<Teach
Me
Skills/>

Lesson 10

Препроцессоры и сборщики. Введение.



Препроцессоры и сборщики

- Понятие 'css-препроцессор'
- Понятие 'синтаксический сахар'
- Node.js
- Менеджер пакетов прт
- Parcel
- Разновидности препроцессоров Sass, Less, Stylus

CSS-препроцессоры

CSS препроцессор - это надстройка над CSS, которая добавляет ранее недоступные возможности для CSS, с помощью новых синтаксических конструкций.

Основная задача препроцессора - это предоставление удобных синтаксических конструкций для разработчика, чтобы упростить, и тем самым, ускорить разработку и поддержу стилей в проектах.

CSS препроцессоры преобразуют код, написанный с использованием препроцессорного языка, в чистый и валидный CSS-код.

При помощи препроцессоров вы можете писать код, который нацелен на:

- :1 читабельность для человека
- 2 структурированность и логичность
- 3 производительность





CSS-препроцессоры

Синтаксический сахар - это дополнения синтаксиса языка программирования, которые не вносят каких-то существенных изменений или новых возможностей, но делают этот язык более читабельным для человека.

Синтаксический сахар вводит в язык альтернативные варианты записи заложенных в этот язык конструкций.

Под альтернативными вариантами записи стоит понимать более короткие или удобные конструкции для человека, которые в конечном итоге будут преобразовываться препроцессором в исходный язык, без синтаксического сахара.



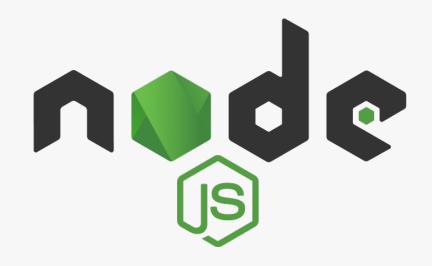




Node.js - введение

Так, как css-препроцессоры не могут исполняться в простой среде разработки, они зависят от дополнительных программных решений, таких как: Node,js, Npm, Parcel, Webpack, Gulp etc.

Таким образом складывается следующая последовательность: Node.js + Npm + Pacrcel.

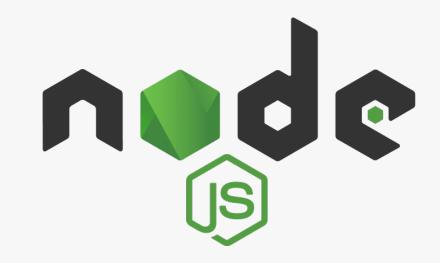




Node.js - введение

Node.js - это просто другой способ выполнять код на вашем компьютере. Это среда выполнения языка JavaScript.

Пакетом в Node із называется один или несколько JavaScript-файлов, представляющих собой какую-то библиотеку или инструмент.





node.is

Node.js - введение

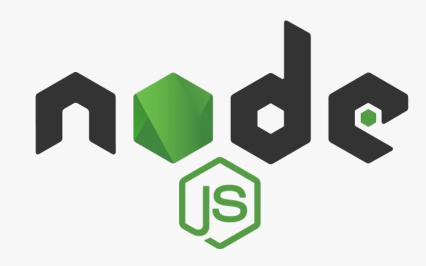
Особенности Node.js:

Скорость

Одной из основных привлекательных особенностей Node.js является скорость. JavaScript-код, выполняемый в среде Node.js, может быть в два раза быстрее, чем код, написанный на компилируемых языках, вроде С или Java, и на порядки быстрее интерпретируемых языков наподобие Python или Ruby.

Простота

Платформа Node.js проста в освоении и использовании. На самом деле, она прямо-таки очень проста, особенно это заметно в сравнении с некоторыми другими серверными платформами.





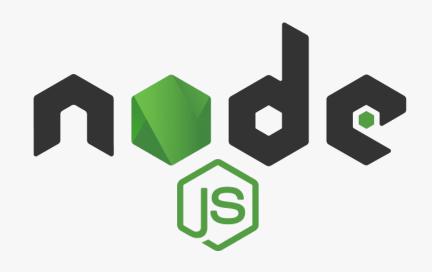
Node.js - введение

Особенности Node.js:

Движок V8

В основе Node.js, помимо других решений, лежит опенсорсный JavaScript-движок V8 от Google, применяемый в браузере Google Chrome и в других браузерах.

Ссылка на страницу загрузки Node.js





Менеджер пакетов прт

npm (аббр. node package manager) - это стандартный менеджер пакетов, автоматически устанавливающийся вместе с Node.js.

Используется для скачивания пакетов из облачного сервера:npm,:либо:для:загрузки пакетов на эти сервера.

Применяя сборщики проектов, прт позволяет добавлять пакеты зависимостей, которые необходимы для организации рабочего процесса. Например пакет расширения SASS, благодаря которому SCSS код может выполняться в нашем проекте.

Ссылка на официальный ресурс





Parcel - это упаковщик для веб-приложений для разработчиков с различным опытом. Он предлагает великолепную быструю работу с использованием многоядерной обработки и не требует сложной настройки.



<u>Ссылка на официальную документацик</u>



Для начала работы необходимо инициализировать наш проект. Для этого, находясь в папке того самого проекта, нужно ввести в терминале команду npm init. Данная команда создаст раскаде json файл, который будет являться отправной точкой всего проекта.

В данный файл будет записываться вся информация, касающаяся установки всех зависимостей, а так же будет содержать скрипты, которые позволят запускать Live Server и компилировать файлы для последующей выгрузки на хостинг.

```
"name": "my-project",
"source": "src/index.html",
"version": "1.0.0",
"description": "project description",
"scripts": {
"author": "",
"license": "ISC"
```



Далее необходимо установить все зависимости, которые позволят функционировать нашему проекту.

