// Type definitions for Chart.js 2.9

// Project: https://github.com/nnnick/Chart.js, https://www.chartjs.org

// Definitions by: Alberto Nuti <https://github.com/anuti>

// Fabien Lavocat <https://github.com/FabienLavocat>

// KentarouTakeda <https://github.com/KentarouTakeda>

// Larry Bahr <https://github.com/larrybahr>

// Daniel Luz <https://github.com/mernen>

// Joseph Page <https://github.com/josefpaij>

// Dan Manastireanu <https://github.com/danmana>

// Guillaume Rodriguez <https://github.com/guillaume-ro-fr>

// Simon Archer <https://github.com/archy-bold>

// Ken Elkabany <https://github.com/braincore>

// Francesco Benedetto <https://github.com/frabnt>

// Alexandros Dorodoulis <https://github.com/alexdor>

// Manuel Heidrich <https://github.com/mahnuh>

// Conrad Holtzhausen <https://github.com/Conrad777>

// Adrián Caballero <https://github.com/adripanico>

// wertzui <https://github.com/wertzui>

// Martin Trobäck <https://github.com/lekoaf>

// Elian Cordoba <https://github.com/ElianCordoba>

// Takuya Uehara <https://github.com/indigolain>

// Ricardo Mello <https://github.com/ricmello>

// Ray Nicholus <https://github.com/rnicholus>

// Oscar Cabrera <https://github.com/mrjack88>

// Carlos Anoceto <https://github.com/canoceto>

// Nobuhiko Futagami <https://github.com/nobu222>

// Marco Ru <https://github.com/Marcoru97>

// Tony Liu <https://github.com/tonybadguy>

// Mathias Helminger <https://github.com/Ilmarinen100>

// Mostafa Sameti <https://github.com/IVIosi>

// Samar Mohan <https://github.com/samarmohan>

// Definitions: https://github.com/DefinitelyTyped/DefinitelyTyped

// TypeScript Version: 2.3

import { Moment } from 'moment';

declare class Chart {

static readonly Chart: typeof Chart;

constructor(

context: string | CanvasRenderingContext2D | HTMLCanvasElement | ArrayLike<CanvasRenderingContext2D | HTMLCanvasElement>,

options: Chart.ChartConfiguration

);

config: Chart.ChartConfiguration;

data: Chart.ChartData;

destroy: () => {};

update: ({duration, lazy, easing}?: Chart.ChartUpdateProps) => {};

render: ({duration, lazy, easing}?: Chart.ChartRenderProps) => {};

stop: () => Chart;

resize: () => Chart;

clear: () => Chart;

toBase64Image: () => string;

generateLegend: () => {};

getElementAtEvent: (e: any) => [{}];

getElementsAtEvent: (e: any) => Array<{}>;

getElementsAtXAxis: (e: any) => Array<{}>;

getDatasetAtEvent: (e: any) => Array<{}>;

getDatasetMeta: (index: number) => Meta;

getVisibleDatasetCount: () => number;

isDatasetVisible: (datasetIndex: number) => boolean;

setDatasetVisibility: (datasetIndex: number, visible: boolean) => void;

ctx: CanvasRenderingContext2D | null;

canvas: HTMLCanvasElement | null;

width: number | null;

height: number | null;

aspectRatio: number | null;

options: Chart.ChartOptions;

chartArea: Chart.ChartArea;

static pluginService: PluginServiceStatic;

static plugins: PluginServiceStatic;

static defaults: {

global: Chart.ChartOptions & Chart.ChartFontOptions & {

tooltips: Chart.ChartTooltipOptions

};

[key: string]: any;

};

static controllers: {

[key: string]: any;

};

static helpers: {

[key: string]: any;

};

static platform: {

disableCSSInjection: boolean

};

static scaleService: {

updateScaleDefaults: (type: Chart.ScaleType, updates: Chart.ChartScales) => void;

};

// Tooltip Static Options

static Tooltip: Chart.ChartTooltipsStaticConfiguration;

static readonly instances: {

[key: string]: Chart;

};

}

type Plugin = Chart.PluginServiceGlobalRegistration & Chart.PluginServiceRegistrationOptions;

interface PluginDescriptor {

plugin: Plugin;

options: Chart.ChartPluginsOptions;

