


ХОЧУ ПОМОЧЬ ПРОЕКТУ



practicum.yandex.ru

РЕКЛАМА • 16+ Я

Курс «React-разработчик» от Яндекс Практикума

Наставники из Яндекса • Много практики • Сертификат от Яндекса • Код-ревью

Узнать больше

gb.ru

Курсы РНР для начинающих. 3 месяца Бесплатно

Освой востребованную профессию в ИТ «РНР-разработчик». 3 месяца обучения Бесплатно!

Скидка 60% >

Трудоустройство через 9 мес. >

Год английского Бесплатно >

90% обучения - практика >

Узнать больше

- настраиваемое отображение из собственных стилей .docx в HTML. Например, можно преобразовать заголовок WarningHeading в h1.warning, предоставив соответствующее сопоставление стилей.
- Таблицы. Форматирование самой таблицы, такое как границы, в настоящее время игнорируется, но форматирование текста обрабатывается так же, как и в остальной части документа.
- Сноски и концевые примечания.
- Изображения.
- Жирный шрифт, курсив, подчеркивание, зачеркивание, верхний и подстрочный индексы.
- Связи.
- Разрывы строк.
- Текстовые поля. Содержимое текстового поля обрабатывается как отдельный абзац, который появляется после абзаца, содержащего текстовое поле.
- Комментарии.

Установка модуля mammoth в виртуальное окружение:

```
# создаем виртуальное окружение
$ python3 -m venv .venv --prompt VirtualEnv
# активируем виртуальное окружение
$ source .venv/bin/activate
# ставим модуль mammoth
(VirtualEnv) :~$ python3 -m pip install mammoth --upgrade
```

Содержание:

- [Преобразование DOCX в HTML в командной строке;](#)
 - [Извлечение изображений из документа `DOCX`;](#)
 - [Встраивание стилей в итоговый HTML-фрагмент;](#)
- [Преобразование DOCX в HTML в коде Python;](#)
 - [Функция mammoth.convert to html\(\);](#)
 - [Пользовательский обработчик стилей в выходном HTML;](#)
 - [Функция mammoth.embed style map\(\);](#)
 - [Изменение стилей документа Bold, Italic, Underline и т.д. в выходном HTML;](#)
 - [Пользовательские обработчики изображений;](#)
- [Написание пользовательских карт стилей.](#)

Вверх

Преобразование DOCX в HTML в командной строке.

Чтобы преобразовать DOCX-файлы в HTML, используя командную строку, необходимо команде mammoth передать путь к файлу с расширением .docx и указать выходной файл. Например:

РЕКЛАМА · 16+

GeekBrains

РНР

gb.ru

Курсы РНР для начинающих. 3 месяца Бесплатно

Освой востребованную профессию в ИТ «РНР-разработчик». 3 месяца обучения Бесплатно!

Скидка 60%

Трудоустройство через 9 мес.

Год английского Бесплатно

90% обучения - практика

Узнать больше

python mammoth document.docx > output.html

Выходные данные записываются в стандартный вывод консоли.

Файл output.html кодирован в кодировке UTF-8, а не полный HTML-документ. Так как кодировка явно не задана в HTML-файла в веб-браузере может привести к неправильному отображению символов Unicode, если браузер не поддерживает UTF-8.

Извлечь HTML-фрагмент из документа DOCX.

Если не указан выходной файл, то результат будет выведен в stdout. Если указан выходной каталог параметром --output-dir, то файлы будут записаны в указанную папку в отдельные файлы (очень удобно для извлечения изображений из документа DOCX).

Пользовательские стили задаются с помощью параметра --style-map. Например:

python mammoth --style-map=custom-style-map.txt document.docx --output-dir=img-dir

Если не указан выходной файл, то результат будет выведен в stdout. Если указан выходной каталог параметром --output-dir, то файлы будут записаны в указанную папку в отдельные файлы (очень удобно для извлечения изображений из документа DOCX).

