

Convert HTML to PDF in Python with 5 Popular Libraries (Updated 2025)



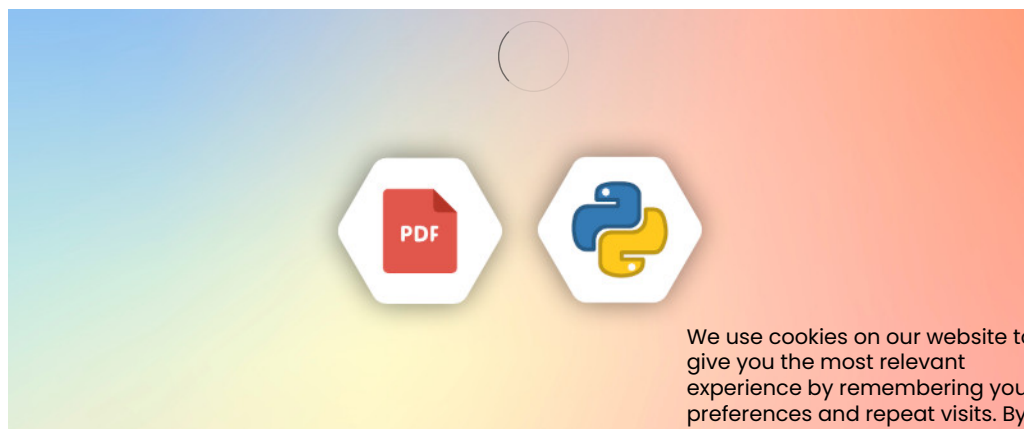
By Manoj Ahirwar



January 6, 2025(<https://apitemplate.io/2025/01/06/>)

We often encounter the need to create PDFs based on content. While there is no right or wrong way to generate PDFs, some approaches are more efficient and quicker to build than others.

Previously, we had to write all the boilerplate code to generate PDFs in our applications.



However, now we have many great libraries and tools that can help us quickly implement this feature.

The most important part of generating PDFs is the input data. The most common and useful approach is to generate PDFs from HTML content or based on a website URL.

In this article, we will look into some approaches that we can take to generate PDFs from HTML.

We use cookies on our website to give you the most relevant experience by remembering your preferences and repeat visits. By clicking "Accept All", you consent to the use of ALL the cookies.

To get more information about these cookies and the processing of your personal data, check our [Privacy Policy](https://apitemplate.io/privacy-policy/) (<https://apitemplate.io/privacy-policy/>).

Accept All

Table of Contents

Why generate PDF from HTML?

HTML to PDF using Python Libraries

- Pyppeteer
- xhtml2pdf
- python-pdfkit
- Playwright
- WeasyPrint

Comparison of All 5 Popular Libraries

HTML to PDF using APITemplate.io

- Template-based PDF generation
- Generate PDF from the website URL
- Generate PDF from custom HTML content

Conclusion

TL;DR: We provide a robust REST API (<https://apitemplate.io/apiv2/>) designed for seamless PDF generation with popular programming languages like Python.

In addition, our free HTML to PDF converter (<https://apitemplate.io/pdf-tools/convert-html-to-pdf/>) lets you instantly transform HTML into high-quality PDFs. Try our free HTML to PDF converter (<https://apitemplate.io/pdf-tools/convert-html-to-pdf/>) online today!

Why generate PDF from HTML?

Before we move on to the libraries, first let's see why we prefer HTML as input data for generating PDFs. Some of the reasons are as follows

1. **Open and Mature Technology:** HTML is an open standard, which ensures that tools and technologies built around it are widely available and well-understood. Its maturity also means that most of the challenges and quirks are well-documented, making troubleshooting easier.
2. **Cost-effective:** There are a plethora of tools, libraries, and APIs available (both free and paid) that can convert HTML to PDF, reducing the need for specialized software for PDF creation.
3. **Embed Multimedia:** HTML supports the embedding of multimedia such as images, videos, and audio. Although not all of these can be directly translated into a PDF, having a source in HTML provides options for creating rich, multimedia-enhanced documents.
4. **Styling with CSS:** Cascading Style Sheets (CSS) provide powerful styling options for HTML content, allowing for branding, theming, and visual consistency. These can then be reflected in the resulting PDF.
5. **Easy to Learn and Use:** Learning the basics of HTML can be done quickly, making it accessible for many users to create content.

In summary, converting PDFs from HTML combines the best of both worlds: the flexibility, accessibility, and interactivity of HTML with the portability and Standardization of PDFs.

HTML to PDF using Python Libraries

There are many libraries available in Python that allow the generation of PDFs from HTML content, some of them are explained below.

When generating HTML to PDF in Python, we need libraries and solutions which does not compromise the formatting of the PDF. With the following Open Source libraries you don't need to worry about losing formatting because all the below solutions take care of the formatting when generating HTML to PDF using Python.

We use cookies on our website to give you the most relevant experience by remembering your preferences and repeat visits. By clicking "Accept All", you consent to the use of ALL the cookies.

