{"version":3,"sources":["chartist.js"],"names":["root","factory","define","amd","exports","module","this","Chartist","version","window","document","namespaces","svg","xmlns","xhtml","xlink","ct","noop","n","alphaNumerate","String","fromCharCode","extend","target","i","source","sourceProp","arguments","length","prop","Array","replaceAll","str","subStr","newSubStr","replace","RegExp","ensureUnit","value","unit","quantity","input","match","exec","undefined","querySelector","query","Node","times","apply","sum","previous","current","mapMultiply","factor","num","mapAdd","addend","serialMap","arr","cb","result","Math","max","map","e","forEach","index","args","roundWithPrecision","digits","precision","pow","round","escapingMap","&","<",">","\"","'","serialize","data","JSON","stringify","Object","keys","reduce","key","deserialize","parse","createSvg","container","width","height","className","prototype","slice","call","querySelectorAll","filter","getAttributeNS","removeChild","Svg","attr","addClass","style","appendChild","\_node","normalizeData","reverse","multi","labelCount","output","raw","normalized","series","getDataArray","every","labels","push","reverseData","safeHasProperty","object","property","hasOwnProperty","isDataHoleValue","isNaN","recursiveConvert","multiValue","getNumberOrUndefined","y","x","normalizePadding","padding","fallback","top","right","bottom","left","getMetaData","meta","orderOfMagnitude","floor","log","abs","LN10","projectLength","axisLength","bounds","range","getAvailableHeight","options","chartPadding","axisX","offset","getHighLow","dimension","recursiveHighLow","findHigh","highLow","high","findLow","low","toUpperCase","Number","MAX\_VALUE","referenceValue","min","isNumeric","isFinite","isFalseyButZero","isMultiValue","getMultiValue","rho","gcd","p","q","f","divisor","x1","x2","getBounds","scaleMinSpace","onlyInteger","safeIncrement","increment","EPSILON","newMin","newMax","optimizationCounter","valueRange","oom","step","ceil","numberOfSteps","scaleUp","smallestFactor","Error","values","polarToCartesian","centerX","centerY","radius","angleInDegrees","angleInRadians","PI","cos","sin","createChartRect","fallbackPadding","hasAxis","axisY","yAxisOffset","xAxisOffset","normalizedPadding","chartRect","y1","y2","position","createGrid","axis","group","classes","eventEmitter","positionalData","units","pos","counterUnits","gridElement","elem","join","emit","type","element","createGridBackground","gridGroup","gridBackground","createLabel","axisOffset","labelOffset","useForeignObject","labelElement","len","content","foreignObject","text","getSeriesOption","name","seriesOptions","optionsProvider","responsiveOptions","updateCurrentOptions","mediaEvent","previousOptions","currentOptions","baseOptions","mql","matchMedia","matches","removeMediaQueryListeners","mediaQueryListeners","removeListener","addListener","getCurrentOptions","splitIntoSegments","pathCoordinates","valueData","defaultOptions","increasingX","fillHoles","segments","hole","Interpolation","none","path","Path","currX","currY","currData","move","line","simple","d","prevX","prevY","prevData","curve","cardinal","tension","t","c","paths","segment","z","iLen","monotoneCubic","xs","ys","ms","ds","dys","dxs","postpone","EventEmitter","addEventHandler","event","handler","handlers","removeEventHandler","splice","indexOf","starHandler","listToArray","list","properties","superProtoOverride","superProto","Class","proto","create","cloneDefinitions","constr","instance","fn","constructor","getOwnPropertyNames","propName","defineProperty","getOwnPropertyDescriptor","update","override","initializeTimeoutId","createChart","detach","clearTimeout","removeEventListener","resizeListener","on","off","initialize","addEventListener","bind","plugins","plugin","Base","supportsForeignObject","isSupported","supportsAnimations","\_\_chartist\_\_","setTimeout","attributes","parent","insertFirst","Element","createElementNS","xmlns:ct","firstChild","insertBefore","ns","getAttribute","namespacedAttribute","split","setAttributeNS","setAttribute","parentNode","SVGElement","node","nodeName","selector","foundNode","foundNodes","List","getNode","createElement","innerHTML","fnObj","createTextNode","empty","remove","newElement","replaceChild","append","trim","names","concat","self","removeClass","removedClasses","removeAllClasses","getBoundingClientRect","animate","animations","guided","attribute","createAnimate","animationDefinition","timeout","easing","attributeProperties","Easing","begin","dur","calcMode","keySplines","keyTimes","fill","from","attributeName","beginElement","err","to","params","SvgList","nodeList","svgElements","prototypeProperty","feature","implementation","hasFeature","easingCubicBeziers","easeInSine","easeOutSine","easeInOutSine","easeInQuad","easeOutQuad","easeInOutQuad","easeInCubic","easeOutCubic","easeInOutCubic","easeInQuart","easeOutQuart","easeInOutQuart","easeInQuint","easeOutQuint","easeInOutQuint","easeInExpo","easeOutExpo","easeInOutExpo","easeInCirc","easeOutCirc","easeInOutCirc","easeInBack","easeOutBack","easeInOutBack","command","pathElements","relative","pathElement","toLowerCase","forEachParam","pathElementIndex","elementDescriptions","paramName","paramIndex","SvgPath","close","count","arc","rx","ry","xAr","lAf","sf","chunks","pop","elements","chunk","shift","description","spliceArgs","accuracyMultiplier","accuracy","scale","translate","transform","transformFnc","transformed","clone","splitByCommand","joinedPath","j","m","l","a","Axis","ticks","axisUnits","rectEnd","rectStart","gridOffset","rectOffset","createGridAndLabels","labelGroup","chartOptions","axisOptions","projectedValues","projectValue","labelValues","labelInterpolationFnc","projectedValue","labelLength","showGrid","classNames","grid","dir","showLabel","label","AutoScaleAxis","axisUnit","FixedScaleAxis","sort","b","stepLength","StepAxis","calc","stretch","chart","seriesGroup","fullWidth","showGridBackground","seriesIndex","seriesElement","ct:series-name","ct:meta","pathData","valueIndex","lineSmooth","showPoint","showLine","showArea","areaBase","smoothing","point","ct:value","seriesMeta","areaBaseProjected","pathSegment","solidPathSegments","firstElement","lastElement","areaPath","area","Line","vertical","