Пользовательские стили задаются с помощью параметра --style-map. Например:

python mammoth --style-map=custom-style-map.txt document.docx --output-dir=img-dir

Файл с описанием пользовательских стилей выглядит примерно так:

```
p[style-name='Aside Heading'] => div.aside > h2:fresh
p[style-name='Aside Text'] => div.aside > p:fresh
```

Описание синтаксиса для составления карт стилей можно посмотреть в разделе "[Написание пользовательских карт стилей](#)".

Преобразование DOCX в HTML в коде Python.

Чтобы преобразовать существующий файл .docx в HTML, необходимо передать [файлоподобный объект](#) в функцию mammoth.convert_to_html(). Файл должен быть открыт в бинарном режиме. Например:

```
import mammoth

with open("document.docx", "rb") as docx_file:
    result = mammoth.convert_to_html(docx_file)
    # сгенерированный HTML
    html = result.value
    # предупреждения во время конвертации
    messages = result.messages
```

Также можно извлечь необработанный текст документа с помощью функции mammoth.extract_raw_text(). Эта функция игнорирует все форматирование в документе. За каждым абзацем следует две новые строки.

```
with open("document.docx", "rb") as docx_file:
    result = mammoth.extract_raw_text(docx_file)
    # Необработанный текст
    text = result.value
    # предупреждения во время конвертации
    messages = result.messages # Any messages
```

Функция `mammoth.convert_to_html(fileobj, **kwargs)`:


Функция mammoth.convert_to_html() преобразует исходный документ DOCX в HTML.


Принимаемые аргументы:


- fileobj: [файлоподобный объект](#), содержащий исходный документ. Файлы должны открываться в двоичном режиме.
- style_map: строка, задающая отображение стилей Word в HTML. Описание синтаксиса приведено в разделе "[Написание пользовательских карт стилей](#)".

- `include_embedded_style_map`: по умолчанию, если документ содержит встроенную карту стилей, то она объединяется с картой стилей по умолчанию. Чтобы игнорировать любые встроенные карты стилей, передайте `include_embedded_style_map=False`.
- `include_default_style_map`: по умолчанию карта стилей, переданная в `style_map`, объединяется с картой стилей по умолчанию. Чтобы отказаться от использования карты стилей по умолчанию, передайте `include_default_style_map=False`.

РЕКЛАМА • 16+

**РНР**



 gb.ru

Курсы РНР для начинающих. 3 месяца Бесплатно

Освой востребованную профессию в ИТ «РНР-разработчик». 3 месяца обучения Бесплатно!

Скидка 60%

Трудоустройство через 9 мес.

Год английского Бесплатно

90% обучения - практика

Узнать больше

Изображения преобразуются в элементы `` с указанием источника, встроенного в атрибут `src` в значение конвертера изображений, чтобы переопределить поведение по умолчанию. По умолчанию пустые абзацы игнорируются. Установите для этого параметра значение `False`, чтобы не включать их в выходные данные.

В зависимости от любых сгенерированным идентификаторам, таким как те, которые используются в примечаниях. По умолчанию используется пустая строка.

Свойства:

Такие как ошибки и предупреждения, сгенерированные во время преобразования.

Следующие свойства:

- `type`: тип сообщения, например "warning".
- `text`: фактическое сообщение.

Карты стилей в выходном HTML.

Модуль `mammoth` предоставляет некоторые общие стили `.docx` с элементами HTML. Например, абзац с названием стиля `Section Title` будет преобразован в элемент `<h1>`. Можно передать пользовательскую карту стилей, в качестве второго аргумента функции `convert_to_html()`. Допустим, что необходимо преобразовать абзацы с названием стиля `Section Title` в элемент `<h1>` и абзацы с названием стиля `Subsection Title` в элемент `<h2>`:

```
import mammoth

style_map = """
p[style-name='Section Title'] => h1:fresh
p[style-name='Subsection Title'] => h2:fresh
"""

with open("document.docx", "rb") as docx_file:
    html = mammoth.convert_to_html(docx_file, style_map=style_map)
```

Пользовательские сопоставления `style_map` будут использоваться вместо сопоставлений стилей по умолчанию. Чтобы вообще отказаться от сопоставления стилей по умолчанию, передайте третий ключевой аргумент `include_default_style_map=False`:

```
html = mammoth.convert_to_html(docx_file, style_map=style_map, include_default_style_map=False)
```

Описание синтаксиса для составления карт стилей можно посмотреть в разделе "[Написание пользовательских карт стилей](#)".

