Руководство пользователя

Установка и настройка

Требования к системе

Для использования плеера требуется операционная система поддерживающая работу Node.js. Например: FreeBSD или Debian

Установка Node.js

Для работы ПО в систему необходимо установить Node.js не ниже 16 версии. Для установки можно воспользоваться официальными установщиками https://nodejs.org/en/download, через пакетные менеджеры https://nodejs.org/en/download/package-manager или собрать самостоятельно из дистрибутивов https://github.com/nodejs/node/blob/main/BUILDING.md

Инициализация проекта

В системе необходимо завести папку проекта

```
mkdir course-project && cd course-project
```

Инициализируем проект. Вводим команду `npm init` и указываем базовую информацию

```
npm init
```

Система спросит несколько базовых вопросов. На этом этапе можно оставить значения по умолчанию и вернуться к ним позже

```
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (course-project) course-project
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to .../course-project/package.json:

{
  "name": "course-project",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this OK? (yes) yes
```

Установка плеера

Запустите команду ниже, она создаст файл `.npmrc` в корне проекта и добавит в него информацию, где искать нужные пакеты

```
echo "@htmlacademy:registry=https://npm.pkg.github.com/htmlacademy" > .npmrc
```

> Обратите внимание, что для установки пакета потребуется токен. Для получения токена необходимо обратиться в службу поддержки

Замените `_TOKEN_` на полученный вами токен в команде ниже и запустите ее

```
echo "//npm.pkg.github.com/:_authToken=_TOKEN_" >> .npmrc
```

Установите плеер в качестве npm-зависимости

```
npm i @htmlacademy/content-helper -DE
```

Добавьте скрипты в `package.json` для старта дев-сервера и сборки

```
{
    "scripts": {
        "release": "helper-cli release", // собирает релиз для публикации курса
        "start": "helper-cli watch -e dev" // запускает локальный сервер
    }
}
```

Создайте файл `releaserc.json` для конфигурации проекта

```
echo "{}" >> releaserc.json
```

Запуск плеера

Теперь если запустить `npm start` то поднимется дев-окружение и на локальной машине по адресу http://localhost:3000 можно посмотреть курса. Базовая настройка готова.

Конфигурация проекта

В зависимости от структуры курса плееру можно передать отличные от настроек по умолчанию папки для поиска контента. Для этого в корне проекта есть конфигурационный файл `releaserc.json`

```
{
  "projectsDir": "название папки с проектами",
  "tasksDir": "название папки с учебником"
};
```

Ниже будут описаны все возможные настройки проекта

Перезапуск плеера

В некоторых случаях в процессе работы с проектом в консоли могут упасть ошибки сборки. В этом случае необходимо перезапустить плеер.

Остановите его при помощи `Ctrl + C` и еще раз запустите `npm start` .

Управление курсом

Структура курса задается в виде набора файлов и папок в папке с проектом.

Формат конфигурации

Плеер поддерживает два формата конфига `JSON` и `Yaml`. Система ищет нужную папку и проверяет наличие файла `config.json`. Если такой файл существует, то конфигурация считывается с него. Если его нет, то система проверяет наличие файла `config.yml` и пробует использовать его. Сама структура конфигурации полностью одинакова.

Пример конфига проектов

В JSON

```
{
  "title": "Барбершоп",
  "description": "Сайт мужской парикмахерской",
  "dificulty": "Легкий проект",
  "learning": "1",
  "layout": {
    "PSD": "https://htmlacademy.ru",
    "Sketch": "https://htmlacademy.ru"
  },
  "overview": "",
  "fonts": "«PT Sans Narrow»"
}
```

Аналогичная конфигурация в Yaml

```
title: Барбершоп
description: Сайт мужской парикмахерской
dificulty: Легкий проект
learning: 1
layout:
  PSD: https://htmlacademy.ru
  Sketch: https://htmlacademy.ru
overview:
fonts: «PT Sans Narrow»
```

Далее в описании приводятся примеры только для одного варианта конфигурации, но второй вариант тоже будет рабочим. Главное требование — соблюдать необходимую структуру конфигурации

Управление проектами курса

Проекты используются студентами для работы с ними на курсе. По умолчанию собираются из папки `projects` проекта. Папку можно переопределить в файле `releaserc.json` - ключ `projectsDir`

Под каждый проект на курсе внутри папки projects необходимо завести отдельную папку. Название папки будет использоваться как сервисное имя проекта в системе.

Конфигурация проекта располагается в файле `config.json` внутри папки проекта

Пример конфигурации

```
{
  "title": "Барбершоп",
  "description": "Сайт мужской парикмахерской",
  "dificulty": "Легкий проект",
  "learning": "1",
  "layout": {
    "PSD": "https://htmlacademy.ru",
    "Sketch": "https://htmlacademy.ru"
  },
  "overview": "",
  "fonts": "«PT Sans Narrow»"
}
```

Описание настроек проекта:

- `title` (обязательный) - название проекта
- `description` (необязательный) - краткое описание проекта
- `learning` (необязательный, по умолчанию false) - флаг того, что этот проект является учебным. При выборе основного проекта студенту будет автоматически создаваться этот учебный проект, чтобы можно было выполнять задания по нему.
- `readonly` (необязательный, по умолчанию false) - флаг того, что проект только для ознакомления и с ним не предназначается работать в течение курса. В этом состоянии страница проекта доступна для просмотра, но выбрать проект и создать репозиторий по нему нельзя
- `dificulty` (необязательный) - описание сложности проекта
- `layout` (необязательный) - набор макетов по проекту, где ключ — тип макета, значение — ссылка на скачивание макетов
- `overview` (необязательный) - ссылка на видеообзор проекта
- `fonts` (необязательный) - дополнительная информация о шрифтах, используемых в проекте
- `the_simplest` (необязательный) - отмечает простой проект для стандартных студентов группового формата

В описании конфигурации есть дополнительная возможность переопределять значения по мере открытия разделов. Пример:

```
{
  "title": "Барбершоп",
  "description": "Сайт мужской парикмахерской",
  "dificulty": "Легкий проект",
  "learning": true,
  "readonly": true,
  "layout": {
    "PSD": "https://htmlacademy.ru",
    "Sketch": "https://htmlacademy.ru"
  },
  "overview": "",
  "fonts": "«PT Sans Narrow»",
  "module": {
    "4": {
      "dificulty": "Уже не легкий проект",
      "layout": {
        "PSD": null,
        "Sketch": null,
        "PSD_ALL": "https://htmlacademy.ru",
        "Sketch_ALL": "https://htmlacademy.ru"
      }
    }
  }
}
```

В ключе `module` можно определить какие значения обновятся при открытии нужного модуля. В данном примере:

- `4` - номер модуля, после открытия которого применятся описанные обновления
- указав значение `null` очистит ранее определенные значения и скроет их из интерфейса
- для макетов добавив к типу постфикс `_ALL` в системе автоматически добавит подпись `все макеты`

В этом примере после 4 модуля скроются мобильные версии макетов и откроются полные версии всех макетов

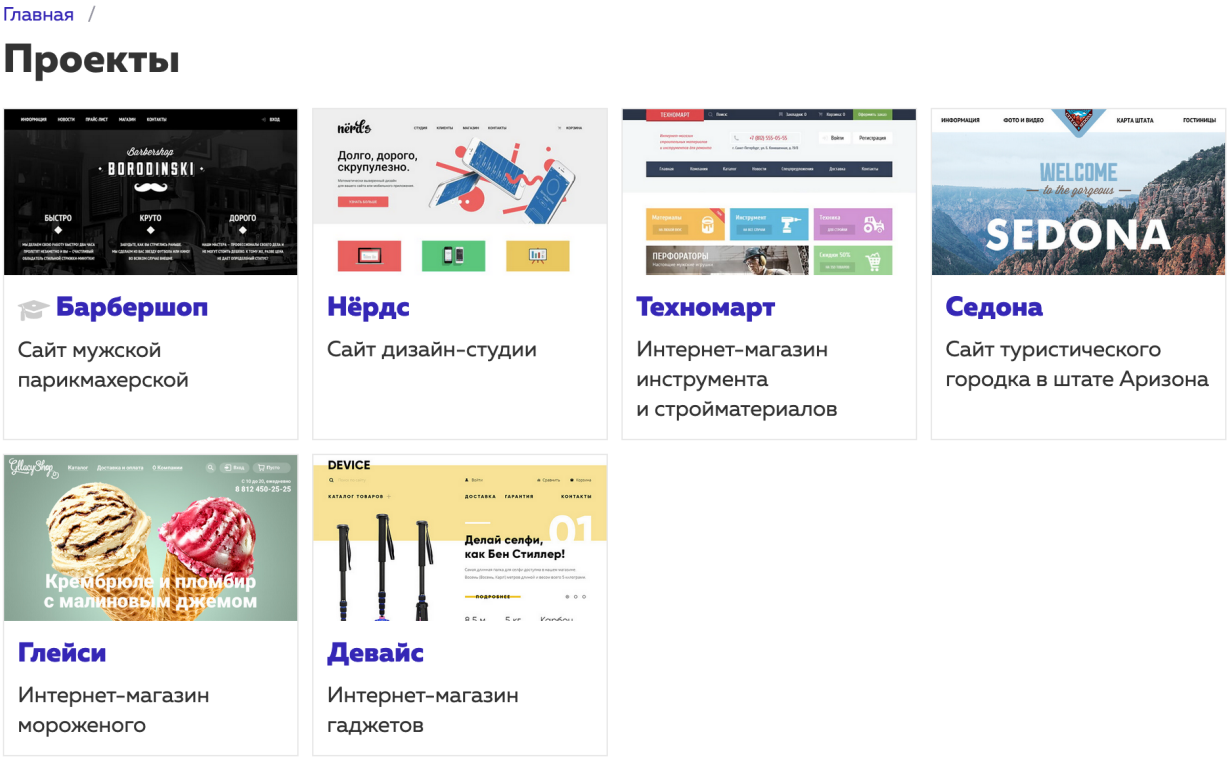
В режиме разработки переопределение не работает, это переопределение используется только в режиме интеграции.

