White Paper

Choosing the right charting component for your product

FusionCharts

Data to delight... in minutes

Abstract

Software products, whether hosted or delivered on-premise, assimilate and store ever-increasing amounts of data today. Despite this, most businesses are data-rich but information-poor. Even businesses that want to make data-driven business decisions are restricted by the limited reporting capabilities of the software products they use. As an ISV, if your customers are facing this problem too, you need to rethink the data visualization capabilities offered by your product.

Today, your end users need real-time access to relevant information pertaining to their business, even when they are on the move. Your products need to be able to process and visualize heaps of data in the form of reports or dashboards, which could be accessed on a variety of devices, be it computers, tablets or smartphones. Whether it's an ERP application or a social media analytics tool, your end-users expect your product to process and convert data into easily identifiable trends and patterns. To visualize these trends and patterns effectively, you need a charting component. A good charting component not only provides the correct visualization for the data but also builds an immersive experience for your end-user.

This white paper highlights the factors you should evaluate when choosing a charting component for your software product.

Questions to ask yourself before you start evaluating a charting component

Before you start evaluating a charting component, you should have a clear idea of what data you want to represent and how. The following questions should help you identify that:

- What data would you need to visualize in your application? Will the visualization provide something that raw data cannot?
- Will your visualizations be powered by historic data to drive strategic decisions or will they have real-time data for decision-making on the field?
- Which functions and roles would be consuming these visualizations? What would they derive from the visualization?
- What would be the right chart type to represent that data?
- What level of data do you want to represent? Will the reports show only summarized data or allow users to drill-down to more details?
- Which data entities are related to each other and can they be aggregated to provide more meaningful insights?
- What amount of flexibility should you give the users to view different perspectives of data, or slice and dice it?

Once these questions are answered, your next step is to start finding a charting component. It is recommended that the charting component be acquired right at the start of a development initiative, so that both the user interface and implementation can be planned in light of that. That way, developers get ample time to learn the implementation of the component and designers to understand the cosmetic capabilities of the component.

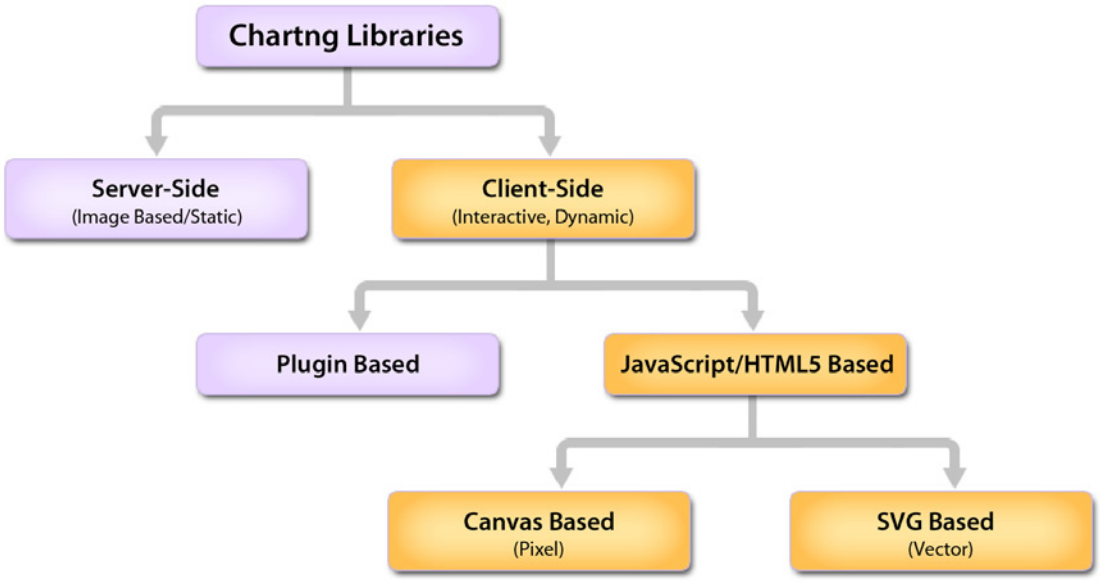
The best way to choose a charting component is to see what is available in the market, consider all the factors listed in the next section, find answers to them and finally make a decision.