}

declare class PluginServiceStatic {

register(plugin: Plugin): void;

unregister(plugin: Plugin): void;

clear(): void;

count(): number;

getAll(): Plugin[];

notify(chart: Chart, hook: keyof Chart.PluginServiceRegistrationOptions, args: any): boolean;

descriptors(chart: Chart): PluginDescriptor[];

}

interface Meta {

type: Chart.ChartType;

data: MetaData[];

dataset?: Chart.ChartDataSets | undefined;

controller: { [key: string]: any; };

hidden?: boolean | undefined;

total?: string | undefined;

xAxisID?: string | undefined;

yAxisID?: string | undefined;

"$filler"?: { [key: string]: any; } | undefined;

}

interface MetaData {

\_chart: Chart;

\_datasetIndex: number;

\_index: number;

\_model: Model;

\_start?: any;

\_view: Model;

\_xScale: Chart.ChartScales;

\_yScale: Chart.ChartScales;

hidden?: boolean | undefined;

}

// NOTE: This model is generic with a bunch of optional properties to represent all types of chart models.

// Each chart type defines their own unique model structure so some of these optional properties

// might always have values depending on the chart type.

interface Model {

backgroundColor: string;

borderAlign?: Chart.BorderAlignment | undefined;

borderColor: string;

borderWidth?: number | undefined;

circumference?: number | undefined;

controlPointNextX: number;

controlPointNextY: number;

controlPointPreviousX: number;

controlPointPreviousY: number;

endAngle?: number | undefined;

hitRadius: number;

innerRadius?: number | undefined;

outerRadius?: number | undefined;

pointStyle: string;

radius: string;

skip?: boolean | undefined;

startAngle?: number | undefined;

steppedLine?: undefined;

tension: number;

x: number;

y: number;

base: number;

head: number;

}

declare namespace Chart {

type ChartType = 'line' | 'bar' | 'horizontalBar' | 'radar' | 'doughnut' | 'polarArea' | 'bubble' | 'pie' | 'scatter';

type TimeUnit = 'millisecond' | 'second' | 'minute' | 'hour' | 'day' | 'week' | 'month' | 'quarter' | 'year';

type ScaleType = 'category' | 'linear' | 'logarithmic' | 'time' | 'radialLinear';

type PointStyle = 'circle' | 'cross' | 'crossRot' | 'dash' | 'line' | 'rect' | 'rectRounded' | 'rectRot' | 'star' | 'triangle';

type PositionType = 'left' | 'right' | 'top' | 'bottom' | 'chartArea';

// Allow extending the IteractionMode type alias

// see https://github.com/microsoft/TypeScript/issues/28078#issuecomment-432339564

interface InteractionModeRegistry {

'point': 'point';

'nearest': 'nearest';

'single': 'single';

'label': 'label';

'index': 'index';

'x-axis': 'x-axis';

'dataset': 'dataset';

'x': 'x';

'y': 'y';

}

type InteractionMode = InteractionModeRegistry[keyof InteractionModeRegistry];

type Easing = 'linear' | 'easeInQuad' | 'easeOutQuad' | 'easeInOutQuad' | 'easeInCubic' | 'easeOutCubic' | 'easeInOutCubic' |

'easeInQuart' | 'easeOutQuart' | 'easeInOutQuart' | 'easeInQuint' | 'easeOutQuint' | 'easeInOutQuint' | 'easeInSine' | 'easeOutSine' |

'easeInOutSine' | 'easeInExpo' | 'easeOutExpo' | 'easeInOutExpo' | 'easeInCirc' | 'easeOutCirc' | 'easeInOutCirc' | 'easeInElastic' |

'easeOutElastic' | 'easeInOutElastic' | 'easeInBack' | 'easeOutBack' | 'easeInOutBack' | 'easeInBounce' | 'easeOutBounce' | 'easeInOutBounce';

type TextAlignment = 'left' | 'center' | 'right';

type BorderAlignment = 'center' | 'inner';

type BorderWidth = number | { [key in PositionType]?: number };

interface ChartArea {

top: number;

right: number;

bottom: number;

left: number;

}

interface ChartLegendItem {

text?: string | undefined;

fillStyle?: string | undefined;

hidden?: boolean | undefined;

index?: number | undefined;

lineCap?: 'butt' | 'round' | 'square' | undefined;

lineDash?: number[] | undefined;

lineDashOffset?: number | undefined;

lineJoin?: 'bevel' | 'round' | 'miter' | undefined;

lineWidth?: number | undefined;

strokeStyle?: string | undefined;

pointStyle?: PointStyle | undefined;

