You don't have any verified emails. We recommend verifying at least one email.
Email verification helps our support team verify ownership if you lose account access and allows you to receive all the notifications you ask for.

gitbrent / PptxGenJS

JavaScript library that creates PowerPoint (pptx) presentations https://gitbrent.github.io/PptxGenJS/

#powerpoint-presentations #powerpoint #powerpoint-utilities #javascript-library #pptx #powerpoint-generation #powerpoint-library #powerpoint-javascript-library
#javascript-powerpoint #powerpoint-charts #node-powerpoint #powerpoint-tables #javascript-powerpoint-charts #node-powerpoint-charts #angular-powerpoint

748 commits | **5** branches | **21** releases | **11** contributors | MIT

Branch: **master** | New pull request | Create new file | Upload files | Find file | Clone or download

GitBrent Updated 2.0 section — Latest commit `a13d690` 5 days ago

dist	Renamed marginPt to margin	5 days ago
examples	Renamed marginPt to margin	5 days ago
libs	Removed FileSaver.js from libs and README	10 months ago
.gitignore	Add series data point colors	5 months ago
CHANGELOG.md	Updated 2.0 section	5 days ago
LICENSE	20160318	2 years ago
README.md	Updated 2.0 section	5 days ago
bower.json	Updates for v1.9.0	3 months ago
gulpfile.js	New gulp build script (Issue #107)	7 months ago

package.json Updated version 2 months ago

README.md

PptxGenJS

JavaScript library that produces PowerPoint (pptx) presentations

Quickly and easily create PowerPoint presentations with a few simple JavaScript commands in client web browsers or Node desktop apps.

Main Features

- Widely Supported: Creates and downloads presentations on all current web browsers (Chrome, Edge, Firefox, etc.) and IE11
- Full Featured: Slides can include Charts, Images, Media, Shapes, Tables and Text (plus Master Slides/Templates)
- Easy To Use: Entire PowerPoint presentations can be created in a few lines of code
- Modern: Pure JavaScript solution - everything necessary to create PowerPoint PPT exports is included

Additional Features

- Use the unique Table-to-Slides feature to copy an HTML table into 1 or more Slides with a single command

Table of Contents (*generated with DocToc*)

Live Demo

Use JavaScript to Create a PowerPoint presentation with your web browser right now: https://gitbrent.github.io/PptxGenJS

Installation

Client-Side

Include Local Scripts

```
<script lang="javascript" src="PptxGenJS/libs/jquery.min.js"></script>
<script lang="javascript" src="PptxGenJS/libs/jszip.min.js"></script>
<script lang="javascript" src="PptxGenJS/dist/pptxgen.js"></script>
```

IE11 support requires a Promises polyfill as well (included in the libs folder)

Include Bundled Script

```
<script lang="javascript" src="PptxGenJS/dist/pptxgen.bundle.js"></script>
```

Install With Bower

```
bower install pptxgen
```

Node.js

PptxGenJS NPM Homepage

```
npm install pptxgenjs

var pptx = require("pptxgenjs");
```

- Desktop: Compatible with Electron applications!

Presentations: Usage and Options

PptxGenJS PowerPoint presentations are created via JavaScript by following 4 basic steps:

1. Create a new Presentation
2. Add a Slide
3. Add one or more objects (Tables, Shapes, Images, Text and Media) to the Slide
4. Save the Presentation

```
var pptx = new PptxGenJS();
var slide = pptx.addNewSlide();
slide.addText('Hello World!', { x:1.5, y:1.5, font_size:18, color:'363636' });
pptx.save('Sample Presentation');
```

That's really all there is to it!

Creating a Presentation

A Presentation is a single `.pptx` file. When creating more than one Presentation, declare the pptx again to start with a new, empty Presentation.

Client Browser:

```
var pptx = new PptxGenJS();
```

Node.js:

```
var pptx = require("pptxgenjs");
```

Presentation Properties

There are several optional properties that can be set:

```
pptx.setAuthor('Brent Ely');
pptx.setCompany('S.T.A.R. Laboratories');
pptx.setRevision('15');
```

```
pptx.setSubject('Annual Report');
pptx.setTitle('PptxGenJS Sample Presentation');
```

Presentation Layouts

Setting the Layout (applies to all Slides in the Presentation):

```
pptx.setLayout('LAYOUT_WIDE');
```

Presentation Layout Options

Layout Name	Default	Layout Slide Size
`LAYOUT_16x9`	Yes	10 x 5.625 inches
`LAYOUT_16x10`	No	10 x 6.25 inches
`LAYOUT_4x3`	No	10 x 7.5 inches
`LAYOUT_WIDE`	No	13.3 x 7.5 inches
`LAYOUT_USER`	No	user defined - see below (inches)

Custom user defined Layout sizes are supported - just supply a `name` and the size in inches.

- Defining a new Layout using an object will also set this new size as the current Presentation Layout

```
// Defines and sets this new layout for the Presentation
pptx.setLayout({ name:'A3', width:16.5, height:11.7 });
```

Presentation Text Direction

Right-to-Left (RTL) text is supported. Simply set the RTL mode at the Presentation-level.

```
pptx.setRTL(true);
```

Adding a Slide

Syntax:

```
var slide = pptx.addNewSlide();
```

Slide Formatting

```
slide.back  = 'F1F1F1';
slide.color = '696969';
```

Slide Formatting Options

Option	Type	Unit	Default	Description	Possible Values
`bkgd`	string		`FFFFFF`	background color	hex color code or scheme color constant.
`color`	string		`000000`	default text color	hex color code or scheme color constant.

Applying Master Slides / Branding

```
// Create a new Slide that will inherit properties from a pre-defined master page (margins, logos, text, background,
var slide1 = pptx.addNewSlide('TITLE_SLIDE');

// The background color can be overridden on a per-slide basis:
var slide2 = pptx.addNewSlide('TITLE_SLIDE', {bkgd:'FFFCCC'});
```

Adding Slide Numbers

```
slide.slideNumber({ x:1.0, y:'90%' });
// Slide Numbers can be styled:
slide.slideNumber({ x:1.0, y:'90%', fontFace:'Courier', fontSize:32, color:'CF0101' });
```

Slide Number Options

Option	Type	Unit	Default	Description	Possible Values
`x`	number	inches	`0.3`	horizontal location	0-n OR 'n%'. (Ex: `{x:'10%'}` places number 10% from left edge)
`y`	number	inches	`90%`	vertical location	0-n OR 'n%'. (Ex: `{y:'90%'}` places number 90% down the Slide)
`color`	string			text color	hex color code or scheme color constant. Ex: `{color:'0088CC'}`
`fontFace`	string			font face	any available font. Ex: `{fontFace:Arial}`
`fontSize`	number	points		font size	8-256. Ex: `{fontSize:12}`

Slide Return Value

The Slide object returns a reference to itself, so calls can be chained.

Example:

```
slide
.addImage({ path:'images/logo1.png', x:1, y:2, w:3, h:3 })
.addImage({ path:'images/logo2.jpg', x:5, y:3, w:3, h:3 })
.addImage({ path:'images/logo3.png', x:9, y:4, w:3, h:3 });
```

Saving a Presentation

Presentations require nothing more than passing a filename to `save()`. Node.js users have more options available examples of which can be found below.

