html academy
интерактивные
онлайн-курсы

Раздел 7:

Продакшен

План

План

– Переменные окружения

План

– Переменные окружения

– Логгинг

План

– Переменные окружения

– Логгинг

– Сборка приложения

План

- Переменные окружения
- Логгинг
- Сборка приложения
- Запуск приложения и отслеживание

План

- Переменные окружения
- Логгинг
- Сборка приложения
- Запуск приложения и отслеживание
- Поиски обнаружение проблем

План

– Переменные окружения

– Логгинг

– Сборка приложения

– Запуск приложения и отслеживание

– Поиски обнаружение проблем

– Профилирование

Переменная среды (*Environment Variable*)

Способ передать/распространить параметры между процессами и/или операционной системой

Для чего нужны переменные среды

en.wikipedia.org/wiki/Environment_variable

Для чего нужны переменные среды

– Универсальный способ передавать настройки между процессами

en.wikipedia.org/wiki/Environment_variable

Для чего нужны переменные среды

– Универсальный способ передавать настройки между процессами

– Переменные окружения наследуются между подпроцессами

en.wikipedia.org/wiki/Environment_variable

Для чего нужны переменные среды

– Универсальный способ передавать настройки между процессами

– Переменные окружения наследуются между подпроцессами

– Стандартный способ переопределения значений по умолчанию

en.wikipedia.org/wiki/Environment_variable

Для чего нужны переменные среды

– Универсальный способ передавать настройки между процессами

– Переменные окружения наследуются между подпроцессами

– Стандартный способ переопределения значений по умолчанию

– Управление режимами приложения

en.wikipedia.org/wiki/Environment_variable

Переменная среды

Переменная среды

– Традиционно записываются и читаются в верхнем регистре (*SNAKE_UPPER_CASE*)

Переменная среды

– Традиционно записываются и читаются в верхнем регистре (*SNAKE_UPPER_CASE*)

– Регистр зависит от ОС:

Переменная среды

- Традиционно записываются и читаются в верхнем регистре (*SNAKE_UPPER_CASE*)
- Регистр зависит от ОС:
  - Windows — регистронезависимый

Переменная среды

- Традиционно записываются и читаются в верхнем регистре (*SNAKE_UPPER_CASE*)
- Регистр зависит от ОС:
  - Windows — регистронезависимый
  - Unix-base — регистрозависимый

Переменная среды

- Традиционно записываются и читаются в верхнем регистре (*SNAKE_UPPER_CASE*)
- Регистр зависит от ОС:
  - Windows — регистронезависимый
  - Unix-base — регистрозависимый
  - Остальные — указано в документации

dotenv

маленькая библиотека, которая загружает переменные среды из специального файла *.env* в объект *process.env*

github.com/motdotla/dotenv

.env

```
# Настройки сервера
SERVER_HOST=localhost
SERVER_PORT=3000
SERVER_LOG_LEVEL=INFO

# Настройки Базы Данных
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=s3cr3t
```

Лог

система учёта дополнительной информации к событиям в программном обеспечении

Зачем нужен лог

Зачем нужен лог

– Обратная связь от приложения

Зачем нужен лог

– Обратная связь от приложения

– Поиск и устранение возникающих проблем

Зачем нужен лог

– Обратная связь от приложения

– Поиск и устранение возникающих проблем

– Уведомление о непредвиденных или неожиданных ситуациях

Что отражает лог

Что отражает лог

– Когда произошло событие

Что отражает лог

– Когда произошло событие

– Какое событие произошло

Что отражает лог

– Когда произошло событие

– Какое событие произошло

– Почему наступило данное событие

Что отражает лог

– Когда произошло событие

– Какое событие произошло

– Почему наступило данное событие

– Какое окружение (контекст) был во время события

Где хранятся записи лога

Где хранятся записи лога

– В файле

Где хранятся записи лога

– В файле

– В памяти

Где хранятся записи лога

– В файле

– В памяти

– В специальном хранилище (базе данных)

Где хранятся записи лога

– В файле

– В памяти

– В специальном хранилище (базе данных)

– Нигде

Логгер

Библиотека, которая предоставляет универсальный интерфейс для создания лог-записей

Логгер

Логгер

– Разделяет записи на уровни:

Логгер

- Разделяет записи на уровни:
  - ERROR — уровень критической ошибки

Логгер

- Разделяет записи на уровни:
  - ERROR — уровень критической ошибки
  - LOG/INFO — информационный уровень

Логгер

- Разделяет записи на уровни:
  - ERROR — уровень критической ошибки
  - LOG/INFO — информационный уровень
  - TRACE/DEBUG — уровень подробностей