Дополнительные файлы, которые считываются из папки проекта:

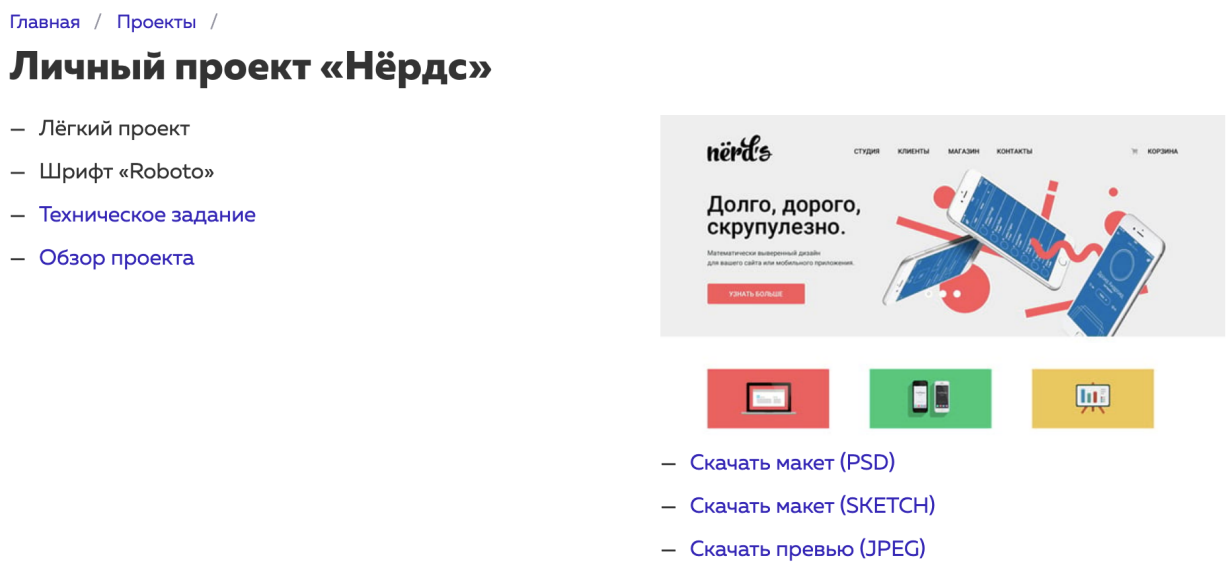
- `specification.md` - техническое задание на проект в формате Markdown

- `image.<jpeg|png>` - изображение проекта
- `preview.<jpeg|png>` - маленькое превью проекта

Пример страницы с проектами



Пример страницы отдельного проекта



Управление критериями курса

Критерии на курсе используются для проведения процесса защиты. Выполненный студентом проект должен соответствовать этим критериям

По умолчанию критерии собираются из папки `criteria`. Папку можно переопределить в файле `releaserc.json` - ключ `criteriaDir`

Есть два варианта заведения критериев.

Единый файл со списком критериев

В этом формате внутри папки должне лежать файл `readme.md`, хранящий в себе весь список критериев. Структура файла имеет следующий вид

```
Вводная часть критериев

# Критерии
```

````
## Базовые критерии

### Первая категория

#### Б1. Первый критерий

Описание критерия

---
Дополнительная инструкция для наставника

#### Б2. Второй критерий

```html
<p>
  <a href=""></a>
</p>
```

### Вторая категория

#### Б3. Третий критерий

## Дополнительные критерии

### Первая категория

#### Д1. Первый дополнительный критерий

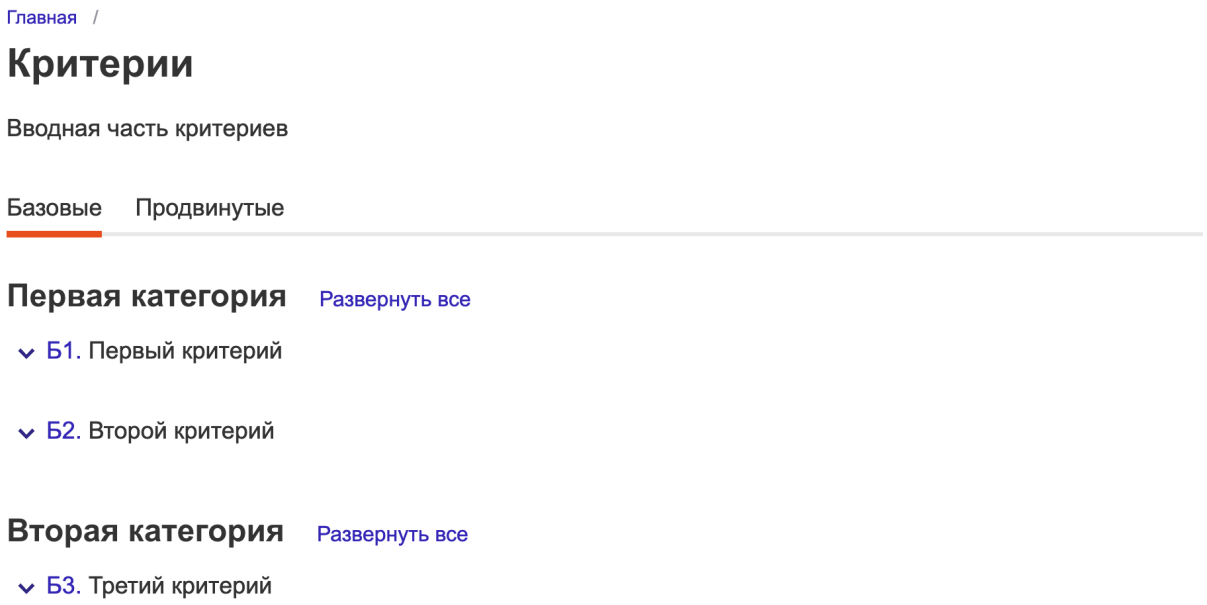
Описание критерия
````

В описании структуры действуют следующие правила:

- Все, что находится до заголовка первого уровня `# Критерии` является вводным текстом к критериям. Все, что находится после этого заголовка, уже считается блоком с списком критериев
- Заголовки второго уровня разделяют весь список на два типа критериев: базовые и дополнительные
- Заголовки третьего уровня определяют категории
- Заголовки четвертого уровня обозначают сам критерий
- Все, что идет после заголовка четвертого уровня, является описанием критерия

- Внутри описания можно выделить вспомогательный блок с инструкцией для наставника, он отделяется тройным дефисом `---`
- В описании критериев можно использовать любые конструкции маркдауна.

Пример отображения критериев заведенных по такой структуре



Нумерация критериев.

По умолчанию система считает, что в названии критерия прописан его порядковый номер. Если номер не прописан, то можно настроить так, чтобы система проставляла их автоматически. Для этого необходимо в настройках сборки `releaserc.json` передать ключ `isCriteriaOrdered` со значением `false`

Набор папок и файлов под отдельные критерии

Альтернативный вариант заведения критериев - это формирование набора папок, описывающих их структуру. Чтобы включить этот вариант, необходимо внутри папки с критериями создать файл с конфигурацией `config.json` и добавить в нем ключ `structure` со значением `folder`

```
{
  "structure": "folder"
}
```

В этом случае необходимо придерживаться следующей стурктуры

- Файл `readme.md` содержит вводную информацию о критериях
- Под каждую категорию критериев необходимо завести отдельную папку. Порядок категорий определяется префиксом в названии папки. Например
  - 01-TEST - первая категория TEST
  - 02-PROJ - вторая категория PROJ
- Внутри папки с категорией в файле `readme.md` можно добавить название и описание категории. Название будет считываться из заголовка файла, а все остальное будет описанием категории
- Внутри категории под каждый критерий необходимо завести отдельную папку. Порядок критериев определяется префиксом в названии папки. Например
  - 01-codeguide - первый критерий в категории TEST
  - 02-naming - второй критерий в категории TEST
- На основании структуры папок критерии получают уникальные сервисные имена состоящие из сервисного имени категории и порядкового номера критерия: `TEST-01`, `TEST-02`
- Внутри папки с критерием система ждет файл `readme.md`. Заголовок в этом файле будет названием критерия, а все, что находится под заголовком - его описанием
- Дополнительно для критерия можно добавить файл с конфигурацией `config.json`:

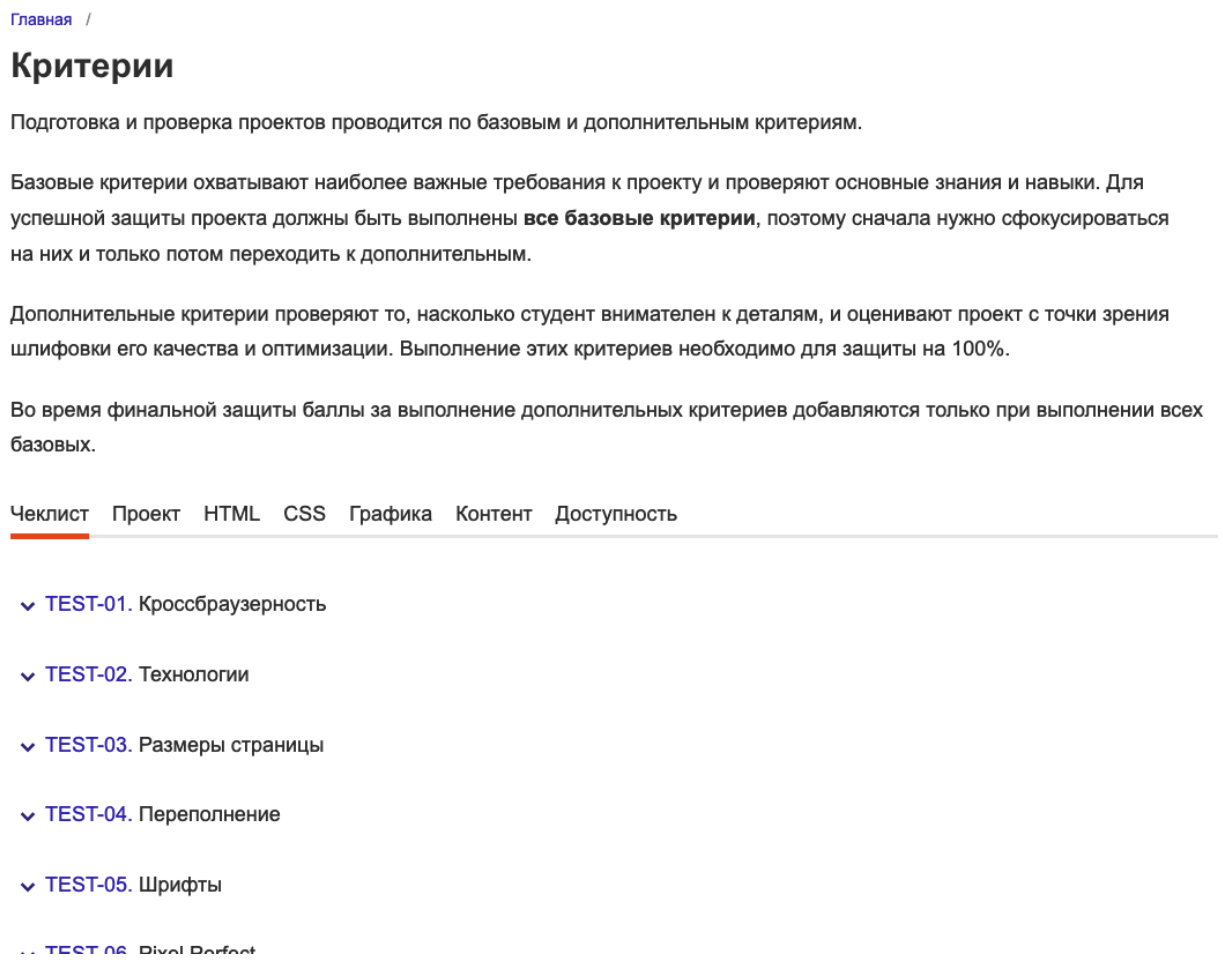
  ```
  {
    "optional": true,
    "check": {
      "name": "project/naming"
    }
  }
  ```

Где

- `optional` - флаг того, что критерий является необязательным
- `check` - настройки для автоматической проверки критерия сервисом автопроверок

Всю графику, которая используется в описании критериев можно положить в отдельную папку `assets` внутри папки с критериями. Подробнее о том, как использовать графику, будет описан ниже в разделе «Использование графики»

Пример критериев, собранных по такой структуре



Управление учебником курса

По умолчанию собирается из папки `tutorial`. Папку можно переопределить в файле `releaserc.json`. Ключ `bookDir`.

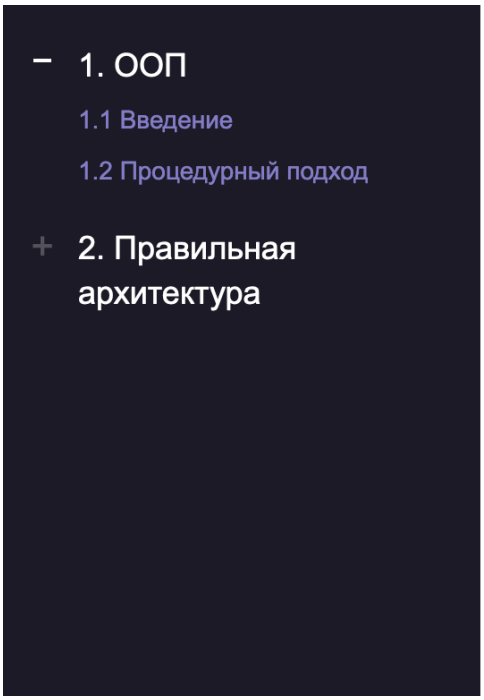
Главы учебника определяются папками и их вложенностью.

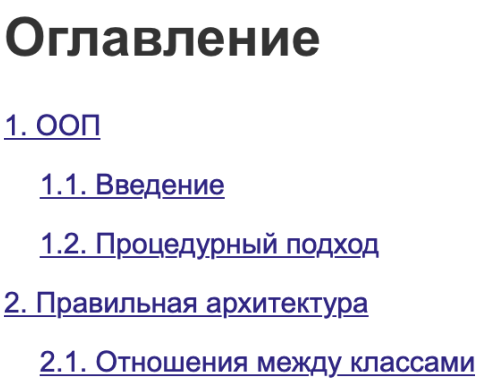
Пример структуры папок:

```
01-oop/
   01-intro/
   02-procedures/
02-architecture/
   01-class-relations/
assets/
```

Приведённая на примере структура определит учебник из двух глав. У первой главы будет 2 подглавы, у второй - одна подглава.

Пример отображения этого учебника





Номер в префиксе названия папки определяет номер главы или подглавы и также отвечает за порядок глав в учебнике.

Внутри главы система ищет следующие файлы:

- `config.json` - файл с конфигурацией главы.

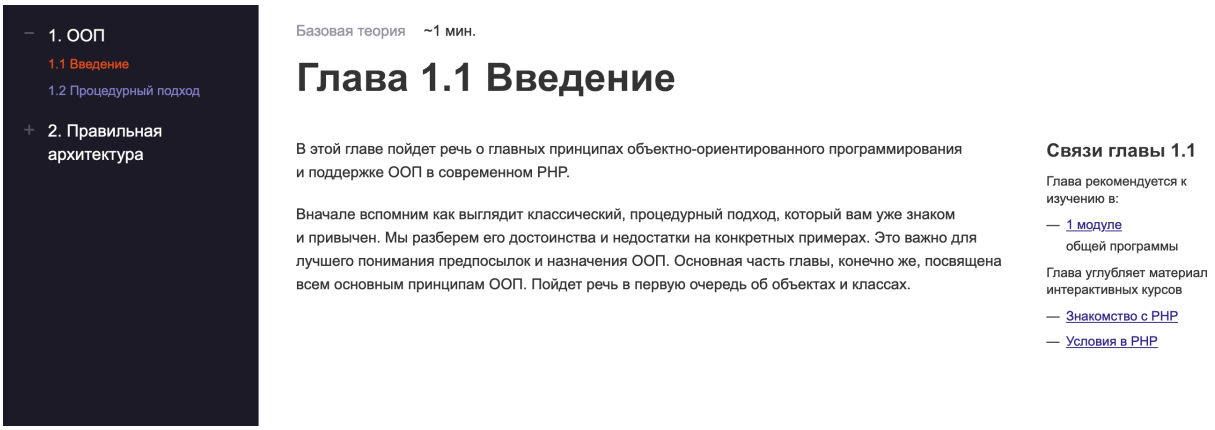
  Пример

```
{
  "title": "Введение",
  "type": "Базовая теория",
  "criteria": ["b1", "d2"]
}
```

Возможные ключи конфигурации:

- `title` - название главы
- `type` - вспомогательный тип для вывода на странице глав
- `criteria` - список критериев, которые затрагивают материалы этой главы. Необходимо использовать сервисное имя критерия
- `hidden` - флаг, скрывающий главу для студентов.

- `readme.md` - основной файл с контентом главы
- `recommend.md` - блок вспомогательных рекомендаций и связей главы



В папке `assets` хранятся все вспомогательные картинки, скрипты, изображения для встраивания в учебнике.

Время чтения страницы считается автоматически из расчета 120 слов в минуту.

Альтернативный вариант создания учебника

Альтернативный вариант позволяет задать конфигурацию для всего учебника. Файл `config.json` должен лежать в корне папки учебника.

Он представляет собой массив объектов, описывающих главы учебника всех уровней. В новом формате упразднены префиксы в названиях папок — теперь последовательность глав и, соответственно, номер каждой определяется порядком элементов массива в конфигурации.

Ключи конфигурации:

- `path` — путь к папке с файлами главы (полные путь от корня учебника).
- `pages` — внутренние главы, если они есть. В качестве значения должен быть передан массив с путями к папкам глав. Предусмотрено два формата — простой массив со строковыми значениями или массив с объектами с такой же структурой, как для глав верхнего уровня (ключи `path`, `pages` и тд).
- `title` — альтернативный заголовок для главы. Может быть использован для сокращённого варианта заголовка, который будет отображаться в навигации. При этом в самом тексте главы заголовок будет по-прежнему подтягиваться из конфигурации главы, если он был указан. Например, если верхнеуровневая глава просто собирает страницы вместе, можно не создавать внутри нее отдельный файл с конфигурацией, а прописать ее параметры в общей конфигурации учебника

Пример:

```
[
  {
    "path": "mailing-features",
    "title": "Глава ознакомительная",
    "pages": [
      "mailing-features/mailing",
      "mailing-features/markup-start",
      "mailing-features/markup-content"
    ]
  },
  {
    "path": "mailing-style",
    "pages": [
      {
        "path": "mailing-style/style",
        "title": "Альтернативный заголовок"
      }
    ]
```

```
  }
]
```

Если файла `config.json` в корне учебника нет, то учебник собирается по старой схеме.

Управление модулями курса

По умолчанию система собирает модули из папки `modules`. Папку можно переопределить в файле `releaserc.json`. Ключ `modulesDir`

Каждый модуль лежит в отдельной папке. Имя папки должно начинаться с числового префикса. Префиксы должны идти по порядку. Этот префикс выполняет функцию идентификации модуля и является его порядковым номером. Остальная часть названия не принципиальна, но может использоваться для дополнительного обозначения, чему посвящен этот модуль

Пример структуры модулей

```
modules/
   01-intro/
   02-oop/
```

Внутри папки модуля система ищет следующие файлы и папки

- `config.json` - файл с конфигурацией модуля.

  Пример

  ```
  {
    "title": "Привет, мир!",
    "description": "Знакомимся с рабочим процессом на интенсиве и базовыми понятиями PHP."
  }
  ```

  Возможные ключи:

  - `title` - название
  - `description` - Краткое описание модуля

    - `opens_in` (необязательный) - указывает через сколько дней от старта курса должен открыться модуль. Система автоматически просчитает дату открытия

- `program.json` - JSON файл с описанием последовательной программы модуля. Структура описана ниже
- `intro.md` - файл с вводной информацией о том, как лучше проходить этот модуль

Управление программой модуля

Внутри папки с модулем система смотрит на файл `program.json`, где лежит вся структура программы модуля и представляет собой набор блоков

Пример программы

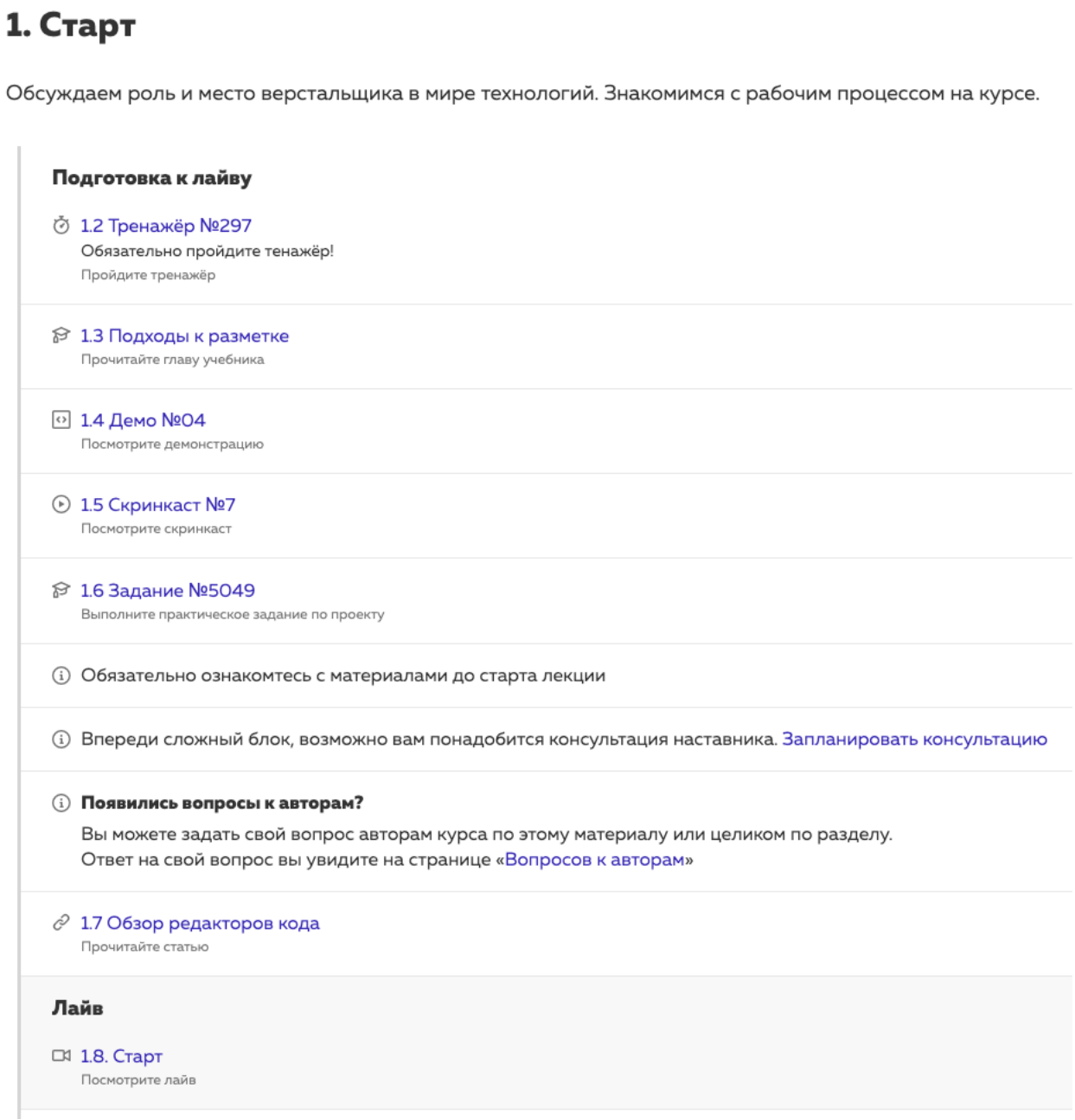
```
[
  {
    "title": "Подготовка к лайву",
    "items": [...]
  },
  {
    "type": "live",
    "title": "Лайв"
  },
  {
    "title": "Домашние задания",
    "items": [...]
  },
  {
    "type": "optional",
    "title": "Дополнительные материалы",
    "items": [...]
  }
]
```

Настройки блока:

- `type`
    - По умолчанию отсутствует
    - `live` — для лекций

    - `optional` — блок с необязательными заданиями по модулю. Скрывает вложенные элементы группы на главной странице
- `title` — название блока
- `items` — список материалов внутри блока

В результате получается примерно такая программа.