horizontal","start","end","distributeSeries","horizontalBars","stackBars","serialSums","prev","curr","valueAxis","labelAxisTicks","labelAxis","zeroPoint","stackedBarValues","periodHalfLength","biPol","projected","bar","previousStack","labelAxisValueIndex","seriesBarDistance","positions","stackMode","metaData","Bar","determineAnchorPosition","center","direction","toTheRight","labelsGroup","labelRadius","totalDataSum","seriesGroups","startAngle","donut","chartDonut","chartPie","total","previousValue","currentValue","donutWidth","labelPosition","hasSingleValInSeries","val","ignoreEmptyValues","endAngle","overlappigStartAngle","sliceDonut","slicePie","rawValue","interpolatedValue","dx","dy","text-anchor","labelDirection","Pie"],"mappings":";;;;;;;CAAC,SAAUA,EAAMC,GACO,kBAAXC,SAAyBA,OAAOC,IAEzCD,OAAO,cAAgB,WACrB,MAAQF,GAAe,SAAIC,MAED,gBAAZG,SAIhBC,OAAOD,QAAUH,IAEjBD,EAAe,SAAIC,KAErBK,KAAM,WAaR,GAAIC,IACFC,QAAS,SAk1IX,OA/0IC,UAAUC,EAAQC,EAAUH,GAC3B,YAQAA,GAASI,YACPC,IAAK,6BACLC,MAAO,gCACPC,MAAO,+BACPC,MAAO,+BACPC,GAAI,6CAUNT,EAASU,KAAO,SAAUC,GACxB,MAAOA,IAUTX,EAASY,cAAgB,SAAUD,GAEjC,MAAOE,QAAOC,aAAa,GAAKH,EAAI,KAWtCX,EAASe,OAAS,SAAUC,GAC1B,GAAIC,GAAGC,EAAQC,CAGf,KAFAH,EAASA,MAEJC,EAAI,EAAGA,EAAIG,UAAUC,OAAQJ,IAAK,CACrCC,EAASE,UAAUH,EACnB,KAAK,GAAIK,KAAQJ,GACfC,EAAaD,EAAOI,GACM,gBAAfH,IAA0C,OAAfA,GAAyBA,YAAsBI,OAGnFP,EAAOM,GAAQH,EAFfH,EAAOM,GAAQtB,EAASe,OAAOC,EAAOM,GAAOH,GAOnD,MAAOH,IAYThB,EAASwB,WAAa,SAASC,EAAKC,EAAQC,GAC1C,MAAOF,GAAIG,QAAQ,GAAIC,QAAOH,EAAQ,KAAMC,IAW9C3B,EAAS8B,WAAa,SAASC,EAAOC,GAKpC,MAJoB,gBAAVD,KACRA,GAAgBC,GAGXD,GAUT/B,EAASiC,SAAW,SAASC,GAC3B,GAAqB,gBAAVA,GAAoB,CAC7B,GAAIC,GAAQ,kBAAoBC,KAAKF,EACrC,QACEH,OAASI,EAAM,GACfH,KAAMG,EAAM,IAAME,QAGtB,OAASN,MAAOG,IAUlBlC,EAASsC,cAAgB,SAASC,GAChC,MAAOA,aAAiBC,MAAOD,EAAQpC,EAASmC,cAAcC,IAUhEvC,EAASyC,MAAQ,SAASpB,GACxB,MAAOE,OAAMmB,MAAM,KAAM,GAAInB,OAAMF,KAWrCrB,EAAS2C,IAAM,SAASC,EAAUC,GAChC,MAAOD,IAAYC,EAAUA,EAAU,IAUzC7C,EAAS8C,YAAc,SAASC,GAC9B,MAAO,UAASC,GACd,MAAOA,GAAMD,IAWjB/C,EAASiD,OAAS,SAASC,GACzB,MAAO,UAASF,GACd,MAAOA,GAAME,IAYjBlD,EAASmD,UAAY,SAASC,EAAKC,GACjC,GAAIC,MACAjC,EAASkC,KAAKC,IAAId,MAAM,KAAMU,EAAIK,IAAI,SAASC,GAC7C,MAAOA,GAAErC,SAWf,OARArB,GAASyC,MAAMpB,GAAQsC,QAAQ,SAASD,EAAGE,GACzC,GAAIC,GAAOT,EAAIK,IAAI,SAASC,GAC1B,MAAOA,GAAEE,IAGXN,GAAOM,GAASP,EAAGX,MAAM,KAAMmB,KAG1BP,GAWTtD,EAAS8D,mBAAqB,SAAS/B,EAAOgC,GAC5C,GAAIC,GAAYT,KAAKU,IAAI,GAAIF,GAAU/D,EAASgE,UAChD,OAAOT,MAAKW,MAAMnC,EAAQiC,GAAaA,GASzChE,EAASgE,UAAY,EAQrBhE,EAASmE,aACPC,IAAK,QACLC,IAAK,OACLC,IAAK,OACLC,IAAK,SACLC,IAAM,UAWRxE,EAASyE,UAAY,SAASC,GAC5B,MAAY,QAATA,GAA0BrC,SAATqC,EACXA,GACiB,gBAATA,GACfA,EAAO,GAAGA,EACc,gBAATA,KACfA,EAAOC,KAAKC,WAAWF,KAAMA,KAGxBG,OAAOC,KAAK9E,EAASmE,aAAaY,OAAO,SAASzB,EAAQ0B,GAC/D,MAAOhF,GAASwB,WAAW8B,EAAQ0B,EAAKhF,EAASmE,YAAYa,KAC5DN,KAUL1E,EAASiF,YAAc,SAASP,GAC9B,GAAmB,gBAATA,GACR,MAAOA,EAGTA,GAAOG,OAAOC,KAAK9E,EAASmE,aAAaY,OAAO,SAASzB,EAAQ0B,GAC/D,MAAOhF,GAASwB,WAAW8B,EAAQtD,EAASmE,YAAYa,GAAMA,IAC7DN,EAEH,KACEA,EAAOC,KAAKO,MAAMR,GAClBA,EAAqBrC,SAAdqC,EAAKA,KAAqBA,EAAKA,KAAOA,EAC7C,MAAMhB,IAER,MAAOgB,IAaT1E,EAASmF,UAAY,SAAUC,EAAWC,EAAOC,EAAQC,GACvD,GAAIlF,EAwBJ,OAtBAgF,GAAQA,GAAS,OACjBC,EAASA,GAAU,OAInB/D,MAAMiE,UAAUC,MAAMC,KAAKN,EAAUO,iBAAiB,QAAQC,OAAO,SAAkCvF,GACrG,MAAOA,GAAIwF,eAAe7F,EAASI,WAAWE,MAAO,QACpDqD,QAAQ,SAA+BtD,GACxC+E,EAAUU,YAAYzF,KAIxBA,EAAM,GAAIL,GAAS+F,IAAI,OAAOC,MAC5BX,MAAOA,EACPC,OAAQA,IACPW,SAASV,GAAWS,MACrBE,MAAO,UAAYb,EAAQ,aAAeC,EAAS,MAIrDF,EAAUe,YAAY9F,EAAI+F,OAEnB/F,GASTL,EAASqG,cAAgB,SAAS3B,EAAM4B,EAASC,GAC/C,GAAIC,GACAC,GACFC,IAAKhC,EACLiC,cAmCF,OA/BAF,GAAOE,WAAWC,OAAS5G,EAAS6G,cAClCD,OAAQlC,EAAKkC,YACZN,EAASC,GAQVC,EAJEC,EAAOE,WAAWC,OAAOE,MAAM,SAAS/E,GACxC,MAAOA,aAAiBR,SAGbgC,KAAKC,IAAId,MAAM,KAAM+D,EAAOE,WAAWC,OAAOnD,IAAI,SAASmD,GACtE,MAAOA,GAAOvF,UAIHoF,EAAOE,WAAWC,OAAOvF,OAGxCoF,EAAOE,WAAWI,QAAUrC,EAAKqC,YAActB,QAE/ClE,MAAMiE,UAAUwB,KAAKtE,MACnB+D,EAAOE,WAAWI,OAClB/G,EAASyC,MAAMc,KAAKC,IAAI,EAAGgD,EAAaC,EAAOE,WAAWI,OAAO1F,SAASoC,IAAI,WAC5E,MAAO,MAIR6C,GACDtG,EAASiH,YAAYR,EAAOE,YAGvBF,GAUTzG,EAASkH,gBAAkB,SAASC,EAAQC,GAC1C,MAAkB,QAAXD,GACa,gBAAXA,IACPA,EAAOE,eAAeD,IAS1BpH,EAASsH,gBAAkB,SAASvF,GAClC,MAAiB,QAAVA,GACKM,SAAVN,GACkB,gBAAVA,IAAsBwF,MAAMxF,IASxC/B,EAASiH,YAAc,SAASvC,GAC9BA,EAAKqC,OAAOT,UACZ5B,EAAKkC,OAAON,SACZ,KAAK,GAAIrF,GAAI,EAAGA,EAAIyD,EAAKkC,OAAOvF,OAAQJ,IACR,gBAApByD,GAAKkC,OAAO3F,IAA4CoB,SAAxBqC,EAAKkC,OAAO3F,GAAGyD,KACvDA,EAAKkC,OAAO3F,GAAGyD,KAAK4B,UACZ5B,EAAKkC,OAAO3F,YAAcM,QAClCmD,EAAKkC,OAAO3F,GAAGqF,WAcrBtG,EAAS6G,aAAe,SAASnC,EAAM4B,EAASC,GAG9C,QAASiB,GAAiBzF,GACxB,GAAG/B,EAASkH,gBAAgBnF,EAAO,SAEjC,MAAOyF,GAAiBzF,EAAMA,MACzB,IAAG/B,EAASkH,gBAAgBnF,EAAO,QAExC,MAAOyF,GAAiBzF,EAAM2C,KACzB,IAAG3C,YAAiBR,OAEzB,MAAOQ,GAAM0B,IAAI+D,EACZ,KAAGxH,EAASsH,gBAAgBvF,GAA5B,CAML,GAAGwE,EAAO,CACR,GAAIkB,KAcJ,OAToB,gBAAVlB,GACRkB,EAAWlB,GAASvG,EAAS0H,qBAAqB3F,GAElD0F,EAAWE,EAAI3H,EAAS0H,qBAAqB3F,GAG/C0F,EAAWG,EAAI7F,EAAMsF,eAAe,KAAOrH,EAAS0H,qBAAqB3F,EAAM6F,GAAKH,EAAWG,EAC/FH,EAAWE,EAAI5F,EAAMsF,eAAe,KAAOrH,EAAS0H,qBAAqB3F,EAAM4F,GAAKF,EAAWE,EAExFF,EAIP,MAAOzH,GAAS0H,qBAAqB3F,IAK3C,MAAO2C,GAAKkC,OAAOnD,IAAI+D,IAWzBxH,EAAS6H,iBAAmB,SAASC,EAASC,GAG5C,MAFAA,GAAWA,GAAY,EAEG,gBAAZD,IACZE,IAAKF,EACLG,MAAOH,EACPI,OAAQJ,EACRK,KAAML,IAENE,IAA4B,gBAAhBF,GAAQE,IAAmBF,EAAQE,IAAMD,EACrDE,MAAgC,gBAAlBH,GAAQG,MAAqBH,EAAQG,MAAQF,EAC3DG,OAAkC,gBAAnBJ,GAAQI,OAAsBJ,EAAQI,OAASH,EAC9DI,KAA8B,gBAAjBL,GAAQK,KAAoBL,EAAQK,KAAOJ,IAI5D/H,EAASoI,YAAc,SAASxB,EAAQhD,GACtC,GAAI7B,GAAQ6E,EAAOlC,KAAOkC,EAAOlC,KAAKd,GAASgD,EAAOhD,EACtD,OAAO7B,GAAQA,EAAMsG,KAAOhG,QAU9BrC,EAASsI,iBAAmB,SAAUvG,GACpC,MAAOwB,MAAKgF,MAAMhF,KAAKiF,IAAIjF,KAAKkF,IAAI1G,IAAUwB,KAAKmF,OAYrD1I,EAAS2I,cAAgB,SAAUC,EAAYvH,EAAQwH,GACrD,MAAOxH,GAASwH,EAAOC,MAAQF,GAWjC5I,EAAS+I,mBAAqB,SAAU1I,EAAK2I,GAC3C,MAAOzF,MAAKC,KAAKxD,EAASiC,SAAS+G,EAAQ1D,QAAQvD,OAAS1B,EAAIiF,WAAa0D,EAAQC,aAAajB,IAAOgB,EAAQC,aAAaf,QAAUc,EAAQE,MAAMC,OAAQ,IAYhKnJ,EAASoJ,WAAa,SAAU1E,EAAMsE,EAASK,GAY7C,QAASC,GAAiB5E,GACxB,GAAYrC,SAATqC,EAEI,GAAGA,YAAgBnD,OACxB,IAAK,GAAIN,GAAI,EAAGA,EAAIyD,EAAKrD,OAAQJ,IAC/BqI,EAAiB5E,EAAKzD,QAEnB,CACL,GAAIc,GAAQsH,GAAa3E,EAAK2E,IAAc3E,CAExC6E,IAAYxH,EAAQyH,EAAQC,OAC9BD,EAAQC,KAAO1H,GAGb2H,GAAW3H,EAAQyH,EAAQG,MAC7BH,EAAQG,IAAM5H,IAzBpBiH,EAAUhJ,EAASe,UAAWiI,EAASK,EAAYL,EAAQ,OAASK,EAAUO,kBAE9E,IAAIJ,IACAC,KAAuBpH,SAAjB2G,EAAQS,MAAsBI,OAAOC,WAAad,EAAQS,KAChEE,IAAqBtH,SAAhB2G,EAAQW,IAAoBE,OAAOC,WAAad,EAAQW,KAE7DJ,EAA4BlH,SAAjB2G,EAAQS,KACnBC,EAA0BrH,SAAhB2G,EAAQW,GAuDtB,QA/BGJ,GAAYG,IACbJ,EAAiB5E,IAMfsE,EAAQe,gBAA6C,IAA3Bf,EAAQe,kBACpCP,EAAQC,KAAOlG,KAAKC,IAAIwF,EAAQe,eAAgBP,EAAQC,MACxDD,EAAQG,IAAMpG,KAAKyG,IAAIhB,EAAQe,eAAgBP,EAAQG,MAKrDH,EAAQC,MAAQD,EAAQG,MAEN,IAAhBH,EAAQG,IACVH,EAAQC,KAAO,EACND,EAAQG,IAAM,EAEvBH,EAAQC,KAAO,EACND,EAAQC,KAAO,EAExBD,EAAQG,IAAM,GAGdH,EAAQC,KAAO,EACfD,EAAQG,IAAM,IAIXH,GAUTxJ,EAASiK,UAAY,SAASlI,GAC5B,MAAiB,QAAVA,GAAyBmI,SAASnI,IAU3C/B,EAASmK,gBAAkB,SAASpI,GAClC,OAAQA,GAAmB,IAAVA,GAUnB/B,EAAS0H,qBAAuB,SAAS3F,GACvC,MAAO/B,GAASiK,UAAUlI,IAAUA,EAAQM,QAS9CrC,EAASoK,aAAe,SAASrI,GAC/B,MAAwB,gBAAVA,KAAuB,KAAOA,IAAS,KAAOA,KAY9D/B,EAASqK,cAAgB,SAAStI,EAAOsH,GACvC,MAAGrJ,GAASoK,aAAarI,GAChB/B,EAAS0H,qBAAqB3F,EAAMsH,GAAa,MAEjDrJ,EAAS0H,qBAAqB3F,IAWzC/B,EAASsK,IAAM,SAAStH,GAKtB,QAASuH,GAAIC,EAAGC,GACd,MAAID,GAAIC,IAAM,EACLA,EAEAF,EAAIE,EAAGD,EAAIC,GAItB,QAASC,GAAE9C,GACT,MAAOA,GAAIA,EAAI,EAbjB,GAAW,IAAR5E,EACD,MAAOA,EAeT,IAAoB2H,GAAhBC,EAAK,EAAGC,EAAK,CACjB,IAAI7H,EAAM,IAAM,EACd,MAAO,EAGT,GACE4H,GAAKF,EAAEE,GAAM5H,EACb6H,EAAKH,EAAEA,EAAEG,IAAO7H,EAChB2H,EAAUJ,EAAIhH,KAAKkF,IAAImC,EAAKC,GAAK7H,SACd,IAAZ2H,EAET,OAAOA,IAaT3K,EAAS8K,UAAY,SAAUlC,EAAYY,EAASuB,EAAeC,GAuDjE,QAASC,GAAclJ,EAAOmJ,GAK5B,MAHInJ,MAAWA,GAASmJ,KACvBnJ,GAAU,GAAKmJ,EAAY,EAAIC,GAAWA,IAEpCpJ,EA3DT,GAAId,GAEFmK,EACAC,EAFAC,EAAsB,EAGtBzC,GACEY,KAAMD,EAAQC,KACdE,IAAKH,EAAQG,IAGjBd,GAAO0C,WAAa1C,EAAOY,KAAOZ,EAAOc,IACzCd,EAAO2C,IAAMxL,EAASsI,iBAAiBO,EAAO0C,YAC9C1C,EAAO4C,KAAOlI,KAAKU,IAAI,GAAI4E,EAAO2C,KAClC3C,EAAOmB,IAAMzG,KAAKgF,MAAMM,EAAOc,IAAMd,EAAO4C,MAAQ5C,EAAO4C,KAC3D5C,EAAOrF,IAAMD,KAAKmI,KAAK7C,EAAOY,KAAOZ,EAAO4C,MAAQ5C,EAAO4C,KAC3D5C,EAAOC,MAAQD,EAAOrF,IAAMqF,EAAOmB,IACnCnB,EAAO8C,cAAgBpI,KAAKW,MAAM2E,EAAOC,MAAQD,EAAO4C,KAIxD,IAAIpK,GAASrB,EAAS2I,cAAcC,EAAYC,EAAO4C,KAAM5C,GACzD+C,EAAUvK,EAAS0J,EACnBc,EAAiBb,EAAchL,EAASsK,IAAIzB,EAAOC,OAAS,CAGhE,IAAGkC,GAAehL,EAAS2I,cAAcC,EAAY,EAAGC,IAAWkC,EACjElC,EAAO4C,KAAO,MACT,IAAGT,GAAea,EAAiBhD,EAAO4C,MAAQzL,EAAS2I,cAAcC,EAAYiD,EAAgBhD,IAAWkC,EAIrHlC,EAAO4C,KAAOI,MAGd,QAAa,CACX,GAAID,GAAW5L,EAAS2I,cAAcC,EAAYC,EAAO4C,KAAM5C,IAAWkC,EACxElC,EAAO4C,MAAQ,MACV,CAAA,GAAKG,KAAW5L,EAAS2I,cAAcC,EAAYC,EAAO4C,KAAO,EAAG5C,IAAWkC,GAOpF,KALA,IADAlC,EAAO4C,MAAQ,EACZT,GAAenC,EAAO4C,KAAO,IAAM,EAAG,CACvC5C,EAAO4C,MAAQ,CACf,QAMJ,GAAGH,IAAwB,IACzB,KAAM,IAAIQ,OAAM,sEAKtB,GAAIX,GAAU,SAad,KAZAtC,EAAO4C,KAAOlI,KAAKC,IAAIqF,EAAO4C,KAAMN,GAUpCC,EAASvC,EAAOmB,IAChBqB,EAASxC,EAAOrF,IACT4H,EAASvC,EAAO4C,MAAQ5C,EAAOc,KACrCyB,EAASH,EAAcG,EAAQvC,EAAO4C,KAEvC,MAAOJ,EAASxC,EAAO4C,MAAQ5C,EAAOY,MACrC4B,EAASJ,EAAcI,GAASxC,EAAO4C,KAExC5C,GAAOmB,IAAMoB,EACbvC,EAAOrF,IAAM6H,EACbxC,EAAOC,MAAQD,EAAOrF,IAAMqF,EAAOmB,GAEnC,IAAI+B,KACJ,KAAK9K,EAAI4H,EAAOmB,IAAK/I,GAAK4H,EAAOrF,IAAKvC,EAAIgK,EAAchK,EAAG4H,EAAO4C,MAAO,CACvE,GAAI1J,GAAQ/B,EAAS8D,mBAAmB7C,EACpCc,KAAUgK,EAAOA,EAAO1K,OAAS,IACnC0K,EAAO/E,KAAKjF,GAIhB,MADA8G,GAAOkD,OAASA,EACTlD,GAaT7I,EAASgM,iBAAmB,SAAUC,EAASC,EAASC,EAAQC,GAC9D,GAAIC,IAAkBD,EAAiB,IAAM7I,KAAK+I,GAAK,GAEvD,QACE1E,EAAGqE,EAAWE,EAAS5I,KAAKgJ,IAAIF,GAChC1E,EAAGuE,EAAWC,EAAS5I,KAAKiJ,IAAIH,KAapCrM,EAASyM,gBAAkB,SAAUpM,EAAK2I,EAAS0D,GACjD,GAAIC,MAAa3D,EAAQE,QAASF,EAAQ4D,OACtCC,EAAcF,EAAU3D,EAAQ4D,MAAMzD,OAAS,EAC/C2D,EAAcH,EAAU3D,EAAQE,MAAMC,OAAS,EAE/C9D,EAAQhF,EAAIgF,SAAWrF,EAASiC,SAAS+G,EAAQ3D,OAAOtD,OAAS,EACjEuD,EAASjF,EAAIiF,UAAYtF,EAASiC,SAAS+G,EAAQ1D,QAAQvD,OAAS,EACpEgL,EAAoB/M,EAAS6H,iBAAiBmB,EAAQC,aAAcyD,EAGxErH,GAAQ9B,KAAKC,IAAI6B,EAAOwH,EAAcE,EAAkB5E,KAAO4E,EAAkB9E,OACjF3C,EAAS/B,KAAKC,IAAI8B,EAAQwH,EAAcC,EAAkB/E,IAAM+E,EAAkB7E,OAElF,IAAI8E,IACFlF,QAASiF,EACT1H,MAAO,WACL,MAAOtF,MAAK8K,GAAK9K,KAAK6K,IAExBtF,OAAQ,WACN,MAAOvF,MAAKkN,GAAKlN,KAAKmN,IA2B1B,OAvBGP,IAC8B,UAA3B3D,EAAQE,MAAMiE,UAChBH,EAAUE,GAAKH,EAAkB/E,IAAM8E,EACvCE,EAAUC,GAAK1J,KAAKC,IAAI8B,EAASyH,EAAkB7E,OAAQ8E,EAAUE,GAAK,KAE1EF,EAAUE,GAAKH,EAAkB/E,IACjCgF,EAAUC,GAAK1J,KAAKC,IAAI8B,EAASyH,EAAkB7E,OAAS4E,EAAaE,EAAUE,GAAK,IAG3D,UAA3BlE,EAAQ4D,MAAMO,UAChBH,EAAUpC,GAAKmC,EAAkB5E,KAAO0E,EACxCG,EAAUnC,GAAKtH,KAAKC,IAAI6B,EAAQ0H,EAAkB9E,MAAO+E,EAAUpC,GAAK,KAExEoC,EAAUpC,GAAKmC,EAAkB5E,KACjC6E,EAAUnC,GAAKtH,KAAKC,IAAI6B,EAAQ0H,EAAkB9E,MAAQ4E,EAAaG,EAAUpC,GAAK,MAGxFoC,EAAUpC,GAAKmC,EAAkB5E,KACjC6E,EAAUnC,GAAKtH,KAAKC,IAAI6B,EAAQ0H,EAAkB9E,MAAO+E,EAAUpC,GAAK,GACxEoC,EAAUE,GAAKH,EAAkB/E,IACjCgF,EAAUC,GAAK1J,KAAKC,IAAI8B,EAASyH,EAAkB7E,OAAQ8E,EAAUE,GAAK,IAGrEF,GAgBThN,EAASoN,WAAa,SAASD,EAAUvJ,EAAOyJ,EAAMlE,EAAQ9H,EAAQiM,EAAOC,EAASC,GACpF,GAAIC,KACJA,GAAeJ,EAAKK,MAAMC,IAAM,KAAOR,EACvCM,EAAeJ,EAAKK,MAAMC,IAAM,KAAOR,EACvCM,EAAeJ,EAAKO,aAAaD,IAAM,KAAOxE,EAC9CsE,EAAeJ,EAAKO,aAAaD,IAAM,KAAOxE,EAAS9H,CAEvD,IAAIwM,GAAcP,EAAMQ,KAAK,OAAQL,EAAgBF,EAAQQ,KAAK,KAGlEP,GAAaQ,KAAK,OAChBhO,EAASe,QACPkN,KAAM,OACNZ,KAAMA,EACNzJ,MAAOA,EACP0J,MAAOA,EACPY,QAASL,GACRJ,KAaPzN,EAASmO,qBAAuB,SAAUC,EAAWpB,EAAWzH,EAAWiI,GACzE,GAAIa,GAAiBD,EAAUN,KAAK,QAChClG,EAAGoF,EAAUpC,GACbjD,EAAGqF,EAAUE,GACb7H,MAAO2H,EAAU3H,QACjBC,OAAQ0H,EAAU1H,UACjBC,GAAW,EAGdiI,GAAaQ,KAAK,QAChBC,KAAM,iBACNX,MAAOc,EACPF,QAASG,KAoBfrO,EAASsO,YAAc,SAASnB,EAAU9L,EAAQuC,EAAOmD,EAAQsG,EAAMkB,EAAYC,EAAalB,EAAOC,EAASkB,EAAkBjB,GAChI,GAAIkB,GACAjB,IAOJ,IALAA,EAAeJ,EAAKK,MAAMC,KAAOR,EAAWqB,EAAYnB,EAAKK,MAAMC,KACnEF,EAAeJ,EAAKO,aAAaD,KAAOa,EAAYnB,EAAKO,aAAaD,KACtEF,EAAeJ,EAAKK,MAAMiB,KAAOtN,EACjCoM,EAAeJ,EAAKO,aAAae,KAAOpL,KAAKC,IAAI,EAAG+K,EAAa,IAE9DE,EAAkB,CAGnB,GAAIG,GAAU,gBAAkBrB,EAAQQ,KAAK,KAAO,YAClDV,EAAKK,MAAMiB,IAAM,KAAOpL,KAAKW,MAAMuJ,EAAeJ,EAAKK,MAAMiB,MAAQ,OACrEtB,EAAKO,aAAae,IAAM,KAAOpL,KAAKW,MAAMuJ,EAAeJ,EAAKO,aAAae,MAAQ,OACnF5H,EAAOnD,GAAS,SAElB8K,GAAepB,EAAMuB,cAAcD,EAAS5O,EAASe,QACnDmF,MAAO,sBACNuH,QAEHiB,GAAepB,EAAMQ,KAAK,OAAQL,EAAgBF,EAAQQ,KAAK,MAAMe,KAAK/H,EAAOnD,GAGnF4J,GAAaQ,KAAK,OAAQhO,EAASe,QACjCkN,KAAM,QACNZ,KAAMA,EACNzJ,MAAOA,EACP0J,MAAOA,EACPY,QAASQ,EACTI,KAAM/H,EAAOnD,IACZ6J,KAYLzN,EAAS+O,gBAAkB,SAASnI,EAAQoC,EAAShE,GACnD,GAAG4B,EAAOoI,MAAQhG,EAAQpC,QAAUoC,EAAQpC,OAAOA,EAAOoI,MAAO,CAC/D,GAAIC,GAAgBjG,EAAQpC,OAAOA,EAAOoI,KAC1C,OAAOC,GAAc5H,eAAerC,GAAOiK,EAAcjK,GAAOgE,EAAQhE,GAExE,MAAOgE,GAAQhE,IAanBhF,EAASkP,gBAAkB,SAAUlG,EAASmG,EAAmB3B,GAM/D,QAAS4B,GAAqBC,GAC5B,GAAIC,GAAkBC,CAGtB,IAFAA,EAAiBvP,EAASe,UAAWyO,GAEjCL,EACF,IAAKlO,EAAI,EAAGA,EAAIkO,EAAkB9N,OAAQJ,IAAK,CAC7C,GAAIwO,GAAMvP,EAAOwP,WAAWP,EAAkBlO,GAAG,GAC7CwO,GAAIE,UACNJ,EAAiBvP,EAASe,OAAOwO,EAAgBJ,EAAkBlO,GAAG,KAKzEuM,GAAgB6B,GACjB7B,EAAaQ,KAAK,kBAChBsB,gBAAiBA