Логгер

- Разделяет записи на уровни:
  - ERROR — уровень критической ошибки
  - LOG/INFO — информационный уровень
  - TRACE/DEBUG — уровень подробностей
- Сохраняет время события

Логгер

- Разделяет записи на уровни:
  - ERROR — уровень критической ошибки
  - LOG/INFO — информационный уровень
  - TRACE/DEBUG — уровень подробностей
- Сохраняет время события
- Сохраняет контекст события

console.log

Встроенный логгер в Node.js и в Browser API

Плюсы

Плюсы

– Не требует внешних пакетов

Плюсы

– Не требует внешних пакетов

– Не требует дополнительной настройки

Минусы

Минусы

– Может стать синхронным в определённых случаях (https://nodejs.org/dist/latest-v9.x/docs/api/process.html#process_a_note_on_process_i_o)

Минусы

– Может стать синхронным в определённых случаях (https://nodejs.org/dist/latest-v9.x/docs/api/process.html#process_a_note_on_process_i_o)

– Не конфигурируется

Минусы

– Может стать синхронным в определённых случаях (https://nodejs.org/dist/latest-v9.x/docs/api/process.html#process_a_note_on_process_i_o)

– Не конфигурируется

– Не сохраняет время записи

Минусы

- Может стать синхронным в определённых случаях (https://nodejs.org/dist/latest-v9.x/docs/api/process.html#process_a_note_on_process_i_o)
- Не конфигурируется
- Не сохраняет время записи
- Выводит записи в стандартный вывод/консоль

Минусы

- Может стать синхронным в определённых случаях (https://nodejs.org/dist/latest-v9.x/docs/api/process.html#process_a_note_on_process_i_o)
- Не конфигурируется
- Не сохраняет время записи
- Выводит записи в стандартный вывод/консоль
- Не позволяет перенаправть вывод от разных уровней в разные файлы/потоки

util.debuglog

Встроенная функция, которая позволяет записывать сообщения уровня *DEBUG*

nodejs.org/dist/latest-v9.x/docs/api/util.html#util_util_debuglog_section

util.debuglog

util.debuglog

– Преимущества:

util.debuglog

- Преимущества:
  - По умолчанию не выводит *DEBUG* сообщения

util.debuglog

- Преимущества:
  - По умолчанию не выводит *DEBUG* сообщения
  - Не требует внешних зависимостей

util.debuglog

- Преимущества:
  - По умолчанию не выводит *DEBUG* сообщения
  - Не требует внешних зависимостей
  - Позволяет отлаживать встроенные модули в *node.js*

util.debuglog

- Преимущества:
  - По умолчанию не выводит *DEBUG* сообщения
  - Не требует внешних зависимостей
  - Позволяет отлаживать встроенные модули в *node.js*
  - Настраивается переменной окружения *NODE_DEBUG=<имя модуля>*

util.debuglog

- Преимущества:
  - По умолчанию не выводит *DEBUG* сообщения
  - Не требует внешних зависимостей
  - Позволяет отлаживать встроенные модули в *node.js*
  - Настраивается переменной окружения *NODE_DEBUG=<имя модуля>*
- Недостатки:

util.debuglog

- Преимущества:
  - По умолчанию не выводит *DEBUG* сообщения
  - Не требует внешних зависимостей
  - Позволяет отлаживать встроенные модули в *node.js*
  - Настраивается переменной окружения *NODE_DEBUG=<имя модуля>*
- Недостатки:
  - Все сообщения выводит исключительно в *console.error*

util.debuglog

- Преимущества:
  - По умолчанию не выводит *DEBUG* сообщения
  - Не требует внешних зависимостей
  - Позволяет отлаживать встроенные модули в *node.js*
  - Настраивается переменной окружения *NODE_DEBUG=<имя модуля>*
- Недостатки:
  - Все сообщения выводит исключительно в *console.error*
  - Не настраивается

util.debuglog

- Преимущества:
  - По умолчанию не выводит *DEBUG* сообщения
  - Не требует внешних зависимостей
  - Позволяет отлаживать встроенные модули в *node.js*
  - Настраивается переменной окружения *NODE_DEBUG=<имя модуля>*
- Недостатки:
  - Все сообщения выводит исключительно в *console.error*
  - Не настраивается
  - Выводит сообщения в фиксированном формате:
    FOO-BAR 3257: hi there, it's foo-bar [2333]

winston

сторонний инструмент для работы с логами

github.com/winstonjs/winston

Уровни лог-записи

Уровни лог-записи

– *error* (0) — произошла ошибка, предоставляет подробности к ошибке

Уровни лог-записи

– *error* (0) — произошла ошибка, предоставляет подробности к ошибке

– *warn* (1) — предупреждение, возможно редкая или непредвиденная ситуация, позволяет понять когда происходит то или иное событие, после чего можно перевести в статус ошибки или информативного сообщения