Client Browser

- Simply provide a filename

```
pptx.save('Demo-Media');
```

Node.js

- Node can accept a callback function that will return the filename once the save is complete
- Node can also be used to stream a powerpoint file - simply pass a filename that begins with "http"
- Output type can be specified by passing an optional JSZip output type

```
// Example A: File will be saved to the local working directory (`__dirname`)
pptx.save('Node_Demo');
```

```
// Example B: Inline callback function
pptx.save('Node_Demo', function(filename){ console.log('Created: '+filename); });

// Example C: Predefined callback function
pptx.save('Node_Demo', saveCallback);

// Example D: Use a filename of "http" or "https" to receive the powerpoint binary data in your callback
// (Used for streaming the presentation file via http.  See the `nodejs-demo.js` file for a working example.)
pptx.save('http', streamCallback);

// Example E: Save using various JSZip output types: ['arraybuffer', 'base64', 'binarystring', 'blob', 'nodebuffer',
pptx.save('jszip', saveCallback, 'base64');
```

Saving Multiple Presentations

Client-Side:

- In order to generate a new, unique Presentation just create a new instance of the library then add objects and save as normal.

```
// Presenation 1:
var pptx = new PptxGenJS();
pptx.addNewSlide().addText('Presentation 1', {x:1, y:1});
pptx.save('PptxGenJS-Presentation-1');

// Presenation 2:
pptx = new PptxGenJS();
pptx.addNewSlide().addText('Presentation 2', {x:1, y:1});
pptx.save('PptxGenJS-Presentation-2');
```

Node.js:

- See `examples/nodejs-demo.js` for a working demo with multiple presentations, callbacks, streaming, etc.
- (Note: New 2.0.0 feature that breaks backwards compatibility)

```
PptxGenJS = require("pptxgenjs");

// Presenation 1:
var pptx = new PptxGenJS();
pptx.addNewSlide().addText('Presentation 1', {x:1, y:1});
pptx.save('PptxGenJS-NodePres-1');

// Presenation 2:
var pptx = new PptxGenJS();
pptx.addNewSlide().addText('Presentation 2', {x:1, y:1});
pptx.save('PptxGenJS-NodePres-2');
```

Presentations: Adding Objects

Objects on the Slide are ordered from back-to-front based upon the order they were added.

For example, if you add an Image, then a Shape, then a Textbox: the Textbox will be in front of the Shape, which is in front of the Image.

Adding Charts

```
// Syntax
slide.addChart({TYPE}, {DATA}, {OPTIONS});
```

Chart Types

- Chart type can be any one of `pptx.charts`
- Currently: `pptx.charts.AREA`, `pptx.charts.BAR`, `pptx.charts.BUBBLE`, `pptx.charts.LINE`, `pptx.charts.SCATTER`, `pptx.charts.PIE`, `pptx.charts.DOUGHNUT`

Multi-Type Charts

- Chart types can be any one of `pptx.charts`, although `pptx.charts.AREA`, `pptx.charts.BAR`, and `pptx.charts.LINE` will give the best results.
- There should be at least two chart-types. There should always be two value axes and category axes.
- Multi Charts have a different function signature than standard. There are two parameters:
- `chartTypes`: Array of objects, each with `type`, `data`, and `options` objects.
- `options`: Standard options as used with single charts. Can include axes options.
- Columns makes the most sense in general. Line charts cannot be rotated to match up with horizontal bars (a PowerPoint limitation).
- Can optionally have a secondary value axis.
- If there is secondary value axis, a secondary category axis is required in order to render, but currently always uses the primary labels. It is recommended to use `catAxisHidden: true` on the secondary category axis.
- Standard options are used, and the chart-type-options are mixed in to each.

```
// Syntax
slide.addChart({MULTI_TYPES_AND_DATA}, {OPTIONS_AND_AXES});
```

Chart Size/Formatting Options

Option	Type	Unit	Default	Description	Possible Values

Option	Type	Unit	Default	Description	Possible Values
`x`	number	inches	`1.0`	horizontal location	0-n OR 'n%'. (Ex: `{x:'50%'}` places object in middle of the Slide)
`y`	number	inches	`1.0`	vertical location	0-n OR 'n%'.
`w`	number	inches	`50%`	width	0-n OR 'n%'. (Ex: `{w:'50%'}` will make object 50% width of the Slide)
`h`	number	inches	`50%`	height	0-n OR 'n%'.
`border`	object			chart border	object with `pt` and `color` values. Ex: `border:{pt:'1', color:'f1f1f1'}`
`chartColors`	array			data colors	array of hex color codes. Ex: `['0088CC','FFCC00']`
`chartColorsOpacity`	number	percent	`100`	data color opacity percent	1-100. Ex: `{ chartColorsOpacity:50 }`
`fill`	string			fill/background color	hex color code. Ex: `{ fill:'0088CC' }`
`holeSize`	number	percent	`50`	doughnut hole size	1-100. Ex: `{ holeSize:50 }`
`invertedColors`	array			data colors for negative numbers	array of hex color codes. Ex: `['0088CC','FFCC00']`

Option	Type	Unit	Default	Description	Possible Values
`legendFontSize`	number	points	`10`	legend font size	1-256. Ex: `{ legendFontSize: 13 }`
`legendColor`	string		`000000`	legend text color	hex color code. Ex: `{ legendColor: '0088CC' }`
`legendPos`	string		`r`	chart legend position	`b` (bottom), `tr` (top-right), `l` (left), `r` (right), `t` (top)
`layout`	object			positioning plot within chart area	object with `x`, `y`, `w` and `h` props, all in range 0-1 (proportionally related to the chart size). Ex: `{x: 0, y: 0, w: 1, h: 1}` fully expands chart within the plot area
`showDataTable`	boolean		`false`	show Data Table under the chart	`true` or `false` (Not available for Pie/Doughnut charts)
`showDataTableKeys`	boolean		`true`	show Data Table Keys (color blocks)	`true` or `false` (Not available for Pie/Doughnut charts)
`showDataTableHorzBorder`	boolean		`true`	show Data Table horizontal borders	`true` or `false` (Not available for Pie/Doughnut charts)

Option	Type	Unit	Default	Description	Possible Values
`showDataTableVertBorder`	boolean		`true`	show Data Table vertical borders	`true` or `false` (Not available for Pie/Doughnut charts)
`showDataTableOutline`	boolean		`true`	show Data Table table outline	`true` or `false` (Not available for Pie/Doughnut charts)
`showLabel`	boolean		`false`	show data labels	`true` or `false`
`showLegend`	boolean		`false`	show chart legend	`true` or `false`
`showPercent`	boolean		`false`	show data percent	`true` or `false`
`showTitle`	boolean		`false`	show chart title	`true` or `false`
`showValue`	boolean		`false`	show data values	`true` or `false`
`title`	string			chart title	a string. Ex: `{ title:'Sales by Region' }`
`titleAlign`	string		`center`	chart title text align	`left` `center` or `right` Ex: `{ titleAlign:'left' }`
`titleColor`	string		`000000`	title color	hex color code. Ex: `{ titleColor:'0088CC' }`

Option	Type	Unit	Default	Description	Possible Values
`titleFontFace`	string		`Arial`	font face	font name. Ex: `{ titleFontFace:'Arial' }`
`titleFontSize`	number	points	`18`	font size	1-256. Ex: `{ titleFontSize:12 }`
`titlePos`	object			title position	object with x and y values. Ex: `{ titlePos:{x: 0, y: 10} }`
`titleRotate`	integer	degrees		title rotation degrees	0-360. Ex: `{ titleRotate:45 }`

Chart Axis Options

Option	Type	Unit	Default	Description	Possible Values
`axisLineColor`	string		`000000`	cat/val axis line color	hex color code. Ex: `{ axisLineColor:'0088CC' }`
`catAxisBaseTimeUnit`	string			category-axis base time unit	`days` `months` or `years`
`catAxisHidden`	boolean		`false`	hide category-axis	`true` or `false`
`catAxisLabelColor`	string		`000000`	category-axis color	hex color code. Ex: `{ catAxisLabelColor:'0088CC' }`

Option	Type	Unit	Default	Description	Possible Values
`catAxisLabelFontFace`	string		`Arial`	category-axis font face	font name. Ex: `{ titleFontFace:'Arial' }`
`catAxisLabelFontSize`	integer	points	`18`	category-axis font size	1-256. Ex: `{ titleFontSize:12 }`
`catAxisLabelFrequency`	integer			PPT "Interval Between Labels"	1-n. Ex: `{ catAxisLabelFrequency: 2 }`
`catAxisLabelPos`	string	string	`nextTo`	axis label position	`low` , `high` , or `nextTo` . Ex: `{ catAxisLabelPos: 'low' }`
`catAxisLineShow`	boolean		true	show/hide category-axis line	`true` or `false`
`catAxisMajorTimeUnit`	string			category-axis major time unit	`days` `months` or `years`
`catAxisMinorTimeUnit`	string			category-axis minor time unit	`days` `months` or `years`

