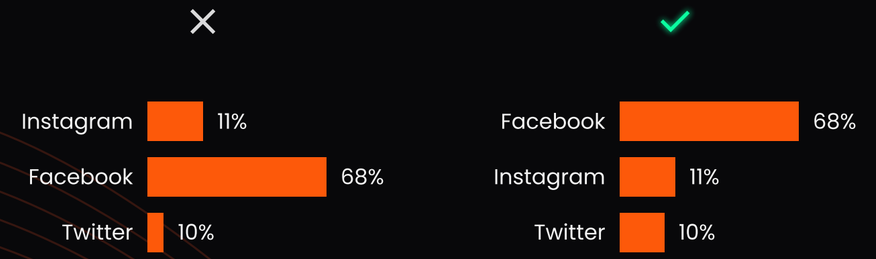
Data Visualization Best Practices for Scaling Businesses

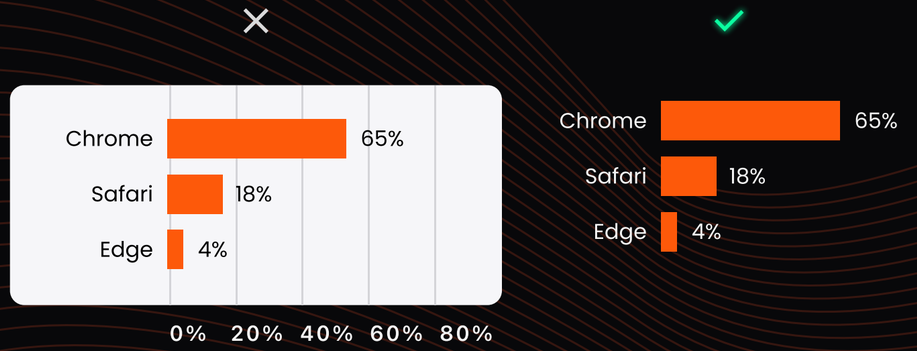
Design Best Practices
for Data Visualization

Design with Clarity and Simplicity

- Highlight the most important data and remove unnecessary details.
- Arrange data logically to tell a clear story and guide users toward insights.
- Use size, color, and placement to draw attention to key points.



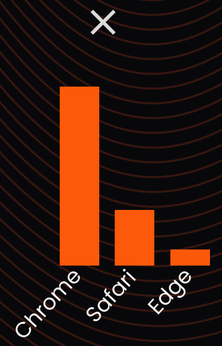
The labels should be order from least to most frequent.

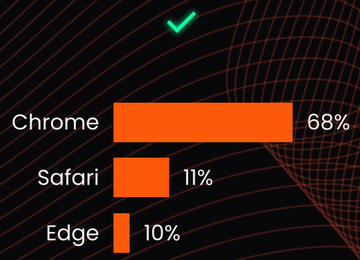


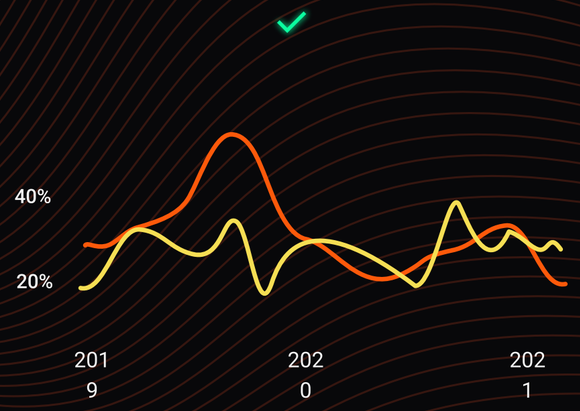
Remove unnecessary styling to improve clarity.

