**PDF file** format is very useful to download bulk data in the web application. It helps the user to download dynamic content in file format for offline use. With export to PDF functionality, the HTML content is converted to PDF document and downloaded as a PDF file. In the dynamic web application, a server-side script is used to convert HTML to PDF and [generate PDF file using PHP](https://www.codexworld.com/convert-html-to-pdf-php-dompdf/).

If you want a client-side solution to generate PDF document, JavaScript is the easiest way to **convert HTML to PDF**. There are various JavaScript library is available for generating PDF from HTML. **jsPDF** is one of the best library to convert HTML to PDF using JavaScript. In this tutorial, we will show you how to generate PDF document and **convert HTML to PDF using jQuery** and jsPDF library.

**Include jQuery and jsPDF Library**

Include the jQuery and jsPDF library files to use the jsPDF class.

<!-- jQuery library -->

<script src="js/jquery.min.js"></script>

<!-- jsPDF library -->

<script src="js/jsPDF/dist/jspdf.min.js"></script>

Note that: You don’t need to download the jsPDF library separately, all the required files are included in our source code package.

**Instantiate jsPDF Class**

Use the following line of code to instantiate and use the jsPDF object in JavaScript.

var doc = new jsPDF();

**Generate PDF using JavaScript**

The following example shows how to use the jsPDF library to generate PDF file using JavaScript.

* Specify the content in **text()** method of jsPDF object.
* Use the **addPage()** method to add new page to PDF.
* Use the **save()** method to generate and download PDF file.

var doc = new jsPDF();

doc.text(20, 20, 'Hello world!');

doc.text(20, 30, 'This is client-side Javascript to generate a PDF.');

// Add new page

doc.addPage();

doc.text(20, 20, 'Visit CodexWorld.com');

// Save the PDF

doc.save('document.pdf');

**Convert HTML Content to PDF using JavaScript**

The following example shows how to use the jsPDF library to **convert HTML to PDF** and generate PDF file from HTML content using JavaScript.

* Retrieve the HTML content from the specific element by ID or class.
* Convert HTML content of the specific part of the web page and generate PDF.
* Save and download the HTML content as a PDF file.

**HTML Code:**

<div id="content">

<!-- HTML contnet goes here -->

</div>

<div id="elementH"></div>

**JavaScript Code:**

var doc = new jsPDF();

var elementHTML = $('#contnet').html();

var specialElementHandlers = {

'#elementH': function (element, renderer) {

return true;

}

};

doc.fromHTML(elementHTML, 15, 15, {

'width': 170,

'elementHandlers': specialElementHandlers

});

// Save the PDF

doc.save('sample-document.pdf');

**Useful Configurations**

The jsPDF library provides various methods and options to configure the PDF creation. Some of the useful methods of jsPDF class are given below that are commonly used to export HTML to PDF using jQuery.

**Change Paper Orientation:**  
Use the orientation option to set the paper orientation of the PDF.

var doc = new jsPDF({

orientation: 'landscape'

});

doc.text(20, 20, 'Hello world!');

doc.text(20, 30, 'This is client-side Javascript to generate a PDF.');

// Add new page

doc.addPage();

doc.text(20, 20, 'Visit CodexWorld.com');

// Save the PDF

doc.save('document.pdf');

**Change Text Font:**  
Use setFont() and setFontType() methods to set text font and font-style in the PDF.

var doc = new jsPDF();

doc.text(20, 20, 'This is the default font.');

doc.setFont("courier");

doc.setFontType("normal");

doc.text(20, 30, 'This is courier normal.');

doc.setFont("times");

doc.setFontType("italic");

doc.text(20, 40, 'This is times italic.');

doc.setFont("helvetica");

doc.setFontType("bold");

doc.text(20, 50, 'This is helvetica bold.');

doc.setFont("courier");

doc.setFontType("bolditalic");

doc.text(20, 60, 'This is courier bolditalic.');

// Save the PDF

doc.save('document.pdf');

**Change Font Size:**  
Use setFontSize() method to set font size of the text in the PDF.

var doc = new jsPDF();

doc.setFontSize(24);

doc.text(20, 20, 'This is a title');

doc.setFontSize(16);

doc.text(20, 30, 'This is some normal sized text underneath.');