Option	Type	Unit	Default	Description	Possible Values
`catAxisMajorUnit`	integer			category-axis major unit	Positive integer. Ex: `{ catAxisMajorUnit:12 }`
`catAxisMinorUnit`	integer			category-axis minor unit	Positive integer. Ex: `{ catAxisMinorUnit:1 }`
`catAxisOrientation`	string		`minMax`	category-axis orientation	`maxMin` (high->low) or `minMax` (low->high)
`catAxisTitle`	string		`Axis Title`	axis title	a string. Ex: `{ catAxisTitle:'Regions' }`
`catAxisTitleColor`	string		`000000`	title color	hex color code. Ex: `{ catAxisTitleColor:'0088CC' }`
`catAxisTitleFontFace`	string		`Arial`	font face	font name. Ex: `{ catAxisTitleFontFace:'Arial' }`
`catAxisTitleFontSize`	integer	points		font size	1-256. Ex: `{ catAxisTitleFontSize:12 }`
`catAxisTitleRotate`	integer	degrees		title rotation degrees	0-360. Ex: `{ catAxisTitleRotate:45 }`
`catGridLine`	object		`none`	category grid line style	object with properties `size` (pt), `color` and `style` (`'solid'`, `'dash'` or `'dot'`) or `'none'` to hide

Option	Type	Unit	Default	Description	Possible Values
`showCatAxisTitle`	boolean		`false`	show category (vert) title	`true` or `false` . Ex: `{ showCatAxisTitle:true }`
`showValAxisTitle`	boolean		`false`	show values (horiz) title	`true` or `false` . Ex: `{ showValAxisTitle:true }`
`valAxisHidden`	boolean		`false`	hide value-axis	`true` or `false`
`valAxisLabelColor`	string		`000000`	value-axis color	hex color code. Ex: `{ valAxisLabelColor:'0088CC' }`
`valAxisLabelFontFace`	string		`Arial`	value-axis font face	font name. Ex: `{ titleFontFace:'Arial' }`
`valAxisLabelFontSize`	integer	points	`18`	value-axis font size	1-256. Ex: `{ titleFontSize:12 }`
`valAxisLabelFormatCode`	string		`General`	value-axis number format	format string. Ex: `{ axisLabelFormatCode:'#,##0' }` MicroSoft Number Format Codes
`valAxisLineShow`	boolean		true	show/hide value-axis line	`true` or `false`
`valAxisMajorUnit`	number		`1.0`	value-axis tick steps	Float or whole number. Ex: `{ majorUnit:0.2 }`

Option	Type	Unit	Default	Description	Possible Values
`valAxisMaxVal`	number			value-axis maximum value	1-N. Ex: `{ valAxisMaxVal:125 }`
`valAxisMinVal`	number			value-axis minimum value	1-N. Ex: `{ valAxisMinVal: -10 }`
`valAxisOrientation`	string		`minMax`	value-axis orientation	`maxMin` (high->low) or `minMax` (low->high)
`valAxisTitle`	string		`Axis Title`	axis title	a string. Ex: `{ valAxisTitle:'Sales (USD)' }`
`valAxisTitleColor`	string		`000000`	title color	hex color code. Ex: `{ valAxisTitleColor:'0088CC' }`
`valAxisTitleFontFace`	string		`Arial`	font face	font name. Ex: `{ valAxisTitleFontFace:'Arial' }`
`valAxisTitleFontSize`	number	points		font size	1-256. Ex: `{ valAxisTitleFontSize:12 }`
`valAxisTitleRotate`	integer	degrees		title rotation degrees	0-360. Ex: `{ valAxisTitleRotate:45 }`
`valGridLine`	object			value grid line style	object with properties `size` (pt), `color` and `style` (`'solid'`, `'dash'` or `'dot'`) or `'none'` to hide

Chart Data Options

Option	Type	Unit	Default	Description	Possible Values
`barDir`	string		`col`	bar direction	(*Bar Chart*) `bar` (horizontal) or `col` (ver `{barDir:'bar'}`
`barGapWidthPct`	number	percent	`150`	width % between bar groups	(*Bar Chart*) 0-999. Ex: `{ barGapWidthPct:`
`barGrouping`	string		`clustered`	bar grouping	(*Bar Chart*) `clustered` or `stacked` or `percentStacked` .
`dataBorder`	object			data border	object with `pt` and `color` values. Ex: `b` `{pt:'1', color:'f1f1f1'}`
`dataLabelColor`	string		`000000`	data label color	hex color code. Ex: `{ dataLabelColor:'0`
`dataLabelFormatCode`	string			format to show data value	format string. Ex: `{ dataLabelFormatCode` MicroSoft Number Format Codes
`dataLabelFontFace`	string		`Arial`	value-axis font face	font name. Ex: `{ titleFontFace:'Arial'`
`dataLabelFontSize`	number	points	`18`	value-axis font size	1-256. Ex: `{ titleFontSize:12 }`
`dataLabelPosition`	string		`bestFit`	data label position	`bestFit` , `b` , `ctr` , `inBase` , `inEnd` , `l` , `outl`

Option	Type	Unit	Default	Description	Possible Values
`dataNoEffects`	boolean		`false`	whether to omit effects on data	(*Doughnut/Pie Charts*) `true` or `false`
`gridLineColor`	string		`000000`	grid line color	hex color code. Ex: `{ gridLineColor:'00`
`lineDataSymbol`	string		`circle`	symbol used on line marker	`circle` , `dash` , `diamond` , `dot` , `none` , `squa`
`lineDataSymbolSize`	number	points	`6`	size of line data symbol	1-256. Ex: `{ lineDataSymbolSize:12 }`
`lineDataSymbolLineSize`	number	points	`0.75`	size of data symbol outline	1-256. Ex: `{ lineDataSymbolLineSize:12`
`lineDataSymbolLineColor`	number	points	`0.75`	size of data symbol outline	1-256. Ex: `{ lineDataSymbolLineSize:12`
`lineSize`	number	points	`2`	thickness of data line (0 is no line)	0-256. Ex: `{ lineSize: 1 }`

Option	Type	Unit	Default	Description	Possible Values
`lineSmooth`	boolean		`false`	whether to smooth lines	`true` or `false`
`shadow`	object			data element shadow options	`'none'` or shadow options
`valueBarColors`	boolean		`false`	forces chartColors on multi-data-series	`true` or `false`

Chart Element Shadow Options

Option	Type	Unit	Default	Description	Possible Values
`type`	string		`outer`	shadow type	`outer` or `inner` . Ex: `{ type:'outer' }`
`angle`	number	degrees	`90`	shadow angle	0-359. Ex: `{ angle:90 }`
`blur`	number	points	`3`	blur size	1-256. Ex: `{ blur:3 }`
`color`	string		`000000`	shadow color	hex color code. Ex: `{ color:'0088CC' }`
`offset`	number	points	`1.8`	offset size	1-256. Ex: `{ offset:2 }`
`opacity`	number	percent	`0.35`	opacity	0-1. Ex: `{ opacity:0.35 }`

Chart Multi-Type Options

Option	Type	Default	Description	Possible Values
`catAxes`	array		array of two axis options objects	See example below
`secondaryCatAxis`	boolean	`false`	If data should use secondary category axis (or primary)	`true` or `false`
`secondaryValAxis`	boolean	`false`	If data should use secondary value axis (or primary)	`true` or `false`
`valAxes`	array		array of two axis options objects	See example below

