📊 Chart.js Complete Detailed Guide

Table of Contents

What is Chart.js?

Chart.js is a JavaScript library that helps you create beautiful, interactive charts on your website.

Think of it like this:

- You have data: [10, 20, 30, 40]
- Chart.js turns it into visual charts (bars, lines, pies, etc.)
- Users can see your data in a beautiful, easy-to-understand way

---

Setup

Step 1: Include Chart.js Library

You need to load Chart.js before you can use it. Add this line in your HTML `<head>` section:

```
html
```

```
<!DOCTYPE html>
<html>
<head>
  <title>My Charts</title>
  <!-- ✅ This loads Chart.js from the internet -->
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
  <!-- Your content here -->
</body>
</html>
```

What this does:

- Loads Chart.js library from a CDN (Content Delivery Network)
- Makes the `Chart` object available to use in your JavaScript
- No need to download anything!

Basic Concepts

1. Canvas Element

Charts are drawn on an HTML `<canvas>` element.

```
html

<canvas id="myChart"></canvas>
```

Explanation:

- `<canvas>` is like a blank drawing board
- Chart.js will draw the chart on this canvas
- `id="myChart"` gives it a unique name so JavaScript can find it

Better with a container:

```
html
```

```
<div style="width: 600px; height: 400px;">
  <canvas id="myChart"></canvas>
</div>
```

Why?

- The container controls the size of your chart
- Without it, chart might be too big or too small

2. Getting the Context

Before creating a chart, you need to get the "2d context" of the canvas:

```javascript
const ctx = document.getElementById('myChart').getContext('2d');
```

Breaking it down:

- `document.getElementById('myChart')` - Finds the canvas with id="myChart"
- `.getContext('2d')` - Tells the canvas we want to draw 2D graphics
- `ctx` - Short for "context", this is what Chart.js needs to draw

---

Creating Your First Chart

Complete Example with Explanation

```html
```

```
<!DOCTYPE html>
<html>
<head>
    <title>My First Chart</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
    <h1>Sales Chart</h1>

    <!-- Container for the chart -->
    <div style="width: 600px; margin: 50px auto;">
        <canvas id="myChart"></canvas>
    </div>

    <script>
        // STEP 1: Get the canvas context
        const ctx = document.getElementById('myChart').getContext('2d');

        // STEP 2: Create the chart
        const myChart = new Chart(ctx, {
            // ═══════════════════════════════════════════
            // Chart Type
            // ═══════════════════════════════════════════
            type: 'bar', // This makes it a bar chart

            // ═══════════════════════════════════════════
            // Data Configuration
            // ═══════════════════════════════════════════
            data: {
                // X-axis labels (what shows at the bottom)
                labels: ['January', 'February', 'March', 'April', 'May'],

                // The actual data to display
                datasets: [{
                    label: 'Monthly Sales',       // Legend label
                    data: [12, 19, 3, 5, 8],      // The numbers to show
                    backgroundColor: 'rgba(54, 162, 235, 0.5)', // Bar color
                    borderColor: 'rgba(54, 162, 235, 1)',       // Border color
                    borderWidth: 2                              // Border thickness
                }]
            },

            // ═══════════════════════════════════════════
```

```
        // Chart Options
        //═══════════════════════════════════════════════
        options: {
          responsive: true,  // Makes chart resize with window
          scales: {
            y: {
              beginAtZero: true  // Y-axis starts at 0
            }
          }
        }
      });
    </script>
</body>
</html>
```

Understanding Each Part:

1. Chart Type

```
javascript

type: 'bar'
```

- `'bar'` = Vertical bars
- `'line'` = Line graph
- `'pie'` = Pie chart
- `'doughnut'` = Donut chart

2. Labels (X-axis)

```
javascript

labels: ['January', 'February', 'March']
```

- These appear on the bottom of the chart
- Can be anything: months, names, categories, etc.

