию о приложении, в частности о необходимых пакетах.

```
npm install some_pack --save
```

Команда `npm init` – позволяет создать файл *package.json*. Использование ключа `--save` добавляет устанавливаемый пакет в файл *package.json*.

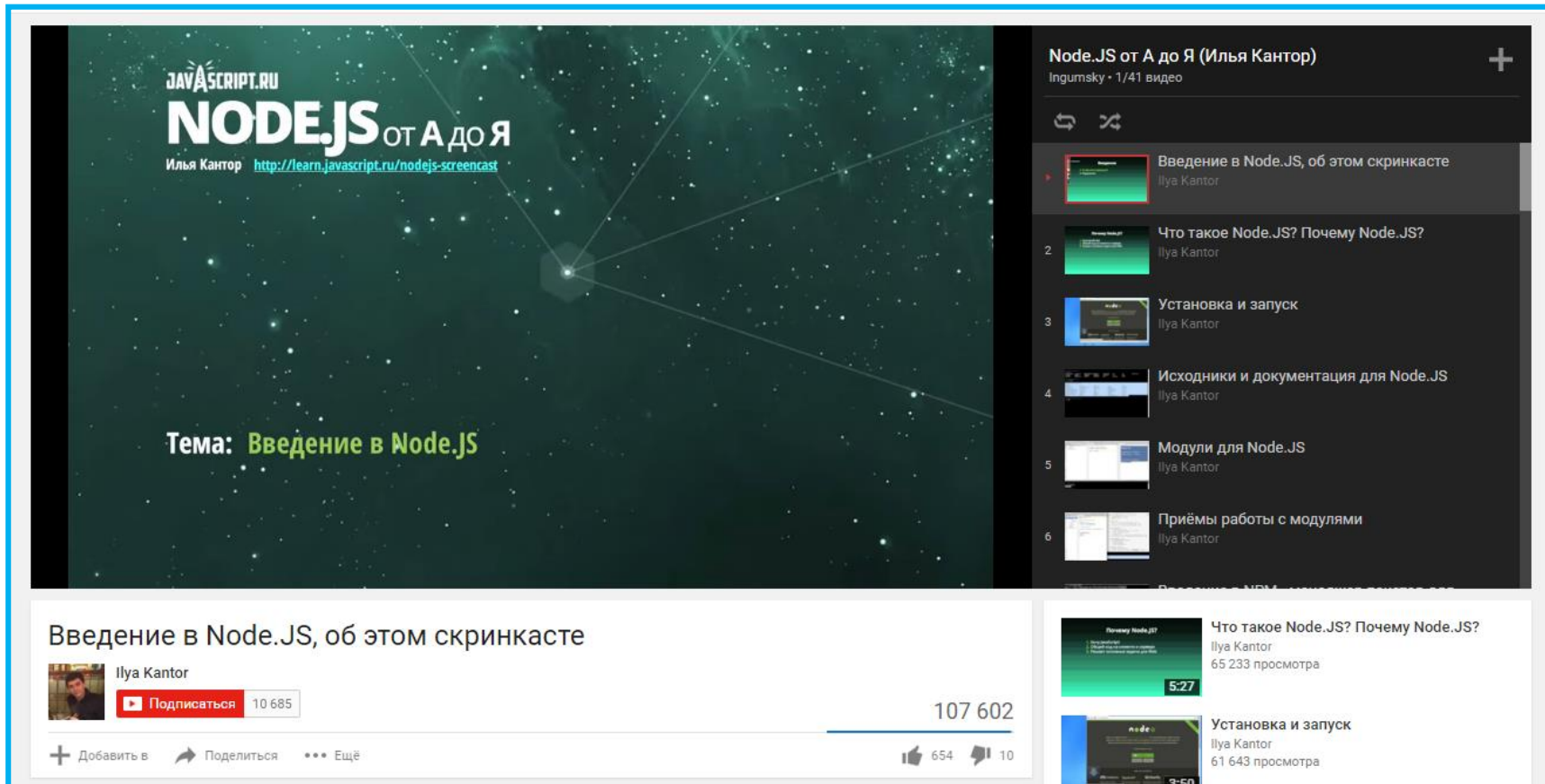
```
npm install
```

Использование команды `npm install` позволяет установить все пакеты из *package.json*, если они еще не установлены.