}

interface ChartLegendLabelItem extends ChartLegendItem {

datasetIndex?: number | undefined;

}

interface ChartTooltipItem {

label?: string | undefined;

value?: string | undefined;

xLabel?: string | number | undefined;

yLabel?: string | number | undefined;

datasetIndex?: number | undefined;

index?: number | undefined;

x?: number | undefined;

y?: number | undefined;

}

interface ChartTooltipLabelColor {

borderColor: ChartColor;

backgroundColor: ChartColor;

}

interface ChartTooltipCallback {

beforeTitle?(item: ChartTooltipItem[], data: ChartData): string | string[];

title?(item: ChartTooltipItem[], data: ChartData): string | string[];

afterTitle?(item: ChartTooltipItem[], data: ChartData): string | string[];

beforeBody?(item: ChartTooltipItem[], data: ChartData): string | string[];

beforeLabel?(tooltipItem: ChartTooltipItem, data: ChartData): string | string[];

label?(tooltipItem: ChartTooltipItem, data: ChartData): string | string[];

labelColor?(tooltipItem: ChartTooltipItem, chart: Chart): ChartTooltipLabelColor;

labelTextColor?(tooltipItem: ChartTooltipItem, chart: Chart): string;

afterLabel?(tooltipItem: ChartTooltipItem, data: ChartData): string | string[];

afterBody?(item: ChartTooltipItem[], data: ChartData): string | string[];

beforeFooter?(item: ChartTooltipItem[], data: ChartData): string | string[];

footer?(item: ChartTooltipItem[], data: ChartData): string | string[];

afterFooter?(item: ChartTooltipItem[], data: ChartData): string | string[];

}

interface ChartAnimationParameter {

chartInstance?: any;

animationObject?: any;

}

interface ChartPoint {

x?: number | string | Date | Moment | undefined;

y?: number | string | Date | Moment | undefined;

r?: number | undefined;

t?: number | string | Date | Moment | undefined;

}

interface ChartConfiguration {

type?: ChartType | string | undefined;

data?: ChartData | undefined;

options?: ChartOptions | undefined;

plugins?: PluginServiceRegistrationOptions[] | undefined;

}

interface ChartData {

labels?: Array<string | string[] | number | number[] | Date | Date[] | Moment | Moment[]> | undefined;

datasets?: ChartDataSets[] | undefined;

}

interface RadialChartOptions extends ChartOptions {

scale?: RadialLinearScale | undefined;

}

interface ChartSize {

height: number;

width: number;

}

interface ChartOptions {

responsive?: boolean | undefined;

responsiveAnimationDuration?: number | undefined;

aspectRatio?: number | undefined;

maintainAspectRatio?: boolean | undefined;

events?: string[] | undefined;

legendCallback?(chart: Chart): string;

onHover?(this: Chart, event: MouseEvent, activeElements: Array<{}>): any;

onClick?(event?: MouseEvent, activeElements?: Array<{}>): any;

onResize?(this: Chart, newSize: ChartSize): void;

title?: ChartTitleOptions | undefined;

legend?: ChartLegendOptions | undefined;

tooltips?: ChartTooltipOptions | undefined;

hover?: ChartHoverOptions | undefined;

animation?: ChartAnimationOptions | undefined;

elements?: ChartElementsOptions | undefined;

layout?: ChartLayoutOptions | undefined;

scale?: RadialLinearScale | undefined;

scales?: ChartScales | LinearScale | LogarithmicScale | TimeScale | undefined;

showLines?: boolean | undefined;

spanGaps?: boolean | undefined;

cutoutPercentage?: number | undefined;

circumference?: number | undefined;

rotation?: number | undefined;

devicePixelRatio?: number | undefined;

plugins?: ChartPluginsOptions | undefined;

defaultColor?: ChartColor | undefined;

}

interface ChartFontOptions {

defaultFontColor?: ChartColor | undefined;

defaultFontFamily?: string | undefined;

defaultFontSize?: number | undefined;

defaultFontStyle?: string | undefined;

}

interface ChartTitleOptions {

display?: boolean | undefined;

position?: PositionType | undefined;

fullWidth?: boolean | undefined;

fontSize?: number | undefined;

fontFamily?: string | undefined;

fontColor?: ChartColor | undefined;

fontStyle?: string | undefined;

padding?: number | undefined;

lineHeight?: number | string | undefined;

text?: string | string[] | undefined;