Chart Examples

```
var pptx = new PptxGenJS();
pptx.setLayout('LAYOUT_WIDE');

var slide = pptx.addNewSlide();

// Chart Type: BAR
var dataChartBar = [
  {
    name  : 'Region 1',
    labels: ['May', 'June', 'July', 'August'],
    values: [26, 53, 100, 75]
  },
  {
    name  : 'Region 2',
    labels: ['May', 'June', 'July', 'August'],
    values: [43.5, 70.3, 90.1, 80.05]
  }
```

```
];
slide.addChart( pptx.charts.BAR, dataChartBar, { x:1.0, y:1.0, w:12, h:6 } );

// Chart Type: AREA
// Chart Type: LINE
var dataChartAreaLine = [
  {
    name  : 'Actual Sales',
    labels: ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec'],
    values: [1500, 4600, 5156, 3167, 8510, 8009, 6006, 7855, 12102, 12789, 10123, 15121]
  },
  {
    name  : 'Projected Sales',
    labels: ['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec'],
    values: [1000, 2600, 3456, 4567, 5010, 6009, 7006, 8855, 9102, 10789, 11123, 12121]
  }
];
slide.addChart( pptx.charts.AREA, dataChartAreaLine, { x:1.0, y:1.0, w:12, h:6 } );
slide.addChart( pptx.charts.LINE, dataChartAreaLine, { x:1.0, y:1.0, w:12, h:6 } );

// Chart Type: PIE
var dataChartPie = [
  { name:'Location', labels:['DE','GB','MX','JP','IN','US'], values:[35,40,85,88,99,101] }
];
slide.addChart( pptx.charts.PIE, dataChartPie, { x:1.0, y:1.0, w:6, h:6 } );

// Chart Type: XY SCATTER
var dataChartScatter = [
        { name:'X-Axis',    values:[1,2,3,4,5,6,7,8,9,10] },
        { name:'Y-Value 1', values:[13, 20, 21, 25] },
        { name:'Y-Value 2', values:[21, 22, 25, 49] }
];
slide.addChart( pptx.charts.SCATTER, dataChartScatter, { x:1.0, y:1.0, w:6, h:4 } );

// Chart Type: BUBBLE
var dataChartBubble = [
        { name:'X-Axis',   values:[1, 2, 3, 4, 5, 6] },
```

```
        { name:'Airplane', values:[33, 20, 51, 65, 71, 75], sizes:[10,10,12,12,15,20] },
        { name:'Train',    values:[99, 88, 77, 89, 99, 99], sizes:[20,20,22,22,25,30] },
        { name:'Bus',      values:[21, 22, 25, 49, 59, 69], sizes:[11,11,13,13,16,21] }
];
slide.addChart( pptx.charts.BUBBLE, dataChartBubble, { x:1.0, y:1.0, w:6, h:4 } );

// Chart Type: Multi-Type
// NOTE: use the same labels for all types
var labels = ['Q1', 'Q2', 'Q3', 'Q4', 'OT'];
var chartTypes = [
  {
    type: pptx.charts.BAR,
    data: [{
      name: 'Projected',
      labels: labels,
      values: [17, 26, 53, 10, 4]
    }],
    options: { barDir: 'col' }
  },
  {
    type: pptx.charts.LINE,
    data: [{
      name: 'Current',
      labels: labels,
      values: [5, 3, 2, 4, 7]
    }],
    options: {
      // NOTE: both are required, when using a secondary axis:
      secondaryValAxis: true,
      secondaryCatAxis: true
    }
  }
];
var multiOpts = {
  x:1.0, y:1.0, w:6, h:6,
  showLegend: false,
  valAxisMaxVal: 100,
```

```
  valAxisMinVal: 0,
  valAxisMajorUnit: 20,
  valAxes:[
    {
      showValAxisTitle: true,
      valAxisTitle: 'Primary Value Axis'
    },
    {
      showValAxisTitle: true,
      valAxisTitle: 'Secondary Value Axis',
      valAxisMajorUnit: 1,
      valAxisMaxVal: 10,
      valAxisMinVal: 1,
      valGridLine: 'none'
    }
  ],
  catAxes: [{ catAxisTitle: 'Primary Category Axis' }, { catAxisHidden: true }]
};

slide.addChart(chartTypes, multiOpts);

pptx.save('Demo-Chart');
```

Adding Text

```
// Syntax
slide.addText('TEXT', {OPTIONS});
slide.addText('Line 1\nLine 2', {OPTIONS});
slide.addText([ {text:'TEXT', options:{OPTIONS}} ]);
```

Text Options

Option	Type	Unit	Default	Description	Possible Values
`x`	number	inches	`1.0`	horizontal location	0-n OR 'n%'. (Ex: `{x:'50%'}` will place object in the middle of the Slide)
`y`	number	inches	`1.0`	vertical location	0-n OR 'n%'.
`w`	number	inches		width	0-n OR 'n%'. (Ex: `{w:'50%'}` will make object 50% width of the Slide)
`h`	number	inches		height	0-n OR 'n%'.
`align`	string		`left`	alignment	`left` or `center` or `right`
`autoFit`	boolean		`false`	"Fit to Shape"	`true` or `false`
`bold`	boolean		`false`	bold text	`true` or `false`
`breakLine`	boolean		`false`	appends a line break	`true` or `false` (only applies when used in text options) Ex: `{text:'hi', options: {breakLine:true}}`
`bullet`	boolean		`false`	bulleted text	`true` or `false`
`bullet`	object			bullet options (number type or choose any unicode char)	object with `type` or `code`. Ex: `bullet: {type:'number'}`. Ex: `bullet: {code:'2605'}`
`color`	string			text color	hex color code or scheme color constant. Ex: `{ color:'0088CC' }`
`fill`	string			fill/bkgd color	hex color code or scheme color constant. Ex: `{ color:'0088CC' }`

Option	Type	Unit	Default	Description	Possible Values
`fontFace`	string			font face	Ex: `{ fontFace:'Arial'}`
`font_size`	number	points		font size	1-256. Ex: `{ font_size:12 }`
`hyperlink`	string			add hyperlink	object with `url` or `slide` (`tooltip` optional). Ex: `{ hyperlink: {url:'https://github.com'} }`
`indentLevel`	number	level	`0`	bullet indent level	1-32. Ex: `{ indentLevel:1 }`
`inset`	number	inches		inset/padding	1-256. Ex: `{ inset:1.25 }`
`isTextBox`	boolean		`false`	PPT "Textbox"	`true` or `false`
`italic`	boolean		`false`	italic text	`true` or `false`
`lang`	string		`en-US`	language setting	Ex: `{ lang:'zh-TW' }` (Set this when using non-English fonts like Chinese)
`lineSpacing`	number	points		line spacing points	1-256. Ex: `{ lineSpacing:28 }`
`margin`	number	points		margin	0-99 (ProTip: use the same value from CSS `padding`)
`rectRadius`	number	inches		rounding radius	rounding radius for `ROUNDED_RECTANGLE` text shapes
`rotate`	integer	degrees	`0`	text rotation degrees	0-360. Ex: `{rotate:180}`
`rtlMode`	boolean		`false`	enable Right-to-Left mode	`true` or `false`

Option	Type	Unit	Default	Description	Possible Values
`shadow`	object			text shadow options	see options below. Ex: `shadow:{ type:'outer' }`
`strike`	boolean		`false`	text strikethrough	`true` or `false`
`subscript`	boolean		`false`	subscript text	`true` or `false`
`superscript`	boolean		`false`	superscript text	`true` or `false`
`underline`	boolean		`false`	underline text	`true` or `false`
`valign`	string			vertical alignment	`top` or `middle` or `bottom`
`vert`	string		`horz`	text direction	`eaVert` or `horz` or `mongolianVert` or `vert` or `vert270` or `wordArtVert` or `wordArtVertRtl`

Text Shadow Options

Option	Type	Unit	Default	Description	Possible Values
`type`	string		outer	shadow type	`outer` or `inner`
`angle`	number	degrees		shadow angle	0-359. Ex: `{ angle:180 }`
`blur`	number	points		blur size	1-256. Ex: `{ blur:3 }`
`color`	string			text color	hex color code or scheme color constant. Ex: `{ color:'0088CC' }`

Option	Type	Unit	Default	Description	Possible Values
`offset`	number	points		offset size	1-256. Ex: `{ offset:8 }`
`opacity`	number	percent		opacity	0-1. Ex: `opacity:0.75`