3. Datasets

```
javascript
```

```
datasets: [{
  label: 'Monthly Sales',
  data: [12, 19, 3, 5, 8]
}]
```

- `label` = Name that appears in the legend
- `data` = Array of numbers to display
- Each number corresponds to a label

Example:

- January → 12
- February → 19
- March → 3

4. Colors

```
javascript

backgroundColor: 'rgba(54, 162, 235, 0.5)'
borderColor: 'rgba(54, 162, 235, 1)'
```

Understanding RGBA:

- `rgba(Red, Green, Blue, Alpha)`
- Red, Green, Blue: 0-255
- Alpha: 0-1 (transparency)
  - 0 = fully transparent
  - 1 = fully opaque
  - 0.5 = 50% transparent

Examples:

- Red: `rgba(255, 0, 0, 1)`
- Green: `rgba(0, 255, 0, 1)`
- Blue: `rgba(0, 0, 255, 1)`

- Semi-transparent blue: `rgba(0, 0, 255, 0.5)`

Chart Types in Detail

1. Bar Chart (Vertical Bars)

javascript

```
new Chart(ctx, {
    type: 'bar',
    data: {
        labels: ['Red', 'Blue', 'Yellow'],
        datasets: [{
            label: 'Colors',
            data: [12, 19, 7]
        }]
    }
});
```

Use for:

- Comparing different categories
- Showing quantities
- Monthly sales, department counts, etc.

2. Horizontal Bar Chart

javascript

```javascript
new Chart(ctx, {
    type: 'bar',
    data: {
        labels: ['Product A', 'Product B', 'Product C'],
        datasets: [{
            label: 'Sales',
            data: [50, 75, 30]
        }]
    },
    options: {
        indexAxis: 'y'  // ← This makes it horizontal!
    }
});
```

Use for:

- Long category names
- Rankings
- Comparisons

3. Line Chart

```javascript
new Chart(ctx, {
    type: 'line',
    data: {
        labels: ['Jan', 'Feb', 'Mar', 'Apr', 'May'],
        datasets: [{
            label: 'Temperature',
            data: [20, 22, 25, 23, 27],
            borderColor: 'rgb(255, 99, 132)',
            tension: 0.4  // Makes line curved
        }]
    }
});
```

Use for:

- Trends over time
- Stock prices

- Temperature changes
- Progress tracking

4. Pie Chart

javascript

```javascript
new Chart(ctx, {
    type: 'pie',
    data: {
        labels: ['Red', 'Blue', 'Yellow'],
        datasets: [{
            data: [300, 50, 100],
            backgroundColor: [
                'rgb(255, 99, 132)',
                'rgb(54, 162, 235)',
                'rgb(255, 205, 86)'
            ]
        }]
    }
});
```

Use for:

- Showing percentages
- Parts of a whole
- Market share
- Status distribution

5. Doughnut Chart

javascript

```
new Chart(ctx, {
    type: 'doughnut',  // Same as pie but with hole in middle
    data: {
        labels: ['Processing', 'Completed', 'Waiting'],
        datasets: [{
            data: [10, 5, 3],
            backgroundColor: [
                '#4CAF50',
                '#2196F3',
                '#FF9800'
            ]
        }]
    }
});
```

Use for:

- Same as pie chart
- Looks more modern
- Center can show total count

Customization Options

1. Chart Title

javascript

```
options: {
    plugins: {
        title: {
            display: true,
            text: 'Monthly Sales Report',
            font: {
                size: 20
            }
        }
    }
}
```

2. Legend Position

```javascript
options: {
  plugins: {
    legend: {
      position: 'bottom'  // 'top', 'left', 'right', 'bottom'
    }
  }
}
```

3. Hide Legend

```javascript
options: {
  plugins: {
    legend: {
      display: false
    }
  }
}
```

4. Grid Lines

```javascript
options: {
  scales: {
    y: {
      grid: {
        display: false  // Hides horizontal grid lines
      }
    },
    x: {
      grid: {
        display: false  // Hides vertical grid lines
      }
    }
  }
}
```