}

interface ChartLegendOptions {

align?: 'center' | 'end' | 'start' | undefined;

display?: boolean | undefined;

position?: PositionType | undefined;

fullWidth?: boolean | undefined;

onClick?(event: MouseEvent, legendItem: ChartLegendLabelItem): void;

onHover?(event: MouseEvent, legendItem: ChartLegendLabelItem): void;

onLeave?(event: MouseEvent, legendItem: ChartLegendLabelItem): void;

labels?: ChartLegendLabelOptions | undefined;

reverse?: boolean | undefined;

rtl?: boolean | undefined;

textDirection?: string | undefined;

}

interface ChartLegendLabelOptions {

boxWidth?: number | undefined;

fontSize?: number | undefined;

fontStyle?: string | undefined;

fontColor?: ChartColor | undefined;

fontFamily?: string | undefined;

padding?: number | undefined;

generateLabels?(chart: Chart): ChartLegendLabelItem[];

filter?(legendItem: ChartLegendLabelItem, data: ChartData): any;

usePointStyle?: boolean | undefined;

}

interface ChartTooltipOptions {

axis?: 'x'|'y'|'xy' | undefined;

enabled?: boolean | undefined;

custom?: ((tooltipModel: ChartTooltipModel) => void) | undefined;

mode?: InteractionMode | undefined;

intersect?: boolean | undefined;

backgroundColor?: ChartColor | undefined;

titleAlign?: TextAlignment | undefined;

titleFontFamily?: string | undefined;

titleFontSize?: number | undefined;

titleFontStyle?: string | undefined;

titleFontColor?: ChartColor | undefined;

titleSpacing?: number | undefined;

titleMarginBottom?: number | undefined;

bodyAlign?: TextAlignment | undefined;

bodyFontFamily?: string | undefined;

bodyFontSize?: number | undefined;

bodyFontStyle?: string | undefined;

bodyFontColor?: ChartColor | undefined;

bodySpacing?: number | undefined;

footerAlign?: TextAlignment | undefined;

footerFontFamily?: string | undefined;

footerFontSize?: number | undefined;

footerFontStyle?: string | undefined;

footerFontColor?: ChartColor | undefined;

footerSpacing?: number | undefined;

footerMarginTop?: number | undefined;

xPadding?: number | undefined;

yPadding?: number | undefined;

caretSize?: number | undefined;

cornerRadius?: number | undefined;

multiKeyBackground?: string | undefined;

callbacks?: ChartTooltipCallback | undefined;

filter?(item: ChartTooltipItem, data: ChartData): boolean;

itemSort?(itemA: ChartTooltipItem, itemB: ChartTooltipItem, data?: ChartData): number;

position?: string | undefined;

caretPadding?: number | undefined;

displayColors?: boolean | undefined;

borderColor?: ChartColor | undefined;

borderWidth?: number | undefined;

rtl?: boolean | undefined;

textDirection?: string | undefined;

}

interface ChartTooltipModel {

afterBody: string[];

backgroundColor: string;

beforeBody: string[];

body: ChartTooltipModelBody[];

bodyFontColor: string;

bodyFontSize: number;

bodySpacing: number;

borderColor: string;

borderWidth: number;

caretPadding: number;

caretSize: number;

caretX: number;

caretY: number;

cornerRadius: number;

dataPoints: ChartTooltipItem[];

displayColors: boolean;

footer: string[];

footerFontColor: string;

footerFontSize: number;

footerMarginTop: number;

footerSpacing: number;

height: number;

labelColors: ChartTooltipLabelColor[];

labelTextColors: string[];

legendColorBackground: string;

opacity: number;

title: string[];

titleFontColor: string;

titleFontSize: number;

titleMarginBottom: number;

titleSpacing: number;

width: number;

x: number;

xAlign: string;

xPadding: number;

y: number;

yAlign: string;

yPadding: number;

\_bodyAlign: string;

\_bodyFontFamily: string;

\_bodyFontStyle: string;

\_footerAlign: string;

\_footerFontFamily: string;

\_footerFontStyle: string;

\_titleAlign: string;

\_titleFontFamily: string;

\_titleFontStyle: string;

}

interface ChartTooltipModelBody {

before: string[];

lines: string[];

after: string[];

}

// NOTE: declare plugin options as interface instead of inline '{ [plugin: string]: any }'