Questions to ask during the evaluation of a charting component

- What devices, browsers and platforms does your product need to support?
- How easy is it to implement the component or library?
- What chart types will you need to show the correct representation of data?
- What reporting capabilities do you need to provide your users to empower their business decisions?
- What happens when you get stuck with a technical query during implementation? What support options does the vendor offer?

What devices, browsers and platforms does your product need to support?

In the early 1990s, when the web was still in its nascent stage, it was common to see *Best Viewed In* footers across the web specifying the set of browsers a website would be best rendered in. But today, when web applications are replacing desktop applications right from small business use cases to large enterprise-wide applications, the days of *Best Viewed In* are over. People want access to their applications on their PCs at work, on their tablets in meetings and on their smartphones on the move. As such your product and all its features need to be supported on a majority, if not all, devices and browsers.



There are two options that you can go with to get the pervasiveness required for your reporting:

1. Server-side charting libraries that generate image-based charts
2. Client-side charting libraries that (typically) generate interactive charts

Server-side charting libraries

These are platform-specific charting libraries that accept data on server through their APIs, create images to represent the chart and then stream these images as output to the user. The advantage of using these libraries is that they deliver the same experience on all devices by virtue of being images and if you are just using a chart or two in your product, they are light on the bandwidth as well. Examples of such components are Telerik (.NET), Infragistics (.NET), ComponentArt (.NET), ChartFX (Java and .NET), Steema (.NET), pChart (PHP) and jPGraph (PHP). Most of them are mature libraries, so they offer a wide range of chart types as well.

The disadvantage of using these libraries is that when you have a lot of charts in your reports and dashboards, they consume considerable resources on the server-side when generating the charts as images, and more so when you have concurrent users. Secondly, because they are image-based, they have little or no interactivity like drill-down, tooltips and clickable data points that most users have to come to expect by default in their reports. And lastly, because they are platform-specific, when you have to change technologies, you would have to switch to a different charting library that would result in a different feel and feature set for the charts.

Client-side charting libraries

The second option is to use client-side charting libraries using JavaScript, CSS, Flash, Silverlight or Java Applets. Considering that iOS devices do not support any proprietary plugins (Flash, Java and Silverlight), it is safe to ignore plugin-based components as that's a huge chunk of the market you wouldn't want to miss out on. Pure CSS components are very light-weight and can be used for very basic visualizations, so unless what you need is very basic, you can ignore them too.

That brings us to JavaScript/HTML5 based charting components, technologies that are called the future of the web. They have been adopted across a majority of devices and browsers, and have been embraced by industry leaders including Google, Apple and Microsoft, so the title is quite apt. The JavaScript/HTML5 charting components can again be divided under two heads:

- Canvas-based components
- SVG based components

In general, Canvas and SVG are both web technologies that allow you to render rich graphics inside the browser. SVG has been around since 1999 and Canvas since 2005. While they sound pretty similar, there is a very important fundamental difference between them — SVG is vector-based while Canvas is pixel-based, i.e. each shape drawn using SVG is remembered as an individual object whereas with Canvas, only the complete drawing is remembered by the system, not the individual shapes. So graphics rendered using SVG are better suited for web interfaces since they need to be interactive and dynamic as opposed to Canvas where for any kind of interactivity, the complete graphic will have to be redrawn. That makes SVG the right choice for creating interactive charts.

However, just like Canvas, older versions of IE — 6, 7 and 8 — do not support SVG. While most charting libraries have decided not to support these older browsers, over 6% of the world* (and over 15% in China) still use them. So it's not wise to ignore them as that would again mean a sizable dent in your market opportunity. To combat this situation, enterprise-grade charting libraries use VML, which is pervasively used in MS Office 2007 and from IE 5 to 9, for rendering exactly the same graphics within IE as well.

So an SVG-based library that uses VML as fallback for older versions of IE covers you on all major devices, browsers and platforms.