Создание элементов блока программы

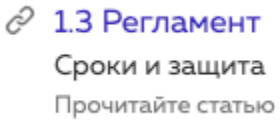
Внутри ключа `items` блока заводятся отдельные материалы программы

```
"items": [
  {
    "type": "course",
    "id": 297,
    "description": "Обязательно пройдите тенажёр!"
  },
  {
    "type": "tutorial",
    "id": "2.1"
  },
  {
    "type": "demo",
    "path": "01-intro/demo-1",
    "id": "04"
  },
  {
    "type": "screencast",
    "title": "Инструменты разработчика. Chrome DevTools",
    "link": "https://player.vimeo.com/video/234321295"
  },
  {
    "type": "task",
    "path": "01-intro/task-1",
    "id": 5049
  },
  {
    "type": "info",
    "title": "Обязательно ознакомтесь с материалами до старта лекции"
  },
  {
    "type": "consultation",
    "title": "Впереди сложный блок, возможно вам понадобится консультация наставника."
  },
  {
    "type": "question"
  }
]
```

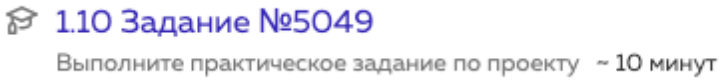
Общие параметры

Могут использоваться у любого из элементов

- `description` — (необязательный) - подпись под элементом с дополнительным указанием



- `duration` — (необязательный) - указание, сколько времени понадобится для прохождения пункта



Типы элементов программы

Тип «Тренажёр»

Прохождение интерактивного тренажера внутри интерфейса курса

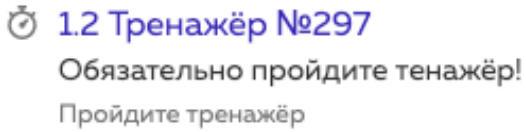
Обязательные параметры:

- `type` со значением `course`
- `id` тренажера на сайте https://htmlacademy.ru

Название тренажёра подтягивается автоматически

```
{
  "type": "course",
  "id": 297,
  "description": "Обязательно пройдите тенажёр!"
}
```

Пример отображения



Тип «Испытания из тренажёра»

Прохождение набора отдельных заданий или испытаний тренажера внутри интерфейса курса

Обязательные параметры:

- `type` со значением `challenges`
- `tasks` id заданий из тренажёров с сайта https://htmlacademy.ru
- `title` - название списка с заданиями

```
{
  "type": "challenges",
  "title": "Сборник испытаний по CSS",
  "tasks": [6205, 6200, 6301],
  "description": "В этом сборнике 5 испытаний. Выполните их для закрепления теории"
}
```

Тип «Глава учебника»

Вставляет в программу модуля статью из учебника

Обязательные параметры:

- `type` со значением `tutorial`
- `id` с номером главы. Параметр может быть передан как число или строка. Но в случае значений вида `1.10`, `2.20` и тд ноль у числа будет обрезаться, поэтому в таких случаях значение нужно передавать как строку — `"id": "1.10"`
- Если учебник настроен на работу с альтернативной структурой, то в программе можно ссылаться на путь к главе используя ключ `path`. Например `"path": "mailing-features/mailing"`

Название главы подтягивается автоматически.

Как создавать главы учебника описано в разделе «Управление учебником курса»

```
{
  "type": "tutorial",
  "id": "1.9"
},
```

Пример отображения

Прочитайте главу учебника

1.9 Что такое email-рассылка

Тип «Демонстрация»

Интерактивная пошаговая демострация примеров кода

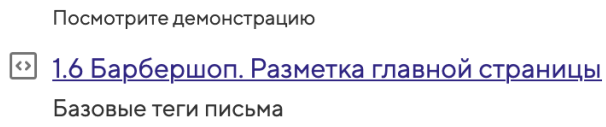
Обязательные параметры:

- `type` со значением `demo`
- `path` - путь к папке с демонстрацией внутри проекта. По умолчанию ищет в папке `modules`

Как создавать демонстрации описано в разделе «Управление демонстрациями»

```
{
  "type": "demo",
  "path": "01-intro/demo-1",
  "description": "Базовые теги письма",
  "id": 1234
}
```

Пример отображения



Тип «Лайв»

Встраивание в программу лайвов и ретроспектив

Обязательные параметры:

- `type` со значением `live`
- `title` - название лайва

Дополнительные параметры

- `isRetro` со значением `true` - если лайв является ретроспективой

Тип «Скринкаст»

Видео с разбором отдельных тем на курсе

Обязательные параметры:

- `type` со значением `screencast`
- `title` - название скринкаста
- `link` - ссылка на скринкаст на YouTube, Vimeo или Кинескоп
- `path` - путь к текстовому описанию скринкаста

```
{
  "type": "screencast",
  "title": "Инструменты разработчика. Chrome DevTools",
  "link": "https://player.vimeo.com/video/234321295"
}
```

Так же поддерживается сокращенный формат записи

```
{
  "screencast": "https://player.vimeo.com/video/234321295",
  "title": "Инструменты разработчика. Chrome DevTools"
}
```

Дополнительные параметры

- `navigation` - параметры навигации по видео. Массив временных меток (`time` в формате `чч:мм:сс`) и заголовков `title`

```
{
  "screencast": "https://player.vimeo.com/video/234321295",
  "title": "Инструменты разработчика. Chrome DevTools",
  "navigation": [
    {
       "time": "00:01:29",
       "title": "Организационные вопросы"
    },
    {
       "time": "00:03:11",
       "title": "Проекты"
    }
  ]
}
```

Пример отображения

Посмотрите скринкаст

1.5 Инструменты разработчика. Chrome DevTools

Тип «Статья»

Отдельная текстовая статья

Обязательные параметры:

- `type` со значением `article`
- `path` - путь к папке со статьей внутри проекта. По умолчанию ищет в папке `modules`

Название статьи подтягивается автоматически. Как создавать статьи описано в разделе «Управление статьями»

Пример отображения

Прочитайте статью

1.4 Email письма

Тип «Задание»

Практическое задание для выполнения студентом

Обязательные параметры:

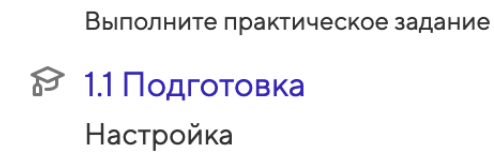
- `type` со значением `task`
- `path` - путь к папке с заданием внутри проекта. По умолчанию ищет в папке `modules`

Название задания подтягивается автоматически. Как создавать задания описано в разделе «Управление практическими заданиями»

```
{
  "type": "task",
```

```
  "path": "01-module/task-01",
  "description": "Настройка"
}
```

Пример отображения



Тип «Ссылка»

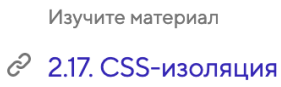
Ссылка на внешний материал

Обязательные параметры:

- `title` с названием внешнего ресурса
- `link` - ссылка на внешний ресурс

```
{
  "title": "CSS-изоляция",
  "link": "https://css-live.ru/articles/css-izolyaciya.html"
}
```

Пример отображения



Тип «Опрос»

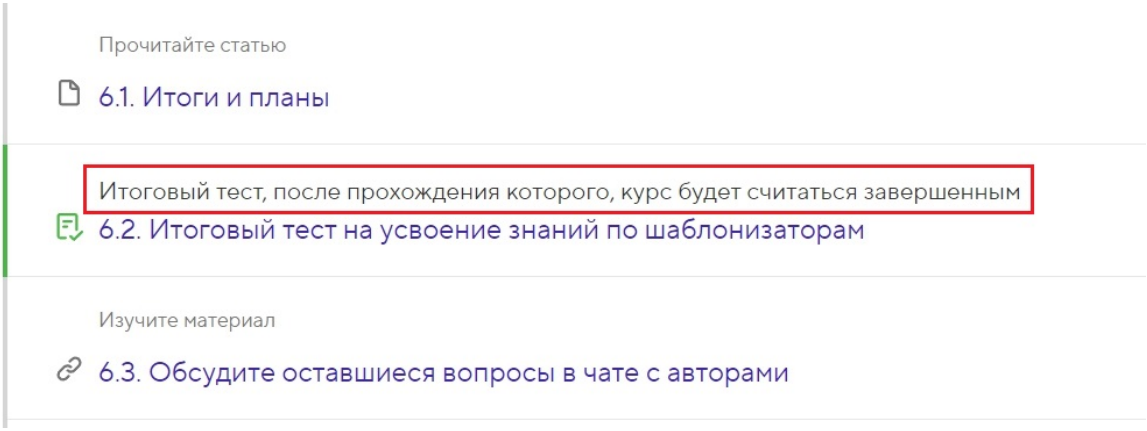
Тест или опрос по пройденным материалам

Обязательные параметры:

- `type` со значением `quiz`
- `path` - путь к папке с опросом внутри проекта.

Необязательные параметры:

- `info` - добавляет название опроса
- `isFinal` - отметка о том что тест является финальным (значение `true`), после прохождения которого (независимо от результата) курс отмечается завершенным.
  В интерфейсе курса (только у админов) отображается с текстовой подсказкой:



Название опроса подтягивается автоматически. Как создавать опросы описано в разделе «Управление опросами и тестами»

```
{
  "type": "quiz",
  "path": "01-intro/08-quiz",
  "info": "Базовые теги",
  "isFinal": true
}
```

Вспомогательные элементы программы

Для более понятного восприятия программы можно добавлять вспомогательные элементы в программу

Информационный блок

Выводит подсказку

Обязательные параметры:

- `type` со значением `info`

- `title` - текст подсказки

```
{
  "type": "info",
  "title": "Обязательно ознакомтесь с материалами до старта лекции"
}
```

Пример отображения

Консультация

Обязательные параметры:

- `type` со значением `consultation`
- `title` - Подпись на подсказке, ссылка на запись формируется автоматически

Подсказка о том, что возможно понадобится консультация наставника

```
{
  "type": "consultation",
  "title": "Впереди сложный блок, возможно вам понадобится консультация наставника."
}
```

Вопрос автору

Блок с возможностью задать вопрос авторам

Обязательные параметры:

- `type` со значением `question`

```
{
  "type": "question"
```

```
}
```

Появились вопросы к авторам?