// to allow module augmentation in case some plugins want to strictly type their options.

interface ChartPluginsOptions {

[pluginId: string]: any;

}

interface ChartTooltipsStaticConfiguration {

positioners: { [mode: string]: ChartTooltipPositioner };

}

type ChartTooltipPositioner = (elements: any[], eventPosition: Point) => Point;

interface ChartHoverOptions {

mode?: InteractionMode | undefined;

animationDuration?: number | undefined;

intersect?: boolean | undefined;

axis?: 'x' | 'y' | 'xy' | undefined;

onHover?(this: Chart, event: MouseEvent, activeElements: Array<{}>): any;

}

interface ChartAnimationObject {

currentStep?: number | undefined;

numSteps?: number | undefined;

easing?: Easing | undefined;

render?(arg: any): void;

onAnimationProgress?(arg: any): void;

onAnimationComplete?(arg: any): void;

}

interface ChartAnimationOptions {

duration?: number | undefined;

easing?: Easing | undefined;

onProgress?(chart: any): void;

onComplete?(chart: any): void;

animateRotate?: boolean | undefined;

animateScale?: boolean | undefined;

}

interface ChartElementsOptions {

point?: ChartPointOptions | undefined;

line?: ChartLineOptions | undefined;

arc?: ChartArcOptions | undefined;

rectangle?: ChartRectangleOptions | undefined;

}

interface ChartArcOptions {

angle?: number | Scriptable<number> | undefined;

backgroundColor?: ChartDataSets["backgroundColor"] | undefined;

borderAlign?: BorderAlignment | Scriptable<BorderAlignment> | undefined;

borderColor?: ChartColor | Scriptable<ChartColor> | undefined;

borderWidth?: number | Scriptable<number> | undefined;

}

type CubicInterpolationMode = 'default' | 'monotone';

type FillMode = 'zero' | 'top' | 'bottom' | boolean;

interface ChartLineOptions {

cubicInterpolationMode?: CubicInterpolationMode | Scriptable<CubicInterpolationMode> | undefined;

tension?: number | Scriptable<number> | undefined;

backgroundColor?: ChartDataSets["backgroundColor"] | undefined;

borderWidth?: number | Scriptable<number> | undefined;

borderColor?: ChartColor | Scriptable<ChartColor> | undefined;

borderCapStyle?: string | Scriptable<string> | undefined;

borderDash?: any[] | Scriptable<any[]> | undefined;

borderDashOffset?: number | Scriptable<number> | undefined;

borderJoinStyle?: string | Scriptable<string> | undefined;

capBezierPoints?: boolean | Scriptable<boolean> | undefined;

fill?: FillMode | Scriptable<FillMode> | undefined;

stepped?: boolean | Scriptable<boolean> | undefined;

}

interface ChartPointOptions {

radius?: number | Scriptable<number> | undefined;

pointStyle?: PointStyle | Scriptable<PointStyle> | undefined;

rotation?: number | Scriptable<number> | undefined;

backgroundColor?: ChartDataSets["backgroundColor"] | undefined;

borderWidth?: number | Scriptable<number> | undefined;

borderColor?: ChartColor | Scriptable<ChartColor> | undefined;

hitRadius?: number | Scriptable<number> | undefined;

hoverRadius?: number | Scriptable<number> | undefined;

hoverBorderWidth?: number | Scriptable<number> | undefined;

}

interface ChartRectangleOptions {

backgroundColor?: ChartDataSets["backgroundColor"] | undefined;

borderWidth?: number | Scriptable<number> | undefined;

borderColor?: ChartColor | Scriptable<ChartColor> | undefined;

borderSkipped?: string | Scriptable<string> | undefined;

}

interface ChartLayoutOptions {

padding?: ChartLayoutPaddingObject | number | undefined;

}

interface ChartLayoutPaddingObject {

top?: number | undefined;

right?: number | undefined;

bottom?: number | undefined;

left?: number | undefined;

}

interface GridLineOptions {

display?: boolean | undefined;

circular?: boolean | undefined;

color?: ChartColor | undefined;

borderDash?: number[] | undefined;

borderDashOffset?: number | undefined;

lineWidth?: number | number[] | undefined;

drawBorder?: boolean | undefined;

drawOnChartArea?: boolean | undefined;

drawTicks?: boolean | undefined;

tickMarkLength?: number | undefined;

zeroLineWidth?: number | undefined;

zeroLineColor?: ChartColor | undefined;

zeroLineBorderDash?: number[] | undefined;

zeroLineBorderDashOffset?: number | undefined;

offsetGridLines?: boolean | undefined;

z?: number | undefined;