Text Examples

```
var pptx = new PptxGenJS();
var slide = pptx.addNewSlide();

// EX: Dynamic location using percentages
slide.addText('^ (50%/50%)', {x:'50%', y:'50%'});

// EX: Basic formatting
slide.addText('Hello',  { x:0.5, y:0.7, w:3, color:'0000FF', font_size:64 });
slide.addText('World!', { x:2.7, y:1.0, w:5, color:'DDDD00', font_size:90 });

// EX: More formatting options
slide.addText(
    'Arial, 32pt, green, bold, underline, 0 inset',
    { x:0.5, y:5.0, w:'90%', margin:0.5, fontFace:'Arial', font_size:32, color:'00CC00', bold:true, underline:true, i
);

// EX: Format some text
slide.addText('Hello World!', { x:2, y:4, fontFace:'Arial', font_size:42, color:'00CC00', bold:true, italic:true, unc

// EX: Multiline Text / Line Breaks - use "\n" to create line breaks inside text strings
slide.addText('Line 1\nLine 2\nLine 3', { x:2, y:3, color:'DDDD00', font_size:90 });

// EX: Format individual words or lines by passing an array of text objects with `text` and `options`
slide.addText(
    [
        { text:'word-level', options:{ font_size:36, color:'99ABCC', align:'r', breakLine:true } },
        { text:'formatting', options:{ font_size:48, color:'FFFF00', align:'c' } }
```

```
    ],
    { x:0.5, y:4.1, w:8.5, h:2.0, fill:'F1F1F1' }
);

// EX: Bullets
slide.addText('Regular, black circle bullet', { x:8.0, y:1.4, w:'30%', h:0.5, bullet:true });
// Use line-break character to bullet multiple lines
slide.addText('Line 1\nLine 2\nLine 3', { x:8.0, y:2.4, w:'30%', h:1, fill:'F2F2F2', bullet:{type:'number'} });
// Bullets can also be applied on a per-line level
slide.addText(
    [
        { text:'I have a star bullet'    , options:{bullet:{code:'2605'}, color:'CC0000'} },
        { text:'I have a triangle bullet', options:{bullet:{code:'25BA'}, color:'00CD00'} },
        { text:'no bullets on this line' , options:{font_size:12} },
        { text:'I have a normal bullet'  , options:{bullet:true, color:'0000AB'} }
    ],
    { x:8.0, y:5.0, w:'30%', h:1.4, color:'ABABAB', margin:1 }
);

// EX: Hyperlink: Web
slide.addText(
    [{
        text: 'PptxGenJS Project',
        options: { hyperlink:{ url:'https://github.com/gitbrent/pptxgenjs', tooltip:'Visit Homepage' } }
    }],
    { x:1.0, y:1.0, w:5, h:1 }
);
// EX: Hyperlink: Slide in Presentation
slide.addText(
    [{
        text: 'Slide #2',
        options: { hyperlink:{ slide:'2', tooltip:'Go to Summary Slide' } }
    }],
    { x:1.0, y:2.5, w:5, h:1 }
);

// EX: Drop/Outer Shadow
```

```
slide.addText(
    'Outer Shadow',
    {
        x:0.5, y:6.0, font_size:36, color:'0088CC',
        shadow: {type:'outer', color:'696969', blur:3, offset:10, angle:45}
    }
);

// EX: Formatting can be applied at the word/line level
// Provide an array of text objects with the formatting options for that `text` string value
// Line-breaks work as well
slide.addText(
    [
        { text:'word-level\nformatting', options:{ font_size:36, fontFace:'Courier New', color:'99ABCC', align:'r', b
        { text:'...in the same textbox', options:{ font_size:48, fontFace:'Arial', color:'FFFF00', align:'c' } }
    ],
    { x:0.5, y:4.1, w:8.5, h:2.0, margin:0.1, fill:'232323' }
);

pptx.save('Demo-Text');
```

Adding Tables

Syntax:

```
slide.addTable( [rows] );
slide.addTable( [rows], {any Layout/Formatting OPTIONS} );
```

Table Layout Options

Option	Type	Unit	Default	Description	Possible Values
`x`	number	inches	`1.0`	horizontal location	0-n OR 'n%'. (Ex: `{x:'50%'}` will place object in the middle of the Slide)
`y`	number	inches	`1.0`	vertical location	0-n OR 'n%'.
`w`	number	inches		width	0-n OR 'n%'. (Ex: `{w:'50%'}` will make object 50% width of the Slide)
`h`	number	inches		height	0-n OR 'n%'.
`colW`	integer	inches		width for every column	Ex: Width for every column in table (uniform) `2.0`
`colW`	array	inches		column widths in order	Ex: Width for each of 5 columns `[1.0, 2.0, 2.5, 1.5, 1.0]`
`rowH`	integer	inches		height for every row	Ex: Height for every row in table (uniform) `2.0`
`rowH`	array	inches		row heights in order	Ex: Height for each of 5 rows `[1.0, 2.0, 2.5, 1.5, 1.0]`

Table Auto-Paging Options

Option	Type	Default	Description	Possible Values
`autoPage`	boolean	`true`	auto-page table	`true` or `false` . Ex: `{autoPage:false}`
`lineWeight`	float	0	line weight value	-1.0 to 1.0. Ex: `{lineWeight:0.5}`

Option	Type	Default	Description	Possible Values
`newPageStartY`	number/string		starting `y` value for tables on new Slides	0-n OR 'n%'. Ex: `{newPageStartY:0.5}`

Option Details:

- `autoPage` : allows the auto-paging functionality (as table rows overflow the Slide, new Slides will be added) to be disabled.
- `lineWeight` : adjusts the calculated height of lines. If too much empty space is left under each table, then increase lineWeight value. Conversely, if the tables are overflowing the bottom of the Slides, then reduce the lineWeight value. Also helpful when using some fonts that do not have the usual golden ratio.
- `newPageStartY` : provides the ability to specify where new tables will be placed on new Slides. For example, you may place a table halfway down a Slide, but you wouldn't that to be the starting location for subsequent tables. Use this option to ensure there is no wasted space and to guarantee a professional look.

Table Auto-Paging Notes

- New slides will be created as tables overflow. The table will start at either `newPageStartY` (if present) or the Slide's top `margin` .
- Tables will retain their existing `x` , `w` , and `colW` values as they are rendered onto subsequent Slides.
- Auto-paging is not an exact science! Try using different `lineWeight` and Slide margin values if your tables are overflowing the Slide.
- There are many examples of auto-paging in the `examples` folder.

Table Formatting Options

Option	Type	Unit	Default	Description	Possible Values
`align`	string		`left`	alignment	`left` or `center` or `right` (or `l` `c` `r`)

Option	Type	Unit	Default	Description	Possible Values
`bold`	boolean		`false`	bold text	`true` or `false`
`border`	object			cell border	object with `pt` and `color` values. Ex: `{pt:'1', color:'f1f1f1'}`
`border`	array			cell border	array of objects with `pt` and `color` values in TRBL order.
`color`	string			text color	hex color code or scheme color constant. Ex: `{color:'0088CC'}`
`colspan`	integer			column span	2-n. Ex: `{colspan:2}`
`fill`	string			fill/bkgd color	hex color code or scheme color constant. Ex: `{color:'0088CC'}`
`fontFace`	string			font face	Ex: `{fontFace:'Arial'}`
`font_size`	number	points		font size	1-256. Ex: `{font_size:12}`
`italic`	boolean		`false`	italic text	`true` or `false`
`margin`	number	points		margin	0-99 (ProTip: use the same value from CSS `padding`)
`margin`	array	points		margin	array of integer values in TRBL order. Ex: `margin: [5,10,5,10]`
`rowspan`	integer			row span	2-n. Ex: `{rowspan:2}`
`underline`	boolean		`false`	underline text	`true` or `false`
`valign`	string			vertical alignment	`top` or `middle` or `bottom` (or `t` `m` `b`)

Table Formatting Notes

- **Formatting Options** passed to `slide.addTable()` apply to every cell in the table
- You can selectively override formatting at a cell-level providing any **Formatting Option** in the cell `options`

Table Cell Formatting

- Table cells can be either a plain text string or an object with text and options properties
- When using an object, any of the formatting options above can be passed in `options` and will apply to that cell only
- Cell borders can be removed (aka: borderless table) by passing a 'none' (Ex: `border:'none'`)

Bullets and word-level formatting are supported inside table cells. Passing an array of objects with text/options values as the `text` value allows fine-grained control over the text inside cells.