For more information about these cookies and the processing of your personal data, check our privacy policy (<https://apitemplate.io/privacy-policy/>).

i. Pyppeteer

Pyppeteer (<https://github.com/pyppeteer/pyppeteer>) is a Python port of the Node library Puppeteer, which provides a high-level API over the Chrome DevTools Protocol. it's like you are running a browser in your code that can do similar things that your browser can do. Puppeteer can be used to scrap data from websites, take screenshots for a website, and much more. Let's see how we can utilize pyppeteer to generate PDFs from HTML.

First, we need to install pyppeteer with the following command:

```
pip install pyppeteer
```

Copy

Generate PDF from a website URL

```
import asyncio
from pyppeteer import launch

async def generate_pdf(url, pdf_path):
    browser = await launch()
    page = await browser.newPage()

    await page.goto(url)

    await page.pdf({'path': pdf_path, 'format': 'A4'})

    await browser.close()

# Run the function
asyncio.get_event_loop().run_until_complete(generate_pdf('https://example.
```

Copy

In the above code, if you see the `generate_pdf` method, we are doing the following things

1. Launching a new headless browser instance
2. Opens a new tab or page in the headless browser and waits for it to be ready.
3. Navigate to the URL specified in the `url` argument and wait for the page to load.
4. Generates a PDF of the webpage. The PDF is saved at the location specified in `pdf_path`, and the format is set to `A4`.
5. Closes the headless browser.

Generate PDF from Custom HTML content

```
import asyncio
from pyppeteer import launch

async def generate_pdf_from_html(html_content, pdf_path):
    browser = await launch()
    page = await browser.newPage()

    await page.setContent(html_content)

    await page.pdf({'path': pdf_path, 'format': 'A4'})

    await browser.close()

# HTML content
html_content = '''
<!DOCTYPE html>
<html>
<head>
  <title>PDF Example</title>
</head>
<body>
  <h1>Hello World</h1>
```

We use cookies on our website to give you the most relevant experience by remembering your preferences and repeat visits. By clicking "Accept All", you consent to the use of ALL the cookies

To get more information about these cookies and the processing of your personal data, check our [Privacy Policy](https://apitemplate.io/privacy-policy/) (<https://apitemplate.io/privacy-policy/>).

Accept All

Above is another example using Pyppeteer on how we can use our own custom HTML content to generate PDFs. Let's see what is happening in the method `generate_pdf_from_html`

1. Launching a new headless browser instance
2. Opens a new tab or page in the headless browser and waits for it to be ready.
3. Now we are explicitly setting the content of the page to our HTML content
4. Generates a PDF of the webpage. The PDF is saved at the location specified in `pdf_path`, and the format is set to `'A4'`.

5. Closes the headless browser.

ii. xhtml2pdf

xhtml2pdf (<https://xhtml2pdf.readthedocs.io/en/latest/>) is another Python library that lets you generate PDFs from HTML content. Let's see xhtml2pdf in action.

The following command is to install xhtml2pdf:

```
pip install xhtml2pdf requests
```

Copy

To generate PDF from a website URL

Note that xhtml2pdf does not have an in-built feature to parse the URL, but we can use requests in Python to get the content from a URL.

```
from xhtml2pdf import pisa
import requests

def convert_url_to_pdf(url, pdf_path):
    # Fetch the HTML content from the URL
    response = requests.get(url)
    if response.status_code != 200:
        print(f"Failed to fetch URL: {url}")
        return False

    html_content = response.text

    # Generate PDF
    with open(pdf_path, "wb") as pdf_file:
        pisa_status = pisa.CreatePDF(html_content, dest=pdf_file)

    return not pisa_status.err

# URL to fetch
url_to_fetch = "https://google.com"

# PDF path to save
```

Copy

We use cookies on our website to give you the most relevant experience by remembering your preferences and repeat visits. By clicking "Accept All", you consent to the use of ALL the cookies

To get more information about these cookies and the processing of your personal data, check our [Privacy Policy](https://apitemplate.io/privacy-policy/) (<https://apitemplate.io/privacy-policy/>).

In the above code, we are doing the following things in our method

convert_url_to_pdf

1. First, we are using `requests` to get the webpage content from the URL. Accept All
2. Once we get the content, we select the text part from the response using `response.text`
3. Now the generating PDF part comes, we are using `pisa.CreatePDF` and pass our HTML content and PDF file name for the output.

Generating PDF from custom HTML content

```

from xhtml2pdf import pisa

def convert_html_to_pdf(html_string, pdf_path):
    with open(pdf_path, "wb") as pdf_file:
        pisa_status = pisa.CreatePDF(html_string, dest=pdf_file)

    return not pisa_status.err

# HTML content
html_content = '''
<!DOCTYPE html>
<html>
<head>
    <title>PDF Example</title>
</head>
<body>
    <h1>Hello, world!</h1>
</body>
</html>
'''

# Generate PDF

```

Copy

Generating PDF from custom HTML content is also similar to what we have done for the URL part, the only change here is, that we are passing the actual HTML content to our generating method. Now it will use our custom HTML content and generate PDF from it.

iii. python-pdfkit

python-pdfkit (<https://pypi.org/project/pdfkit/>) is a python wrapper for *wkhtmltopdf* utility to convert HTML to PDF using Webkit.