* Source: *w3schools.com*

How easy is it to implement the component or library?

Charting for you is a part of your product spectrum, an important part of it, but definitely not the entire spectrum. It is a means to an end and hence rapid developer adoption, agile implementation and quick results is what you need from the component you choose.

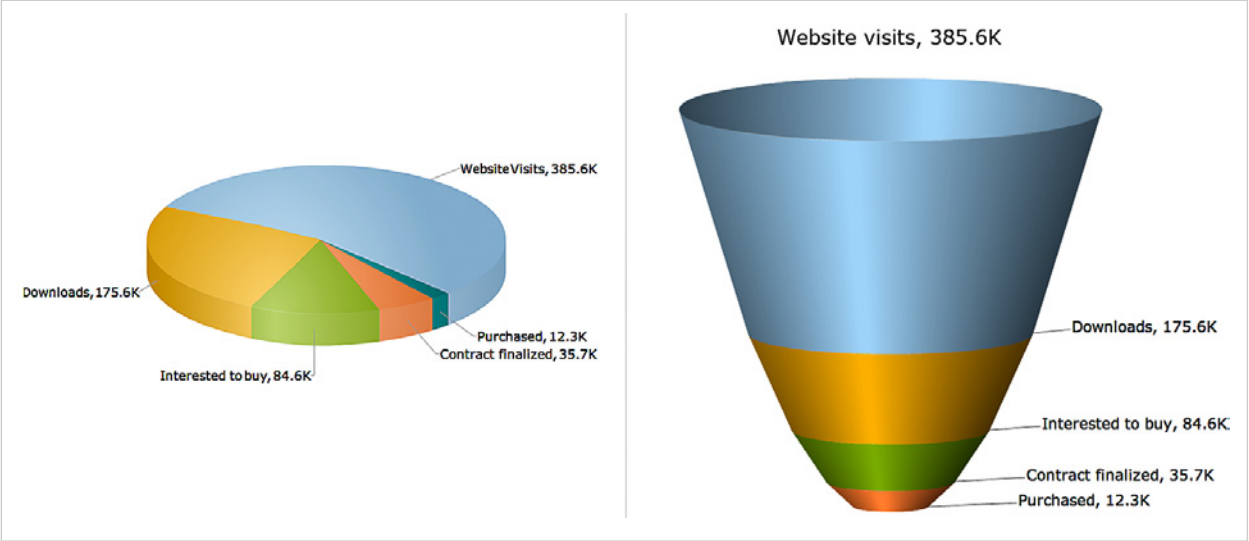
So it is vital to choose a charting component where the developer can get started just by copy-pasting a couple of lines of code. The charts should have the best usability practices built right in, so the developer doesn't have to spend any time on that.

Thereafter, to connect the chart to its data source and configure it to meet your requirements, the component must have a well-defined and easy-to-understand API. The documentation must have detailed API references covering all aspects of the product, have real-life examples to demonstrate the business use case of each feature, integration examples with different technologies and common troubleshooting issues.

Apart from the documentation, it is nice to have ready-to-use demos and sample applications that can be plugged right into to get a complete dashboard and serve as an inspiration for building a rich reporting experience.

What chart types will you need to show the correct representation of data?

Each chart type facilitates a certain type of data analysis. For example, line charts are used to see trends over time (change in employment rates over the last decade) and pie charts to understand how a data value breaks down into its constituents (breakdown of website traffic into individual sources). But none of them are apt for showing phased reduction in a leads management application, which is best represented by the funnel chart.



Pie charts are inappropriate for showing phased reduction – wrong or makeshift choices like these are often made due to unavailability of specialized chart types

Therefore, it is critical to know the different types of data analysis you will need to facilitate, and the charts needed for each of them, so that you can check for their availability in the component you are evaluating. While most vendors offer the basic charts like column, line, bar and pie, advanced charts like funnel, radar, waterfall, heat map, candlestick, Gantt and speedometer charts are supported only by enterprise-grade charting components.