Функция `mammoth.embed_style_map(fileobj, style_map)`:

Функция `mammoth.embed_style_map()` встраивает карту стилей `style_map` в файл `fileobj`. Когда `mammoth` читает файловый объект, он будет использовать встроенную карту стилей (т.е. дополнительно передавать ее аргументом `style_map` в функцию `mammoth.convert_to_html` уже будет не надо).

Принимаемые аргументы:

- `fileobj`: [файлоподобный объект](#), содержащий исходный документ. Файлы должны быть открыты для чтения и записи в двоичном режиме.
- `style_map`: карта стилей для встраивания.

Возвращает `None`.

Изменение стилей документа Bold, Italic, Underline и т.д. в выходном HTML.

Полужирный текст.


По умолчанию полужирный текст заключен в теги ``. Это поведение можно изменить, добавив отображение стиля для `b`. Например, чтобы заключить полужирный текст в теги ``:


Вверх


```
style_map = "b => em"

with open("document.docx", "rb") as docx_file:
    result = mammoth.convert_to_html(docx_file, style_map=style_map)
```

РЕКЛАМА • 16+

**PHP**



 gb.ru

Курсы PHP для начинающих. 3 месяца Бесплатно

Освой востребованную профессию в ИТ «PHP-разработчик». 3 месяца обучения Бесплатно!

Скидка 60% >

Трудоустройство через 9 мес. >

Год английского Бесплатно >

90% обучения - практика >

Узнать больше

курсивом, заключен в теги ``. Это поведение можно изменить, добавив сопоставление стилей текст, выделенный курсивом, в теги ``:

```
as docx_file:
    result = mammoth.convert_to_html(docx_file, style_map=style_map)
```

о текста игнорируется, поскольку подчеркивание можно спутать со ссылками в HTML - изменить, добавив сопоставление стилей для `u`. Например, предположим, что в исходном тексте есть подчеркивание. Код ниже любой явно подчеркнутый исходный текст заключает в теги `<u>`:

```
as docx_file:
    result = mammoth.convert_to_html(docx_file, style_map=style_map)
```

зачеркнутый текст в теги `<s>`. Это поведение можно изменить, добавив сопоставление стилей для `del` в теги ``:

```
style_map = "strike => del"

with open("document.docx", "rb") as docx_file:
    result = mammoth.convert_to_html(docx_file, style_map=style_map)
```

Комментарии в тексте.

По умолчанию комментарии игнорируются. Чтобы включить комментарии в сгенерированный HTML-код, добавьте сопоставление стилей для `comment-reference`. Например:

```
style_map = "comment-reference => sup"

with open("document.docx", "rb") as docx_file:
    result = mammoth.convert_to_html(docx_file, style_map=style_map)
```

Комментарии будут добавлены в конец документа со ссылками на комментарии, обернутыми с использованием указанного сопоставления стилей.

Пользовательские обработчики изображений.

По умолчанию изображения преобразуются в элементы `` с включенным источником в атрибут `src`. Это поведение можно изменить, задав для аргумента `convert_image` значение конвертера изображений.

Например, следующий код будет воспроизводить поведение по умолчанию:

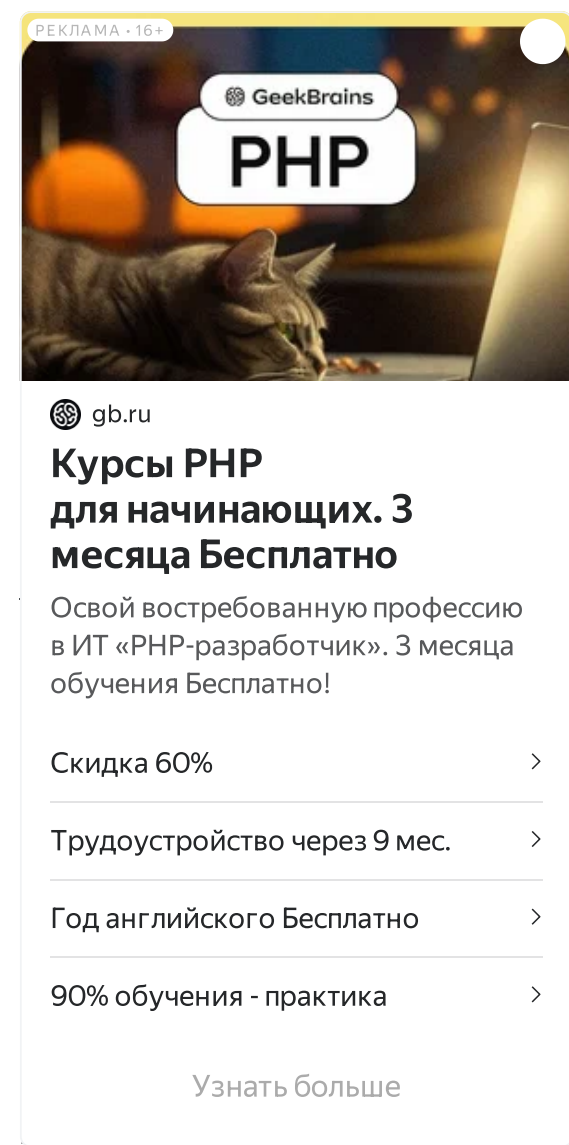
```
def convert_image(image):
    with image.open() as image_bytes:
        encoded_src = base64.b64encode(image_bytes.read()).decode("ascii")
    return {
        "src": f"data:{image.content_type};base64,{encoded_src}"
    }

with open("document.docx", "rb") as docx_file:
    result = mammoth.convert_to_html(docx_file, convert_image=mammoth.images.img_element(convert_image))
```

Конвертер изображений по умолчанию `mammoth.images.data_uri`.

Вверх

Конвертер изображений можно создать, вызвав `mammoth.images.img_element(func)`. Это создаст элемент `` для каждого изображения в исходном `.docx` файле. Аргумент `func` должен быть функцией, принимающей один аргумент `image`. Этот аргумент является преобразуемым элементом изображения и обладает следующими свойствами:



ажения. Возвращает [файлоподобный объект](#).
содержимого изображения, например `image/png`.

Список атрибутов для элемента ``. Как минимум, это должно включать атрибут `src`. Если какой-либо альтернативный текст, он будет автоматически добавлен к атрибутам элемента.

Изображения *WMF* по умолчанию не обрабатываются модулем *mammoth*.

Сопоставление стилей карт стилей.

Сопоставлений стилей, каждая из которых начинается с новой строки. Пустые строки и строки,

Сопоставление частей:

Средство сопоставления элементов документа.
тег.

Маммوت находит первое сопоставление стилей, в котором средство сопоставления элементов соответствует абзацу. Затем маммот гарантирует, что HTML-тег будет удовлетворен.

Сопоставление элементов.

Это полезно понимать "свежесть" HTML-элемента. При генерации HTML маммот закрывает HTML-тег в том случае, если элемент используется повторно.

Если из указанных сопоставлений стилей является `p[style-name='Heading 1'] => h1`. Если маммот встречает абзац в `.docx` с названием стиля `Heading 1`, то абзац `.docx` преобразуется в элемент `<h1>` с тем же текстом. Если следующий абзац в `.docx` также имеет название стиля `Heading 1`, то текст этого абзаца будет добавлен к существующему тегу `<h1>`, и не приведет к созданию нового HTML-элемента `<h1>`.

В большинстве случаев, необходимо будет сгенерировать новый элемент `h1`. Это можно сделать с помощью модификатора `:fresh`:

```
p[style-name='Heading 1'] => h1:fresh
```

Два последовательных абзаца с `Heading 1` будут преобразованы в два отдельных элемента `h1`.