}

interface ScaleTitleOptions {

display?: boolean | undefined;

labelString?: string | undefined;

lineHeight?: number | string | undefined;

fontColor?: ChartColor | undefined;

fontFamily?: string | undefined;

fontSize?: number | undefined;

fontStyle?: string | undefined;

padding?: ChartLayoutPaddingObject | number | undefined;

}

interface TickOptions extends NestedTickOptions {

minor?: NestedTickOptions | false | undefined;

major?: MajorTickOptions | false | undefined;

}

interface NestedTickOptions {

autoSkip?: boolean | undefined;

autoSkipPadding?: number | undefined;

backdropColor?: ChartColor | undefined;

backdropPaddingX?: number | undefined;

backdropPaddingY?: number | undefined;

beginAtZero?: boolean | undefined;

/\*\*

\* If the callback returns null or undefined the associated grid line will be hidden.

\*/

callback?(value: number | string, index: number, values: number[] | string[]): string | number | null | undefined;

display?: boolean | undefined;

fontColor?: ChartColor | undefined;

fontFamily?: string | undefined;

fontSize?: number | undefined;

fontStyle?: string | undefined;

labelOffset?: number | undefined;

lineHeight?: number | undefined;

max?: any;

maxRotation?: number | undefined;

maxTicksLimit?: number | undefined;

min?: any;

minRotation?: number | undefined;

mirror?: boolean | undefined;

padding?: number | undefined;

precision?: number | undefined;

reverse?: boolean | undefined;

/\*\*

\* The number of ticks to examine when deciding how many labels will fit.

\* Setting a smaller value will be faster, but may be less accurate

\* when there is large variability in label length.

\* Deault: `ticks.length`

\*/

sampleSize?: number | undefined;

showLabelBackdrop?: boolean | undefined;

source?: 'auto' | 'data' | 'labels' | undefined;

stepSize?: number | undefined;

suggestedMax?: number | undefined;

suggestedMin?: number | undefined;

}

interface MajorTickOptions extends NestedTickOptions {

enabled?: boolean | undefined;

}

interface AngleLineOptions {

display?: boolean | undefined;

color?: ChartColor | undefined;

lineWidth?: number | undefined;

borderDash?: number[] | undefined;

borderDashOffset?: number | undefined;

}

interface PointLabelOptions {

callback?(arg: any): any;

fontColor?: ChartColor | undefined;

fontFamily?: string | undefined;

fontSize?: number | undefined;

fontStyle?: string | undefined;

lineHeight?: number|string | undefined;

}

interface LinearTickOptions extends TickOptions {

maxTicksLimit?: number | undefined;

stepSize?: number | undefined;

precision?: number | undefined;

suggestedMin?: number | undefined;

suggestedMax?: number | undefined;

}

// tslint:disable-next-line no-empty-interface

interface LogarithmicTickOptions extends TickOptions {

}

type ChartColor = string | CanvasGradient | CanvasPattern | string[];

type Scriptable<T> = (ctx: {

chart?: Chart | undefined;

dataIndex?: number | undefined;

dataset?: ChartDataSets | undefined

datasetIndex?: number | undefined;

}) => T;

interface ChartDataSets {

cubicInterpolationMode?: CubicInterpolationMode | Scriptable<CubicInterpolationMode> | undefined;

backgroundColor?: ChartColor | ChartColor[] | Scriptable<ChartColor> | undefined;

barPercentage?: number | undefined;

barThickness?: number | "flex" | undefined;

borderAlign?: BorderAlignment | BorderAlignment[] | Scriptable<BorderAlignment> | undefined;

borderWidth?: BorderWidth | BorderWidth[] | Scriptable<BorderWidth> | undefined;

borderColor?: ChartColor | ChartColor[] | Scriptable<ChartColor> | undefined;

borderCapStyle?: 'butt' | 'round' | 'square' | undefined;

borderDash?: number[] | undefined;

borderDashOffset?: number | undefined;

borderJoinStyle?: 'bevel' | 'round' | 'miter' | undefined;

borderSkipped?: PositionType | PositionType[] | Scriptable<PositionType> | undefined;

categoryPercentage?: number | undefined;

data?: Array<number | null | undefined | number[]> | ChartPoint[] | undefined;