// Save the PDF

doc.save('document.pdf');

**Change Text Color:**  
Use setTextColor() method to set the color of the text in the PDF.

var doc = new jsPDF();

doc.setTextColor(100);

doc.text(20, 20, 'This is gray.');

doc.setTextColor(150);

doc.text(20, 30, 'This is light gray.');

doc.setTextColor(255,0,0);

doc.text(20, 40, 'This is red.');

doc.setTextColor(0,255,0);

doc.text(20, 50, 'This is green.');

doc.setTextColor(0,0,255);

doc.text(20, 60, 'This is blue.');

// Save the PDF

doc.save('document.pdf');

[**Create PDF with Watermark in PHP using Dompdf**](https://www.codexworld.com/create-pdf-with-watermark-in-php-using-dompdf/)

**Conclusion**

Our example code helps you to convert HTML to PDF and generate PDF file using JavaScript. You can easily add the **Export to PDF** functionality on the web page without depending on the server-side script. The PDF creation functionality can be enhanced with jsPDF configuration options as per your needs. Download our source code package to get all the required files including the jsPDF JavaScript library.

Stack overflow link : https://stackoverflow.com/questions/18191893/generate-pdf-from-html-in-div-using-javascript

**jsPDF is able to use plugins.** In order to enable it to print HTML, you have to include certain plugins and therefore have to do the following:

1. Go to <https://github.com/MrRio/jsPDF> and download the latest Version.
2. Include the following Scripts in your project:
   * jspdf.js
   * jspdf.plugin.from\_html.js
   * jspdf.plugin.split\_text\_to\_size.js
   * jspdf.plugin.standard\_fonts\_metrics.js

If you want to ignore certain elements, you have to mark them with an ID, which you can then ignore in a special element handler of jsPDF. Therefore your HTML should look like this:

<!DOCTYPE html>

<html>

<body>

<p id="ignorePDF">don't print this to pdf</p>

<div>

<p><font size="3" color="red">print this to pdf</font></p>

</div>

</body>

</html>

Then you use the following JavaScript code to open the created PDF in a PopUp:

var doc = new jsPDF();

var elementHandler = {

'#ignorePDF': function (element, renderer) {

return true;

}

};

var source = window.document.getElementsByTagName("body")[0];

doc.fromHTML(

source,

15,

15,

{

'width': 180,'elementHandlers': elementHandler

});

doc.output("dataurlnewwindow");

For me this created a nice and tidy PDF that only included the line 'print this to pdf'.

Please note that the special element handlers only deal with IDs in the current version, which is also stated in a [GitHub Issue](https://github.com/MrRio/jsPDF/issues/34). It states:

Because the matching is done against every element in the node tree, my desire was to make it as fast as possible. In that case, it meant "Only element IDs are matched" The element IDs are still done in jQuery style "#id", but it does not mean that all jQuery selectors are supported.

Therefore replacing '#ignorePDF' with class selectors like '.ignorePDF' did not work for me. Instead you will have to add the same handler for each and every element, which you want to ignore like:

var elementHandler = {

'#ignoreElement': function (element, renderer) {

return true;

},

'#anotherIdToBeIgnored': function (element, renderer) {

return true;

}

};

From the [examples](http://mrrio.github.io/jsPDF/examples/basic.html) it is also stated that it is possible to select tags like 'a' or 'li'. That might be a little bit to unrestrictive for the most usecases though:

We support special element handlers. Register them with jQuery-style ID selector for either ID or node name. ("#iAmID", "div", "span" etc.) There is no support for any other type of selectors (class, of compound) at this time.

**One very important thing to add is that you lose all your style information (CSS). Luckily jsPDF is able to nicely format h1, h2, h3 etc., which was enough for my purposes. Additionally it will only print text within text nodes, which means that it will not print the values of textareas and the like. Example:**

<body>

<ul>

<!-- This is printed as the element contains a textnode -->

<li>Print me!</li>

</ul>

<div>

<!-- This is not printed because jsPDF doesn't deal with the value attribute -->

<input type="textarea" value="Please print me, too!">

</div>

</body>