- Available formatting options are here: Text Options
- See below for examples or view the `examples/pptxgenjs-demo.html` page for lots more

Table Cell Formatting Examples

```
// TABLE 1: Cell-level Formatting
var rows = [];
// Row One: cells will be formatted according to any options provided to `addTable()`
rows.push( ['First', 'Second', 'Third'] );
// Row Two: set/override formatting for each cell
rows.push([
    { text:'1st', options:{color:'ff0000'} },
    { text:'2nd', options:{color:'00ff00'} },
    { text:'3rd', options:{color:'0000ff'} }
]);
slide.addTable( rows, { x:0.5, y:1.0, w:9.0, color:'363636' } );

// TABLE 2: Using word-level formatting inside cells
```

```
// NOTE: An array of text/options objects provides fine-grained control over formatting
var arrObjText = [
    { text:'Red ',   options:{color:'FF0000'} },
    { text:'Green ', options:{color:'00FF00'} },
    { text:'Blue',   options:{color:'0000FF'} }
];
// EX A: Pass an array of text objects to `addText()`
slide.addText( arrObjText, { x:0.5, y:2.75, w:9, h:2, margin:0.1, fill:'232323' } );

// EX B: Pass the same objects as a cell's `text` value
var arrTabRows = [
    [
        { text:'Cell 1 A',  options:{fontFace:'Arial'  } },
        { text:'Cell 1 B',  options:{fontFace:'Courier'} },
        { text: arrObjText, options:{fill:'232323'}      }
    ]
];
slide.addTable( arrTabRows, { x:0.5, y:5, w:9, h:2, colW:[1.5,1.5,6] } );
```

Table Examples

```
var pptx = new PptxGenJS();
var slide = pptx.addNewSlide();
slide.addText('Demo-03: Table', { x:0.5, y:0.25, font_size:18, fontFace:'Arial', color:'0088CC' });

// TABLE 1: Single-row table
// --------
var rows = [ 'Cell 1', 'Cell 2', 'Cell 3' ];
var tabOpts = { x:0.5, y:1.0, w:9.0, fill:'F7F7F7', font_size:14, color:'363636' };
slide.addTable( rows, tabOpts );

// TABLE 2: Multi-row table (each rows array element is an array of cells)
// --------
var rows = [
    ['A1', 'B1', 'C1'],
```

```
    ['A2', 'B2', 'C2']
];
var tabOpts = { x:0.5, y:2.0, w:9.0, fill:'F7F7F7', font_size:18, color:'6f9fc9' };
slide.addTable( rows, tabOpts );

// TABLE 3: Formatting at a cell level - use this to selectively override table's cell options
// --------
var rows = [
    [
        { text:'Top Lft', options:{ valign:'t', align:'l', fontFace:'Arial'   } },
        { text:'Top Ctr', options:{ valign:'t', align:'c', fontFace:'Verdana' } },
        { text:'Top Rgt', options:{ valign:'t', align:'r', fontFace:'Courier' } }
    ],
];
var tabOpts = { x:0.5, y:4.5, w:9.0, rowH:0.6, fill:'F7F7F7', font_size:18, color:'6f9fc9', valign:'m'} };
slide.addTable( rows, tabOpts );

// Multiline Text / Line Breaks - use either "\r" or "\n"
slide.addTable( ['Line 1\nLine 2\nLine 3'], { x:2, y:3, w:4 });

pptx.save('Demo-Tables');
```

Adding Shapes

Syntax (no text):

```
slide.addShape({SHAPE}, {OPTIONS});
```

Syntax (with text):

```
slide.addText("some string", {SHAPE, OPTIONS});
```

Check the `pptxgen.shapes.js` file for a complete list of the hundreds of PowerPoint shapes available.

Shape Options

Option	Type	Unit	Default	Description	Possible Values
`x`	number	inches	`1.0`	horizontal location	0-n OR 'n%'. (Ex: `{x:'50%'}` will place object in the middle of the Slide)
`y`	number	inches	`1.0`	vertical location	0-n OR 'n%'.
`w`	number	inches		width	0-n OR 'n%'. (Ex: `{w:'50%'}` will make object 50% width of the Slide)
`h`	number	inches		height	0-n OR 'n%'.
`align`	string		`left`	alignment	`left` or `center` or `right`
`fill`	string			fill/bkgd color	hex color code or scheme color constant. Ex: `{color:'0088CC'}`
`fill`	object			fill/bkgd color	object with `type`, `color`, `alpha` (opt). Ex: `fill: {type:'solid', color:'0088CC', alpha:25}`
`flipH`	boolean			flip Horizontal	`true` or `false`
`flipV`	boolean			flip Vertical	`true` or `false`
`line`	string			border line color	hex color code or scheme color constant. Ex: `{line:'0088CC'}`

Option	Type	Unit	Default	Description	Possible Values
`lineDash`	string		`solid`	border line dash style	`dash` , `dashDot` , `lgDash` , `lgDashDot` , `lgDashDotDot` , `solid` , `sysDash` or `sysDot`
`line_head`	string			border line ending	`arrow` , `diamond` , `oval` , `stealth` , `triangle` or `none`
`lineSize`	number	points		border line size	1-256. Ex: {lineSize:4}
`line_tail`	string			border line heading	`arrow` , `diamond` , `oval` , `stealth` , `triangle` or `none`
`rectRadius`	number	inches		rounding radius	rounding radius for `ROUNDED_RECTANGLE` text shapes
`rotate`	integer	degrees		rotation degrees	0-360. Ex: `{rotate:180}`

Shape Examples

```
var pptx = new PptxGenJS();
pptx.setLayout('LAYOUT_WIDE');

var slide = pptx.addNewSlide();

// LINE
slide.addShape(pptx.shapes.LINE,      { x:4.15, y:4.40, w:5, h:0, line:'FF0000', lineSize:1 });
slide.addShape(pptx.shapes.LINE,      { x:4.15, y:4.80, w:5, h:0, line:'FF0000', lineSize:2, line_head:'triangle' });
slide.addShape(pptx.shapes.LINE,      { x:4.15, y:5.20, w:5, h:0, line:'FF0000', lineSize:3, line_tail:'triangle' });
slide.addShape(pptx.shapes.LINE,      { x:4.15, y:5.60, w:5, h:0, line:'FF0000', lineSize:4, line_head:'triangle', li
// DIAGONAL LINE
slide.addShape(pptx.shapes.LINE,      { x:0, y:0, w:5.0, h:0, line:'FF0000', rotate:45 });
```

```
// RECTANGLE
slide.addShape(pptx.shapes.RECTANGLE, { x:0.50, y:0.75, w:5, h:3, fill:'FF0000' });
// OVAL
slide.addShape(pptx.shapes.OVAL,      { x:4.15, y:0.75, w:5, h:2, fill:{ type:'solid', color:'0088CC', alpha:25 } });

// Adding text to Shapes:
slide.addText('RIGHT-TRIANGLE', { shape:pptx.shapes.RIGHT_TRIANGLE, align:'c', x:0.40, y:4.3, w:6, h:3, fill:'0088CC'
slide.addText('RIGHT-TRIANGLE', { shape:pptx.shapes.RIGHT_TRIANGLE, align:'c', x:7.00, y:4.3, w:6, h:3, fill:'0088CC'

pptx.save('Demo-Shapes');
```

Adding Images

Syntax:

```
slide.addImage({OPTIONS});
```

Animated GIFs can be included in Presentations in one of two ways:

- Using Node.js: use either `data` or `path` options (Node can encode any image into base64)
- Client Browsers: pre-encode the gif and add it using the `data` option (encoding images into GIFs is beyond any current browser)

Image Options

Option	Type	Unit	Default	Description	Possible Values
`x`	number	inches	`1.0`	horizontal location	0-n

Option	Type	Unit	Default	Description	Possible Values
`y`	number	inches	`1.0`	vertical location	0-n
`w`	number	inches	`1.0`	width	0-n
`h`	number	inches	`1.0`	height	0-n
`data`	string			image data (base64)	base64-encoded image string. (either `data` or `path` is required)
`hyperlink`	string			add hyperlink	object with `url` or `slide` (`tooltip` optional). Ex: `{ hyperlink:{url:'https://github.com'} }`
`path`	string			image path	Same as used in an (img src="") tag. (either `data` or `path` is required)
`sizing`	object			transforms image	See Image Sizing

NOTES

- SVG images are not currently supported in PowerPoint or PowerPoint Online (even when encoded into base64). PptxGenJS does properly encode and include SVG images, so they will begin showing once Microsoft adds support for this image type.
- Using `path` to add remote images (images from a different server) is not currently supported.