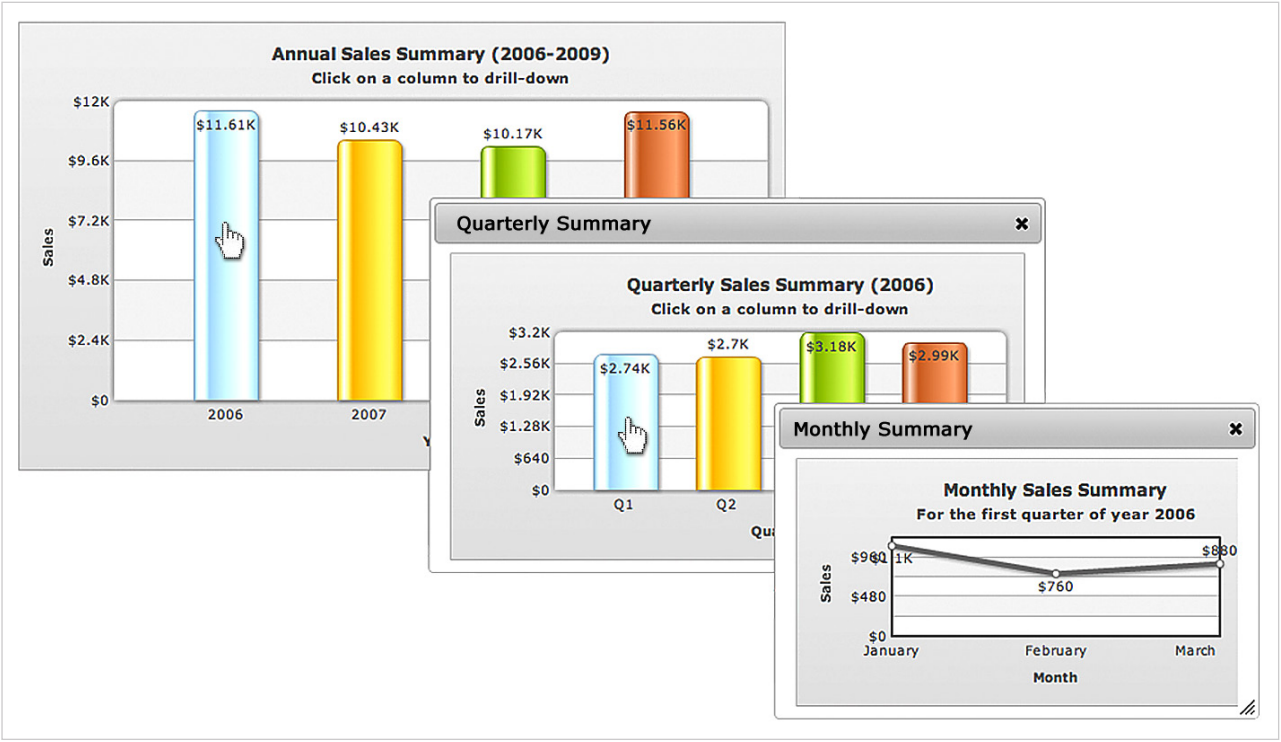
What kind of reporting capabilities will the customers expect in your product?

A rich reporting experience is a must for any application that deals with heaps of business data and a power-packed charting component helps facilitate that for your users. Depending on how you want that experience to be, you will require specific features in your charting component.

Will you just show summarized data to your users or do they also need to drill-down to see further details?

Drill-down allows presentation of large data sets without overwhelming the users. When a user clicks on a data value, you have to register that click, fetch the detailed data from the database, create a new chart to show the detailed data and give the user an option to return to the parent chart.