Вы можете задать свой вопрос авторам курса по этому материалу или целиком по разделу. Ответ на свой вопрос вы увидите на странице «Вопросов к авторам»

Кнопка

Интерактивная кнопка

Обязательные параметры:

`type` - со значением `button`

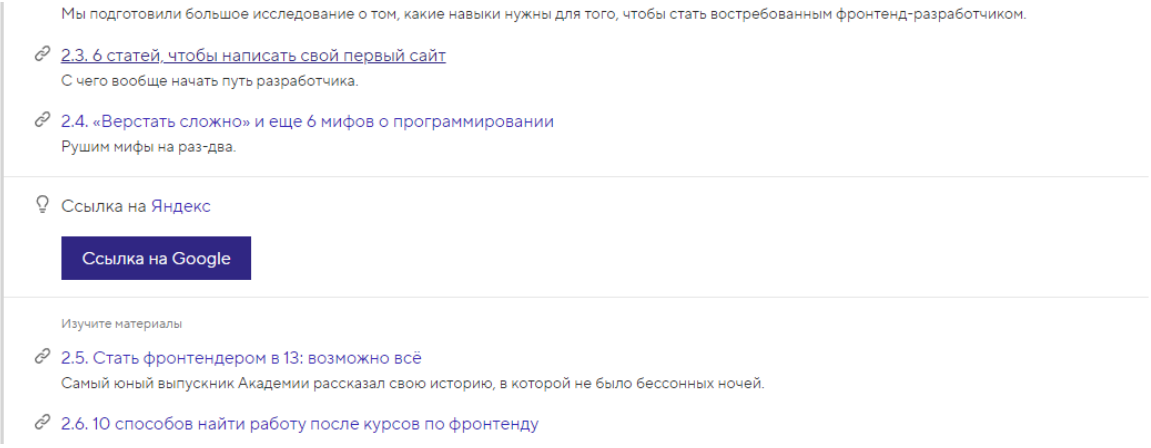
`title` - текст на кнопке

`link` - ссылка куда ведёт кнопка по нажатию

Необязательные параметры:

`description` - можно не указывать, в таком случае будет отображаться только лишь кнопка. Внутри можно положить html теги, как показано в примере.
Ссылка на кнопке всегда открывается в новой вкладке

```
{
  type: "button",
  title: "Ссылка на Google"
  link: "https://google.com"
  description: "Ссылка на <a href='https://yandex.ru' target='_blank'>Яндекс</a>"
}
```



Управление практическими заданиями

Задания считываются из папки `modules`. Папку можно переопределить в файле `releaserc.json`. Ключ `tasksDir`

Каждое задание находится в отдельной папке. Само название папки ни на что не влияет, оно используется только для адресации в программе и более легкого ориентирования по заданиям

Пример структуры:

```
modules/           // папка модулей
  02-intro/        // модуль
    task-intro     // задание
      assets/      // папка с ассетами
        img.png
      readme.md    // контент задания
      config.json  // конфиг для задания
    task-practice  // задание
      readme.md    // контент задания
      config.json  // конфиг для задания
```

В данном примере внутри модуля 2 добавлено 2 задания. Подключение заданий к модулю и настройка позиции задания в программе описано в разделе «Управление программой модуля»

Конфигурация задания

Конфигурация задания хранится в файле `config.json`

Пример конфигурации

```
{
  "title": "Название задания"
  "type": "pullrequest",
  "branch": "module1-task2",
  "learning": false,
  "additional": false,
  "dependsOn": "01-markup-learning",
  "materials": [
      {
        "title": "Скачать Node.js 14.17.5 для Windows",
        "link": "https://nodejs.org/dist/v14.17.5/node-v14.17.5-x64.msi"
      },
      {
        "title": "Скачать Python 3.9.6 для Windows",
```

```
        "link": "https://www.python.org/ftp/python/3.9.6/python-3.9.6-amd64.exe"
      }
  ]
}
```

Доступные настройки

Обязательные параметры:

- `type` - тип задания. Возможные значения:
  - `pullrequest` - классическое задание с созданием пулреквеста
  - `choose` - задание на выбор наставника и проекта
  - `training` - тренировочное задание по проекту без проверки наставником
  - `precommit` - задание на получение обновлений в репозиторий студента
  - `autocheck` - задание с автоматической проверкой
  - `custom` - свободное задание не привязанное к проекту

Необязательные параметры:

- `branch` - название ветки задания (только для задания с типом `pullrequest`)
- `learning` - флаг, что задание относится к учебному проекту
- `additional` - флаг, что задание необязательное
- `dependsOn` - указание зависимости, после какого задания студенту открывается это задание. Указывается в формате `название папки модуля/название папки задания`. аналогично адресации к заданию из программы модуля. Пример: `01-intro/01-markup-learning`
- `title` - название задания. Может быть передано через файл с описанием
- `criteria` - массив с идентификаторами критериев. Например, `["HTML-01", "HTML-01", "CSS-01"]`. Перечень критериев, которые должны быть выполнены в этом задании
- `materials` - массив ссылок для отображения вспомогательных материалов (необязательный ключ). В каждой ссылке необходимо передать `title` - название ссылки и `link` - адрес ссылки
- `checks` - массив автопроверок, которые должны запускаться на коде студента

Описание задания

- Описаниеи задания считывается из файла `readme.md`
- Название задания выделяется заголовком 1 уровня (может указываться в файле `config.json`)
- Пояснения для наставников отделяются символом `---` . Если пояснений для наставника к заданию нет, то отбивать этим символом ничего не надо

Пример:

```
# Название задания

Текст задания для студента

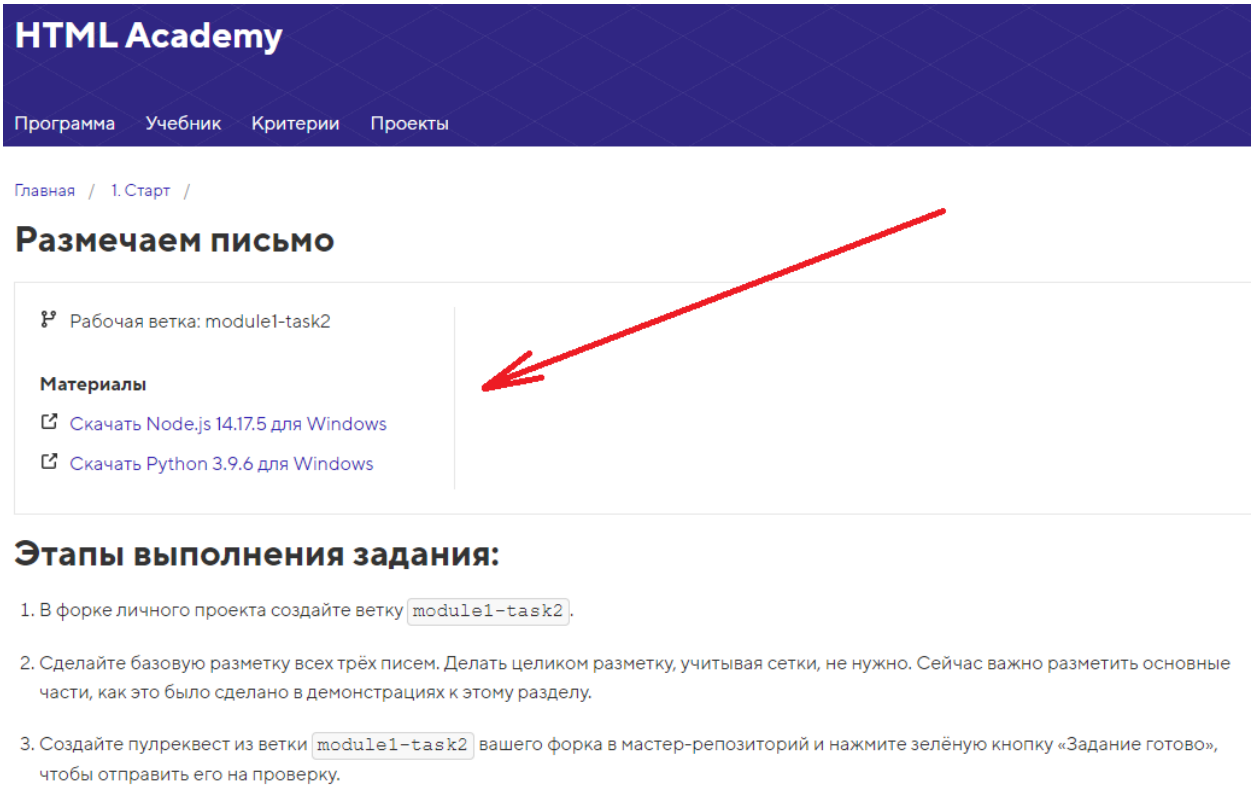
![Пример картинки](assets/img.png)

---

Комментарий по проверке для наставника
```

Внутри папки с заданием может лежать папка `assets` со вспомогательной графикой и любыми другими материалами, которые используются в задании. Пример обращения к таким файлам из текста есть в примере выше

Пример отображения задании



Управление статьями

Статьи считываются из папки `modules` . Папку можно переопределить в файле `releaserc.json`. Ключ `articlesDir`

Каждая статья находится в отдельной папке. Само название папки ни на что не влияет, оно используется только для адресации в программе и более легкого ориентирования по статьям

Пример структуры:

```
modules/           // папка модулей
  02-intro/        // модуль
    materials-1    // статья
      assets/      // папка с ассетами
  img.png
      readme.md    // контент статьи
    materials-2    // статья
      readme.md    // контент статьи
```

В данном примере внутри модуля 2 добавлено 2 статьи. Подключение статей к модулю и настройка позиции статьи в программе описано в разделе «Управление программой модуля»

- Контент статьи находится в файле `readme.md`
- Название статьи выделяется заголовком 1 уровня

Пример:

```
# Название статьи

Текст статьи

![Пример картинки](assets/img.png)
```

Внутри папки со статьей может лежать папка `assets` со вспомогательной графикой и любыми другими материалами, которые используются в статье. Пример обращения к таким файлам из текста есть в примере выше

Управление демонстрациями

Демонстрации позволяют пошагово показывать студенту формирование кода, сопровождая его описанием и наглядным отображением результата

По умолчанию считываются из папки `modules`. Папку можно переопределить в файле `releaserc.json`. Ключ `demosDir`

Каждая демонстрация собирается в отдельной папке. Само название папки ни на что не влияет, оно используется только для адресации в программе и более легкого ориентирования по демонстрациям