Deprecation Warning Old positional parameters (e.g.: `slide.addImage('images/chart.png', 1, 1, 6, 3)`) are now deprecated as of 1.1.0

Image Examples

```
var pptx = new PptxGenJS();
var slide = pptx.addNewSlide();

// Image by path
slide.addImage({ path:'images/chart_world_peace_near.png', x:1.0, y:1.0, w:8.0, h:4.0 });
// Image by data (base64-encoding)
slide.addImage({ data:'image/png;base64,iVtDafDrBF[...]=', x:3.0, y:5.0, w:6.0, h:3.0 });

// NOTE: Slide API calls return the same slide, so you can chain calls:
slide.addImage({ path:'images/cc_license_comp_chart.png', x:6.6, y:0.75, w:6.30, h:3.70 })
     .addImage({ path:'images/cc_logo.jpg',               x:0.5, y:3.50, w:5.00, h:3.70 })
     .addImage({ path:'images/cc_symbols_trans.png',      x:6.6, y:4.80, w:6.30, h:2.30 });

// Image with Hyperlink
slide.addImage({
  x:1.0, y:1.0, w:8.0, h:4.0,
  hyperlink:{ url:'https://github.com/gitbrent/pptxgenjs', tooltip:'Visit Homepage' },
  path:'images/chart_world_peace_near.png',
});

pptx.save('Demo-Images');
```

Image Sizing

The `sizing` option provides cropping and scaling an image to a specified area. The property expects an object with the following structure:

Property	Type	Unit	Default	Description	Possible Values
`type`	string			sizing algorithm	`'crop'`, `'contain'` or `'cover'`

Property	Type	Unit	Default	Description	Possible Values
`w`	number	inches	`w` of the image	area width	0-n
`h`	number	inches	`h` of the image	area height	0-n
`x`	number	inches	`0`	area horizontal position related to the image	0-n (effective for `crop` only)
`y`	number	inches	`0`	area vertical position related to the image	0-n (effective for `crop` only)

Particular `type` values behave as follows:

- `contain` works as CSS property `background-size` — shrinks the image (ratio preserved) to the area given by `w` and `h` so that the image is completely visible. If the area's ratio differs from the image ratio, an empty space will surround the image.
- `cover` works as CSS property `background-size` — shrinks the image (ratio preserved) to the area given by `w` and `h` so that the area is completely filled. If the area's ratio differs from the image ratio, the image is centered to the area and cropped.
- `crop` cuts off a part specified by image-related coordinates `x`, `y` and size `w`, `h`.

NOTES:

- If you specify an area size larger than the image for the `contain` and `cover` type, then the image will be stretched, not shrunken.
- In case of the `crop` option, if the specified area reaches out of the image, then the covered empty space will be a part of the image.

- When the `sizing` property is used, its `w` and `h` values represent the effective image size. For example, in the following snippet, width and height of the image will both equal to 2 inches and its top-left corner will be located at [1 inch, 1 inch]:

```
slide.addImage({
  path: '...',
  w: 4,
  h: 3,
  x: 1,
  y: 1,
  sizing: {
    type: 'contain',
    w: 2,
    h: 2
  }
});
```

Adding Media (Audio/Video/YouTube)

Syntax:

```
slide.addMedia({OPTIONS});
```

Both Video (mpg, mov, mp4, m4v, etc.) and Audio (mp3, wav, etc.) are supported (list of supported formats)

- Using Node.js: use either `data` or `path` options (Node can encode any media into base64)
- Client Browsers: pre-encode the media and add it using the `data` option (encoding video/audio is beyond any current browser)

Online video (YouTube embeds, etc.) is supported in both client browser and in Node.js

Media Options

Option	Type	Unit	Default	Description	Possible Values
`x`	number	inches	`1.0`	horizontal location	0-n
`y`	number	inches	`1.0`	vertical location	0-n
`w`	number	inches	`1.0`	width	0-n
`h`	number	inches	`1.0`	height	0-n
`data`	string			media data (base64)	base64-encoded string
`path`	string			media path	relative path to media file
`link`	string			online url/link	link to online video. Ex: `link:'https://www.youtube.com/embed/blahBlah'`
`type`	string			media type	media type: `audio` or `video` (reqs: `data` or `path`) or `online` (reqs: `link`)

Media Examples

```
var pptx = new PptxGenJS();
var slide = pptx.addNewSlide();

// Media by path (Node.js only)
```

```
slide.addMedia({ type:'audio', path:'../media/sample.mp3', x:1.0, y:1.0, w:3.0, h:0.5 });
// Media by data (client browser or Node.js)
slide.addMedia({ type:'audio', data:'audio/mp3;base64,iVtDafDrBF[...]=', x:3.0, y:1.0, w:6.0, h:3.0 });
// Online by link (client browser or Node.js)
slide.addMedia({ type:'online', link:'https://www.youtube.com/embed/Dph6ynRVyUc', x:1.0, y:4.0, w:8.0, h:4.5 });

pptx.save('Demo-Media');
```

Master Slides and Corporate Branding

Slide Masters

Generating sample slides like those shown above is great for demonstrating library features, but the reality is most of us will be required to produce presentations that have a certain design or corporate branding.

PptxGenJS allows you to define Slide Master Layouts via objects that can then be used to provide branding functionality.

Slide Masters are created by calling the `defineSlideMaster()` method along with an options object (same style used in Slides). Once defined, you can pass the Master title to `addNewSlide()` and that Slide will use the Layout previously defined. See the demo under /examples for several working examples.

The defined Masters become first-class Layouts in the exported PowerPoint presentation and can be changed via View > Slide Master and will affect the Slides created using that layout.

Slide Master Object Options

Option	Type	Unit	Default	Description	Possible Values

Option	Type	Unit	Default	Description	Possible Values
`bkgd`	string		`ffffff`	color	hex color code or scheme color constant. Ex: `{ bkgd:'0088CC' }`
`bkgd`	object			image	object with path OR data. Ex: `{path:'img/bkgd.png'}` OR `{data:'image/png;base64,iVBORwTwB[...]='}`
`slideNumber`	object			Show slide numbers	ex: `{ x:1.0, y:'50%' }` `x` and `y` can be either inches or percent
`margin`	number	inches	`1.0`	Slide margins	0.0 through Slide.width
`margin`	array			Slide margins	array of numbers in TRBL order. Ex: `[0.5, 0.75, 0.5, 0.75]`
`objects`	array			Objects for Slide	object with type and options. Type: `chart` , `image` , `line` , `rect` or `text` . Example
`title`	string			Layout title/name	some title

TIP: Pre-encode your images (base64) and add the string as the optional data key/val (see `bkgd` above)

Slide Master Examples

```
var pptx = new PptxGenJS();
pptx.setLayout('LAYOUT_WIDE');

pptx.defineSlideMaster({
  title: 'MASTER_SLIDE',
  bkgd:  'FFFFFF',
```

```
    objects: [
      { 'line':  { x: 3.5, y:1.00, w:6.00, line:'0088CC', lineSize:5 } },
      { 'rect':  { x: 0.0, y:5.30, w:'100%', h:0.75, fill:'F1F1F1' } },
      { 'text':  { text:'Status Report', options:{ x:3.0, y:5.30, w:5.5, h:0.75 } } },
      { 'image': { x:11.3, y:6.40, w:1.67, h:0.75, path:'images/logo.png' } }
    ],
    slideNumber: { x:0.3, y:'90%' }
  });

  var slide = pptx.addNewSlide('MASTER_SLIDE');
  slide.addText('How To Create PowerPoint Presentations with JavaScript', { x:0.5, y:0.7, font_size:18 });

  pptx.save();
```

Table-to-Slides Feature

Syntax:

```
slide.addSlidesForTable(htmlElementID);
slide.addSlidesForTable(htmlElementID, {OPTIONS});
```

Any variety of HTML tables can be turned into a series of slides (auto-paging) by providing the table's ID.