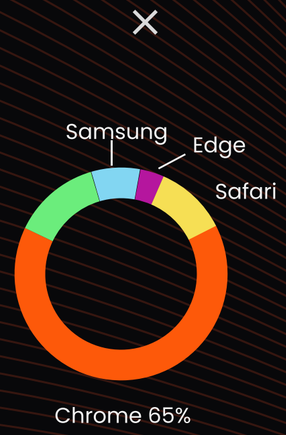
Choose the Right Chart Type

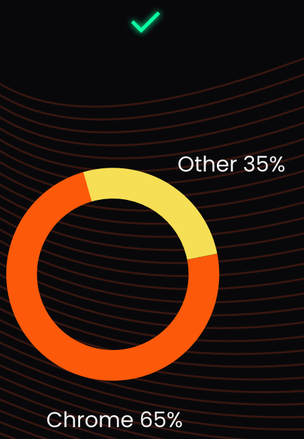
- Bar Charts: Ideal for comparisons; horizontal bars work well for long labels.
- Line Charts: Best for trends over time with fewer than five lines; highlight key data in bold.
- Pie/Donut Charts: Use for part-to-whole comparisons, avoiding complex data.





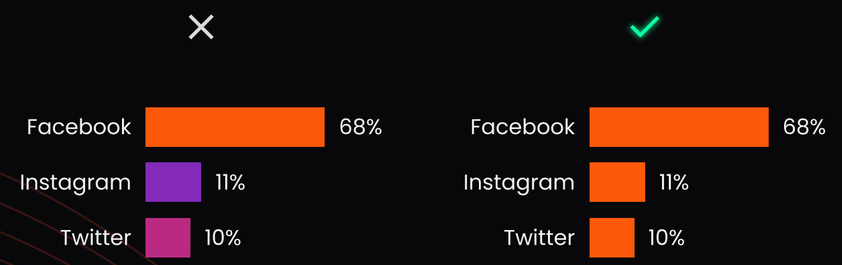




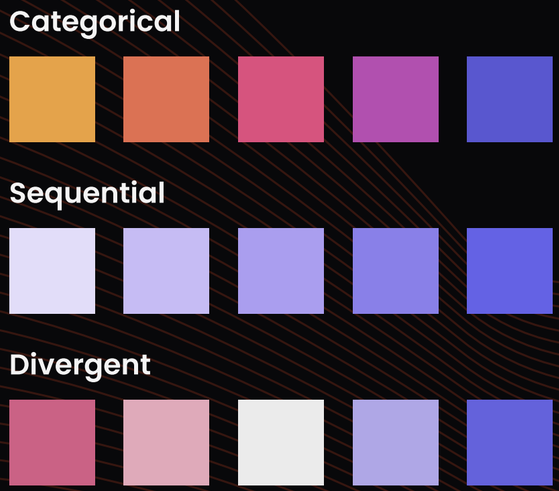


Color with Purpose

- Use color sparingly to highlight key information or differences.
- Limit the number of colors to make visual distinctions easier.
- Be mindful of cultural differences in color meanings, especially in financial contexts.



If color doesn't service a purpose, don't use it.



Three types of color schemes when designing with data.

Tables: Charts' Trusted Sidekick

Layout and Structure: Tables need clear headers and descriptions to help users understand their purpose. Fixed headers and frozen columns keep context intact when scrolling, while responsive designs allow for horizontal scrolling and customizable column arrangements.

Data Organization and Presentation: Effective tables use text styles and contrast to highlight hierarchy and ensure headers stand out. Align text to the left and numbers to the right for better readability, and use zebra striping or hover effects to improve scannability.

Functionality and Interaction: Interactive tables allow users to sort columns, apply filters, and search for specific data with ease. Expandable rows reveal additional details without clutter, and bulk actions make tasks like selecting multiple rows more efficient.

Views, Defaults, & Data Manipulation

Views

- Let users toggle between tables (detailed data) and charts (trends/summaries).
- Suggest the best view based on data or user goals.

Defaults

- Display relevant data with pre-set filters (e.g., recent or top categories).
- Allow easy reset or customization of defaults.

Data Manipulation

- Support multi-filters with real-time updates and tooltips for guidance.
- Include robust search and sorting by key attributes (e.g., alphabetical, numerical).

Development Best Practices
for Data Visualization

Choosing a Data Visualization Library

Define Needs: Match the library to your data complexity, visualization type, and interactivity needs.

Performance: Use optimized libraries like ECharts for large datasets; canvas for heavy visuals, SVG for smaller ones.

Ease of Use: Pick tools with good docs, community support, and compatibility with your tech stack.

Interactivity: Choose Plotly or D3.js for interactive, responsive designs.

Budget: Compare open-source (e.g., D3.js) vs. paid tools (e.g., Highcharts) and check licensing terms.