Подробнее о Node.js

Введение в Node.JS от Ильи Кантора*

*создателя *javascript.ru*



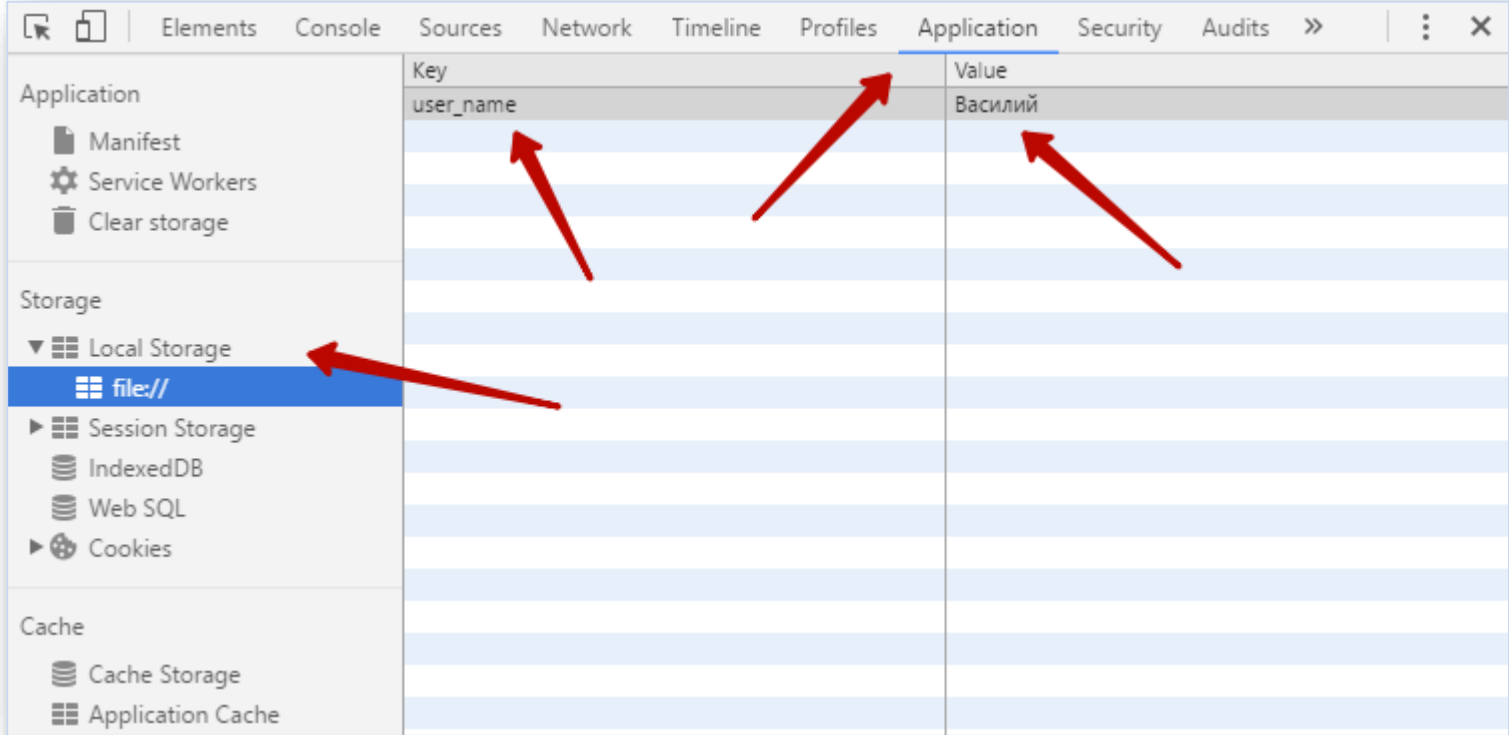
The screenshot displays a YouTube video player interface. The main video area shows a dark green background with a constellation pattern and the text "JAVASCRIPT.RU NODE.JS ОТ А ДО Я" and "Илья Кантор <http://learn.javascript.ru/nodejs-screencast>". Below this, it says "Тема: Введение в Node.JS". The video title is "Введение в Node.JS, об этом скринкасте" by Ilya Kantor, with 107,602 views. The video player includes a progress bar, a "Подписаться" (Subscribe) button, and a "10 685" subscriber count. To the right, a playlist is visible with six items: "Введение в Node.JS, об этом скринкасте", "Что такое Node.JS? Почему Node.JS?", "Установка и запуск", "Исходники и документация для Node.JS", "Модули для Node.JS", and "Приёмы работы с модулями". The first item is highlighted, and the second item is shown with a duration of 5:27. The third item is also shown with a duration of 3:50.