Конфигурация демонстрации

Основной конфиг

Основная конфигурация демонстрации хранится в файле `config.json`

Пример

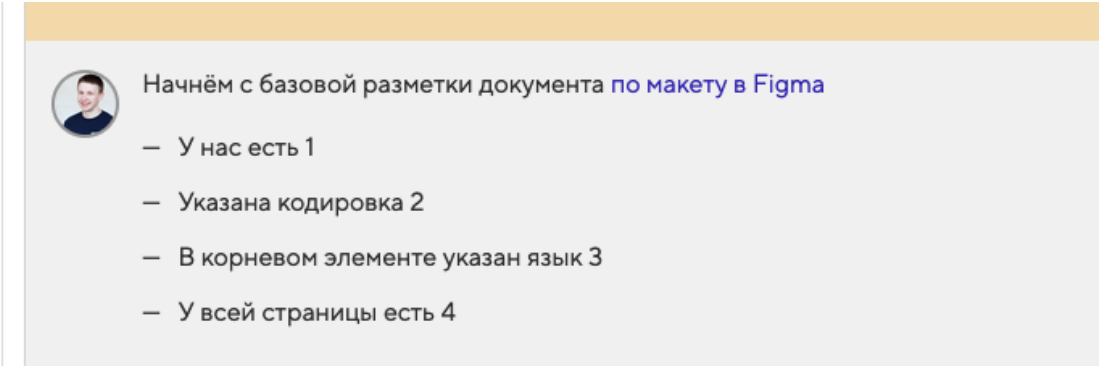
```
{
  "title": "Заголовок тестовой демки",
```

```
  "description": "Демку можно создавать и редактировать в репозитории. Это ли не чудо?"
}
```

Возможные настройки:

- `title` - название демонстрации
- `description` - Краткое описание демонстрации
- `base` - путь к статике демонстрации. По умолчанию (если `base` не указан) статика собираются из папки `assets`, которая лежит внутри папки с демонстрацией. Если для нескольких демонстраций используется один и тот же набор статики, то эту статику можно положить в глобальную папку со статикой на весь курс .(`assets` в корне репозитория) и добавить `base`, тогда вся статика будет браться из той папки. Например `/assets/demos/barbershop/`
- `author` - автор демонстрации. Принимает в себя два параметра
  - `image` - ссылка на картинку (может быть локальной, может быть извне). Путь к картинке прописывается либо абсолютный (со схемой и доменом) либо относительно папки со статикой. То есть если `image` указать `author/kolia.jpg`, то если `base` не указан, то картинка должна лежать в папке `assets` внутри демонстрации, а там уже `author/kolia.jpg` . А если `base` указан, то внутри папки `<base>/author/kolia.jpg`
  - `name` - имя автора

  Подтянется в блок с описанием шага таким образом



- `isTextDemo` - необязательный параметр, созданный для показа информации в виде текстовой демонстрации. В параметр можно передать `true` или `false`.

Когда флаг активирован, то движок демонстрации считывает только общий `config.json` и `readme.md` в папке каждого шага. Весь контент в `readme.md` считывается как контент шага, чтобы указать заголовок необходимо написать первую строку с решёткой (`#`). Так же заголовок можно указать в конфигурации каждого шага (`config.json`), он тоже подтянется. Вот пример `readme.md`:

```
# Это хедер!!

Такую простую с виду задачу на самом деле можно решить десятком способов, при этом на ум сначала приходят совершенно неоптимальные решения.
Перед тем, как переходить к следующим шагам, попробуйте прийти к самому оптимальному из них.
```

Папка `assets` для текстовых демонстраций должна находиться в корневой директории самой демонстрации, там где `config.json`.

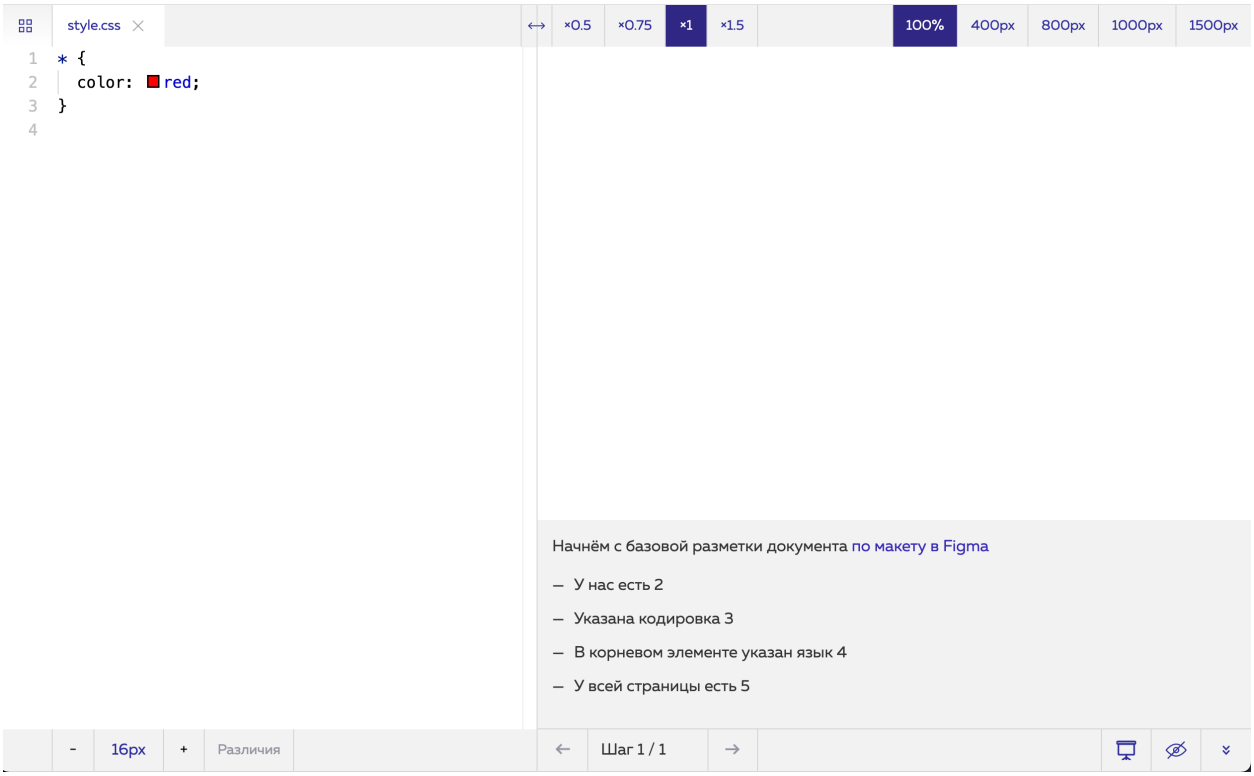
- `tab` - Открытая вкладка по умолчанию. Возможные значения: Browser, Layout. Может быть переопределено в настройках отдельного шага
- `delimiterPosition` - Изменяет ширину панели редактирования, Возможные значение: от 0 до 1. Может быть переопределено в настройках отдельного шага
- `viewportSize` - Размер вьюпорта браузера. Возможные значения: 100%, 400px, 800px, 1000px, 1500px. Может быть переопределено в настройках отдельного шага
- `viewportZoom` - Значение масштабирования браузера. Возможные значения: 0.5, 0.75, 1, 1.5. Может быть переопределено в настройках отдельного шага

Контент демонстрации

- `content.json` – файл с контентом демонстрации. Система автоматически туда сохраняет код. Руками лучше не редактировать, чтобы не сломать. Вместо этого файла система так же проверяет на формат хранения в виде файловой структуры. Структура описана ниже

- `assets` – папка с ассетами для демонстрации (картинки, вспомогательные файлы)

Пример демонстрации



Управление шагами демонстрации

Шаги демонстрации размещаются в отдельных пронумерованных папках.

В каждой папке должен быть файл `config.json` с конфигурацией для этого шага демонстрации. Значения ключей, используемых в этом файле, переопределяют значения, указанные в основном файле конфигурации.

В файле `readme.md` хранится описание шага демки.

Пример структуры шагов

```
modules/              // папка модулей
  02-intro/           // модуль
    demo-1            // демонстрация
      assets/         // папка с ассетами
      01/             // первый шаг
        code/         // папка с кодом шага
        config.json   // кофиг шага
```

```
    readme.md      // описание шага
  02/              // второй шаг
    code/          // папка с кодом шага
    config.json    // кофиг шага
    readme.md      // описание шага
  03/              // третий шаг
    code/          // папка с кодом шага
    config.json    // кофиг шага
    readme.md      // описание шага
  config.json      // кофиг демонстрации
```

Настройки шага

- `title` - Название шага демки
- `mode` - Режим шага. Сейчас доступен только "render-single-html"
- `visibleTabs` - Список путей до файлов, для которых будут созданы табы с редакторами. Например, ["catalog.html", "js/index.js"]
- `activeEditor` - Путь к файлу, для которого открыт редактор и таб с которым будет открытым. Например, "catalog.html"
- `tab` - Открытая вкладка по умолчанию. Возможные значения: Browser, Layout.
- `delimiterPosition` - Изменяет ширину панели редактирования, Возможные значение: от 0 до 1.
- `viewportSize` - Размер вьюпорта браузера. Возможные значения: 100%, 400px, 800px, 1000px, 1500px.
- `viewportZoom` - Значение масштабирования браузера. Возможные значения: 0.5, 0.75, 1, 1.5.
- `console` - Активирует консоль в шаге. "active" – активирует консоль в закрытом режиме; "opened" – активация в открытом виде

Значения валидируются при построении демонстрации, результат выводится в консоль.

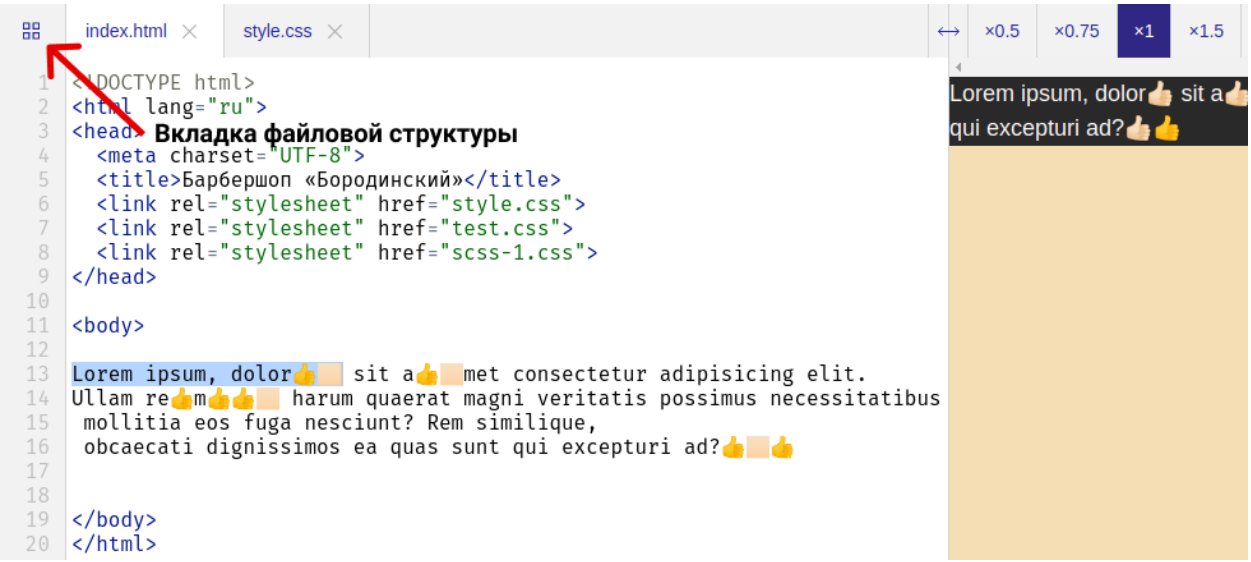
Подключение ресурсов демонстрации

Ресурсы, расположенные в папке `assets` внутри основной папки демонстрации, доступны по пути `assets`.