First, we need to install python-pdfkit with pip:

```
pip install pdfkit
```

We use cookies on our website to give you the most relevant experience by remembering your preferences and repeat visits. By clicking “Accept All”, you consent to the use of ALL the cookies

Copy

To get more information about these cookies and the processing of your personal data, check our [Privacy Policy](https://apitemplate.io/privacy-policy/) (<https://apitemplate.io/privacy-policy/>).

To generate PDF from website URL

```

import pdfkit

def convert_url_to_pdf(url, pdf_path):
    try:
        pdfkit.from_url(url, pdf_path)
        print(f"PDF generated and saved at {pdf_path}")
    except Exception as e:
        print(f"PDF generation failed: {e}")

# URL to fetch
url_to_fetch = 'https://example.com'

# PDF path to save
pdf_path = 'example_from_url.pdf'

# Generate PDF
convert_url_to_pdf(url_to_fetch, pdf_path)

```

Copy

Accept All

pdfkit supports generating PDFs from website URLs out of the box just like Pyppeteer.

In the above code, as you can see, **pdfkit** is generating pdf just from one line code. `pdfkit.from_url` is all you need to generate a PDF.

Generating PDF from custom HTML content

```
import pdfkit

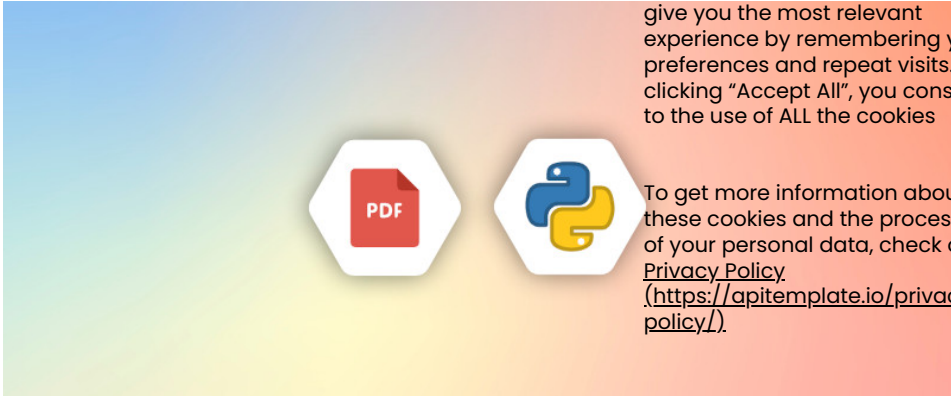
def convert_html_to_pdf(html_content, pdf_path):
    try:
        pdfkit.from_string(html_content, pdf_path)
        print(f"PDF generated and saved at {pdf_path}")
    except Exception as e:
        print(f"PDF generation failed: {e}")

# HTML content
html_content = '''
<!DOCTYPE html>
<html>
<head>
    <title>PDF Example</title>
</head>
<body>
    <h1>Hello, world!</h1>
</body>
</html>
'''
```

Copy

For generating PDF from custom HTML content, we only need to use `pdfkit.from_string` and provide HTML content and a pdf file path.

We find out more about python-pdfkit, we have a detailed article in the following link:




We use cookies on our website to give you the most relevant experience by remembering your preferences and repeat visits. By clicking "Accept All", you consent to the use of ALL the cookies

To get more information about these cookies and the processing of your personal data, check our [Privacy Policy \(https://apitemplate.io/privacy-policy/\)](https://apitemplate.io/privacy-policy/).

Accept All

How to Generate PDFs from HTML with Python-PDFKit (HTML string, HTML File and URL)

In this article, we will introduce another tool that can be used to automate PDF generation from HTML in Python. PDFKit is another Python library that can be used to convert HTML pages into PDFs. It wraps the wkhtmltopdf command line tool. PDFKit is easy to use and can handle complex PDF generation tasks like tables and multi-page documents.

 [API Template.io](https://apitemplate.io)

iv. Playwright

Playwright is a modern and lightweight library for using a headless browser in your application. Playwright is primarily used in automation testing with its powerful offering of integrations with modern browsers. Currently, Playwright supports Firefox, Chromium, Edge & Safari. Playwright is Cross-platform, cross-browser, and cross-language.

In this article, we will look into how we can convert HTML to PDF in Python using Playwright without losing formatting and quality.

the first step is to install the Playwright library

```
pip install playwright
playwright install
```

Copy

playwright install command will make sure to install a headless browser in your system which will be used to convert HTML to PDF in Python.

Generate PDF from website URL

```
import asyncio
from playwright.async_api import async_playwright

async def url_to_pdf(url, output_path):
    async with async_playwright() as p:
        browser = await p.chromium.launch()
        page = await browser.new_page()
        await page.goto(url)
        await page.pdf(path=output_path)
        await browser.close()

# Example usage
url = 'https://google.com'
output_path = 'html-to-pdf-output.pdf'

asyncio.run(url_to_pdf(url, output_path))
```

Copy

In the above code, we have **url_to_pdf** method which takes the URL of the website and output path as input parameters. It creates an async function to run a headless browser.

we use **chromium.launch()** to launch a new instance of the browser and create a new page with **browser.new_page()**

once we have a new page ready, we then load our URL which will be the source for HTML to PDF in this case. Once everything is done, we call **browser.close()** to close the browser instance.