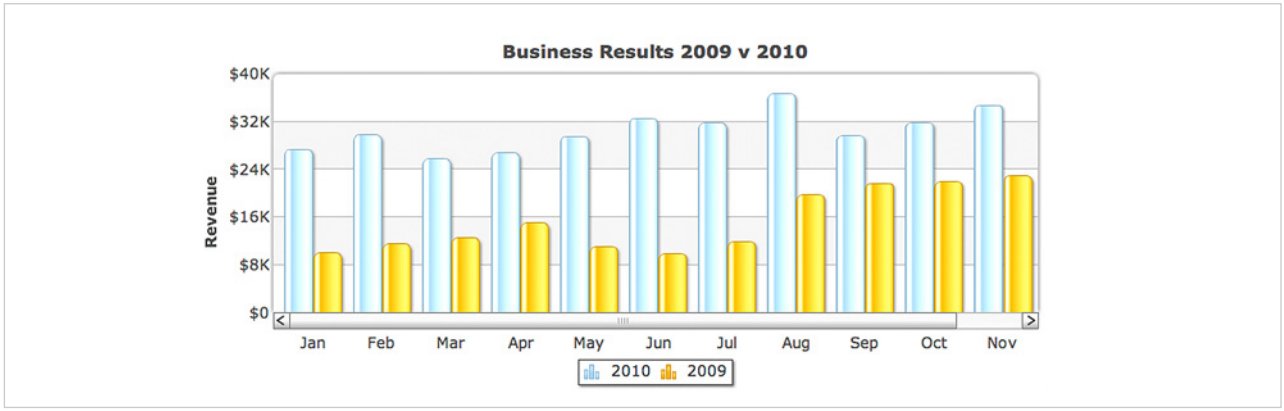
For simple drill-downs, this is not difficult to implement. But for complex multi-level drill-downs, the process can get very cumbersome in which case a charting component having in-built drill-down capabilities is required. The only thing your developer has to do is set the number of levels of drill-down required and provide the data associated with all the levels, and everything else will be taken care of by the charting component in a very user-friendly manner.



Complex drill-downs can get cumbersome to implement. So it is best to choose a charting component that has in-built drill-down capabilities

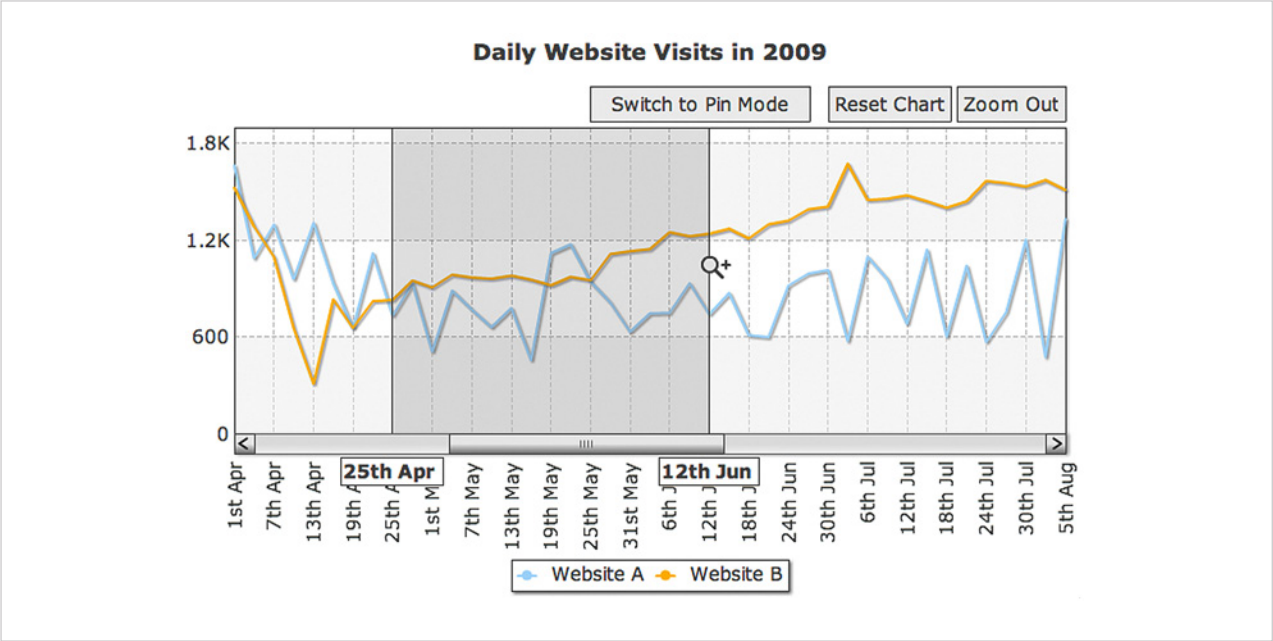
Would you need to plot large amounts of data on the charts, with zooming and scrolling capabilities?