fill?: boolean | number | string | undefined;

hitRadius?: number | number[] | Scriptable<number> | undefined;

hoverBackgroundColor?: ChartColor | ChartColor[] | Scriptable<ChartColor> | undefined;

hoverBorderColor?: ChartColor | ChartColor[] | Scriptable<ChartColor> | undefined;

hoverBorderWidth?: number | number[] | Scriptable<number> | undefined;

hoverRadius?: number | undefined;

label?: string | undefined;

lineTension?: number | undefined;

maxBarThickness?: number | undefined;

minBarLength?: number | undefined;

steppedLine?: 'before' | 'after' | 'middle' | boolean | undefined;

order?: number | undefined;

pointBorderColor?: ChartColor | ChartColor[] | Scriptable<ChartColor> | undefined;

pointBackgroundColor?: ChartColor | ChartColor[] | Scriptable<ChartColor> | undefined;

pointBorderWidth?: number | number[] | Scriptable<number> | undefined;

pointRadius?: number | number[] | Scriptable<number> | undefined;

pointRotation?: number | number[] | Scriptable<number> | undefined;

pointHoverRadius?: number | number[] | Scriptable<number> | undefined;

pointHitRadius?: number | number[] | Scriptable<number> | undefined;

pointHoverBackgroundColor?: ChartColor | ChartColor[] | Scriptable<ChartColor> | undefined;

pointHoverBorderColor?: ChartColor | ChartColor[] | Scriptable<ChartColor> | undefined;

pointHoverBorderWidth?: number | number[] | Scriptable<number> | undefined;

pointStyle?: PointStyle

| HTMLImageElement

| HTMLCanvasElement

| Array<PointStyle

| HTMLImageElement

| HTMLCanvasElement>

| Scriptable<PointStyle

| HTMLImageElement

| HTMLCanvasElement>

| undefined;

radius?: number | number[] | Scriptable<number> | undefined;

rotation?: number | number[] | Scriptable<number> | undefined;

xAxisID?: string | undefined;

yAxisID?: string | undefined;

type?: ChartType | string | undefined;

hidden?: boolean | undefined;

hideInLegendAndTooltip?: boolean | undefined;

showLine?: boolean | undefined;

stack?: string | undefined;

spanGaps?: boolean | undefined;

weight?: number | undefined;

}

interface ChartScales {

type?: ScaleType | string | undefined;

display?: boolean | undefined;

position?: PositionType | string | undefined;

gridLines?: GridLineOptions | undefined;

scaleLabel?: ScaleTitleOptions | undefined;

ticks?: TickOptions | undefined;

xAxes?: ChartXAxe[] | undefined;

yAxes?: ChartYAxe[] | undefined;

}

interface CommonAxe {

bounds?: string | undefined;

type?: ScaleType | string | undefined;

display?: boolean | string | undefined;

id?: string | undefined;

labels?: string[] | undefined;

stacked?: boolean | undefined;

position?: string | undefined;

ticks?: TickOptions | undefined;

gridLines?: GridLineOptions | undefined;

scaleLabel?: ScaleTitleOptions | undefined;

time?: TimeScale | undefined;

offset?: boolean | undefined;

beforeUpdate?(scale?: any): void;

beforeSetDimension?(scale?: any): void;

beforeDataLimits?(scale?: any): void;

beforeBuildTicks?(scale?: any): void;

beforeTickToLabelConversion?(scale?: any): void;

beforeCalculateTickRotation?(scale?: any): void;

beforeFit?(scale?: any): void;

afterUpdate?(scale?: any): void;

afterSetDimension?(scale?: any): void;

afterDataLimits?(scale?: any): void;

afterBuildTicks?(scale: any, ticks: number[]): number[];

afterTickToLabelConversion?(scale?: any): void;

afterCalculateTickRotation?(scale?: any): void;

afterFit?(scale?: any): void;

}

interface ChartXAxe extends CommonAxe {

distribution?: 'linear' | 'series' | undefined;

}

// tslint:disable-next-line no-empty-interface

interface ChartYAxe extends CommonAxe {

}

interface LinearScale extends ChartScales {

ticks?: LinearTickOptions | undefined;

}

interface LogarithmicScale extends ChartScales {

ticks?: LogarithmicTickOptions | undefined;