We use cookies on our website to give you the most relevant experience by remembering your preferences and repeat visits. By clicking "Accept All", you consent to the use of All the cookies.

To get more information about these cookies and the processing of your personal data, check our [Privacy Policy](https://apitemplate.io/privacy-policy/) (<https://apitemplate.io/privacy-policy/>).

Accept All

Generate PDF from custom HTML content

We can also use Playwright to generate custom HTML to PDF along with directly loading HTML content from the website URL. To generate HTML to PDF from your custom HTML, we will do the following

```
import asyncio
from playwright.async_api import async_playwright

async def html_to_pdf(html_content, output_path):
    async with async_playwright() as p:
        browser = await p.chromium.launch()
        page = await browser.new_page()
        await page.set_content(html_content)
        await page.pdf(path=output_path)
        await browser.close()

html_content = '''
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sample HTML</title>
</head>
<body>
    <h1>Hello, World!</h1>
    <p>This is a sample HTML content to be converted to PDF.</p>
</body>
</html>'''
```

In the above code snippet, we are passing our custom HTML content to **html_to_pdf** method. We use Playwright to load that custom HTML in a new tab of the headless browser. and then generate PDFs using our custom HTML.

v. WeasyPrint

WeasyPrint (<https://weasyprint.org/>) is a visual rendering engine that converts HTML and CSS into PDFs, focusing on adhering to web standards for printing. It is freely available under a BSD license.

Unlike some other libraries which rely on browsers such as Chrome or Firefox, it is built on several libraries but does not use a full rendering engine.

In this section, we will look into how we can use WeasyPrint to convert HTML to PDF using Python.

Install WeasyPrint

```
pip install WeasyPrint
```

Generate PDF from website URL

To generate a PDF from the website URL, we will use **.HTML()** method given by the WeasyPrint library.

```
from weasyprint import HTML

def url_to_pdf(url, output_path):
    HTML(url).write_pdf(output_path)

# Example usage
url = 'https://google.com'
output_path = 'output_url.pdf'

url_to_pdf(url, output_path)
```


In the above code snippet, we are using the WeasyPrint .write_pdf() method which converts the Website URL to PDF in a simple step.

While using WeasyPrint, we don't need to worry about setting up any other things and we can just call the method and get PDFs directly from the website URL.

Generate PDF from custom HTML content

```
from weasyprint import HTML

def html_to_pdf(html_content, output_path):
    HTML(string=html_content).write_pdf(output_path)

html_content = '''
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sample HTML</title>
</head>
<body>
    <h1>Hello, World!</h1>
    <p>This is a sample HTML content to be converted to PDF.</p>
</body>
</html>
'''

output_path = 'output_html.pdf'

html_to_pdf(html_content, output_path)
```

Copy

Same as generating PDFs from website URL, we can use `.HTML()` method to load the custom HTML and then use `.write_pdf()` method to create PDF from our custom HTML.

We use cookies on our website to give you the most relevant experience by remembering your preferences and repeat visits. By clicking "Accept All", you consent to the use of ALL the cookies

To get more information about these cookies and the processing of your personal data, check our [Privacy Policy](https://apitemplate.io/privacy-policy/) (https://apitemplate.io/privacy-policy/).

Comparison of All 5 Popular Libraries

While all of these tools serve the primary purpose of HTML to PDF conversion, each brings its unique features and approaches to the table.

Below is a detailed tabular comparison to help you choose the right tool for your needs.

Feature/Aspect	Pyppeteer	xhtml2pdf	python-pdfkit	Playwright	WeasyPrint
Nature	Browser automation library	HTML/CSS to PDF converter	Wrapper around wkhtmltopdf	Lightweight and Modern Browser automation library	Document factory to create PDF documents easily

Feature/Aspect	Pyppeteer	xhtml2pdf	python-pdfkit	Playwright	WeasyPrint
Based On	Puppeteer (Chromium headless)	ReportLab & html5lib	wkhtmltopdf	Playwright (supports Firefox, Chromium, Edge & Safari)	Written in Python and based on various libraries and not a full rendering engine
Dependencies	Requires Chrome/Chromium	Python libraries	Requires wkhtmltopdf	Cross-browser support	Python libraries
Language	Python	Python	Python	Python	Python
Javascript Support	Yes (full browser environment)	No	Yes (limited, via wkhtmltopdf)	Yes (full browser environment)	No
CSS Support	Full (as in Chrome)	Limited	Good (via wkhtmltopdf)	Full (as in browser)	Yes
Performance	May be slower (full browser)	Moderate	Fast (native conversion)	Fast	Fast
Ease of Setup	Moderate (need Chromium)	Easy (pure Python)	Moderate (requires wkhtmltopdf)	Easy to setup	Easy to Setup
API Flexibility	High (full browser automation)	Moderate (focused on PDF)	Moderate (wrapper around tool)	One API for cross-browser support	Simple Python API
Usage	Good for complex web content, SPA, dynamic JS content	Good for simpler HTML/CSS docs	Common for various HTML to PDF tasks	Good for complex automation testing with support of all modern browser	Good for creating PDF documents with rich styling

In conclusion, the choice between Pyppeteer, Playwright, xhtml2pdf, WeasyPrint, and python-pdfkit depends on the specific requirements of your project.