5. Colors for Each Bar

```javascript
datasets: [{
  label: 'Colors',
  data: [12, 19, 7, 5],
  backgroundColor: [
    'rgba(255, 99, 132, 0.5)',   // Red for bar 1
    'rgba(54, 162, 235, 0.5)',   // Blue for bar 2
    'rgba(255, 206, 86, 0.5)',   // Yellow for bar 3
    'rgba(75, 192, 192, 0.5)'    // Green for bar 4
  ]
}]
```

Working with Firebase Data

Example: Status Distribution Chart

```javascript
```

```
import { getDatabase, ref, onValue } from 'firebase/database';

const db = getDatabase();
const requestsRef = ref(db, 'Maintenance');

onValue(requestsRef, (snapshot) => {
    const data = snapshot.val();

    // Convert Firebase object to array
    const requests = Object.values(data);

    //═══════════════════════════════════════════════
    // Count each status
    //═══════════════════════════════════════════════
    const statusCounts = {};

    requests.forEach(request => {
        const status = request.status;  // e.g., "Processing"

        // If status doesn't exist in object, set to 0, then add 1
        statusCounts[status] = (statusCounts[status] || 0) + 1;
    });

    // Result: { "Processing": 5, "Completed": 3, "Waiting": 2 }

    //═══════════════════════════════════════════════
    // Create arrays for Chart.js
    //═══════════════════════════════════════════════
    const labels = Object.keys(statusCounts);    // ["Processing", "Completed", "Waiting"]
    const values = Object.values(statusCounts);  // [5, 3, 2]

    //═══════════════════════════════════════════════
    // Create the chart
    //═══════════════════════════════════════════════
    const ctx = document.getElementById('statusChart').getContext('2d');

    new Chart(ctx, {
        type: 'doughnut',
        data: {
            labels: labels,
            datasets: [{
                data: values,
                backgroundColor: [
```

```
            '#4CAF50',  // Green
            '#2196F3',  // Blue
            '#FF9800'   // Orange
          ]
        }]
      }
    });
});
```

Detailed Breakdown:

Step 1: Get Data from Firebase

```javascript
onValue(requestsRef, (snapshot) => {
    const data = snapshot.val();
    const requests = Object.values(data);
```

- `onValue` listens for data changes
- `snapshot.val()` gets the data
- `Object.values()` converts to array

Step 2: Count Statuses

```javascript
const statusCounts = {};

requests.forEach(request => {
    const status = request.status;
    statusCounts[status] = (statusCounts[status] || 0) + 1;
});
```

How this works:

```javascript
```

```
// Starting with empty object: {}

// Request 1: status = "Processing"
statusCounts["Processing"] = (undefined || 0) + 1 = 1
// Result: { "Processing": 1 }

// Request 2: status = "Processing"
statusCounts["Processing"] = (1 || 0) + 1 = 2
// Result: { "Processing": 2 }

// Request 3: status = "Completed"
statusCounts["Completed"] = (undefined || 0) + 1 = 1
// Result: { "Processing": 2, "Completed": 1 }
```

Step 3: Convert to Arrays

```javascript
const labels = Object.keys(statusCounts);
const values = Object.values(statusCounts);
```

If `statusCounts = { "Processing": 5, "Completed": 3 }`

Then:

- `labels = ["Processing", "Completed"]`
- `values = [5, 3]`

Common Examples

Example 1: Count Requests by Department

```javascript
```

```
onValue(requestsRef, (snapshot) => {
    const requests = Object.values(snapshot.val());

    // Count departments
    const deptCounts = {};
    requests.forEach(req => {
        const dept = req.department || 'N/A';
        deptCounts[dept] = (deptCounts[dept] || 0) + 1;
    });

    // Sort and get top 5
    const sorted = Object.entries(deptCounts)
        .sort((a, b) => b[1] - a[1])  // Sort by count (descending)
        .slice(0, 5);                 // Take first 5

    // Create chart
    const ctx = document.getElementById('deptChart').getContext('2d');
    new Chart(ctx, {
        type: 'bar',
        data: {
            labels: sorted.map(item => item[0]),  // Department names
            data: sorted.map(item => item[1])     // Counts
        }
    });
});
```