Пример подключения в разметке:

```
<link rel="stylesheet" href="assets/style.css">
<link rel="stylesheet" href="assets/styles/style.css">
<img src="assets/ease.png">
<script src="assets/popup.js">
```

Файловая структура в шаге



Все файлы из папки `code` в шаге, кроме игнорируемых сборщиком, попадают в файловую структуру шага в самой демонстрации.

Файлы, отображаются во вкладке с файловой структурой, каждый из них можно открыть в редакторе.

Выделение кода в демонстрации

Выделение кода в редакторах реализовано посредством добавления комментариев.

Пример выделения для CSS

```
.page-header {
  background-color: #242424;
  color: #ffffff;
}


.page-header__logo {
```

```
/* / */outline: 5px solid rgba(255, 0, 0, 0.8);
  outline-offset: -5px;/* \ */
}
```

Пример отображения выделения

```
74  .page-header {
75    background-color: #242424;
76    color: #ffffff;
77  }
78
79  .page-header__logo {
80    outline: 5px solid rgba(255, 0, 0, 0.8);
81    outline-offset: -5px;
82  }
83
```

Такие комментарии при отображении вырезаются из кода. Технически это не комментарии, но сделаны комментариями, чтобы было удобнее с ними работать.

Разные типы комментариев:

- Для html — открывающий `<!-- / -->`, закрывающий `<!-- \ -->`
- Для js — открывающий `/* / */`, закрывающий `/* \ */`
- Для css — открывающий `/* / */`, закрывающий

Пробелы и переносы строк игнорируются.

Управление опросами и тестами

Опросы и тесты позволяют провести срез по навыкам, которые освоил студент

Каждый тест собирается в отдельной папке. Само название папки ни на что не влияет, оно используется только для адресации в программе и более легкого ориентирования по материалам.
Внутри папки необходимо добавить файл `content.json`, описывающий структуру опроса

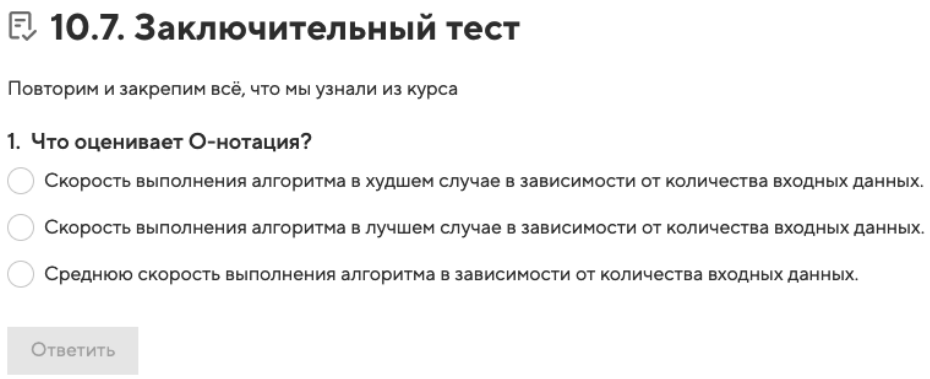
Пример конфига

```
title: Заключительный тест
type: test
description: Повторим и закрепим всё, что мы узнали из курса
wizard: true
questions:
  - id: 1
    title: |
       Что оценивает O-нотация?
    type: single
    options:
      - id: 1
        value: Скорость выполнения алгоритма в худшем случае в зависимости от количества в
ходных данных.
        correct: true
      - id: 2
        value: Скорость выполнения алгоритма в лучшем случае в зависимости от количества в
ходных данных.
        correct: false
        resultMessage: |
          Нет, ведь тогда почти все алгоритмы работали бы максимум за O(n), а оценка такая
пригодилась бы в редких случаях. На всякий случай стоит перечитать первый раздел.
      - id: 3
        value: Среднюю скорость выполнения алгоритма в зависимости от количества входных д
анных.
        correct: false
        resultMessage: |
          Неверно — лучше перечитайте первый раздел. В нём говорилось, что измеряет O-нота
ция и зачем она вообще нужна.
  - id: 2
    title: |
      Какая из сложностей ниже самая большая?
    type: single
    options:
      - id: 1
        value: O(logn).
        correct: false
        resultMessage: |
          Наоборот, это самая быстрая из всех представленных сложностей. Больше та сложнос
ть, у которой выражение в O-нотации больше.
      - id: 2
        value: O(n²).
        correct: false
        resultMessage: |
          Это не самая большая сложность. Сравните все выражения в скобках O-нотации ещё р
аз.
      - id: 3
        value: O(n!).
        correct: true
```

Возможные настройки

- `title` - название теста
- `type` - тип тест `test` (результаты важны) или опрос `survey` (важен только сам факт заполнения)
- `description` - описание
- `wizard` - `true` - отображение по одному вопросу на странице, `false` - отображение всех вопросов сразу
- `successMessage` - сообщение в случае успешного прохождения
- `failureMessage` - сообщение в случае ошибки
- `questions` - список вопросов, где каждый вопрос может содержать следующие настройки
  - `id` - уникальный идентификатор вопроса
  - `title` - текст вопроса
  - `description` - подробное описание вопроса
  - `type` - тип вопроса single - выбор одного варианта ответа, multiple - множественный выбор нескольких вариантов, text - ввод строки с ответом
  - `options` - перечень вариантов ответа, где каждый вариант может содержать следующие настройки:
    - `id` - уникальный идентификатор варианта
    - `value` - занечение для выбора
    - `correct` - флаг, корректный или нет это вариант
    - `description` - описание варианта
    - `resultMessage` - сообщение, если пользователь выбрал этот вариант

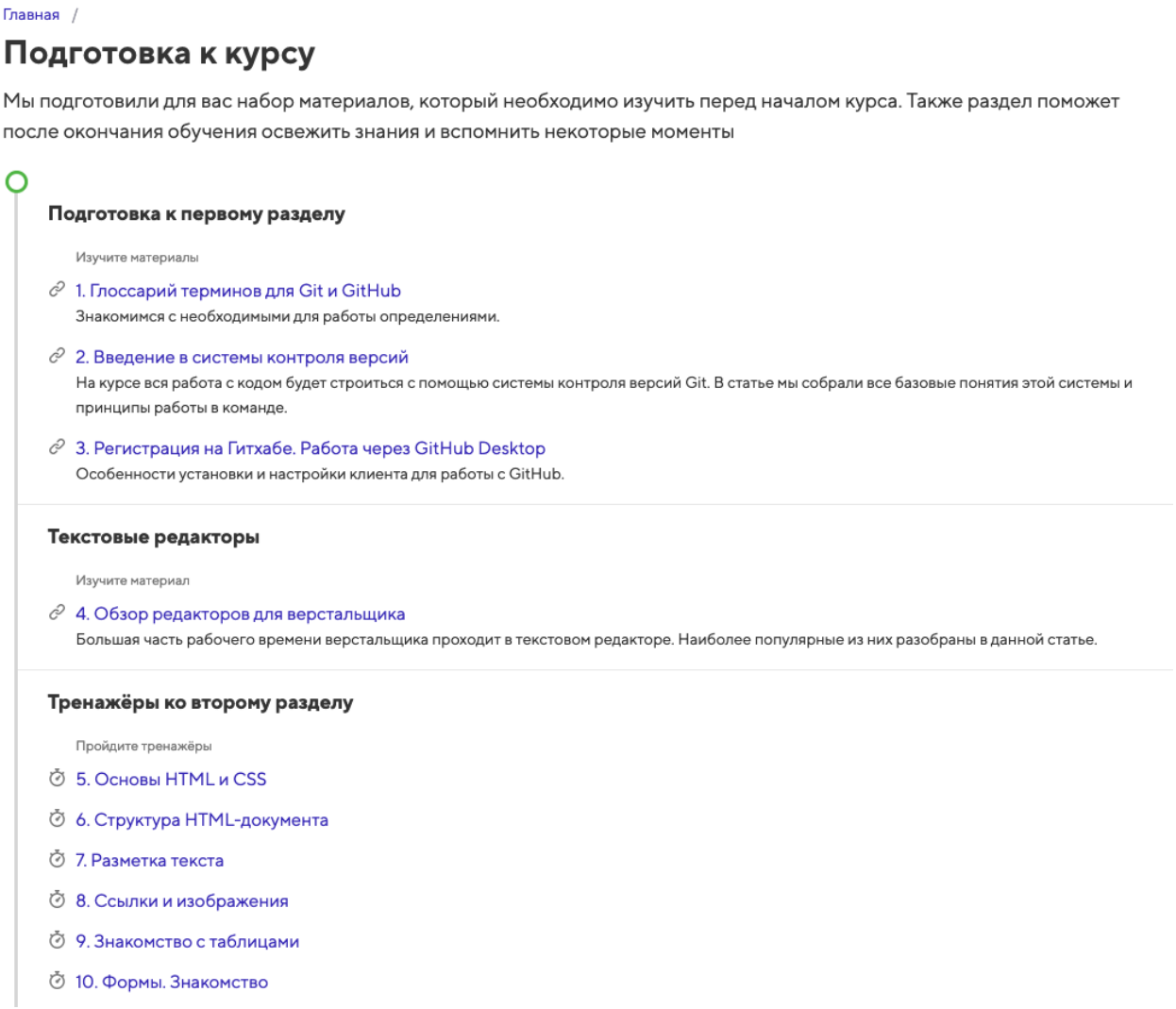
Пример отображения опроса



Управление подготовкой к курсу

Отдельный раздел программы предназначенный для прохождения материалов, которые необходимо изучить до старта курса

Материалы собираются из файла `prepare/program.json` . Структура и формат данных аналогичен модулям



Управление материалами в Markdown

Все текстовые материалы хранятся в формате Markdown. Система автоматически поддерживает все стандартные элементы разметки Markdown https://guides.github.com/features/mastering-markdown/ и автоматически преобразовывает их в HTML-элементы для отображения на страницах курса