While Pyppeteer excels at handling dynamic content due to its full browser automation capabilities, xhtml2pdf offers a straightforward, Python-centric solution for basic conversions.

Playwright is a serious alternative to Pyppeteer as Playwright solves the problems that developers face while using Pupeteer. With the Playwright you get to experience a lightweight and fast library that has support for all the modern browsers.

WeasyPrint is a developer-friendly Python library that does not let you sweat over configurations and setup. One thing to take note is that WeasyPrint is based on various libraries and not a full rendering engine like WebKit or Gecko, so it doesn't support Javasacript.

Python-pdfkit, wrapping around wkhtmltopdf, stands as a versatile middle ground. Developers should weigh the features, setup complexities, and performance of each library against their project's demands to determine the best fit.

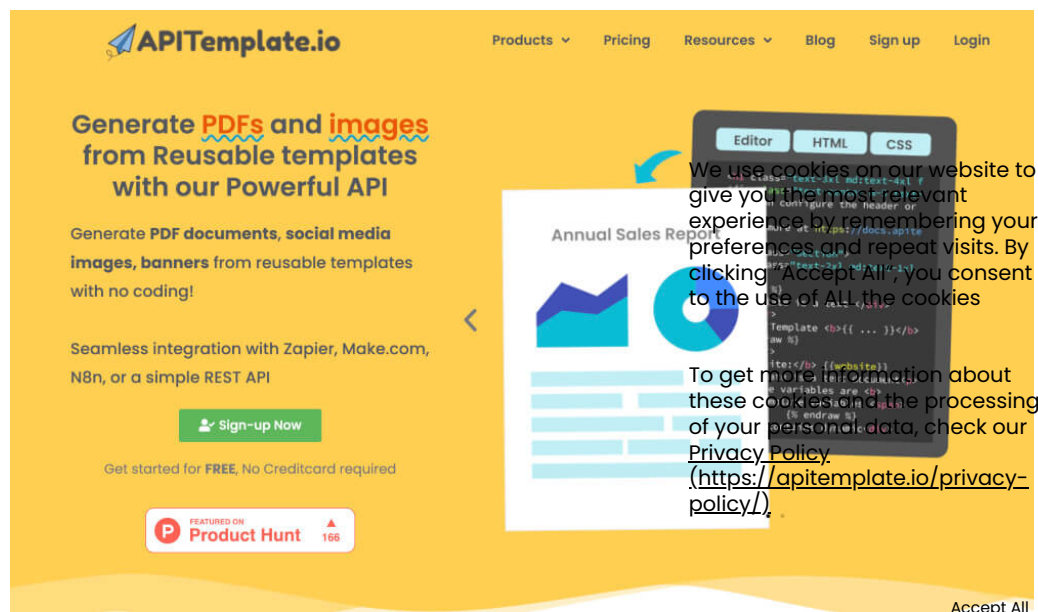
In summary, if you need to render complex HTML, CSS, and JavaScript with full browser compatibility, consider using Pyppeteer, Playwright, or python-pdfkit. For simpler tasks, xhtml2pdf and WeasyPrint are more suitable options.

HTML to PDF using APITemplate.io

Above are some examples of how we can use libraries to convert HTML to PDF and web pages to PDF. but when it comes to generating PDFs using templates or keeping track of generated PDFs, we need to do a lot of extra things to handle all those.

We need to have our own generating pdfs tracker for tracking the files generated. Or if we want to use custom templates such as Invoice generators, we need to create and manage those templates.

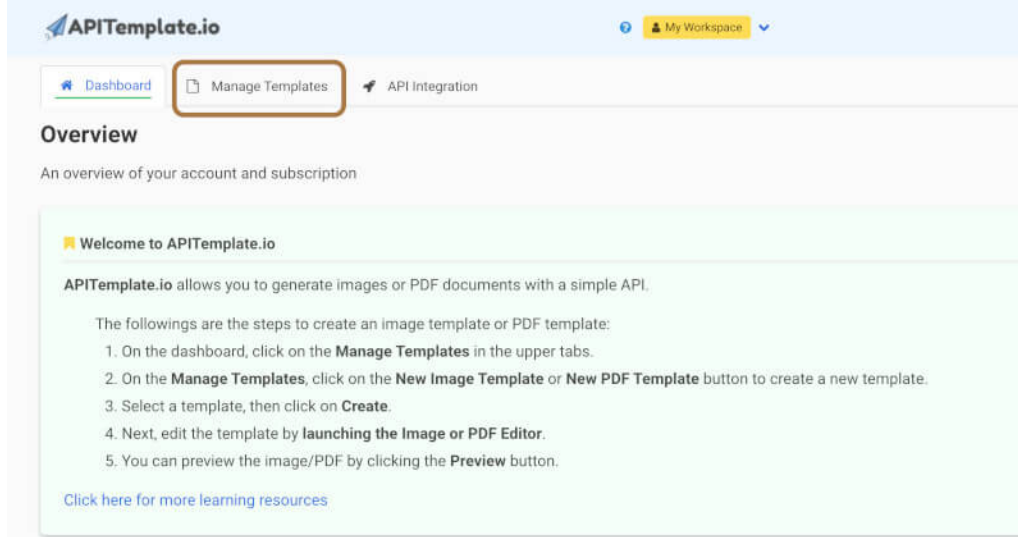
APITemplate.io (<https://apitemplate.io>) is an API-based platform for PDF generation, perfect for the use cases mentioned. Our PDF generation API (<https://apitemplate.io/pdf-generation-api/>) uses a Chromium-based rendering engine, which fully supports JavaScript, CSS, and HTML.