Уровни лог-записи

– *error* (0) — произошла ошибка, предоставляет подробности к ошибке

– *warn* (1) — предупреждение, возможно редкая или непредвиденная ситуация, позволяет понять когда происходит то или иное событие, после чего можно перевести в статус ошибки или информативного сообщения

– *info* (2) — предоставляет нужную пользователю информацию, например, на каком порту доступен сервер

Уровни лог-записи

– *error* (0) — произошла ошибка, предоставляет подробности к ошибке

– *warn* (1) — предупреждение, возможно редкая или непредвиденная ситуация, позволяет понять когда происходит то или иное событие, после чего можно перевести в статус ошибки или информативного сообщения

– *info* (2) — предоставляет нужную пользователю информацию, например, на каком порту доступен сервер

– *verbose* (3) —предоставляет более расширенную информацию пользователю, например, что именно происходит в данный момент и чем занято приложения

Уровни лог-записи

– *error* (0) — произошла ошибка, предоставляет подробности к ошибке

– *warn* (1) — предупреждение, возможно редкая или непредвиденная ситуация, позволяет понять когда происходит то или иное событие, после чего можно перевести в статус ошибки или информативного сообщения

– *info* (2) — предоставляет нужную пользователю информацию, например, на каком порту доступен сервер

– *verbose* (3) —предоставляет более расширенную информацию пользователю, например, что именно происходит в данный момент и чем занято приложения

– *debug* (4) — уровень отладки, содержит отладочные сообщения

Уровни лог-записи

– *error* (0) — произошла ошибка, предоставляет подробности к ошибке

– *warn* (1) — предупреждение, возможно редкая или непредвиденная ситуация, позволяет понять когда происходит то или иное событие, после чего можно перевести в статус ошибки или информативного сообщения

– *info* (2) — предоставляет нужную пользователю информацию, например, на каком порту доступен сервер

– *verbose* (3) —предоставляет более расширенную информацию пользователю, например, что именно происходит в данный момент и чем занято приложения

– *debug* (4) — уровень отладки, содержит отладочные сообщения

– *silly* (5) — совсем безумный уровень подробности

Что лучше не писать в лог-сообщениях

Что лучше не писать в лог-сообщениях

– По возможности не используйте коды ошибок и сокращения

Что лучше не писать в лог-сообщениях

– По возможности не используйте коды ошибок и сокращения

– Не выводите приватную информацию в лог (логины, пароли)

Что лучше не писать в лог-сообщениях

– По возможности не используйте коды ошибок и сокращения

– Не выводите приватную информацию в лог (логины, пароли)

– Не используйте короткие сообщения вида: working, writing

Что лучше не писать в лог-сообщениях

- По возможности не используйте коды ошибок и сокращения
- Не выводите приватную информацию в лог (логины, пароли)
- Не используйте короткие сообщения вида: working, writing
- Не выводите просто объект без каких-либо сообщений и подробностей откуда этот объект и в каком контексте он появился

Что лучше не писать в лог-сообщениях

- По возможности не используйте коды ошибок и сокращения
- Не выводите приватную информацию в лог (логины, пароли)
- Не используйте короткие сообщения вида: working, writing
- Не выводите просто объект без каких-либо сообщений и подробностей откуда этот объект и в каком контексте он появился
- Максимально подробно описывайте, то что произошло и почему

Сборка (build)

Процесс получения информационного продукта из исходного кода для дальнейшего распространения или развертывания на удалённой машине

Деплой

Процесс развертывание собранного пакета (ПО) на серверном оборудовании

Деплой

Деплой

– На локальной машине

Деплой

– На локальной машине

– На удалённой машине:

Деплой

- На локальной машине
- На удалённой машине:
  - вручную

Деплой

- На локальной машине
- На удалённой машине:
  - вручную
  - на платформе/сервисе

Process Manager

Приложение позволяющее разворачивать,
запускать и следить за процессом работы
приложения на удалённом, либо локальном сервере

expressjs.com/en/advanced/pm.html

Cross-Origin Resource Sharing (CORS)