To plot large quantities of data without compromising readability, some vendors provide charting components with scroll and zoom capabilities. This is very handy in reports, dashboards and monitors where a lot of data points need to be plotted on a single screen with limited screen real estate.



Scroll charts are used to display large quantities of data in a compact space; typically used in reports, dashboards and monitors

The zoom feature, relevant to line and area charts, allows data analysis by zooming into a segment of data by selecting it on the chart. Typically these charts plot massive data sets in the order of tens of thousands of data points, so the component should be able to plot that without any performance issues.



The zoom capability provided by client-side charting components allows analysis of data at both macroscopic and microscopic level by zooming into a selected data segment

Another consideration you need to keep in mind with zooming charts is the handling of the data labels on the x-axis. In many charting components, the category labels start overflowing or overlapping when there are a large number of data points. So you should find a component that can intelligently handle data labels by smartly truncating, wrapping or rotating them.

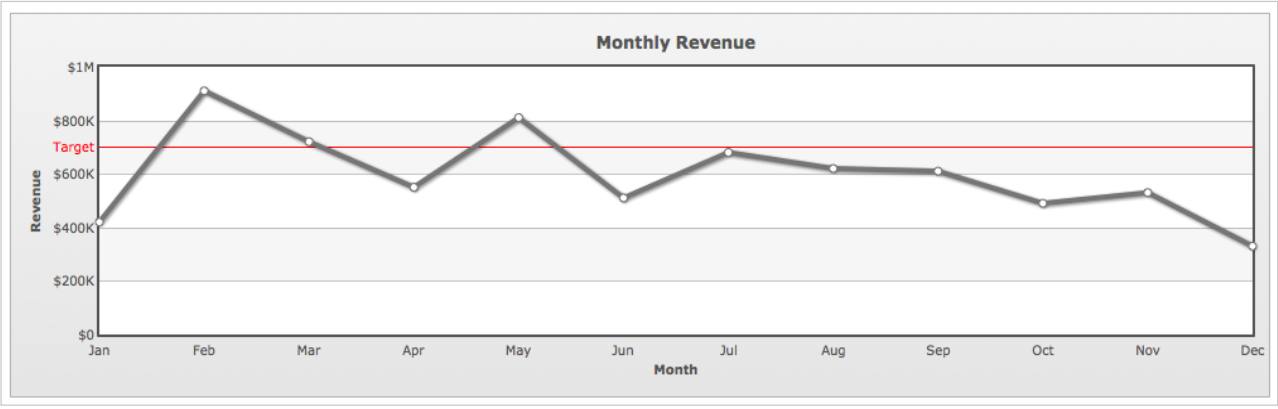
Do you need to format the numbers on the chart in a specific way?

Appropriate formatting of numbers is essential to facilitate easier comparison of data. The charting component you select should be able to convert large numbers to their appropriate formatting using commonly used number scales such as K (thousands) and M (Millions). The component should also allow you to create custom number scales as required by your product and then format the numbers according to that scale.

Number formatting makes the charts more intelligible by converting the input data into easy-to-comprehend numbers – converting input data in seconds to minutes, hours and weeks; converting KB to MB, GB and TB etc.

Do you need to add trendlines to the chart?