Let's see how we can utilize APITemplate.io (<https://apitemplate.io/>) to handle generating PDFs

i. Template-based PDF generation

APITemplate.io (<http://ApiTemplate.io>) allows you to manage your templates. Go to Manage Templates from the dashboard



From Manage Template, You can create your own templates. Following is the sample Invoice template. There are lots of templates available that you can choose and customize based on your requirements.



We use cookies on our website to give you the most relevant experience by remembering your preferences and repeat visits. By clicking “Accept All”, you consent to the use of ALL the cookies

To start using APITemplate.io (<http://APITemplate.io>) API Key which you can get from the API Integration Tab. To get more information about these cookies and how we process your personal data, check our [Privacy Policy](https://apitemplate.io/privacy-policy/) (<https://apitemplate.io/privacy-policy/>).

Accept All

Now that you have your APITemplate account ready, let's get to some actions and integrate it with our application. We will be using the template to generate PDFs.

```
import requests
import json

# Initialize HTTP client
client = requests.Session()

# API URL
url = "https://rest.apitemplate.io/v2/create-pdf?template_id=YOUR_TEMPLATE_ID"

# Payload data
payload = {
    "date": "15/05/2022",
    "invoice_no": "435568799",
    "sender_address1": "3244 Jurong Drive",
    "sender_address2": "Falmouth Maine 1703",
    "sender_phone": "255-781-6789",
    "sender_email": "hello@logos.com",
    "rece_address1": "2354 Lakeside Drive",
    "rece_address2": "New York 234562",
    "rece_phone": "34333-84-223",
    "rece_email": "business@apitemplate.io",
    "item": {

```

and If we check `response_string` we have the following

```
{
  "download_url": "PDF_URL",
  "transaction_ref": "8cd2aced-b2a2-40fb-bd45-392c777d6f6",
  "status": "success",
  "template_id": "YOUR_TEMPLATE_ID"
}
```

In the above code, it's very easy to use APITemplate to convert html to pdf because we don't need to install any other library. Just need to call the simple API and use our data as a request body and that's it!

You can use the `download_url` from the response to download or distribute the generated PDF.

We use cookies on our website to give you the most relevant experience by remembering your preferences and repeat visits. By clicking "Accept All", you consent to the use of ALL the cookies.

To get more information about these cookies and the processing of your personal data, check our [Privacy Policy](https://apitemplate.io/privacy-policy/) (<https://apitemplate.io/privacy-policy/>).

ii. Generate PDF from the website URL

APITemplate also supports generating PDFs from website URLs.

```
import requests, json

def main():
    api_key = "YOUR_API_KEY"
    template_id = "YOUR_TEMPLATE_ID"

    data = {
        "url": "https://en.wikipedia.org/wiki/Sceloporus_malachiticus",
        "settings": {
            "paper_size": "A4",
            "orientation": "1",
            "header_font_size": "9px",
            "margin_top": "40",
            "margin_right": "10",
            "margin_bottom": "40",
            "margin_left": "10",
            "print_background": "1",
            "displayHeaderFooter": true,
            "custom_header": "<style>#header, #footer { padding: 0 !important;
            "custom_footer": "<style>#header, #footer { padding: 0 !important;
        }
    }
```

In the above code, we can provide the URL in the request body along with the settings for the PDF. APITemplate will use this request body to generate a PDF and will return a download URL for your PDF.

iii. Generate PDF from custom HTML content

If you want to generate PDFs using your own custom HTML content, APITemplate supports that as well.

```
import requests, json

def main():
    api_key = "YOUR_API_KEY"
    template_id = "YOUR_TEMPLATE_ID"

    data = {
        "body": "<h1> hello world {{name}} </h1>",
        "css": "<style>.bg{background: red};</style>",
        "data": {
            "name": "This is a title"
        },
        "settings": {
            "paper_size": "A4",
            "orientation": "1",
            "header_font_size": "9px",
            "margin_top": "40",
            "margin_right": "10",
            "margin_bottom": "40",
            "margin_left": "10",
            "print_background": "1",
            "displayHeaderFooter": true
        }
    }
```

Similar to generating PDF from a website URL, the above API request takes body and CSS as part of the payload and it is to generate PDF.