Для минимального функционирования будет достаточно установить два пакета зависимостей.

- 1. npm install --seve-dev parcel
 - 2. npm install --seve-dev sass

```
"name": "my-project",
"source": "src/index.html",
"version": "1.0.0",
"description": "",
"scripts": {
  "start": "parcel",
  "build": "parcel build"
"author": "",
"license": "ISC",
"devDependencies": {
  "@parcel/transformer-sass": "^2.0.0",
  "parcel": "^2.0.0",
  "sass": "^1.43.4"
```

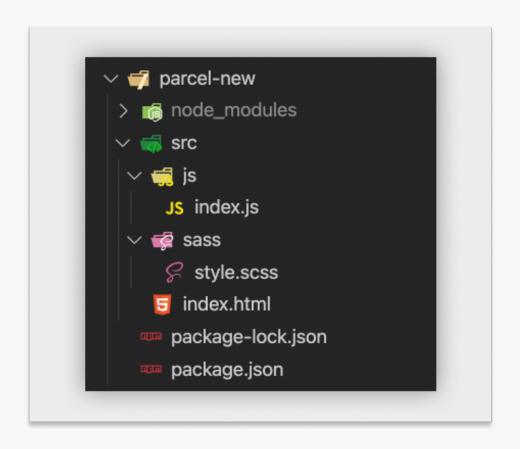


Структура проекта:

состоит из корневой папки src, которая включает в себя папку js, sass, а также папку node_modules, которая добавляется автоматически после установки зависимостей.

При загрузке проекта в репозиторий GitHub - node_modules прописывается в файл gitignore. Таким образом в копировании она не участвует, так как содержит множество мелких файлов.

Клонировав проект на локальную машину, достаточно в нужной нам директории прописать команду npm install. Данная команда создаст локально папку node_modules и подтянет в нее все нужные зависимости из ключевого файла package ison.





Скрипты

Следующим шагом является прописывание скриптов в package.json файл для запуска Live Server и компиляции файлов в готовый проект, после чего мы уже готовы к выгрузке его на хостинг.

```
"name": "my-project",
"source": "src/index.html",
"version": "1.0.0",
"description": "",
▶ Debug
"scripts": {
 "start": "parcel",
 "build": "parcel build"
"author": "",
"license": "ISC",
"devDependencies": {
  "@parcel/transformer-sass": "^2.0.0",
  "parcel": "^2.0.0",
  "sass": "^1.43.4"
```



Подключение файлов

В завершении остается лишь подключить файлы стилей, а так же файл для работы с Java script - index.js.

Запуск Live Server осуществляется командой npm run start

Сборка проекта осуществляется командой npm run build

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content=</pre>
"width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="./sass/style.scss">
  <title>Document</title>
</head>
<body>
  <h1 class="title">Parcel project</h1>
  <script type="module" src="./js/index.js"></script>
</body>
</html>
```



Вложенность

Самым популярным предлагаемым функционалом любого CSS-препроцессора является возможность вкладывать селекторы друг в друга.

На этом этапе вам стоит знать лишь то, что при использовании препроцессоров, можно вкладывать один селектор в другой, а другой в третий,

```
ul {
  width: 50%;
  display: flex;
  li {
    border: 1px solid coral;
      text-decoration: none;
      font-size: 18px;
      font-weight: 400;
```



Использование переменных

Используя перемены, мы можем изменить цветовую палитру всего нашего документа в считанные минуты.

Важно: объявлять переменные необходимо до их использования в документе.

```
$grey: #808080;
$orange: #ffa500;

.block {
  background-color: $orange;
  border: 1px solid $grey;
}
```



Импорт внешних файлов

Также сохраняется возможность импорта внешних файлов. Отдельные файлы стилей, файлы с подключением шрифтов и CDN, миксинов, переменных и т.д. Используем следующий синтаксис.

```
@import 'variables';
@import 'mixins';
@import 'cdn';
```



Миксины

Миксины позволяют создавать целые шаблоны стилей для отдельных элементов, которые невозможно поместить в переменные. В них также можно использовать математические выражения.

```
@mixin card {
   width: 300px / 1140px * 100%;
   border: 1px solid $grey;
   padding: 20px;
   box-shadow: 0 0 25px rgba(0, 0, 0, 0.3);
}
.card {
   @include card();
}
```



Медиазапросы

Благодаря препроцессорам работа с медиа запросами становится максимально компактной и удобный.

```
.block {
 background-color: coral;
 @media (max-width: 968px) {
   background-color: cornflowerblue;
 @media (max-width: 430px) {
   background-color: cyan;
 @media (max-width: 320px) {
   background-color: bisque;
```



Препроцессоры - Less, Stylus

Less

Основан в 2009 году Алексис Сельер (Alexis Sellier) и написан на JavaScript. Имеет все базовые возможности препроцессоров и даже больше, но не имеет условных конструкций и циклов в привычном для нас понимании. Основным плюсом является его простота, практически стандартный для CSS синтаксис и возможность расширения функционала за счёт системы

Stylus

Самый молодой, но в тоже время самый перспективный CSSпрепроцессор. Основан в 2010 году небезызвестной в наших кругах личностью ТЈ Holowaychuk. Написан на JavaScript. Поддерживает уйму вариантов синтаксиса от подобного CSS до упрощённого (отсутствуют:, ;, {} и некоторые скобки).







Ссылки на дополнительные материалы

Node.js

NPM

Parce

Sass