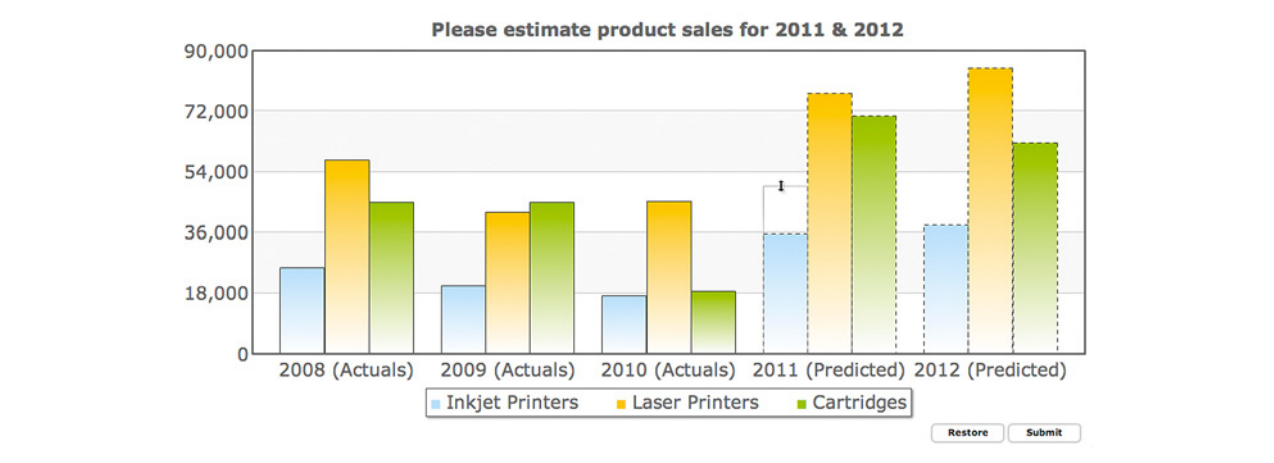
Trendlines are horizontal lines that span the entire width of the chart at a specified point on the chart scale and they add context to the data and make data analysis more meaningful. For a monthly revenue chart, showing average monthly sales from last year or the target for the current year adds much-needed context to the chart and makes the analysis much more meaningful.



Adding trend-lines add context to the chart thereby allowing for a for meaningful analysis

Would you want your users to be able to visually edit the data on the chart?

Visual data editing is needed if your product needs to enable planning or simulation of any sort like sales forecasts or project timeline estimations. The users can change the data on the chart and then submit that back to the database. Some vendors provide editable versions of standard charts like column and line, and gauges like dial (speedometer) which can facilitate visual data editing in your product.



Editable charts are used in planning and decision-making to allow the end user to make changes to the chart directly without having to touch the underlying data

What happens when you get stuck with a technical query during implementation? What support options does the vendor offer?

During the development of a product, engineering teams work under immense pressure with tight deadlines. So when a developer runs into a technical issue, which can happen even with the best of components, finding a fix on community forums and technical blogs can be time-consuming. And sometimes, there are just no solutions available. In the middle of a product development cycle, that is valuable engineering time wasted.

This is a major issue with open-source charting components that don't have a large and active community around them. As such, it is very important to find a vendor who provides personalized technical support as well, typically with a guaranteed turnaround time.

For vendors providing personalized technical support, having an active community is a bonus. The community contributes extensions around the component for specific technologies or usage, shares tips and best practices that can help you make the most of the component.

Conclusion

Choosing a charting component for any organization is a complex task because the selection will have to be made considering both the present and future needs of the organization. You will have to evaluate the alternatives keeping in mind the user experience of your application, ever-changing nature of the technological landscape, the changing needs of your end users and your development team's expertise. The factors described in this white paper will help you understand your options clearly, and you should be able to select the right charting product for your organization.

About FusionCharts

FusionCharts Suite XT is the industry's leading enterprise-grade charting component with delightful JavaScript charts that work across devices, browsers (including IE 6, 7 and 8) and platforms. Using it, you can create your first chart in under 15 minutes and then choose from 90+ chart types and 965+ maps to give your data the best face. It comes with 1,500+ pages of exhaustive documentation, plug-and-play versions of real-world demos and personalized tech support.

21,000 customers and 450,000 developers in 118 countries and Fortune 500 companies like LinkedIn, Microsoft, Cisco, GE, AT&T and World Bank use FusionCharts Suite XT to go from data to delight in minutes.

Learn more about how you can add delight to your products at www.fusioncharts.com