We use cookies on our website to give you the most relevant experience by remembering your preferences and repeat visits. By clicking “Accept All”, you consent to the use of ALL the cookies

Conclusion

PDF generation is now part of every business application and it should not make developers sweat.

To get more information about these cookies and the processing of your personal data, check our [Privacy Policy \(https://apitemplate.io/privacy-policy/\)](https://apitemplate.io/privacy-policy/).

We have seen how we can use third-party libraries to generate PDFs if our use case is simple. But also if we have complex use cases such as maintaining templates, then APITemplate.io (<https://apitemplate.io/>) provides the solution just for that using simple API calls.

Sign up for a free account (<https://app.apitemplate.io/accounts/signup/>) with us now and start automating your PDF generation. (<https://apitemplate.io/pdf-generation-api/>)

Libraries:

- REST API Reference (<https://apitemplate.io/apiv2/>)
- Javascript Client Library (<https://github.com/APITemplate-io/apitemplateio-javascript>)
- Python Client Library (<https://github.com/APITemplate-io/apitemplateio-python>)
- Java Client Library (<https://github.com/APITemplate-io/apitemplateio-java>)
- PHP Client Library (<https://github.com/APITemplate-io/apitemplateio-php>)
- C# Client Library (<https://github.com/APITemplate-io/apitemplateio-csharp>)



Manoj Ahirwar

Hi, I'm Manoj. I am building SaaS products. I love writing about my journey as well as writing on technical stuff. I am also trying to travel to as many countries as I can :)

Share:



Facebook



Twitter



Pinterest



LinkedIn

◀ (https://apitemplate.io/blog/a-guide-to-generate-pdfs-in-p) ▶(https://apitemplate.io/blog/whats-the-dpi-value-of-a-pdf/)

(/category/blog/image/)

Image Generation Tutorials
(/category/blog/image/)

(/category/blog/pdf/)

PDF Generation Tutorials
(/category/blog/pdf/)

(/category/blog/no-code-automation/)

No-code/Automation Tutorials
(/category/blog/no-code-automation/)

Articles for Image Generation



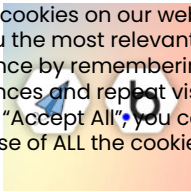
How to leverage APITemplate for effective marketing (https://apitemplate.io/blog/how-to-leverage-apitemplate-for-effective-marketing/)

Ayeesha

Read More » (https://apitemplate.io/blog/how-to-leverage-apitemplate-for-effective-marketing/)

(https://apitemplate.io/blog/how-to-leverage-apitemplate-for-effective-marketing/)

We use cookies on our website to give you the most relevant experience by remembering your preferences and repeat visits. By clicking "Accept All", you consent to the use of ALL the cookies.



How to generate PDF documents and Images with Bubble.io (https://apitemplate.io/blog/how-to-generate-pdf-documents-and-images-with-bubble-io/)

How to generate PDF documents and Images with Bubble.io (https://apitemplate.io/blog/how-to-generate-pdf-documents-and-images-with-bubble-io/)

Imran, Alam

(https://apitemplate.io/blog/how-to-generate-pdf-documents-and-images-with-bubble-io/)

To get more information about these cookies and the processing of your personal data, check our Privacy Policy (https://apitemplate.io/privacy-policy/).

with-bubble-io/)

Read More » (https://apitemplate.io/blog/how-to-generate-pdf-documents-and-images-with-bubble-io/)



Google Sheets

5 Easy Steps to Automate Image Generation API – FREE with Zapier, Google sheets, and APITemplate.io (https://apitemplate.io/blog/5-easy-steps-to-automate-image-generation-api-free-with-zapier-google-sheets-and-apitemplate-io/)

APITemplate.io

Read More » (https://apitemplate.io/blog/5-easy-steps-to-automate-image-generation-api-free-with-zapier-google-sheets-and-apitemplate-io/)

(https://apitemplate.io/blog/5-easy-steps-to-automate-image-generation-api-free-with-zapier-google-sheets-and-apitemplate-io/)



Accept All

Push-button Imagery for Community Good (https://apitemplate.io/blog/push-button-imagery-for-community-good/)

APITemplate.io

(https://apitemplate.io/blog/push-button-imagery-for-community-good/)

Read More » (https://apitemplate.io/blog/push-button-imagery-for-community-good/)



Why Marketing Automation Is Important to You and Its Benefits
(<https://apitemplate.io/blog/why-marketing-automation-is-important-to-you-and-its-benefits/>)

Jacky, Founder

Read More » (<https://apitemplate.io/blog/why-marketing-automation-is-important-to-you-and-its-benefits/>)



How to automatically generate social media images from your Shopify product list
(<https://apitemplate.io/blog/how-to-automatically-generate-social-media-images-from-your-shopify-product-list/>)

Jacky, Founder

Read More » (<https://apitemplate.io/blog/how-to-automatically-generate-social-media-images-from-your-shopify-product-list/>)