}

interface TimeDisplayFormat {

millisecond?: string | undefined;

second?: string | undefined;

minute?: string | undefined;

hour?: string | undefined;

day?: string | undefined;

week?: string | undefined;

month?: string | undefined;

quarter?: string | undefined;

year?: string | undefined;

}

interface DateAdapterOptions {

date?: object | undefined;

}

interface TimeScale extends ChartScales {

adapters?: DateAdapterOptions | undefined;

displayFormats?: TimeDisplayFormat | undefined;

isoWeekday?: boolean | undefined;

max?: string | undefined;

min?: string | undefined;

parser?: string | ((arg: any) => any) | undefined;

round?: TimeUnit | undefined;

tooltipFormat?: string | undefined;

unit?: TimeUnit | undefined;

unitStepSize?: number | undefined;

stepSize?: number | undefined;

minUnit?: TimeUnit | undefined;

}

interface RadialLinearScale {

animate?: boolean | undefined;

position?: PositionType | undefined;

angleLines?: AngleLineOptions | undefined;

pointLabels?: PointLabelOptions | undefined;

ticks?: LinearTickOptions | undefined;

display?: boolean | undefined;

gridLines?: GridLineOptions | undefined;

}

interface Point {

x: number;

y: number;

}

interface PluginServiceGlobalRegistration {

id?: string | undefined;

}

interface PluginServiceRegistrationOptions {

beforeInit?(chartInstance: Chart, options?: any): void;

afterInit?(chartInstance: Chart, options?: any): void;

beforeUpdate?(chartInstance: Chart, options?: any): void;

afterUpdate?(chartInstance: Chart, options?: any): void;

beforeLayout?(chartInstance: Chart, options?: any): void;

afterLayout?(chartInstance: Chart, options?: any): void;

beforeDatasetsUpdate?(chartInstance: Chart, options?: any): void;

afterDatasetsUpdate?(chartInstance: Chart, options?: any): void;

beforeDatasetUpdate?(chartInstance: Chart, options?: any): void;

afterDatasetUpdate?(chartInstance: Chart, options?: any): void;

// This is called at the start of a render. It is only called once, even if the animation will run for a number of frames. Use beforeDraw or afterDraw

// to do something on each animation frame

beforeRender?(chartInstance: Chart, options?: any): void;

afterRender?(chartInstance: Chart, options?: any): void;

// Easing is for animation

beforeDraw?(chartInstance: Chart, easing: Easing, options?: any): void;

afterDraw?(chartInstance: Chart, easing: Easing, options?: any): void;

// Before the datasets are drawn but after scales are drawn

beforeDatasetsDraw?(chartInstance: Chart, easing: Easing, options?: any): void;

afterDatasetsDraw?(chartInstance: Chart, easing: Easing, options?: any): void;

beforeDatasetDraw?(chartInstance: Chart, easing: Easing, options?: any): void;

afterDatasetDraw?(chartInstance: Chart, easing: Easing, options?: any): void;

// Called before drawing the `tooltip`. If any plugin returns `false`,

// the tooltip drawing is cancelled until another `render` is triggered.

beforeTooltipDraw?(chartInstance: Chart, tooltipData?: any, options?: any): void;

// Called after drawing the `tooltip`. Note that this hook will not,

// be called if the tooltip drawing has been previously cancelled.

afterTooltipDraw?(chartInstance: Chart, tooltipData?: any, options?: any): void;

// Called when an event occurs on the chart

beforeEvent?(chartInstance: Chart, event: Event, options?: any): void;

afterEvent?(chartInstance: Chart, event: Event, options?: any): void;

resize?(chartInstance: Chart, newChartSize: ChartSize, options?: any): void;

destroy?(chartInstance: Chart): void;

/\*\* Deprecated since version 2.5.0. Use `afterLayout` instead. \*/

afterScaleUpdate?(chartInstance: Chart, options?: any): void;

}

interface ChartUpdateProps {

duration?: number | undefined;

lazy?: boolean | undefined;

easing?: Easing | undefined;

}

interface ChartRenderProps {

duration?: number | undefined;

lazy?: boolean | undefined;

easing?: Easing | undefined;

}

// Model used with the doughnut chart

interface DoughnutModel {

backgroundColor: ChartColor;

borderAlign: BorderAlignment;

borderColor: string;

borderWidth: number;

circumference: number;

endAngle: number;

innerRadius: number;

outerRadius: number;

startAngle: number;

x: number;

y: number;

}

}

export = Chart;

export as namespace Chart;