- Reproduces an HTML table - background colors, borders, fonts, padding, etc.
- Slide margins are based on either the Master Slide provided or options

NOTE: Nested tables are not supported in PowerPoint, so only the string contents of a single level deep table cell will be reproduced

Table-to-Slides Options

Option	Type	Unit	Description	Possible Values
`x`	number	inches	horizontal location	0-256. Table will be placed here on each Slide
`y`	number	inches	vertical location	0-256. Table will be placed here on each Slide
`w`	number	inches	width	0-256. Default is (100% - Slide margins)
`h`	number	inches	height	0-256. Default is (100% - Slide margins)
`master`	string		master slide to use	Slide Masters name. Ex: `{ master:'TITLE_SLIDE' }`
`addHeaderToEach`	boolean		add table headers to each slide	Ex: `addHeaderToEach:true`
`addImage`	string		add an image to each slide	Ex: `{ addImage:{ path:"images/logo.png", x:10, y:0.5, w:1.2, h:0.75 } }`
`addShape`	string		add a shape to each slide	Use the established syntax
`addTable`	string		add a table to each slide	Use the established syntax
`addText`	string		add text to each slide	Use the established syntax

Table-to-Slides HTML Options

Add an `data` attribute to the table's `<th>` tag to manually size columns (inches)

- minimum column width can be specified by using the `data-pptx-min-width` attribute
- fixed column width can be specified by using the `data-pptx-width` attribute

Example:

```
<table id="tabAutoPaging" class="tabCool">
  <thead>
    <tr>
      <th data-pptx-min-width="0.6" style="width: 5%">Row</th>
      <th data-pptx-min-width="0.8" style="width:10%">Last Name</th>
      <th data-pptx-min-width="0.8" style="width:10%">First Name</th>
      <th data-pptx-width="8.5"     style="width:75%">Description</th>
    </tr>
  </thead>
  <tbody></tbody>
</table>
```

Table-to-Slides Notes

- Default `x`, `y` and `margin` value is 0.5 inches, the table will take up all remaining space by default (h:100%, w:100%)
- Your Master Slides should already have defined margins, so a Master Slide name is the only option you'll need most of the time
- Hidden tables wont auto-size their columns correctly (as the properties are not accurate)

Table-to-Slides Examples

```
// Pass table element ID to addSlidesForTable function to produce 1-N slides
pptx.addSlidesForTable( 'myHtmlTableID' );

// Optionally, include a Master Slide name for pre-defined margins, background, logo, etc.
```

```
pptx.addSlidesForTable( 'myHtmlTableID', { master:'MASTER_SLIDE' } );

// Optionally, add images/shapes/text/tables to each Slide
pptx.addSlidesForTable( 'myHtmlTableID', { addText:{ text:"Dynamic Title", options:{x:1, y:0.5, color:'0088CC'} } } )
pptx.addSlidesForTable( 'myHtmlTableID', { addImage:{ path:"images/logo.png", x:10, y:0.5, w:1.2, h:0.75 } } );
```

Creative Solutions

Design a Master Slide that already contains: slide layout, margins, logos, etc., then you can produce professional looking Presentations with a single line of code which can be embedded into a link or a button:

Add a button to a webpage that will create a Presentation using whatever table data is present:

```
<input type="button" value="Export to PPTX" onclick="{ var pptx=new PptxGenJS(); pptx.addSlidesForTable('tableId'); p
```

SharePoint Integration

Placing a button like this into a WebPart is a great way to add "Export to PowerPoint" functionality to SharePoint. (You'd also need to add the 4 `<script>` includes in the same or another WebPart)

Full PowerPoint Shape Library

If you are planning on creating Shapes (basically anything other than Text, Tables or Rectangles), then you'll want to include the `pptxgen.shapes.js` library.

The shapes file contains a complete PowerPoint Shape object array thanks to the officegen project.

```
<script lang="javascript" src="PptxGenJS/dist/pptxgen.shapes.js"></script>
```

Scheme Colors

Scheme color is a variable that changes its value whenever another scheme palette is selected. Using scheme colors, design consistency can be easily preserved throughout the presentation and viewers can change color theme without any text/background contrast issues.

To use a scheme color, set a color constant as a property value:

```
slide.addText('Hello',  { color: pptx.colors.TEXT1 });
```

The colors file contains a complete PowerPoint palette definition.

```
<script lang="javascript" src="PptxGenJS/dist/pptxgen.colors.js"></script>
```

Performance Considerations

It takes CPU time to read and encode images! The more images you include and the larger they are, the more time will be consumed. The time needed to read/encode images can be completely eliminated by pre-encoding any images (see below).

Pre-Encode Large Images

Pre-encode images into a base64 string (eg: 'image/png;base64,iVBORw[...]=') for use as the `data` option value. This will both reduce dependencies (who needs another image asset to keep track of?) and provide a performance boost (no time will need to be consumed reading and encoding the image).

Integration with Other Libraries

Integration with Angular

Set the browser mode option so the library will use blob file saving instead of detecting your app as a Node.js app (aka: avoid using `fs.writeFile`).

- `pptx.setBrowser(true);`

See Issue #220 for more information

Integration with Webpack/Typescript

- Add to webpack config to avoid a module resolution error: `node: { fs: "empty" }`
- Set browser mode so files will save as blobs via browser: `pptx.setBrowser(true);`

See Issue #72 for more information

Issues / Suggestions

Please file issues or suggestions on the issues page on github, or even better, submit a pull request. Feedback is always welcome!

When reporting issues, please include a code snippet or a link demonstrating the problem. Here is a small jsFiddle that is already configured and uses the latest PptxGenJS code.

Need Help?

Sometimes implementing a new library can be a difficult task and the slightest mistake will keep something from working. We've all been there!

If you are having issues getting a presentation to generate, check out the demos in the `examples` directory. There are demos for both Nodejs and client-browsers that contain working examples of every available library feature.

- Use a pre-configured jsFiddle to test with: PptxGenJS Fiddle
- Use Ask Question on StackOverflow - be sure to tag it with "PptxGenJS"

Version 2.0 Breaking Changes

Please note that version 2.0.0 enabled some much needed cleanup, but may break your previous code... (however, a quick search-and-replace will fix any issues).

While the changes may only impact cosmetic properties, it's recommended you test your solutions thoroughly before upgrading PptxGenJS to the 2.0 version.

All Users

The library `getVersion()` method is now a property: `version`

Option names are now caseCase across all methods:

- `font_face` renamed to `fontFace`
- `font_size` renamed to `fontSize`
- `line_dash` renamed to `lineDash`
- `line_head` renamed to `lineHead`
- `line_size` renamed to `lineSize`
- `line_tail` renamed to `lineTail`

Options deprecated in early 1.0 versions (hopefully nobody still uses these):

- `marginPt` renamed to `margin`

Node Users

Major Change

- `require('pptxgenjs')` no longer returns a singleton instance
- `pptx = new PptxGenJS()` will create a single, unique instance
- Advantage: Creating multiple presentations is much easier now - see Issue #83 for more).

Unimplemented Features

The PptxGenJS library is not designed to replicate all the functionality of PowerPoint, meaning several features are not on the development roadmap.

These include:

- Animations
- Importing Existing Presentations and/or Templates
- Outlines
- SmartArt

Special Thanks

- Officegen Project - For the Shape definitions and XML code
- Dzmitry Dulko - For getting the project published on NPM
- kajda90 - For the new Master Slide Layouts
- PPTX Chart Experts: kajda90, Matt King, Mike Wilcox
- Everyone who has submitted an Issue or Pull Request. :-)

Support Us

Do you like this library and find it useful? Add a link to the PptxGenJS project on your blog, website or social media.

Thanks to everyone who supports this project! <3

License