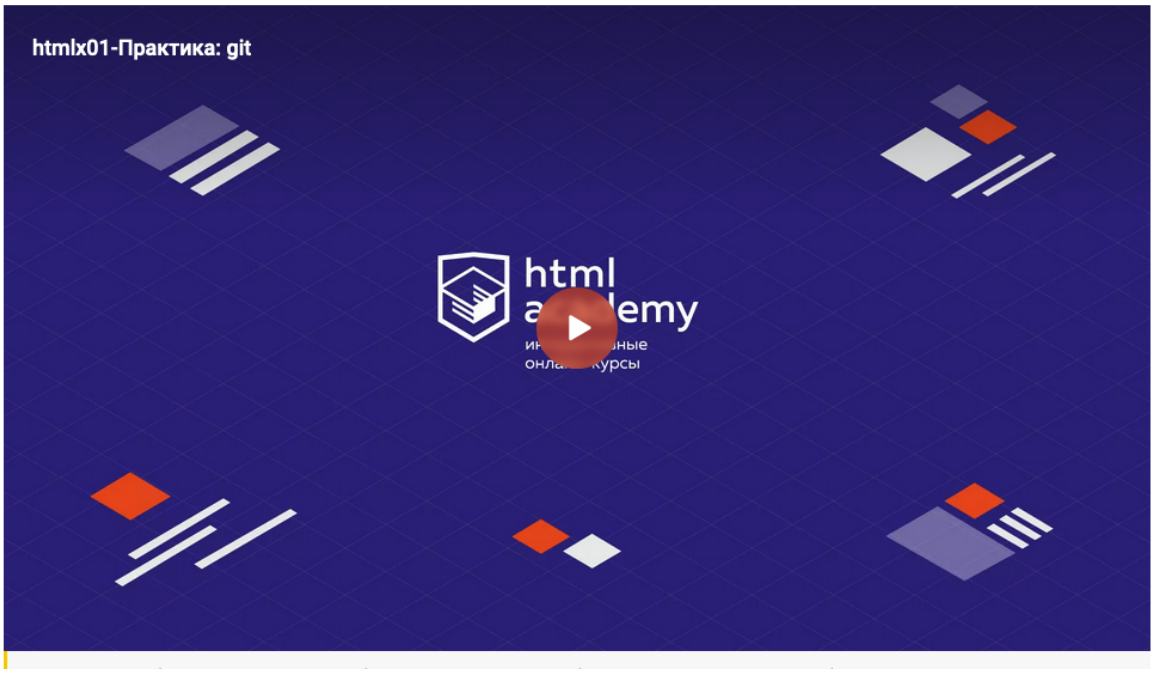
Дополнительные возможности:

- Вставка ссылок на видео

  Ссылки на видео на Vimeo, YouTube или Кинескоп автоматически разбираются. На страницу подставляет плеер для воспроизведения видео

```
![Пример Видео](<https://player.vimeo.com/video/259411563>)
```

Пример отображения



- Подпись к блоку кода

  Дает возможность добавить аккуратную подпись к блоку с кодом

```
```html В данном случае анимация будет применена к элементу circle
<circle …>
    <animate .../>
</circle>
```

Пример отображения

```
<circle …>
    <animate .../>
</circle>
```

В данном случае анимация будет применена к элементу circle

Работа со статикой
```

Всю статику по курсу можно хранить прямо внутри курса, и больше не надо ее закидывать ни в какие другие места.

Есть два типа статики:

- Глобальная
- Локальная

Глобальная статика

В корне проекта можно создать папку `assets` и положить туда статику, которая может использоваться на курсе в нескольких местах.

Например на курсе по верстке, вся статика демок по Барбершопу лежит в глобальной папке. Так как демок по барбершопу несколько, то удобнее не дублировать эту статику, а хранить в одной папке

```
/assets
  /demos
    /barbershop
      /img
        logo.svg
        ...
```

В материалах в данном случае, используя картинки, путь указывается от корня проекта. Например вот так, для использования логотипа в каком-то задании

```
![](/assets/demos/barbershop/img/logo.svg)
```

Локальная статика

Внутри отдельных сущностей может использоваться своя локальная статика. Например внутри отдельного задания может лежать своя папка `assets`

```
/modules
  /01-intro
    /task-1
      /assets
        preview.jpeg
```

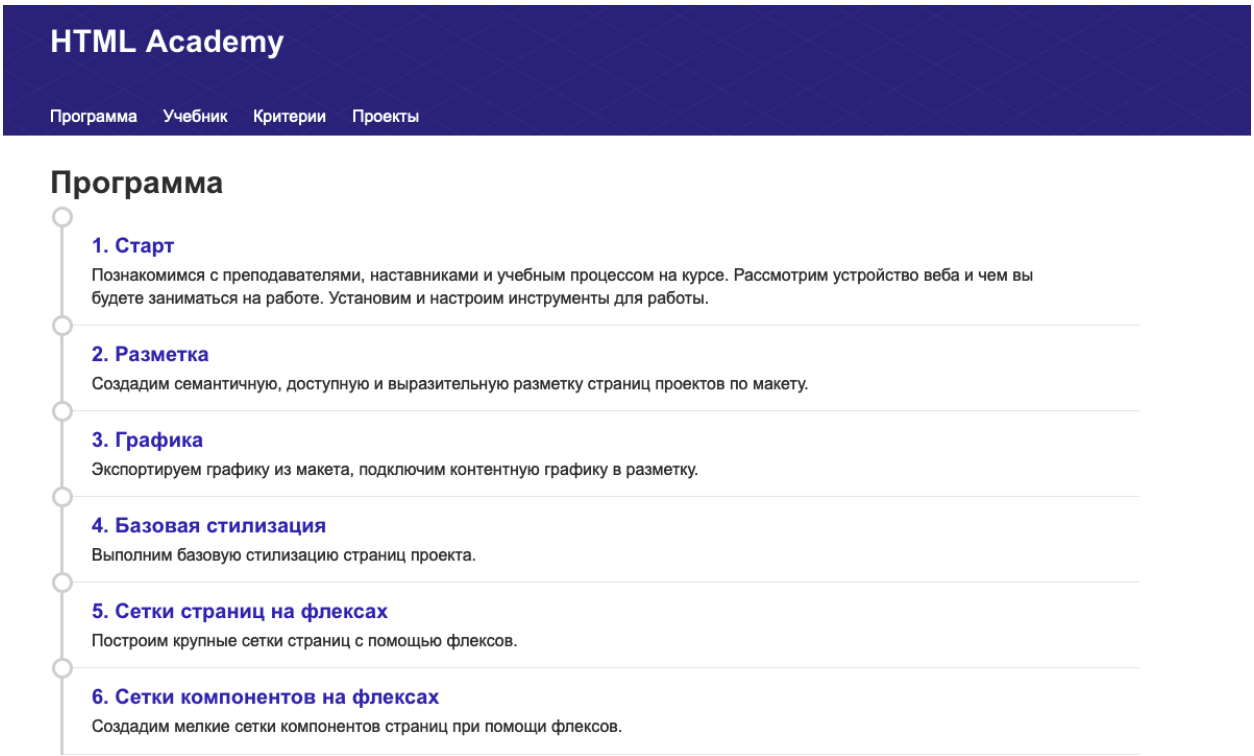
В задании в данном случае, используя картинки, путь указывается следующим образом

```
![](assets/preview.jpeg)
```

Воспроизведение курса

Запустив курс локально или развернув его на сервере его можно просмотреть и пройти.

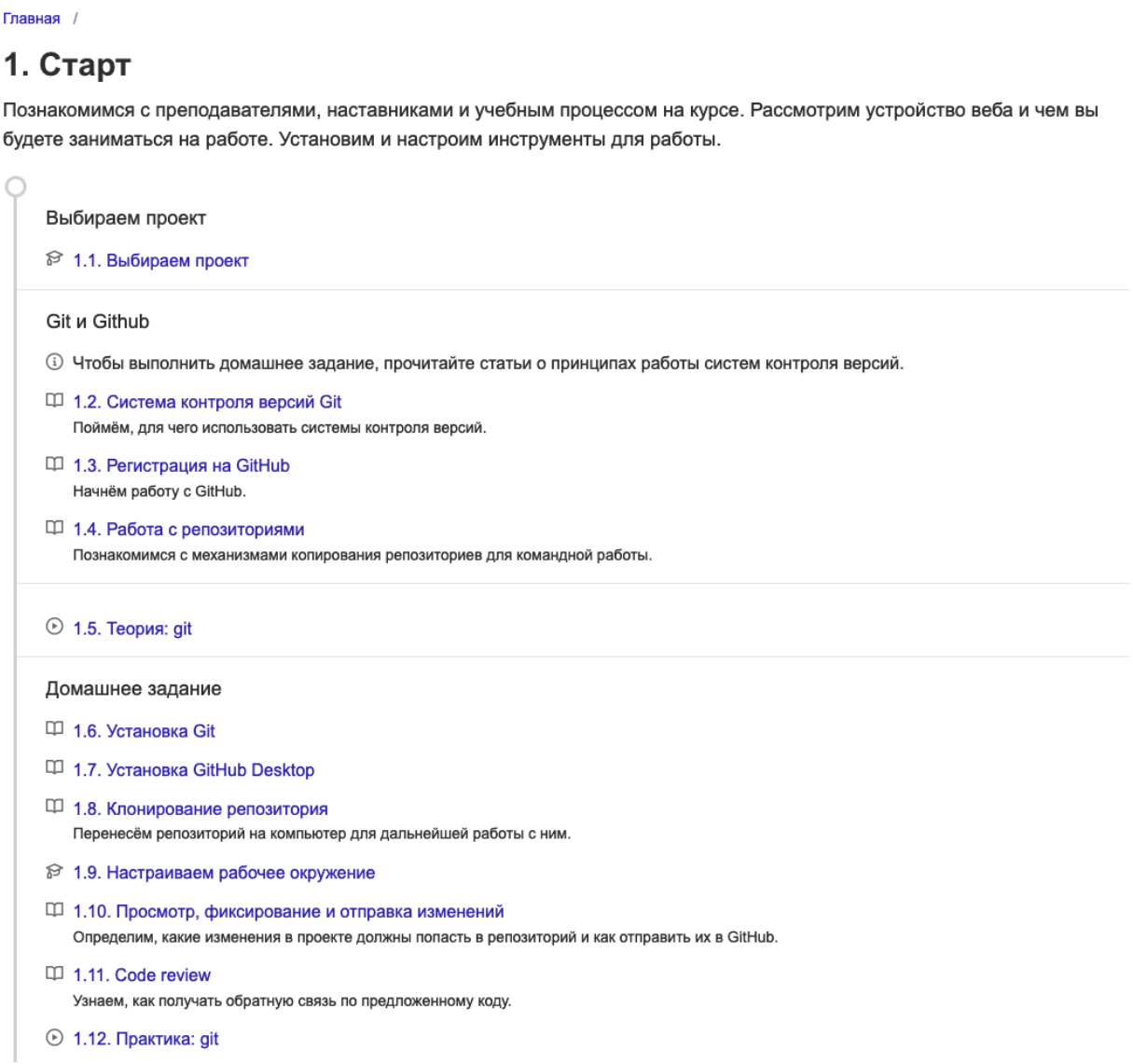
Пример программы курса



На основе созданных в конфигурации материалов система разворачивает курс и настраивает навигацию по нему для пользователей

Все материалы курса доступны для просмотра и прохождения

Пример модуля курса



Локальный запуск курса осуществляется командой `npm start`

На сервере курс доступен по специальному созданному для него адресу