<https://www.youtube.com/watch?v=ILpS4Fq3lmw&list=PLsuEohlthXdkRSxJTkmTstWKHgBHsd3Dx>

LocalStorage

(хранилище данных в браузере)

LocalStorage – локальное хранилище данных



Локальное хранилище (Объект **LocalStorage**) позволяет сохранять данные в браузере (не путать с технологией cookie) и после (при последующих посещениях страницы) считывать ранее сохраненные данные.

LocalStorage на практике

Пополнение телефона

Номер телефона:

Номер банковской карты:

Сумма пополнения:

грн.

Пополнить

Воспользуйтесь заготовкой: [./source/example_01](#)

Сохраним вводимую пользователем информацию, на случай непредвиденного закрытия страницы или её обновления.

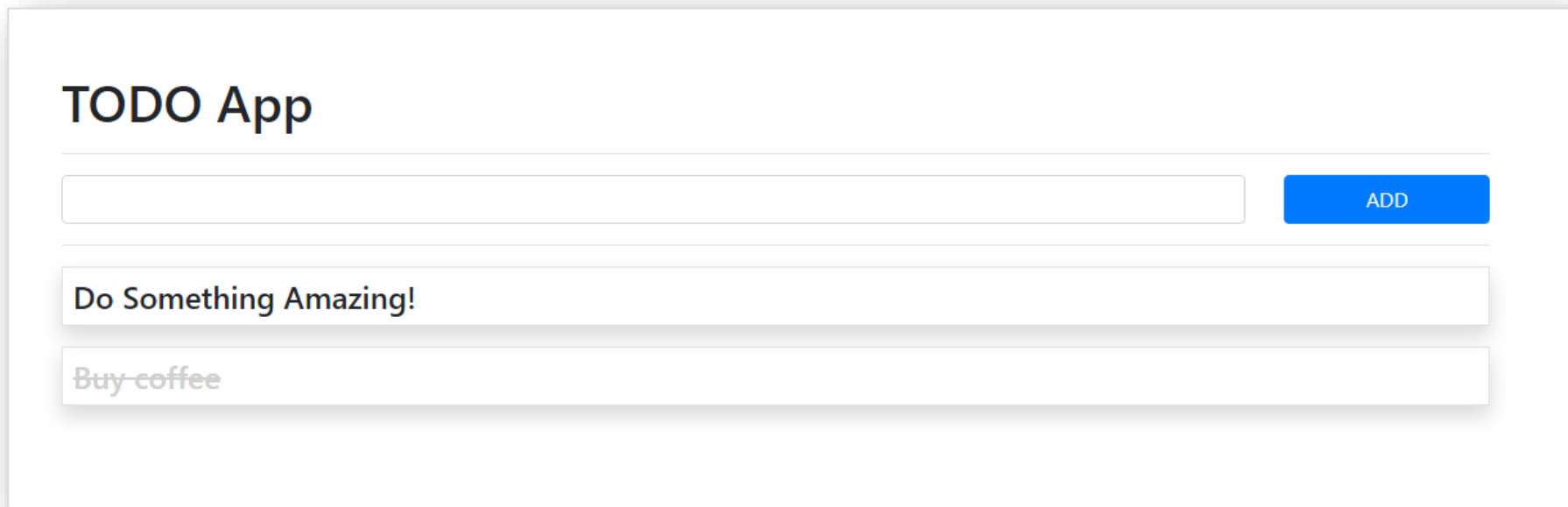
LocalStorage на практике

```
29 <script>
30
31 function soft_safe(element){
32     var title = element.id;
33
34     element.value = !localStorage.getItem(title) ? element.value : localStorage.getItem(title);
35
36     element.oninput = function(e){
37         localStorage.setItem(title, element.value);
38     }
39
40 }
41
42 window.addEventListener("DOMContentLoaded", function(){
43     soft_safe(phone_number);
44     soft_safe(card_number);
45     soft_safe(pay_sum);
46 });
47
48 </script>
```