Instagram Automation: Create a Beautiful Grid Layout on Your Instagram
(<https://apitemplate.io/blog/instagram-automation-create-a-beautiful-grid-layout-on-your-instagram/>)

Jacky, Founder

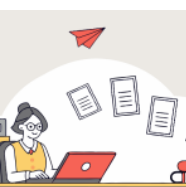
Read More » (<https://apitemplate.io/blog/instagram-automation-create-a-beautiful-grid-layout-on-your-instagram/>)



How to Turn Markdown into PDF Documents with a REST API (<https://apitemplate.io/blog/how-to-turn-markdown-into-pdfs/>)

APITemplate.io

Read More » (<https://apitemplate.io/blog/how-to-turn-markdown-into-pdfs/>)



Convert HTML to PDF for eCommerce Receipts, Labels and Invoices Using APITemplate.io
(<https://apitemplate.io/blog/convert-html-to-pdf-for-ecommerce-receipts-labels-and-invoices/>)



How to Automate Job Posting Images for LinkedIn
(<https://apitemplate.io/blog/how-to-automate-job-posting-images-for-linkedin/>)

Jacky, Founder

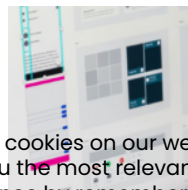
Read More » (<https://apitemplate.io/blog/how-to-automate-job-posting-images-for-linkedin/>)



How to auto-share user feedback on social media with Zapier, Startquestion and APITemplate.io
(<https://apitemplate.io/blog/how-to-auto-share-user-feedback-on-social-media-with-zapier-startquestion-and-apitemplateio/>)

Jacky, Founder

Read More » (<https://apitemplate.io/blog/how-to-auto-share-user-feedback-on-social-media-with-zapier-startquestion-and-apitemplateio/>)



Create a Responsive Image Template using Auto-position
(<https://apitemplate.io/blog/create-a-responsive-image-template-using-auto-position/>)

Jacky, Founder

Read More » (<https://apitemplate.io/blog/create-a-responsive-image-template-using-auto-position/>)

We use cookies on our website to give you the most relevant experience by remembering your preferences and repeat visits. By clicking "Accept", you consent to the use of ALL the cookies.

[To get more information about these cookies and the processing of your personal data, check our Privacy Policy.](#)
(<https://apitemplate.io/privacy-policy/>).

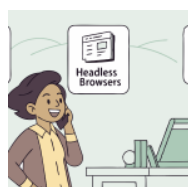
Articles for PDF Generation




























Write Prompts, Not Code: Generating PDFs from HTML via Vibe Coding and APITemplate.io
(<https://apitemplate.io/blog/write-prompts-not-code-generating-pdfs-from-html-via-vibe-coding/>)

APITemplate.io

Read More » (<https://apitemplate.io/blog/write-prompts-not-code-generating-pdfs-from-html-via-vibe-coding/>)



HTML to PDF Conversion Methods: Pros and Cons of Libraries, Headless Browsers, and APIs
(<https://apitemplate.io/blog/html-to-pdf-conversion-methods-pros-and-cons-of-libraries-headless-browsers-and-apis/>)

Registration	Products	Integrations	PDF Tools	PDF Generation Tutorials
 Sig (https://app.apitemplate.io/accounts/signup-gnup/)	PDF Generation API	 Zapier (https://apitemplate.io/pdf-generation-api/)	 Zapier (https://apitemplate.io/integrations/generate-pdf-with-zapier-and-apitemplateio/)	 Zapier (https://docs.apitemplate.io/pdf-with-zapier-and-apitemplateio/)
 Lo (https://app.apitemplate.io/accounts/login/)	Image Generation API	 Bubble.io (https://apitemplate.io/image-generation-api/)	 Bubble.io (https://apitemplate.io/integrations/generate-pdf-and-image-with-bubbleio/)	 Bubble.io (https://apitemplate.io/how-to-generate-pdf-with-bubbleio/)
	Generating PDFs from HTML	 Airtable (https://apitemplate.io/integrations/airtable-generate-images-in-bulk/)	 Airtable (https://apitemplate.io/tools/convert-markdown-to-pdf/)	 Airtable (https://apitemplate.io/charts-into-a-pdf/)
 Ho (https://apitemplate.io/)	Creating PDFs from URL	 Reference v2 (https://apitemplate.io/api/v2/)	 Reference v2 (https://apitemplate.io/pdf-extract/)	 Reference v2 (https://docs.apitemplate.io/language/jinja2.html)
 Pric (https://apitemplate.io/pricing/)	Enterprise Features	 Reference v1 (https://docs.apitemplate.io/reference/api-reference.html)	 Reference v1 (https://docs.apitemplate.io/pdf-from-text/)	 Reference v1 (https://apitemplate.io/blog/add-header-and-footer-to-add-a-logo-to-your-pdf-header-or-footer/)
 FA (https://apitemplate.io/faqs/)				
 About Us (/about-us/)				
 Service Status (https://status.apitemplate.io/)				
	Features	Use-cases		
	 Convert HTML to		 Add Watermark to PDF	 Footer to PDF