Повторное использование элементов полезно при создании более сложных HTML-структур. Например, предположим, что файл `.docx` содержит отступы `aside`. У каждого отступа `aside` может быть заголовок и некоторый основной текст, которые должны содержаться в пределах одного элемента `div.aside`. В этом случае могут быть полезны сопоставления стилей, подобные:

```
p[style-name='Aside Heading'] => div.aside > h2:fresh
# и
p[style-name='Aside Text'] => div.aside > p:fresh
```

Совпадения с абзацем, прогоном и таблицей.

- `p =>` - совпадение с любым абзацем;
- `r =>` - совпадение с любым прогоном;
- `table =>` - совпадение с любой таблицей;

Использование в карте стилей:

```
# обернет абзац в <div>...</div>
p => div
# обернет абзац в <p>...</p>
p => p
# обернет прогон в <span>...</span>
r => span
```

Чтобы сопоставить абзац, прогон или таблицу с определенным стилем документа DOCX, можно сослаться на стиль по имени. Это название стиля, которое отображается в Microsoft Word или LibreOffice. Например, чтобы сопоставить абзац с названием стиля `Heading 1`:


Вверх

```
p[style-name='Heading 1'] => h1
```

Можно также сопоставить имя стиля по префиксу. Например, чтобы соответствовать абзацу, имя стиля которого начинается с Heading:

```
p[style-name^='Heading'] => h1
```

РЕКЛАМА · 16+



Курсы PHP для начинающих. 3 месяца Бесплатно

Освой востребованную профессию в ИТ «PHP-разработчик». 3 месяца обучения Бесплатно!

Скидка 60%

Трудоустройство через 9 мес.

Год английского Бесплатно

90% обучения - практика

Узнать больше

о идентификатору стиля. Это идентификатор, используемый внутри файла .docx. Чтобы определенным идентификатором стиля, нужно добавить точку, за которой следует идентификатор абзаца с идентификатором стиля Heading1:

Bold, Italic, Underline и т.д.

исленные индификаторы соответствуют только тексту, к которому явно применен жирный шрифт. Он не будет соответствовать тексту, который помещен в абзац или прогон со стилем - (т.д.).

- деленному жирным шрифтом тексту;
- деленному курсивом тексту;
- подчеркнутым текстом;
- явно зачеркнутым текстом;
- со всеми заглавными буквами текста;
- т с явно маленькими заглавными буквами;
- ! игнорирует элемент документа. Например, чтобы игнорировать любой абзац со стилем

модификаторов выше, в "[Изменение стилей документа Bold, Italic, Underline и т.д. в](#)

HTML-теги и внедрение пользовательских CSS-классов.

Самый простой путь преобразовать совпадение - это указать один HTML-элемент. Например, чтобы указать элемент:

```
`p.Heading1 => h1`
```

Чтобы присвоить элементу класс CSS, необходимо добавить точку, за которой следует название класса:

```
p.Heading1 => h1.section-title
```

Модификаторы должны использоваться в правильном порядке:

```
p[style-name^='Heading'] => h1.section-title:fresh
```

Чтобы указать разделитель для размещения между содержимым абзацев, которые свернуты вместе, необходимо использовать :separator('SEPARATOR STRING').

Например, предположим, что документ содержит блок кода, где каждая строка кода представляет собой абзац со стилем Code Block. Можно написать сопоставление стилей для сопоставления таких абзацев с элементами <pre>:

```
p[style-name='Code Block'] => pre
```

Поскольку <pre> не помечен как :fresh, то последовательные элементы <pre> будут свернуты вместе. Однако это приводит к тому, что весь код находится в одной строке. Можно использовать :separator для вставки новой строки между каждой строкой кода:

```
p[style-name='Code Block'] => pre:separator('\n')
```

Вложенные HTML-элементы.

Для указания вложенных HTML элементов необходимо использовать >. Например, чтобы указать h2 внутри div.aside:

```
p[style-name^='Heading2'] => div.aside > h2
```

Можно вкладывать элементы на любую глубину.