Сохраним вводимую пользователем информацию, на случай непредвиденного закрытия страницы или её обновления. «Универсальный» вариант решения. Для работы с объектом **localStorage** достаточно использовать два метода **localStorage.setItem(name, value)** и **localStorage.getItem(name)** которые соответственно сохраняет и извлекает из локального хранилища.

TODO приложение

TODO App



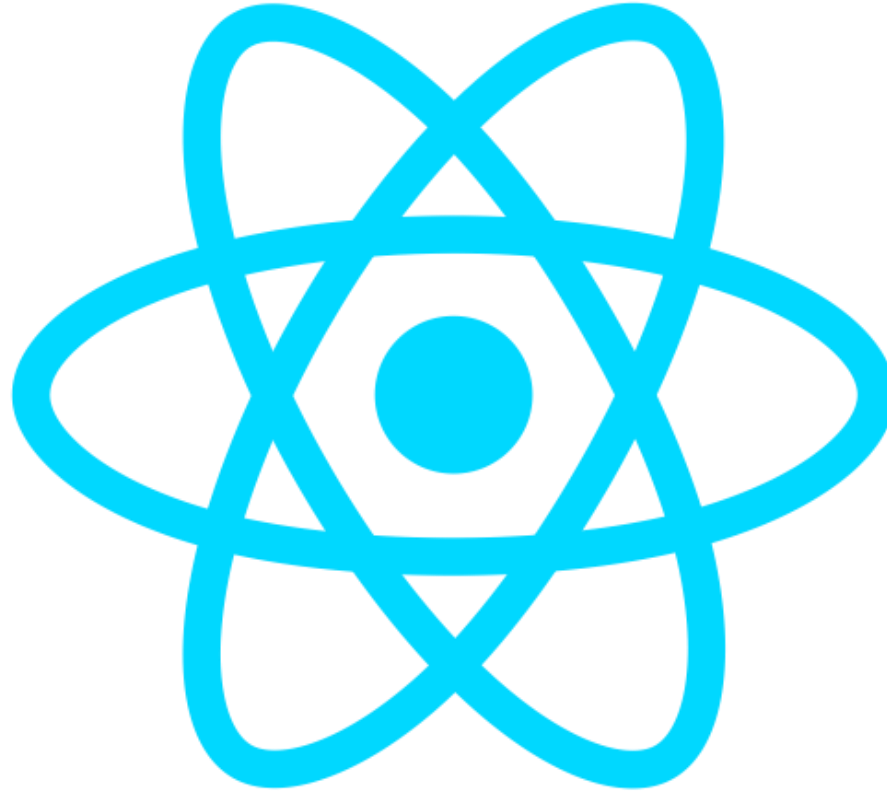
The screenshot shows a web application titled "TODO App". It features a text input field at the top, followed by a blue "ADD" button. Below the input field, there is a list of tasks. The first task is "Do Something Amazing!" in a standard black font. The second task is "Buy coffee" in a lighter gray font, indicating it has been completed. The interface is clean and minimalist, with a white background and subtle shadows.

Воспользуйтесь заготовкой: [./source/example_02](#)

Реализуем простое приложение: **TODO** - список дел. В котором пользователь сможет добавлять новые дела, помечать дела как выполненные (по клику), и удалять дела (по двойному клику). Данные приложения будут храниться в **localStorage**е браузера.

Домашнее задание
/узнать

Познакомиться с React



Посмотрите этот набор из 12 видео:

<https://www.youtube.com/watch?v=ol4OVMJZC1w&list=PLDyvV36pndZEz2unvD0a2Spv7RehBrpDO&index=1>