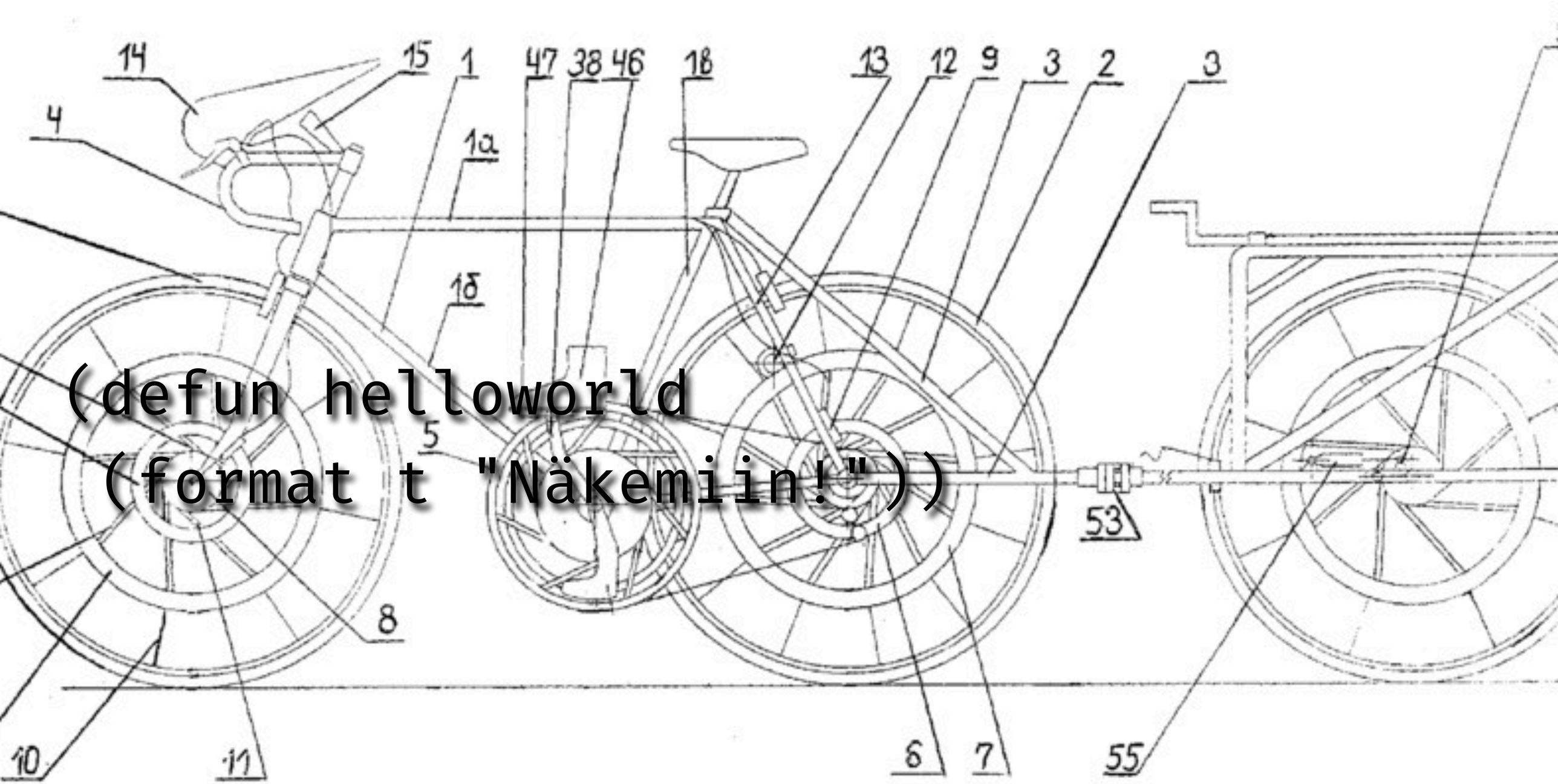
механизм, использующий дополнительные HTTP-заголовки, чтобы дать возможность агенту пользователя получать разрешения на доступ к выбранным ресурсам с сервера на источнике (домене), отличном от того, что сайт использует в данный момент

developer.mozilla.org/ru/docs/Web/HTTP/CORS

Пример *CORS middleware* разрешающий любые подключения

```
app.use((req, res, next) => {
  res.header(`Access-Control-Allow-Origin`, `*`);
  res.header(`Access-Control-Allow-Headers`, `Origin, X-Requested-With, Content-Type, Accept`);
  next();
});
```

Профилирование




```
(defun helloworld
 (format t "Näkemiin!"))
```

Фиг.1