Example 2: Requests Over Time

javascript

```
onValue(requestsRef, (snapshot) => {
    const requests = Object.values(snapshot.val());

    // Group by month
    const monthlyCounts = {};

    requests.forEach(req => {
        const date = new Date(req.timestamp);
        const month = date.getMonth() + 1;  // 1-12
        const year = date.getFullYear();
        const key = `${month}/${year}`;     // "12/2025"

        monthlyCounts[key] = (monthlyCounts[key] || 0) + 1;
    });

    // Create line chart
    const ctx = document.getElementById('timeChart').getContext('2d');
    new Chart(ctx, {
        type: 'line',
        data: {
            labels: Object.keys(monthlyCounts),
            datasets: [{
                label: 'Requests per Month',
                data: Object.values(monthlyCounts),
                borderColor: 'rgb(75, 192, 192)',
                tension: 0.1
            }]
        }
    });
});
```

Updating Charts

Problem: Data Changes

If your Firebase data updates, you need to update the chart.

Solution 1: Destroy and Recreate

```
javascript
```

```javascript
let myChart = null;  // Store chart reference

function createChart(data) {
    // Destroy old chart if it exists
    if (myChart) {
        myChart.destroy();
    }

    // Create new chart
    const ctx = document.getElementById('myChart').getContext('2d');
    myChart = new Chart(ctx, {
        type: 'bar',
        data: data
    });
}

// Use it
onValue(requestsRef, (snapshot) => {
    const newData = processData(snapshot.val());
    createChart(newData);
});
```

Solution 2: Update Chart Data

```javascript
// Update existing chart
myChart.data.labels = newLabels;
myChart.data.datasets[0].data = newData;
myChart.update();  // Refresh the chart
```

Styling Tips

1. Container Sizing

```html
<div style="width: 80%; max-width: 800px; margin: 0 auto; padding: 20px;">
    <canvas id="myChart"></canvas>
</div>
```

2. Card Style

```html
<div style="
    background: white;
    padding: 20px;
    border-radius: 12px;
    box-shadow: 0 2px 8px rgba(0,0,0,0.1);
    margin: 20px;
">
    <h2>Chart Title</h2>
    <canvas id="myChart"></canvas>
</div>
```

3. Responsive Height

```javascript
options: {
    responsive: true,
    maintainAspectRatio: false  // Allows custom height
}
```

```html
<div style="height: 400px;">
    <canvas id="myChart"></canvas>
</div>
```

Common Colors

```javascript
```

```
// Professional Colors
const colors = {
    blue: 'rgba(54, 162, 235, 0.7)',
    green: 'rgba(75, 192, 192, 0.7)',
    red: 'rgba(255, 99, 132, 0.7)',
    orange: 'rgba(255, 159, 64, 0.7)',
    purple: 'rgba(153, 102, 255, 0.7)',
    yellow: 'rgba(255, 205, 86, 0.7)'
};

// Status Colors
const statusColors = {
    approved: '#FFD700',     // Gold
    processing: '#4CAF50',   // Green
    waiting: '#FF9800',      // Orange
    rejected: '#F44336',     // Red
    completed: '#2196F3'     // Blue
};
```

Troubleshooting

Chart not showing?

1. **Check console for errors** (F12 in browser)
2. **Make sure Chart.js is loaded**

```
html

  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
```

3. **Canvas has an ID**

```
html

  <canvas id="myChart"></canvas>
```

4. **Script runs after page loads**

```javascript
window.addEventListener('load', function() {
  // Create chart here
});
```

Chart too small/big?

Set container size:

```html
<div style="width: 600px; height: 400px;">
  <canvas id="myChart"></canvas>
</div>
```

Data not updating?

Make sure to destroy old chart:

```javascript
if (myChart) myChart.destroy();
```

Summary

3 Steps to Create a Chart:

1. **Add Chart.js**

```html
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
```

2. **Add Canvas**

```html
<canvas id="myChart"></canvas>
```

3. **Create Chart**

```javascript
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'bar',
    data: { ... }
});
```

That's it! 🎉