Library	Best At	Good At	Not Good At
D3.js	Highly customizable and powerful for creating unique, interactive visualizations	Handling complex, large datasets and data transformations	Steep learning curve; requires significant development time
Chart.js	Simple, quick-to-use visualizations for common chart types	Lightweight and easy to integrate into projects	Limited customization and interactivity; not ideal for large datasets
Highcharts	Professional-looking, interactive charts with minimal effort	Great documentation and support for exporting charts	Requires a commercial license for most use cases; limited flexibility compared to D3.js
Plotly	Interactive and analytical visualizations, including 3D and multi-axis charts	Python integration and dashboards via Dash	Can be slower with very large datasets; commercial license needed for advanced features in some contexts
ECharts	High performance for large datasets with built-in interactivity	Easy-to-use for common chart types with advanced features	Limited flexibility for custom designs compared to D3.js

D3.js

Best For:

Developers who need full control to build highly customized, unique, and interactive visualizations.

Strengths:

- Extensive flexibility for creating any type of visualization.
- Excellent for handling complex datasets with powerful data manipulation capabilities.
- Great for projects where creativity and bespoke designs are critical.

Weaknesses:

- Requires deep JavaScript knowledge; not beginner-friendly.
- Development time is significantly longer compared to higher-level libraries.
- No built-in components; everything is built from scratch.

Ideal Use Cases:

- Custom dashboards for enterprise clients.
- Visualizing highly specialized or uncommon data structures.
- Research and experimental data visualizations.

Chart.js

Best For:

Simple, common charts like bar, line, and pie with minimal setup and effort.

Strengths:

- Easy to learn and implement, especially for smaller projects.
- Lightweight and integrates seamlessly into JavaScript-based projects.
- Offers responsive visualizations out of the box.

Weaknesses:

- Limited interactivity and customization options.
- Struggles with performance for very large datasets.
- Not designed for highly advanced or complex visualizations.

Ideal Use Cases:

- Quick prototypes or MVPs.
- Simple analytics dashboards for web apps.
- Educational projects or proof-of-concept designs.

Highcharts

Best For:

Professional, polished, and interactive charts that are ready to go with minimal customization.

Strengths:

- Comes with a wide range of prebuilt charts and strong default styling.
- Easy-to-use API and great documentation reduce development time.
- Supports exporting charts to PNG, PDF, or SVG.

Weaknesses:

- Requires a commercial license for non-personal use.
- Less flexible than D3.js for creating completely custom designs.

Ideal Use Cases:

- Business intelligence dashboards with export capabilities.
- Web applications needing high-quality, interactive charts with minimal effort.
- Enterprise projects where licensing costs are not a concern.

Plotly

Best For:

Interactive and analytical visualizations, especially in Python-based workflows.

Strengths:

- Advanced interactivity with features like zoom, hover, and drag.
- Supports 3D visualizations, multi-axis charts, and statistical plotting.
- Tight integration with Python via the Dash framework.

Weaknesses:

- Performance can lag with very large datasets.
- Some advanced features require a commercial license.
- Less customizability compared to D3.js for visual styling.

Ideal Use Cases:

- Data science projects and notebooks.
- Analytical dashboards with Python.
- 3D visualizations or geospatial visualizations.

ECharts

Best For:

Projects with large datasets needing high performance and built-in interactivity.

Strengths:

- Excellent performance with large datasets due to its canvas-based rendering.
- Rich set of built-in features, including animations and interactive filtering.
- Easy-to-use API for creating sophisticated charts with minimal effort.

Weaknesses:

- Customization is somewhat limited compared to D3.js.
- Smaller community and fewer third-party resources compared to libraries like Plotly or Chart.js.

Ideal Use Cases:

- Enterprise dashboards dealing with large, dynamic datasets.
- Real-time monitoring tools or IoT applications.
- Projects where performance is critical but customization is less important.

How to Choose

- Choose D3.js if you need complete flexibility and are willing to invest time in development.
- Opt for Chart.js for quick, simple projects or MVPs that don’t need complex interactivity.
- Select Highcharts for business applications where polished visuals and export capabilities are essential.
- Use Plotly for data science or analytical projects, especially when working with Python.
- Pick ECharts when you need to handle large datasets efficiently with built-in interactivity.

Thank you.
Muchas gracias.