,EACjBC,eAAgBA,IAKtB,QAASK,KACPC,EAAoBlM,QAAQ,SAAS8L,GACnCA,EAAIK,eAAeV,KA5BvB,GACEG,GAEAtO,EAHEuO,EAAcxP,EAASe,UAAWiI,GAEpC6G,IA8BF,KAAK3P,EAAOwP,WACV,KAAM,iEACD,IAAIP,EAET,IAAKlO,EAAI,EAAGA,EAAIkO,EAAkB9N,OAAQJ,IAAK,CAC7C,GAAIwO,GAAMvP,EAAOwP,WAAWP,EAAkBlO,GAAG,GACjDwO,GAAIM,YAAYX,GAChBS,EAAoB7I,KAAKyI,GAM7B,MAFAL,MAGEQ,0BAA2BA,EAC3BI,kBAAmB,WACjB,MAAOhQ,GAASe,UAAWwO,MA8BjCvP,EAASiQ,kBAAoB,SAASC,EAAiBC,EAAWnH,GAChE,GAAIoH,IACFC,aAAa,EACbC,WAAW,EAGbtH,GAAUhJ,EAASe,UAAWqP,EAAgBpH,EAK9C,KAAI,GAHAuH,MACAC,GAAO,EAEHvP,EAAI,EAAGA,EAAIiP,EAAgB7O,OAAQJ,GAAK,EAEQoB,SAAnDrC,EAASqK,cAAc8F,EAAUlP,EAAI,GAAGc,OAErCiH,EAAQsH,YACVE,GAAO,IAGNxH,EAAQqH,aAAepP,GAAK,GAAKiP,EAAgBjP,IAAMiP,EAAgBjP,EAAE,KAE1EuP,GAAO,GAKNA,IACDD,EAASvJ,MACPkJ,mBACAC,eAGFK,GAAO,GAITD,EAASA,EAASlP,OAAS,GAAG6O,gBAAgBlJ,KAAKkJ,EAAgBjP,GAAIiP,EAAgBjP,EAAI,IAC3FsP,EAASA,EAASlP,OAAS,GAAG8O,UAAUnJ,KAAKmJ,EAAUlP,EAAI,IAI/D,OAAOsP,KAETrQ,OAAQC,SAAUH,GAOnB,SAASE,EAAQC,EAAUH,GAC1B,YAEAA,GAASyQ,iBAmBTzQ,EAASyQ,cAAcC,KAAO,SAAS1H,GACrC,GAAIoH,IACFE,WAAW,EAGb,OADAtH,GAAUhJ,EAASe,UAAWqP,EAAgBpH,GACvC,SAAckH,EAAiBC,GAIpC,IAAI,GAHAQ,GAAO,GAAI3Q,GAAS+F,IAAI6K,KACxBJ,GAAO,EAEHvP,EAAI,EAAGA,EAAIiP,EAAgB7O,OAAQJ,GAAK,EAAG,CACjD,GAAI4P,GAAQX,EAAgBjP,GACxB6P,EAAQZ,EAAgBjP,EAAI,GAC5B8P,EAAWZ,EAAUlP,EAAI,EAEiBoB,UAA3CrC,EAASqK,cAAc0G,EAAShP,QAE9ByO,EACDG,EAAKK,KAAKH,EAAOC,GAAO,EAAOC,GAE/BJ,EAAKM,KAAKJ,EAAOC,GAAO,EAAOC,GAGjCP,GAAO,GACExH,EAAQsH,YACjBE,GAAO,GAIX,MAAOG,KA2BX3Q,EAASyQ,cAAcS,OAAS,SAASlI,GACvC,GAAIoH,IACFzF,QAAS,EACT2F,WAAW,EAEbtH,GAAUhJ,EAASe,UAAWqP,EAAgBpH,EAE9C,IAAImI,GAAI,EAAI5N,KAAKC,IAAI,EAAGwF,EAAQ2B,QAEhC,OAAO,UAAgBuF,EAAiBC,GAItC,IAAI,GAFAiB,GAAOC,EAAOC,EADdX,EAAO,GAAI3Q,GAAS+F,IAAI6K,KAGpB3P,EAAI,EAAGA,EAAIiP,EAAgB7O,OAAQJ,GAAK,EAAG,CACjD,GAAI4P,GAAQX,EAAgBjP,GACxB6P,EAAQZ,EAAgBjP,EAAI,GAC5BI,GAAUwP,EAAQO,GAASD,EAC3BJ,EAAWZ,EAAUlP,EAAI,EAEPoB,UAAnB0O,EAAShP,OAEMM,SAAbiP,EACDX,EAAKK,KAAKH,EAAOC,GAAO,EAAOC,GAE/BJ,EAAKY,MACHH,EAAQ/P,EACRgQ,EACAR,EAAQxP,EACRyP,EACAD,EACAC,GACA,EACAC,GAIJK,EAAQP,EACRQ,EAAQP,EACRQ,EAAWP,GACF/H,EAAQsH,YACjBc,EAAQP,EAAQS,EAAWjP,QAI/B,MAAOsO,KA0BX3Q,EAASyQ,cAAce,SAAW,SAASxI,GACzC,GAAIoH,IACFqB,QAAS,EACTnB,WAAW,EAGbtH,GAAUhJ,EAASe,UAAWqP,EAAgBpH,EAE9C,IAAI0I,GAAInO,KAAKyG,IAAI,EAAGzG,KAAKC,IAAI,EAAGwF,EAAQyI,UACtCE,EAAI,EAAID,CAEV,OAAO,SAASF,GAAStB,EAAiBC,GAGxC,GAAII,GAAWvQ,EAASiQ,kBAAkBC,EAAiBC,GACzDG,UAAWtH,EAAQsH,WAGrB,IAAIC,EAASlP,OAGN,CAAA,GAAGkP,EAASlP,OAAS,EAAG,CAG3B,GAAIuQ,KAMN,OAJArB,GAAS5M,QAAQ,SAASkO,GACxBD,EAAM5K,KAAKwK,EAASK,EAAQ3B,gBAAiB2B,EAAQ1B,cAGhDnQ,EAAS+F,IAAI6K,KAAK7C,KAAK6D,GAQ9B,GAJA1B,EAAkBK,EAAS,GAAGL,gBAC9BC,EAAYI,EAAS,GAAGJ,UAGrBD,EAAgB7O,QAAU,EAC3B,MAAOrB,GAASyQ,cAAcC,OAAOR,EAAiBC,EAMxD,KAAK,GAFH2B,GADEnB,GAAO,GAAI3Q,GAAS+F,IAAI6K,MAAOI,KAAKd,EAAgB,GAAIA,EAAgB,IAAI,EAAOC,EAAU,IAGxFlP,EAAI,EAAG8Q,EAAO7B,EAAgB7O,OAAQ0Q,EAAO,GAAKD,EAAI7Q,EAAGA,GAAK,EAAG,CACxE,GAAIuJ,KACD5C,GAAIsI,EAAgBjP,EAAI,GAAI0G,GAAIuI,EAAgBjP,EAAI,KACpD2G,GAAIsI,EAAgBjP,GAAI0G,GAAIuI,EAAgBjP,EAAI,KAChD2G,GAAIsI,EAAgBjP,EAAI,GAAI0G,GAAIuI,EAAgBjP,EAAI,KACpD2G,GAAIsI,EAAgBjP,EAAI,GAAI0G,GAAIuI,EAAgBjP,EAAI,IAEnD6Q,GACG7Q,EAEM8Q,EAAO,IAAM9Q,EACtBuJ,EAAE,IAAM5C,GAAIsI,EAAgB,GAAIvI,GAAIuI,EAAgB,IAC3C6B,EAAO,IAAM9Q,IACtBuJ,EAAE,IAAM5C,GAAIsI,EAAgB,GAAIvI,GAAIuI,EAAgB,IACpD1F,EAAE,IAAM5C,GAAIsI,EAAgB,GAAIvI,GAAIuI,EAAgB,KALpD1F,EAAE,IAAM5C,GAAIsI,EAAgB6B,EAAO,GAAIpK,GAAIuI,EAAgB6B,EAAO,IAQhEA,EAAO,IAAM9Q,EACfuJ,EAAE,GAAKA,EAAE,GACCvJ,IACVuJ,EAAE,IAAM5C,GAAIsI,EAAgBjP,GAAI0G,GAAIuI,EAAgBjP,EAAI,KAI5D0P,EAAKY,MACFG,IAAMlH,EAAE,GAAG5C,EAAI,EAAI4C,EAAE,GAAG5C,EAAI4C,EAAE,GAAG5C,GAAK,EAAM+J,EAAInH,EAAE,GAAG5C,EACrD8J,IAAMlH,EAAE,GAAG7C,EAAI,EAAI6C,EAAE,GAAG7C,EAAI6C,EAAE,GAAG7C,GAAK,EAAMgK,EAAInH,EAAE,GAAG7C,EACrD+J,GAAKlH,EAAE,GAAG5C,EAAI,EAAI4C,EAAE,GAAG5C,EAAI4C,EAAE,GAAG5C,GAAK,EAAM+J,EAAInH,EAAE,GAAG5C,EACpD8J,GAAKlH,EAAE,GAAG7C,EAAI,EAAI6C,EAAE,GAAG7C,EAAI6C,EAAE,GAAG7C,GAAK,EAAMgK,EAAInH,EAAE,GAAG7C,EACrD6C,EAAE,GAAG5C,EACL4C,EAAE,GAAG7C,GACL,EACAwI,GAAWlP,EAAI,GAAK,IAIxB,MAAO0P,GA7DP,MAAO3Q,GAASyQ,cAAcC,aAyFpC1Q,EAASyQ,cAAcuB,cAAgB,SAAShJ,GAC9C,GAAIoH,IACFE,WAAW,EAKb,OAFAtH,GAAUhJ,EAASe,UAAWqP,EAAgBpH,GAEvC,QAASgJ,GAAc9B,EAAiBC,GAG7C,GAAII,GAAWvQ,EAASiQ,kBAAkBC,EAAiBC,GACzDG,UAAWtH,EAAQsH,UACnBD,aAAa,GAGf,IAAIE,EAASlP,OAGN,CAAA,GAAGkP,EAASlP,OAAS,EAAG,CAG3B,GAAIuQ,KAMN,OAJArB,GAAS5M,QAAQ,SAASkO,GACxBD,EAAM5K,KAAKgL,EAAcH,EAAQ3B,gBAAiB2B,EAAQ1B,cAGrDnQ,EAAS+F,IAAI6K,KAAK7C,KAAK6D,GAQ9B,GAJA1B,EAAkBK,EAAS,GAAGL,gBAC9BC,EAAYI,EAAS,GAAGJ,UAGrBD,EAAgB7O,QAAU,EAC3B,MAAOrB,GAASyQ,cAAcC,OAAOR,EAAiBC,EAGxD,IAEElP,GAIA0P,EANEsB,KACFC,KAEAvR,EAAIuP,EAAgB7O,OAAS,EAC7B8Q,KACAC,KAASC,KAAUC,IAKrB,KAAIrR,EAAI,EAAGA,EAAIN,EAAGM,IAChBgR,EAAGhR,GAAKiP,EAAoB,EAAJjP,GACxBiR,EAAGjR,GAAKiP,EAAoB,EAAJjP,EAAQ,EAKlC,KAAIA,EAAI,EAAGA,EAAIN,EAAI,EAAGM,IACpBoR,EAAIpR,GAAKiR,EAAGjR,EAAI,GAAKiR,EAAGjR,GACxBqR,EAAIrR,GAAKgR,EAAGhR,EAAI,GAAKgR,EAAGhR,GACxBmR,EAAGnR,GAAKoR,EAAIpR,GAAKqR,EAAIrR,EASvB,KAHAkR,EAAG,GAAKC,EAAG,GACXD,EAAGxR,EAAI,GAAKyR,EAAGzR,EAAI,GAEfM,EAAI,EAAGA,EAAIN,EAAI,EAAGM,IACP,IAAVmR,EAAGnR,IAA0B,IAAdmR,EAAGnR,EAAI,IAAamR,EAAGnR,EAAI,GAAK,GAAQmR,EAAGnR,GAAK,EAChEkR,EAAGlR,GAAK,GAERkR,EAAGlR,GAAK,GAAKqR,EAAIrR,EAAI,GAAKqR,EAAIrR,MAC3B,EAAIqR,EAAIrR,GAAKqR,EAAIrR,EAAI,IAAMmR,EAAGnR,EAAI,IAClCqR,EAAIrR,GAAK,EAAIqR,EAAIrR,EAAI,IAAMmR,EAAGnR,IAE7BiJ,SAASiI,EAAGlR,MACdkR,EAAGlR,GAAK,GASd,KAFA0P,GAAO,GAAI3Q,GAAS+F,IAAI6K,MAAOI,KAAKiB,EAAG,GAAIC,EAAG,IAAI,EAAO/B,EAAU,IAE/DlP,EAAI,EAAGA,EAAIN,EAAI,EAAGM,IACpB0P,EAAKY,MAEHU,EAAGhR,GAAKqR,EAAIrR,GAAK,EACjBiR,EAAGjR,GAAKkR,EAAGlR,GAAKqR,EAAIrR,GAAK,EAEzBgR,EAAGhR,EAAI,GAAKqR,EAAIrR,GAAK,EACrBiR,EAAGjR,EAAI,GAAKkR,EAAGlR,EAAI,GAAKqR,EAAIrR,GAAK,EAEjCgR,EAAGhR,EAAI,GACPiR,EAAGjR,EAAI,IAEP,EACAkP,EAAUlP,EAAI,GAIlB,OAAO0P,GAtFP,MAAO3Q,GAASyQ,cAAcC,aA+GpC1Q,EAASyQ,cAAchF,KAAO,SAASzC,GACrC,GAAIoH,IACFmC,UAAU,EACVjC,WAAW,EAKb,OAFAtH,GAAUhJ,EAASe,UAAWqP,EAAgBpH,GAEvC,SAAckH,EAAiBC,GAKpC,IAAK,GAFDiB,GAAOC,EAAOC,EAFdX,EAAO,GAAI3Q,GAAS+F,IAAI6K,KAInB3P,EAAI,EAAGA,EAAIiP,EAAgB7O,OAAQJ,GAAK,EAAG,CAClD,GAAI4P,GAAQX,EAAgBjP,GACxB6P,EAAQZ,EAAgBjP,EAAI,GAC5B8P,EAAWZ,EAAUlP,EAAI,EAGPoB,UAAnB0O,EAAShP,OACMM,SAAbiP,EACDX,EAAKK,KAAKH,EAAOC,GAAO,EAAOC,IAE5B/H,EAAQuJ,SAET5B,EAAKM,KAAKJ,EAAOQ,GAAO,EAAOC,GAG/BX,EAAKM,KAAKG,EAAON,GAAO,EAAOC,GAGjCJ,EAAKM,KAAKJ,EAAOC,GAAO,EAAOC,IAGjCK,EAAQP,EACRQ,EAAQP,EACRQ,EAAWP,GACF/H,EAAQsH,YACjBc,EAAQC,EAAQC,EAAWjP,QAI/B,MAAOsO,MAIXzQ,OAAQC,SAAUH,GAOnB,SAAUE,EAAQC,EAAUH,GAC3B,YAEAA,GAASwS,aAAe,WAUtB,QAASC,GAAgBC,EAAOC,GAC9BC,EAASF,GAASE,EAASF,OAC3BE,EAASF,GAAO1L,KAAK2L,GAUvB,QAASE,GAAmBH,EAAOC,GAE9BC,EAASF,KAEPC,GACDC,EAASF,GAAOI,OAAOF,EAASF,GAAOK,QAAQJ,GAAU,GAC3B,IAA3BC,EAASF,GAAOrR,cACVuR,GAASF,UAIXE,GAASF,IAYtB,QAAS1E,GAAK0E,EAAOhO,GAEhBkO,EAASF,IACVE,EAASF,GAAO/O,QAAQ,SAASgP,GAC/BA,EAAQjO,KAKTkO,EAAS,MACVA,EAAS,KAAKjP,QAAQ,SAASqP,GAC7BA,EAAYN,EAAOhO,KAvDzB,GAAIkO,KA4DJ,QACEH,gBAAiBA,EACjBI,mBAAoBA,EACpB7E,KAAMA,KAIV9N,OAAQC,SAAUH,GAOnB,SAASE,EAAQC,EAAUH,GAC1B,YAEA,SAASiT,GAAYC,GACnB,GAAI9P,KACJ,IAAI8P,EAAK7R,OACP,IAAK,GAAIJ,GAAI,EAAGA,EAAIiS,EAAK7R,OAAQJ,IAC/BmC,EAAI4D,KAAKkM,EAAKjS,GAGlB,OAAOmC,GA4CT,QAASrC,GAAOoS,EAAYC,GAC1B,GAAIC,GAAaD,GAAsBrT,KAAKyF,WAAaxF,EAASsT,MAC9DC,EAAQ1O,OAAO2O,OAAOH,EAE1BrT,GAASsT,MAAMG,iBAAiBF,EAAOJ,EAEvC,IAAIO,GAAS,WACX,GACEC,GADEC,EAAKL,EAAMM,aAAe,YAU9B,OALAF,GAAW5T,OAASC,EAAW6E,OAAO2O,OAAOD,GAASxT,KACtD6T,EAAGlR,MAAMiR,EAAUpS,MAAMiE,UAAUC,MAAMC,KAAKtE,UAAW,IAIlDuS,EAOT,OAJAD,GAAOlO,UAAY+N,EACnBG,EAAAA,SAAeL,EACfK,EAAO3S,OAAShB,KAAKgB,OAEd2S,EAIT,QAASD,KACP,GAAI5P,GAAOoP,EAAY7R,WACnBJ,EAAS6C,EAAK,EAYlB,OAVAA,GAAKiP,OAAO,EAAGjP,EAAKxC,OAAS,GAAGsC,QAAQ,SAAUzC,GAChD2D,OAAOiP,oBAAoB5S,GAAQyC,QAAQ,SAAUoQ,SAE5C/S,GAAO+S,GAEdlP,OAAOmP,eAAehT,EAAQ+S,EAC5BlP,OAAOoP,yBAAyB/S,EAAQ6S,QAIvC/S,EAGThB,EAASsT,OACPvS,OAAQA,EACR0S,iBAAkBA,IAGpBvT,OAAQC,SAAUH,GAOnB,SAASE,EAAQC,EAAUH,GAC1B,YAgBA,SAASkU,GAAOxP,EAAMsE,EAASmL,GA6B7B,MA5BGzP,KACD3E,KAAK2E,KAAOA,MACZ3E,KAAK2E,KAAKqC,OAAShH,KAAK2E,KAAKqC,WAC7BhH,KAAK2E,KAAKkC,OAAS7G,KAAK2E,KAAKkC,WAE7B7G,KAAKyN,aAAaQ,KAAK,QACrBC,KAAM,SACNvJ,KAAM3E,KAAK2E,QAIZsE,IACDjJ,KAAKiJ,QAAUhJ,EAASe,UAAWoT,EAAWpU,KAAKiJ,QAAUjJ,KAAKqQ,eAAgBpH,GAI9EjJ,KAAKqU,sBACPrU,KAAKmP,gBAAgBU,4BACrB7P,KAAKmP,gBAAkBlP,EAASkP,gBAAgBnP,KAAKiJ,QAASjJ,KAAKoP,kBAAmBpP,KAAKyN,gBAK3FzN,KAAKqU,qBACPrU,KAAKsU,YAAYtU,KAAKmP,gBAAgBc,qBAIjCjQ,KAQT,QAASuU,KAUP,MAPIvU,MAAKqU,oBAIPlU,EAAOqU,aAAaxU,KAAKqU,sBAHzBlU,EAAOsU,oBAAoB,SAAUzU,KAAK0U,gBAC1C1U,KAAKmP,gBAAgBU,6BAKhB7P,KAUT,QAAS2U,GAAGhC,EAAOC,GAEjB,MADA5S,MAAKyN,aAAaiF,gBAAgBC,EAAOC,GAClC5S,KAUT,QAAS4U,GAAIjC,EAAOC,GAElB,MADA5S,MAAKyN,aAAaqF,mBAAmBH,EAAOC,GACrC5S,KAGT,QAAS6U,KAEP1U,EAAO2U,iBAAiB,SAAU9U,KAAK0U,gBAIvC1U,KAAKmP,gBAAkBlP,EAASkP,gBAAgBnP,KAAKiJ,QAASjJ,KAAKoP,kBAAmBpP,KAAKyN,cAE3FzN,KAAKyN,aAAaiF,gBAAgB,iBAAkB,WAClD1S,KAAKmU,UACLY,KAAK/U,OAIJA,KAAKiJ,QAAQ+L,SACdhV,KAAKiJ,QAAQ+L,QAAQpR,QAAQ,SAASqR,GACjCA,YAAkBzT,OACnByT,EAAO,GAAGjV,KAAMiV,EAAO,IAEvBA,EAAOjV,OAET+U,KAAK/U,OAITA,KAAKyN,aAAaQ,KAAK,QACrBC,KAAM,UACNvJ,KAAM3E,KAAK2E,OAIb3E,KAAKsU,YAAYtU,KAAKmP,gBAAgBc,qBAItCjQ,KAAKqU,oBAAsB/R,OAa7B,QAAS4S,GAAK1S,EAAOmC,EAAM0L,EAAgBpH,EAASmG,GAClDpP,KAAKqF,UAAYpF,EAASsC,cAAcC,GACxCxC,KAAK2E,KAAOA,MACZ3E,KAAK2E,KAAKqC,OAAShH,KAAK2E,KAAKqC,WAC7BhH,KAAK2E,KAAKkC,OAAS7G,KAAK2E,KAAKkC,WAC7B7G,KAAKqQ,eAAiBA,EACtBrQ,KAAKiJ,QAAUA,EACfjJ,KAAKoP,kBAAoBA,EACzBpP,KAAKyN,aAAexN,EAASwS,eAC7BzS,KAAKmV,sBAAwBlV,EAAS+F,IAAIoP,YAAY,iBACtDpV,KAAKqV,mBAAqBpV,EAAS+F,IAAIoP,YAAY,4BACnDpV,KAAK0U,eAAiB,WACpB1U,KAAKmU,UACLY,KAAK/U,MAEJA,KAAKqF,YAEHrF,KAAKqF,UAAUiQ,cAChBtV,KAAKqF,UAAUiQ,aAAaf,SAG9BvU,KAAKqF,UAAUiQ,aAAetV,MAKhCA,KAAKqU,oBAAsBkB,WAAWV,EAAWE,KAAK/U,MAAO,GAI/DC,EAASiV,KAAOjV,EAASsT,MAAMvS,QAC7B8S,YAAaoB,EACb/F,gBAAiB7M,OACjB+C,UAAW/C,OACXhC,IAAKgC,OACLmL,aAAcnL,OACdgS,YAAa,WACX,KAAM,IAAIvI,OAAM,2CAElBoI,OAAQA,EACRI,OAAQA,EACRI,GAAIA,EACJC,IAAKA,EACL1U,QAASD,EAASC,QAClBiV,uBAAuB,KAGzBhV,OAAQC,SAAUH,GAOnB,SAASE,EAAQC,EAAUH,GAC1B,YAaA,SAAS+F,GAAIiJ,EAAMuG,EAAYhQ,EAAWiQ,EAAQC,GAE7CzG,YAAgB0G,SACjB3V,KAAKqG,MAAQ4I,GAEbjP,KAAKqG,MAAQjG,EAASwV,gBAAgB3V,EAASI,WAAWC,IAAK2O,GAGnD,QAATA,GACDjP,KAAKiG,MACH4P,WAAY5V,EAASI,WAAWK,MAKnC8U,GACDxV,KAAKiG,KAAKuP,GAGThQ,GACDxF,KAAKkG,SAASV,GAGbiQ,IACGC,GAAeD,EAAOpP,MAAMyP,WAC9BL,EAAOpP,MAAM0P,aAAa/V,KAAKqG,MAAOoP,EAAOpP,MAAMyP,YAEnDL,EAAOpP,MAAMD,YAAYpG,KAAKqG,QAapC,QAASJ,GAAKuP,EAAYQ,GACxB,MAAyB,gBAAfR,GACLQ,EACMhW,KAAKqG,MAAMP,eAAekQ,EAAIR,GAE9BxV,KAAKqG,MAAM4P,aAAaT,IAInC1Q,OAAOC,KAAKyQ,GAAY5R,QAAQ,SAASqB,GAEvC,GAAuB3C,SAApBkT,EAAWvQ,GAId,GAAIA,EAAI+N,QAAQ,UAAa,CAC3B,GAAIkD,GAAsBjR,EAAIkR,MAAM,IACpCnW,MAAKqG,MAAM+P,eAAenW,EAASI,WAAW6V,EAAoB,IAAKjR,EAAKuQ,EAAWvQ,QAEvFjF,MAAKqG,MAAMgQ,aAAapR,EAAKuQ,EAAWvQ,KAE1C8P,KAAK/U,OAEAA,MAaT,QAAS+N,GAAKkB,EAAMuG,EAAYhQ,EAAWkQ,GACzC,MAAO,IAAIzV,GAAS+F,IAAIiJ,EAAMuG,EAAYhQ,EAAWxF,KAAM0V,GAS7D,QAASD,KACP,MAAOzV,MAAKqG,MAAMiQ,qBAAsBC,YAAa,GAAItW,GAAS+F,IAAIhG,KAAKqG,MAAMiQ,YAAc,KASjG,QAAS5W,KAEP,IADA,GAAI8W,GAAOxW,KAAKqG,MACQ,QAAlBmQ,EAAKC,UACTD,EAAOA,EAAKF,UAEd,OAAO,IAAIrW,GAAS+F,IAAIwQ,GAU1B,QAASjU,GAAcmU,GACrB,GAAIC,GAAY3W,KAAKqG,MAAM9D,cAAcmU,EACzC,OAAOC,GAAY,GAAI1W,GAAS+F,IAAI2Q,GAAa,KAUnD,QAAS/Q,GAAiB8Q,GACxB,GAAIE,GAAa5W,KAAKqG,MAAMT,iBAAiB8Q,EAC7C,OAAOE,GAAWtV,OAAS,GAAIrB,GAAS+F,IAAI6Q,KAAKD,GAAc,KASjE,QAASE,KACP,MAAO9W,MAAKqG,MAad,QAASyI,GAAcD,EAAS2G,EAAYhQ,EAAWkQ,GAGrD,GAAsB,gBAAZ7G,GAAsB,CAC9B,GAAIxJ,GAAYjF,EAAS2W,cAAc,MACvC1R,GAAU2R,UAAYnI,EACtBA,EAAUxJ,EAAUyQ,WAItBjH,EAAQwH,aAAa,QAASpW,EAASI,WAAWE,MAIlD,IAAI0W,GAAQjX,KAAK+N,KAAK,gBAAiByH,EAAYhQ,EAAWkQ,EAK9D,OAFAuB,GAAM5Q,MAAMD,YAAYyI,GAEjBoI,EAUT,QAASlI,GAAK4C,GAEZ,MADA3R,MAAKqG,MAAMD,YAAYhG,EAAS8W,eAAevF,IACxC3R,KAST,QAASmX,KACP,KAAOnX,KAAKqG,MAAMyP,YAChB9V,KAAKqG,MAAMN,YAAY/F,KAAKqG,MAAMyP,WAGpC,OAAO9V,MAST,QAASoX,KAEP,MADApX,MAAKqG,MAAMiQ,WAAWvQ,YAAY/F,KAAKqG,OAChCrG,KAAKyV,SAUd,QAAS5T,GAAQwV,GAEf,MADArX,MAAKqG,MAAMiQ,WAAWgB,aAAaD,EAAWhR,MAAOrG,KAAKqG,OACnDgR,EAWT,QAASE,GAAOpJ,EAASuH,GAOvB,MANGA,IAAe1V,KAAKqG,MAAMyP,WAC3B9V,KAAKqG,MAAM0P,aAAa5H,EAAQ9H,MAAOrG,KAAKqG,MAAMyP,YAElD9V,KAAKqG,MAAMD,YAAY+H,EAAQ9H,OAG1BrG,KAST,QAASwN,KACP,MAAOxN,MAAKqG,MAAM4P,aAAa,SAAWjW,KAAKqG,MAAM4P,aAAa,SAASuB,OAAOrB,MAAM,UAU1F,QAASjQ,GAASuR,GAShB,MARAzX,MAAKqG,MAAMgQ,aAAa,QACtBrW,KAAKwN,QAAQxN,KAAKqG,OACfqR,OAAOD,EAAMD,OAAOrB,MAAM,QAC1BtQ,OAAO,SAASkI,EAAMH,EAAK+J,GAC1B,MAAOA,GAAK3E,QAAQjF,KAAUH,IAC7BI,KAAK,MAGLhO,KAUT,QAAS4X,GAAYH,GACnB,GAAII,GAAiBJ,EAAMD,OAAOrB,MAAM,MAMxC,OAJAnW,MAAKqG,MAAMgQ,aAAa,QAASrW,KAAKwN,QAAQxN,KAAKqG,OAAOR,OAAO,SAASoJ,GACxE,MAAO4I,GAAe7E,QAAQ/D,UAC7BjB,KAAK,MAEDhO,KAST,QAAS8X,KAGP,MAFA9X,MAAKqG,MAAMgQ,aAAa,QAAS,IAE1BrW,KAST,QAASuF,KACP,MAAOvF,MAAKqG,MAAM0R,wBAAwBxS,OAS5C,QAASD,KACP,MAAOtF,MAAKqG,MAAM0R,wBAAwBzS,MA4C5C,QAAS0S,GAAQC,EAAYC,EAAQzK,GA4GnC,MA3GcnL,UAAX4V,IACDA,GAAS,GAGXpT,OAAOC,KAAKkT,GAAYrU,QAAQ,SAAoCuU,GAElE,QAASC,GAAcC,EAAqBH,GAC1C,GACEF,GACAM,EACAC,EAHEC,IAODH,GAAoBE,SAErBA,EAASF,EAAoBE,iBAAkB/W,OAC7C6W,EAAoBE,OACpBtY,EAAS+F,IAAIyS,OAAOJ,EAAoBE,cACnCF,GAAoBE,QAI7BF,EAAoBK,MAAQzY,EAAS8B,WAAWsW,EAAoBK,MAAO,MAC3EL,EAAoBM,IAAM1Y,EAAS8B,WAAWsW,EAAoBM,IAAK,MAEpEJ,IACDF,EAAoBO,SAAW,SAC/BP,EAAoBQ,WAAaN,EAAOvK,KAAK,KAC7CqK,EAAoBS,SAAW,OAI9BZ,IACDG,EAAoBU,KAAO,SAE3BP,EAAoBL,GAAaE,EAAoBW,KACrDhZ,KAAKiG,KAAKuS,GAIVF,EAAUrY,EAASiC,SAASmW,EAAoBK,OAAS,GAAG1W,MAC5DqW,EAAoBK,MAAQ,cAG9BV,EAAUhY,KAAK+N,KAAK,UAAW9N,EAASe,QACtCiY,cAAed,GACdE,IAEAH,GAED3C,WAAW,WAIT,IACEyC,EAAQ3R,MAAM6S,eACd,MAAMC,GAENX,EAAoBL,GAAaE,EAAoBe,GACrDpZ,KAAKiG,KAAKuS,GAEVR,EAAQZ,WAEVrC,KAAK/U,MAAOsY,GAGb7K,GACDuK,EAAQ3R,MAAMyO,iBAAiB,aAAc,WAC3CrH,EAAaQ,KAAK,kBAChBE,QAASnO,KACTgY,QAASA,EAAQ3R,MACjBgT,OAAQhB,KAEVtD,KAAK/U,OAGTgY,EAAQ3R,MAAMyO,iBAAiB,WAAY,WACtCrH,GACDA,EAAaQ,KAAK,gBAChBE,QAASnO,KACTgY,QAASA,EAAQ3R,MACjBgT,OAAQhB,IAITH,IAEDM,EAAoBL,GAAaE,EAAoBe,GACrDpZ,KAAKiG,KAAKuS,GAEVR,EAAQZ,WAEVrC,KAAK/U,OAINiY,EAAWE,YAAsB3W,OAClCyW,EAAWE,GAAWvU,QAAQ,SAASyU,GACrCD,EAAcrD,KAAK/U,MAAMqY,GAAqB,IAC9CtD,KAAK/U,OAEPoY,EAAcrD,KAAK/U,MAAMiY,EAAWE,GAAYD,IAGlDnD,KAAK/U,OAEAA,KAgFT,QAASsZ,GAAQC,GACf,GAAIpG,GAAOnT,IAEXA,MAAKwZ,cACL,KAAI,GAAItY,GAAI,EAAGA,EAAIqY,EAASjY,OAAQJ,IAClClB,KAAKwZ,YAAYvS,KAAK,GAAIhH,GAAS+F,IAAIuT,EAASrY,IAIlD4D,QAAOC,KAAK9E,EAAS+F,IAAIP,WAAWI,OAAO,SAAS4T,GAClD,OAAQ,cACJ,SACA,gBACA,mBACA,UACA,SACA,UACA,SACA,SAASzG,QAAQyG,UACpB7V,QAAQ,SAAS6V,GAClBtG,EAAKsG,GAAqB,WACxB,GAAI3V,GAAOtC,MAAMiE,UAAUC,MAAMC,KAAKtE,UAAW,EAIjD,OAHA8R,GAAKqG,YAAY5V,QAAQ,SAASuK,GAChClO,EAAS+F,IAAIP,UAAUgU,GAAmB9W,MAAMwL,EAASrK,KAEpDqP,KAtGblT,EAAS+F,IAAM/F,EAASsT,MAAMvS,QAC5B8S,YAAa9N,EACbC,KAAMA,EACN8H,KAAMA,EACN0H,OAAQA,EACR/V,KAAMA,EACN6C,cAAeA,EACfqD,iBAAkBA,EAClBkR,QAASA,EACThI,cAAeA,EACfC,KAAMA,EACNoI,MAAOA,EACPC,OAAQA,EACRvV,QAASA,EACT0V,OAAQA,EACR/J,QAASA,EACTtH,SAAUA,EACV0R,YAAaA,EACbE,iBAAkBA,EAClBvS,OAAQA,EACRD,MAAOA,EACP0S,QAASA,IAUX/X,EAAS+F,IAAIoP,YAAc,SAASsE,GAClC,MAAOtZ,GAASuZ,eAAeC,WAAW,sCAAwCF,EAAS,OAQ7F,IAAIG,IACFC,YAAa,IAAM,EAAG,KAAO,MAC7BC,aAAc,IAAM,KAAO,KAAO,GAClCC,eAAgB,KAAO,IAAM,IAAM,KACnCC,YAAa,IAAM,KAAO,IAAM,KAChCC,aAAc,IAAM,IAAM,IAAM,KAChCC,eAAgB,KAAO,IAAM,KAAO,MACpCC,aAAc,IAAM,KAAO,KAAO,KAClCC,cAAe,KAAO,IAAM,KAAO,GACnCC,gBAAiB,KAAO,KAAO,KAAO,GACtCC,aAAc,KAAO,IAAM,KAAO,KAClCC,cAAe,KAAO,IAAM,IAAM,GAClCC,gBAAiB,IAAM,EAAG,KAAO,GACjCC,aAAc,KAAO,IAAM,KAAO,KAClCC,cAAe,IAAM,EAAG,IAAM,GAC9BC,gBAAiB,IAAM,EAAG,IAAM,GAChCC,YAAa,IAAM,IAAM,KAAO,MAChCC,aAAc,IAAM,EAAG,IAAM,GAC7BC,eAAgB,EAAG,EAAG,EAAG,GACzBC,YAAa,GAAK,IAAM,IAAM,MAC9BC,aAAc,KAAO,IAAM,KAAO,GAClCC,eAAgB,KAAO,KAAO,IAAM,KACpCC,YAAa,QAAY,KAAO,MAChCC,aAAc,KAAO,KAAO,IAAM,OAClCC,eAAgB,SAAa,KAAO,MAGtCpb,GAAS+F,IAAIyS,OAASoB,EAwCtB5Z,EAAS+F,IAAI6Q,KAAO5W,EAASsT,MAAMvS,QACjC8S,YAAawF,KAEfnZ,OAAQC,SAAUH,GAOnB,SAASE,EAAQC,EAAUH,GAC1B,YA0BA,SAASkO,GAAQmN,EAASjC,EAAQkC,EAAc3N,EAAK4N,EAAU7W,GAC7D,GAAI8W,GAAcxb,EAASe,QACzBsa,QAASE,EAAWF,EAAQI,cAAgBJ,EAAQzR,eACnDwP,EAAQ1U,GAASA,KAAMA,MAE1B4W,GAAaxI,OAAOnF,EAAK,EAAG6N,GAG9B,QAASE,GAAaJ,EAAcjY,GAClCiY,EAAa3X,QAAQ,SAAS6X,EAAaG,GACzCC,EAAoBJ,EAAYH,QAAQI,eAAe9X,QAAQ,SAASkY,EAAWC,GACjFzY,EAAGmY,EAAaK,EAAWF,EAAkBG,EAAYR,OAa/D,QAASS,GAAQC,EAAOhT,GACtBjJ,KAAKub,gBACLvb,KAAK4N,IAAM,EACX5N,KAAKic,MAAQA,EACbjc,KAAKiJ,QAAUhJ,EAASe,UAAWqP,EAAgBpH,GAUrD,QAASmE,GAASQ,GAChB,MAAWtL,UAARsL,GACD5N,KAAK4N,IAAMpK,KAAKC,IAAI,EAAGD,KAAKyG,IAAIjK,KAAKub,aAAaja,OAAQsM,IACnD5N,MAEAA,KAAK4N,IAWhB,QAASwJ,GAAO8E,GAEd,MADAlc,MAAKub,aAAaxI,OAAO/S,KAAK4N,IAAKsO,GAC5Blc,KAaT,QAASiR,GAAKpJ,EAAGD,EAAG4T,EAAU7W,GAK5B,MAJAwJ,GAAQ,KACNtG,GAAIA,EACJD,GAAIA,GACH5H,KAAKub,aAAcvb,KAAK4N,MAAO4N,EAAU7W,GACrC3E,KAaT,QAASkR,GAAKrJ,EAAGD,EAAG4T,EAAU7W,GAK5B,MAJAwJ,GAAQ,KACNtG,GAAIA,EACJD,GAAIA,GACH5H,KAAKub,aAAcvb,KAAK4N,MAAO4N,EAAU7W,GACrC3E,KAiBT,QAASwR,GAAM3G,EAAIqC,EAAIpC,EAAIqC,EAAItF,EAAGD,EAAG4T,EAAU7W,GAS7C,MARAwJ,GAAQ,KACNtD,IAAKA,EACLqC,IAAKA,EACLpC,IAAKA,EACLqC,IAAKA,EACLtF,GAAIA,EACJD,GAAIA,GACH5H,KAAKub,aAAcvb,KAAK4N,MAAO4N,EAAU7W,GACrC3E,KAkBT,QAASmc,GAAIC,EAAIC,EAAIC,EAAKC,EAAKC,EAAI3U,EAAGD,EAAG4T,EAAU7W,GAUjD,MATAwJ,GAAQ,KACNiO,IAAKA,EACLC,IAAKA,EACLC,KAAMA,EACNC,KAAMA,EACNC,IAAKA,EACL3U,GAAIA,EACJD,GAAIA,GACH5H,KAAKub,aAAcvb,KAAK4N,MAAO4N,EAAU7W,GACrC3E,KAUT,QAASmF,GAAMyL,GAEb,GAAI6L,GAAS7L,EAAK/O,QAAQ,qBAAsB,SAC7CA,QAAQ,qBAAsB,SAC9BsU,MAAM,UACNnR,OAAO,SAASzB,EAAQ4K,GAMvB,MALGA,GAAQ/L,MAAM,aACfmB,EAAO0D,SAGT1D,EAAOA,EAAOjC,OAAS,GAAG2F,KAAKkH,GACxB5K,MAIuC,OAA/CkZ,EAAOA,EAAOnb,OAAS,GAAG,GAAGuI,eAC9B4S,EAAOC,KAKT,IAAIC,GAAWF,EAAO/Y,IAAI,SAASkZ,GAC/B,GAAItB,GAAUsB,EAAMC,QAClBC,EAAcjB,EAAoBP,EAAQI,cAE5C,OAAOzb,GAASe,QACdsa,QAASA,GACRwB,EAAY9X,OAAO,SAASzB,EAAQuY,EAAWjY,GAEhD,MADAN,GAAOuY,IAAcc,EAAM/Y,GACpBN,UAKTwZ,GAAc/c,KAAK4N,IAAK,EAM5B,OALApM,OAAMiE,UAAUwB,KAAKtE,MAAMoa,EAAYJ,GACvCnb,MAAMiE,UAAUsN,OAAOpQ,MAAM3C,KAAKub,aAAcwB,GAEhD/c,KAAK4N,KAAO+O,EAASrb,OAEdtB,KAST,QAAS6E,KACP,GAAImY,GAAqBxZ,KAAKU,IAAI,GAAIlE,KAAKiJ,QAAQgU,SAEnD,OAAOjd,MAAKub,aAAavW,OAAO,SAAS4L,EAAM6K,GAC3C,GAAIpC,GAASwC,EAAoBJ,EAAYH,QAAQI,eAAehY,IAAI,SAASoY,GAC/E,MAAO9b,MAAKiJ,QAAQgU,SACjBzZ,KAAKW,MAAMsX,EAAYK,GAAakB,GAAsBA,EAC3DvB,EAAYK,IACd/G,KAAK/U,MAEP,OAAO4Q,GAAO6K,EAAYH,QAAUjC,EAAOrL,KAAK,MAChD+G,KAAK/U,MAAO,KAAOA,KAAKic,MAAQ,IAAM,IAW5C,QAASiB,GAAMrV,EAAGD,GAIhB,MAHA+T,GAAa3b,KAAKub,aAAc,SAASE,EAAaK,GACpDL,EAAYK,IAA+B,MAAjBA,EAAU,GAAajU,EAAID,IAEhD5H,KAWT,QAASmd,GAAUtV,EAAGD,GAIpB,MAHA+T,GAAa3b,KAAKub,aAAc,SAASE,EAAaK,GACpDL,EAAYK,IAA+B,MAAjBA,EAAU,GAAajU,EAAID,IAEhD5H,KAeT,QAASod,GAAUC,GAOjB,MANA1B,GAAa3b,KAAKub,aAAc,SAASE,EAAaK,EAAWF,EAAkBG,EAAYR,GAC7F,GAAI+B,GAAcD,EAAa5B,EAAaK,EAAWF,EAAkBG,EAAYR,IAClF+B,GAA+B,IAAhBA,KAChB7B,EAAYK,GAAawB,KAGtBtd,KAUT,QAASud,GAAMtB,GACb,GAAIrK,GAAI,GAAI3R,GAAS+F,IAAI6K,KAAKoL,GAASjc,KAAKic,MAM5C,OALArK,GAAEhE,IAAM5N,KAAK4N,IACbgE,EAAE2J,aAAevb,KAAKub,aAAa7V,QAAQhC,IAAI,SAAuB+X,GACpE,MAAOxb,GAASe,UAAWya,KAE7B7J,EAAE3I,QAAUhJ,EAASe,UAAWhB,KAAKiJ,SAC9B2I,EAUT,QAAS4L,GAAelC,GACtB,GAAInF,IACF,GAAIlW,GAAS+F,IAAI6K,KAWnB,OARA7Q,MAAKub,aAAa3X,QAAQ,SAAS6X,GAC9BA,EAAYH,UAAYA,EAAQzR,eAAiE,IAAhDsM,EAAMA,EAAM7U,OAAS,GAAGia,aAAaja,QACvF6U,EAAMlP,KAAK,GAAIhH,GAAS+F,IAAI6K,MAG9BsF,EAAMA,EAAM7U,OAAS,GAAGia,aAAatU,KAAKwU,KAGrCtF,EAaT,QAASnI,GAAK6D,EAAOoK,EAAOhT,GAE1B,IAAI,GADAwU,GAAa,GAAIxd,GAAS+F,IAAI6K,KAAKoL,EAAOhT,GACtC/H,EAAI,EAAGA,EAAI2Q,EAAMvQ,OAAQJ,IAE/B,IAAI,GADA0P,GAAOiB,EAAM3Q,GACTwc,EAAI,EAAGA,EAAI9M,EAAK2K,aAAaja,OAAQoc,IAC3CD,EAAWlC,aAAatU,KAAK2J,EAAK2K,aAAamC,GAGnD,OAAOD,GA3VT,GAAI5B,IACF8B,GAAI,IAAK,KACTC,GAAI,IAAK,KACThM,GAAI,KAAM,KAAM,KAAM,KAAM,IAAK,KACjCiM,GAAI,KAAM,KAAM,MAAO,MAAO,KAAM,IAAK,MASvCxN,GAEF4M,SAAU,EA+UZhd,GAAS+F,IAAI6K,KAAO5Q,EAASsT,MAAMvS,QACjC8S,YAAakI,EACb5O,SAAUA,EACVgK,OAAQA,EACRnG,KAAMA,EACNC,KAAMA,EACNM,MAAOA,EACP2K,IAAKA,EACLe,MAAOA,EACPC,UAAWA,EACXC,UAAWA,EACXjY,MAAOA,EACPN,UAAWA,EACX0Y,MAAOA,EACPC,eAAgBA,IAGlBvd,EAAS+F,IAAI6K,KAAKgL,oBAAsBA,EACxC5b,EAAS+F,IAAI6K,KAAK7C,KAAOA,GACzB7N,OAAQC,SAAUH,GAEnB,SAAUE,EAAQC,EAAUH,GAC3B,YAqBA,SAAS6d,GAAKnQ,EAAOV,EAAW8Q,EAAO9U,GACrCjJ,KAAK2N,MAAQA,EACb3N,KAAK6N,aAAeF,IAAUqQ,EAAUnW,EAAImW,EAAUpW,EAAIoW,EAAUnW,EACpE7H,KAAKiN,UAAYA,EACjBjN,KAAK6I,WAAaoE,EAAUU,EAAMsQ,SAAWhR,EAAUU,EAAMuQ,WAC7Dle,KAAKme,WAAalR,EAAUU,EAAMyQ,YAClCpe,KAAK+d,MAAQA,EACb/d,KAAKiJ,QAAUA,EAGjB,QAASoV,GAAoBhQ,EAAWiQ,EAAY5P,EAAkB6P,EAAc9Q,GAClF,GAAI+Q,GAAcD,EAAa,OAASve,KAAK2N,MAAMC,IAAI/D,eACnD4U,EAAkBze,KAAK+d,MAAMra,IAAI1D,KAAK0e,aAAa3J,KAAK/U,OACxD2e,EAAc3e,KAAK+d,MAAMra,IAAI8a,EAAYI,sBAE7CH,GAAgB7a,QAAQ,SAASib,EAAgBhb,GAC/C,GAOIib,GAPArQ,GACF5G,EAAG,EACHD,EAAG,EAQHkX,GAFCL,EAAgB5a,EAAQ,GAEX4a,EAAgB5a,EAAQ,GAAKgb,EAK7Brb,KAAKC,IAAIzD,KAAK6I,WAAagW,EAAgB,IAIxD5e,EAASmK,gBAAgBuU,EAAY9a,KAAkC,KAAvB8a,EAAY9a,KAMzC,MAAnB7D,KAAK2N,MAAMC,KACZiR,EAAiB7e,KAAKiN,UAAUpC,GAAKgU,EACrCpQ,EAAY5G,EAAI0W,EAAapV,MAAMsF,YAAY5G,EAIZ,UAAhC0W,EAAapV,MAAMiE,SACpBqB,EAAY7G,EAAI5H,KAAKiN,UAAUlF,QAAQE,IAAMsW,EAAapV,MAAMsF,YAAY7G,GAAK8G,EAAmB,EAAI,IAExGD,EAAY7G,EAAI5H,KAAKiN,UAAUC,GAAKqR,EAAapV,MAAMsF,YAAY7G,GAAK8G,EAAmB,EAAI,MAGjGmQ,EAAiB7e,KAAKiN,UAAUC,GAAK2R,EACrCpQ,EAAY7G,EAAI2W,EAAa1R,MAAM4B,YAAY7G,GAAK8G,EAAmBoQ,EAAc,GAIlD,UAAhCP,EAAa1R,MAAMO,SACpBqB,EAAY5G,EAAI6G,EAAmB1O,KAAKiN,UAAUlF,QAAQK,KAAOmW,EAAa1R,MAAM4B,YAAY5G,EAAI7H,KAAKiN,UAAUpC,GAAK,GAExH4D,EAAY5G,EAAI7H,KAAKiN,UAAUnC,GAAKyT,EAAa1R,MAAM4B,YAAY5G,EAAI,IAIxE2W,EAAYO,UACb9e,EAASoN,WAAWwR,EAAgBhb,EAAO7D,KAAMA,KAAKme,WAAYne,KAAKiN,UAAUjN,KAAK6N,aAAae,OAAQP,GACzGkQ,EAAaS,WAAWC,KACxBV,EAAaS,WAAWhf,KAAK2N,MAAMuR,MAClCzR,GAGF+Q,EAAYW,WACblf,EAASsO,YAAYsQ,EAAgBC,EAAajb,EAAO8a,EAAa3e,KAAMwe,EAAYpV,OAAQqF,EAAa6P,GAC3GC,EAAaS,WAAWI,MACxBb,EAAaS,WAAWhf,KAAK2N,MAAMuR,KACT,UAAzBV,EAAYpR,SAAuBmR,EAAaS,WAAWR,EAAYpR,UAAYmR,EAAaS,WAAgB,KAChHtQ,EAAkBjB,KAEvBsH,KAAK/U,OAlGT,GAAIge,IACFnW,GACE+F,IAAK,IACLgB,IAAK,QACLsQ,IAAK,aACLhB,UAAW,KACXD,QAAS,KACTG,WAAY,MAEdxW,GACEgG,IAAK,IACLgB,IAAK,SACLsQ,IAAK,WACLhB,UAAW,KACXD,QAAS,KACTG,WAAY,MAsFhBne,GAAS6d,KAAO7d,EAASsT,MAAMvS,QAC7B8S,YAAagK,EACbO,oBAAqBA,EACrBK,aAAc,SAAS1c,EAAO6B,EAAOc,GACnC,KAAM,IAAIoH,OAAM,uCAIpB9L,EAAS6d,KAAKnQ,MAAQqQ,GAEtB7d,OAAQC,SAAUH,GAuBnB,SAAUE,EAAQC,EAAUH,GAC3B,YAEA,SAASof,GAAcC,EAAU3a,EAAMsI,EAAWhE,GAEhD,GAAIQ,GAAUR,EAAQQ,SAAWxJ,EAASoJ,WAAW1E,EAAMsE,EAASqW,EAAS1R,IAC7E5N,MAAK8I,OAAS7I,EAAS8K,UAAUkC,EAAUqS,EAASrB,SAAWhR,EAAUqS,EAASpB,WAAYzU,EAASR,EAAQ+B,eAAiB,GAAI/B,EAAQgC,aAC5IjL,KAAK+I,OACHkB,IAAKjK,KAAK8I,OAAOmB,IACjBxG,IAAKzD,KAAK8I,OAAOrF,KAGnBxD,EAASof,cAATpf,SAA6B6T,YAAYnO,KAAK3F,KAC5Csf,EACArS,EACAjN,KAAK8I,OAAOkD,OACZ/C,GAGJ,QAASyV,GAAa1c,GACpB,MAAOhC,MAAK6I,aAAe5I,EAASqK,cAActI,EAAOhC,KAAK2N,MAAMC,KAAO5N,KAAK8I,OAAOmB,KAAOjK,KAAK8I,OAAOC,MAG5G9I,EAASof,cAAgBpf,EAAS6d,KAAK9c,QACrC8S,YAAauL,EACbX,aAAcA,KAGhBve,OAAQC,SAAUH,GAqBnB,SAAUE,EAAQC,EAAUH,GAC3B,YAEA,SAASsf,GAAeD,EAAU3a,EAAMsI,EAAWhE,GACjD,GAAIQ,GAAUR,EAAQQ,SAAWxJ,EAASoJ,WAAW1E,EAAMsE,EAASqW,EAAS1R,IAC7E5N,MAAK4K,QAAU3B,EAAQ2B,SAAW,EAClC5K,KAAK+d,MAAQ9U,EAAQ8U,OAAS9d,EAASyC,MAAM1C,KAAK4K,SAASlH,IAAI,SAAS1B,EAAO6B,GAC7E,MAAO4F,GAAQG,KAAOH,EAAQC,KAAOD,EAAQG,KAAO5J,KAAK4K,QAAU/G,GACnEkR,KAAK/U,OACPA,KAAK+d,MAAMyB,KAAK,SAAS3B,EAAG4B,GAC1B,MAAO5B,GAAI4B,IAEbzf,KAAK+I,OACHkB,IAAKR,EAAQG,IACbnG,IAAKgG,EAAQC,MAGfzJ,EAASsf,eAATtf,SAA8B6T,YAAYnO,KAAK3F,KAC7Csf,EACArS,EACAjN,KAAK+d,MACL9U,GAEFjJ,KAAK0f,WAAa1f,KAAK6I,WAAa7I,KAAK4K,QAG3C,QAAS8T,GAAa1c,GACpB,MAAOhC,MAAK6I,aAAe5I,EAASqK,cAActI,EAAOhC,KAAK2N,MAAMC,KAAO5N,KAAK+I,MAAMkB,MAAQjK,KAAK+I,MAAMtF,IAAMzD,KAAK+I,MAAMkB,KAG5HhK,EAASsf,eAAiBtf,EAAS6d,KAAK9c,QACtC8S,YAAayL,EACbb,aAAcA,KAGhBve,OAAQC,SAAUH,GAiBnB,SAAUE,EAAQC,EAAUH,GAC3B,YAEA,SAAS0f,GAASL,EAAU3a,EAAMsI,EAAWhE,GAC3ChJ,EAAS0f,SAAT1f,SAAwB6T,YAAYnO,KAAK3F,KACvCsf,EACArS,EACAhE,EAAQ8U,MACR9U,EAEF,IAAI2W,GAAOpc,KAAKC,IAAI,EAAGwF,EAAQ8U,MAAMzc,QAAU2H,EAAQ4W,QAAU,EAAI,GACrE7f,MAAK0f,WAAa1f,KAAK6I,WAAa+W,EAGtC,QAASlB,GAAa1c,EAAO6B,GAC3B,MAAO7D,MAAK0f,WAAa7b,EAG3B5D,EAAS0f,SAAW1f,EAAS6d,KAAK9c,QAChC8S,YAAa6L,EACbjB,aAAcA,KAGhBve,OAAQC,SAAUH,GASnB,SAASE,EAAQC,EAAUH,GAC1B,YA0GA,SAASqU,GAAYrL,GACnB,GAAItE,GAAO1E,EAASqG,cAActG,KAAK2E,KAAMsE,EAAQ/B,aAAa,EAGlElH,MAAKM,IAAML,EAASmF,UAAUpF,KAAKqF,UAAW4D,EAAQ3D,MAAO2D,EAAQ1D,OAAQ0D,EAAQ+V,WAAWc,MAEhG,IAKI3W,GAAO0D,EALPwB,EAAYrO,KAAKM,IAAIyN,KAAK,KAAK7H,SAAS+C,EAAQ+V,WAAW3Q,WAC3D0R,EAAc/f,KAAKM,IAAIyN,KAAK,KAC5BuQ,EAAate,KAAKM,IAAIyN,KAAK,KAAK7H,SAAS+C,EAAQ+V,WAAWV,YAE5DrR,EAAYhN,EAASyM,gBAAgB1M,KAAKM,IAAK2I,EAASoH,EAAetI,QAIzEoB,GADwB7G,SAAvB2G,EAAQE,MAAM+E,KACP,GAAIjO,GAAS0f,SAAS1f,EAAS6d,KAAKnQ,MAAM9F,EAAGlD,EAAKiC,WAAWC,OAAQoG,EAAWhN,EAASe,UAAWiI,EAAQE,OAClH4U,MAAOpZ,EAAKiC,WAAWI,OACvB6Y,QAAS5W,EAAQ+W,aAGX/W,EAAQE,MAAM+E,KAAKvI,KAAK1F,EAAUA,EAAS6d,KAAKnQ,MAAM9F,EAAGlD,EAAKiC,WAAWC,OAAQoG,EAAWhE,EAAQE,OAI5G0D,EADwBvK,SAAvB2G,EAAQ4D,MAAMqB,KACP,GAAIjO,GAASof,cAAcpf,EAAS6d,KAAKnQ,MAAM/F,EAAGjD,EAAKiC,WAAWC,OAAQoG,EAAWhN,EAASe,UAAWiI,EAAQ4D,OACvHnD,KAAMzJ,EAASiK,UAAUjB,EAAQS,MAAQT,EAAQS,KAAOT,EAAQ4D,MAAMnD,KACtEE,IAAK3J,EAASiK,UAAUjB,EAAQW,KAAOX,EAAQW,IAAMX,EAAQ4D,MAAMjD,OAG7DX,EAAQ4D,MAAMqB,KAAKvI,KAAK1F,EAAUA,EAAS6d,KAAKnQ,MAAM/F,EAAGjD,EAAKiC,WAAWC,OAAQoG,EAAWhE,EAAQ4D,OAG9G1D,EAAMkV,oBAAoBhQ,EAAWiQ,EAAYte,KAAKmV,sBAAuBlM,EAASjJ,KAAKyN,cAC3FZ,EAAMwR,oBAAoBhQ,EAAWiQ,EAAYte,KAAKmV,sBAAuBlM,EAASjJ,KAAKyN,cAEvFxE,EAAQgX,oBACVhgB,EAASmO,qBAAqBC,EAAWpB,EAAWhE,EAAQ+V,WAAW1Q,eAAgBtO,KAAKyN,cAI9F9I,EAAKgC,IAAIE,OAAOjD,QAAQ,SAASiD,EAAQqZ,GACvC,GAAIC,GAAgBJ,EAAYhS,KAAK,IAGrCoS,GAAcla,MACZma,iBAAkBvZ,EAAOoI,KACzBoR,UAAWpgB,EAASyE,UAAUmC,EAAOyB,QAIvC6X,EAAcja,UACZ+C,EAAQ+V,WAAWnY,OAClBA,EAAOrB,WAAayD,EAAQ+V,WAAWnY,OAAS,IAAM5G,EAASY,cAAcqf,IAC9ElS,KAAK,KAEP,IAAImC,MACFmQ,IAEF3b,GAAKiC,WAAWC,OAAOqZ,GAAatc,QAAQ,SAAS5B,EAAOue,GAC1D,GAAI9V,IACF5C,EAAGoF,EAAUpC,GAAK1B,EAAMuV,aAAa1c,EAAOue,EAAY5b,EAAKiC,WAAWC,OAAOqZ,IAC/EtY,EAAGqF,EAAUC,GAAKL,EAAM6R,aAAa1c,EAAOue,EAAY5b,EAAKiC,WAAWC,OAAOqZ,IAEjF/P,GAAgBlJ,KAAKwD,EAAE5C,EAAG4C,EAAE7C,GAC5B0Y,EAASrZ,MACPjF,MAAOA,EACPue,WAAYA,EACZjY,KAAMrI,EAASoI,YAAYxB,EAAQ0Z,MAErCxL,KAAK/U,MAEP,IAAIkP,IACFsR,WAAYvgB,EAAS+O,gBAAgBnI,EAAQoC,EAAS,cACtDwX,UAAWxgB,EAAS+O,gBAAgBnI,EAAQoC,EAAS,aACrDyX,SAAUzgB,EAAS+O,gBAAgBnI,EAAQoC,EAAS,YACpD0X,SAAU1gB,EAAS+O,gBAAgBnI,EAAQoC,EAAS,YACpD2X,SAAU3gB,EAAS+O,gBAAgBnI,EAAQoC,EAAS,aAGlD4X,EAAgD,kBAA7B3R,GAAcsR,WACnCtR,EAAcsR,WAActR,EAAcsR,WAAavgB,EAASyQ,cAAcuB,gBAAkBhS,EAASyQ,cAAcC,OAGrHC,EAAOiQ,EAAU1Q,EAAiBmQ,EAmCtC,IA9BIpR,EAAcuR,WAEhB7P,EAAK2K,aAAa3X,QAAQ,SAAS6X,GACjC,GAAIqF,GAAQX,EAAcpS,KAAK,QAC7BlD,GAAI4Q,EAAY5T,EAChBqF,GAAIuO,EAAY7T,EAChBkD,GAAI2Q,EAAY5T,EAAI,IACpBsF,GAAIsO,EAAY7T,GACfqB,EAAQ+V,WAAW8B,OAAO7a,MAC3B8a,YAAatF,EAAY9W,KAAK3C,MAAM6F,EAAG4T,EAAY9W,KAAK3C,MAAM4F,GAAG/B,OAAO5F,EAASiK,WAAW8D,KAAK,KACjGqS,UAAWpgB,EAASyE,UAAU+W,EAAY9W,KAAK2D,OAGjDtI,MAAKyN,aAAaQ,KAAK,QACrBC,KAAM,QACNlM,MAAOyZ,EAAY9W,KAAK3C,MACxB6B,MAAO4X,EAAY9W,KAAK4b,WACxBjY,KAAMmT,EAAY9W,KAAK2D,KACvBzB,OAAQA,EACRqZ,YAAaA,EACb/W,MAAOA,EACP0D,MAAOA,EACPU,MAAO4S,EACPhS,QAAS2S,EACTjZ,EAAG4T,EAAY5T,EACfD,EAAG6T,EAAY7T,KAEjBmN,KAAK/U,OAGNkP,EAAcwR,SAAU,CACzB,GAAIxP,GAAOiP,EAAcpS,KAAK,QAC5BqD,EAAGR,EAAK/L,aACPoE,EAAQ+V,WAAW9N,MAAM,EAE5BlR,MAAKyN,aAAaQ,KAAK,QACrBC,KAAM,OACNlC,OAAQrH,EAAKiC,WAAWC,OAAOqZ,GAC/BtP,KAAMA,EAAK2M,QACXtQ,UAAWA,EACXpJ,MAAOqc,EACPrZ,OAAQA,EACRqZ,YAAaA,EACbc,WAAYna,EAAOyB,KACnBa,MAAOA,EACP0D,MAAOA,EACPU,MAAO4S,EACPhS,QAAS+C,IAKb,GAAGhC,EAAcyR,UAAY9T,EAAM9D,MAAO,CAGxC,GAAI6X,GAAWpd,KAAKC,IAAID,KAAKyG,IAAIiF,EAAc0R,SAAU/T,EAAM9D,MAAMtF,KAAMoJ,EAAM9D,MAAMkB,KAGnFgX,EAAoBhU,EAAUC,GAAKL,EAAM6R,aAAakC,EAG1DhQ,GAAK4M,eAAe,KAAK3X,OAAO,SAA2Bqb,GAEzD,MAAOA,GAAY3F,aAAaja,OAAS,IACxCoC,IAAI,SAAuByd,GAE5B,GAAIC,GAAeD,EAAkB5F,aAAa,GAC9C8F,EAAcF,EAAkB5F,aAAa4F,EAAkB5F,aAAaja,OAAS,EAMzF,OAAO6f,GAAkB5D,OAAM,GAC5BnQ,SAAS,GACTgK,OAAO,GACPnG,KAAKmQ,EAAavZ,EAAGoZ,GACrB/P,KAAKkQ,EAAavZ,EAAGuZ,EAAaxZ,GAClCwF,SAAS+T,EAAkB5F,aAAaja,OAAS,GACjD4P,KAAKmQ,EAAYxZ,EAAGoZ,KAEtBrd,QAAQ,SAAoB0d,GAG7B,GAAIC,GAAOpB,EAAcpS,KAAK,QAC5BqD,EAAGkQ,EAASzc,aACXoE,EAAQ+V,WAAWuC,MAAM,EAG5BvhB,MAAKyN,aAAaQ,KAAK,QACrBC,KAAM,OACNlC,OAAQrH,EAAKiC,WAAWC,OAAOqZ,GAC/BtP,KAAM0Q,EAAS/D,QACf1W,OAAQA,EACRqZ,YAAaA,EACb/W,MAAOA,EACP0D,MAAOA,EACPI,UAAWA,EACXpJ,MAAOqc,EACP3S,MAAO4S,EACPhS,QAASoT,KAEXxM,KAAK/U,SAET+U,KAAK/U,OAEPA,KAAKyN,aAAaQ,KAAK,WACrBnF,OAAQ+D,EAAM/D,OACdmE,UAAWA,EACX9D,MAAOA,EACP0D,MAAOA,EACPvM,IAAKN,KAAKM,IACV2I,QAASA,IAqFb,QAASuY,GAAKhf,EAAOmC,EAAMsE,EAASmG,GAClCnP,EAASuhB,KAATvhB,SAAoB6T,YAAYnO,KAAK3F,KACnCwC,EACAmC,EACA0L,EACApQ,EAASe,UAAWqP,EAAgBpH,GACpCmG,GArYJ,GAAIiB,IAEFlH,OAEEC,OAAQ,GAERgE,SAAU,MAEVqB,aACE5G,EAAG,EACHD,EAAG,GAGLuX,WAAW,EAEXJ,UAAU,EAEVH,sBAAuB3e,EAASU,KAEhCuN,KAAM5L,QAGRuK,OAEEzD,OAAQ,GAERgE,SAAU,QAEVqB,aACE5G,EAAG,EACHD,EAAG,GAGLuX,WAAW,EAEXJ,UAAU,EAEVH,sBAAuB3e,EAASU,KAEhCuN,KAAM5L,OAEN0I,cAAe,GAEfC,aAAa,GAGf3F,MAAOhD,OAEPiD,OAAQjD,OAERoe,UAAU,EAEVD,WAAW,EAEXE,UAAU,EAEVC,SAAU,EAEVJ,YAAY,EAEZP,oBAAoB,EAEpBrW,IAAKtH,OAELoH,KAAMpH,OAEN4G,cACEjB,IAAK,GACLC,MAAO,GACPC,OAAQ,EACRC,KAAM,IAGR4X,WAAW,EAEX9Y,aAAa,EAEb8X,YACEc,MAAO,gBACPV,MAAO,WACPd,WAAY,YACZzX,OAAQ,YACRqK,KAAM,UACN4P,MAAO,WACPS,KAAM,UACNtC,KAAM,UACN5Q,UAAW,WACXC,eAAgB,qBAChBmT,SAAU,cACVC,WAAY,gBACZC,MAAO,WACPC,IAAK,UA8ST3hB,GAASuhB,KAAOvhB,EAASiV,KAAKlU,QAC5B8S,YAAa0N,EACblN,YAAaA,KAGfnU,OAAQC,SAAUH,GAOnB,SAASE,EAAQC,EAAUH,GAC1B,YA0GA,SAASqU,GAAYrL,GACnB,GAAItE,GACA8E,CAEDR,GAAQ4Y,kBACTld,EAAO1E,EAASqG,cAActG,KAAK2E,KAAMsE,EAAQ/B,YAAa+B,EAAQ6Y,eAAiB,IAAM,KAC7Fnd,EAAKiC,WAAWC,OAASlC,EAAKiC,WAAWC,OAAOnD,IAAI,SAAS1B,GAC3D,OAAQA,MAGV2C,EAAO1E,EAASqG,cAActG,KAAK2E,KAAMsE,EAAQ/B,YAAa+B,EAAQ6Y,eAAiB,IAAM,KAI/F9hB,KAAKM,IAAML,EAASmF,UAClBpF,KAAKqF,UACL4D,EAAQ3D,MACR2D,EAAQ1D,OACR0D,EAAQ+V,WAAWc,OAAS7W,EAAQ6Y,eAAiB,IAAM7Y,EAAQ+V,WAAW8C,eAAiB,IAIjG,IAAIzT,GAAYrO,KAAKM,IAAIyN,KAAK,KAAK7H,SAAS+C,EAAQ+V,WAAW3Q,WAC3D0R,EAAc/f,KAAKM,IAAIyN,KAAK,KAC5BuQ,EAAate,KAAKM,IAAIyN,KAAK,KAAK7H,SAAS+C,EAAQ+V,WAAWV,WAEhE,IAAGrV,EAAQ8Y,WAA+C,IAAlCpd,EAAKiC,WAAWC,OAAOvF,OAAc,CAG3D,GAAI0gB,GAAa/hB,EAASmD,UAAUuB,EAAKiC,WAAWC,OAAQ,WAC1D,MAAOrF,OAAMiE,UAAUC,MAAMC,KAAKtE,WAAWqC,IAAI,SAAS1B;AACxD,MAAOA,KACNgD,OAAO,SAASid,EAAMC,GACvB,OACEra,EAAGoa,EAAKpa,GAAKqa,GAAQA,EAAKra,IAAM,EAChCD,EAAGqa,EAAKra,GAAKsa,GAAQA,EAAKta,IAAM,KAEhCC,EAAG,EAAGD,EAAG,KAGf6B,GAAUxJ,EAASoJ,YAAY2Y,GAAa/Y,EAASA,EAAQ6Y,eAAiB,IAAM,SAIpFrY,GAAUxJ,EAASoJ,WAAW1E,EAAKiC,WAAWC,OAAQoC,EAASA,EAAQ6Y,eAAiB,IAAM,IAIhGrY,GAAQC,MAAQT,EAAQS,OAA0B,IAAjBT,EAAQS,KAAa,EAAID,EAAQC,MAClED,EAAQG,KAAOX,EAAQW,MAAwB,IAAhBX,EAAQW,IAAY,EAAIH,EAAQG,IAE/D,IAEIuY,GACFC,EACAC,EACAlZ,EACA0D,EANEI,EAAYhN,EAASyM,gBAAgB1M,KAAKM,IAAK2I,EAASoH,EAAetI,QAYzEqa,GAHCnZ,EAAQ4Y,kBAAoB5Y,EAAQ8Y,UAGpBpd,EAAKiC,WAAWI,OAAOtB,MAAM,EAAG,GAKhCf,EAAKiC,WAAWI,OAIhCiC,EAAQ6Y,gBAEPK,EAAYhZ,EADY7G,SAAvB2G,EAAQE,MAAM+E,KACK,GAAIjO,GAASof,cAAcpf,EAAS6d,KAAKnQ,MAAM9F,EAAGlD,EAAKiC,WAAWC,OAAQoG,EAAWhN,EAASe,UAAWiI,EAAQE,OACnIM,QAASA,EACTO,eAAgB,KAGEf,EAAQE,MAAM+E,KAAKvI,KAAK1F,EAAUA,EAAS6d,KAAKnQ,MAAM9F,EAAGlD,EAAKiC,WAAWC,OAAQoG,EAAWhN,EAASe,UAAWiI,EAAQE,OAC1IM,QAASA,EACTO,eAAgB,KAKlBqY,EAAYxV,EADYvK,SAAvB2G,EAAQ4D,MAAMqB,KACK,GAAIjO,GAAS0f,SAAS1f,EAAS6d,KAAKnQ,MAAM/F,EAAGjD,EAAKiC,WAAWC,OAAQoG,GACvF8Q,MAAOqE,IAGWnZ,EAAQ4D,MAAMqB,KAAKvI,KAAK1F,EAAUA,EAAS6d,KAAKnQ,MAAM/F,EAAGjD,EAAKiC,WAAWC,OAAQoG,EAAWhE,EAAQ4D,SAIxHwV,EAAYlZ,EADY7G,SAAvB2G,EAAQE,MAAM+E,KACK,GAAIjO,GAAS0f,SAAS1f,EAAS6d,KAAKnQ,MAAM9F,EAAGlD,EAAKiC,WAAWC,OAAQoG,GACvF8Q,MAAOqE,IAGWnZ,EAAQE,MAAM+E,KAAKvI,KAAK1F,EAAUA,EAAS6d,KAAKnQ,MAAM9F,EAAGlD,EAAKiC,WAAWC,OAAQoG,EAAWhE,EAAQE,OAIxHgZ,EAAYtV,EADYvK,SAAvB2G,EAAQ4D,MAAMqB,KACK,GAAIjO,GAASof,cAAcpf,EAAS6d,KAAKnQ,MAAM/F,EAAGjD,EAAKiC,WAAWC,OAAQoG,EAAWhN,EAASe,UAAWiI,EAAQ4D,OACnIpD,QAASA,EACTO,eAAgB,KAGEf,EAAQ4D,MAAMqB,KAAKvI,KAAK1F,EAAUA,EAAS6d,KAAKnQ,MAAM/F,EAAGjD,EAAKiC,WAAWC,OAAQoG,EAAWhN,EAASe,UAAWiI,EAAQ4D,OAC1IpD,QAASA,EACTO,eAAgB,KAMtB,IAAIsY,GAAYrZ,EAAQ6Y,eAAkB7U,EAAUpC,GAAKsX,EAAUzD,aAAa,GAAOzR,EAAUC,GAAKiV,EAAUzD,aAAa,GAEzH6D,IAEJF,GAAUhE,oBAAoBhQ,EAAWiQ,EAAYte,KAAKmV,sBAAuBlM,EAASjJ,KAAKyN,cAC/F0U,EAAU9D,oBAAoBhQ,EAAWiQ,EAAYte,KAAKmV,sBAAuBlM,EAASjJ,KAAKyN,cAE3FxE,EAAQgX,oBACVhgB,EAASmO,qBAAqBC,EAAWpB,EAAWhE,EAAQ+V,WAAW1Q,eAAgBtO,KAAKyN,cAI9F9I,EAAKgC,IAAIE,OAAOjD,QAAQ,SAASiD,EAAQqZ,GAEvC,GAEIsC,GAEArC,EAJAsC,EAAQvC,GAAevb,EAAKgC,IAAIE,OAAOvF,OAAS,GAAK,CAUvDkhB,GAHCvZ,EAAQ4Y,mBAAqB5Y,EAAQ8Y,UAGnBM,EAAUxZ,WAAalE,EAAKiC,WAAWC,OAAOvF,OAAS,EAClE2H,EAAQ4Y,kBAAoB5Y,EAAQ8Y,UAGzBM,EAAUxZ,WAAa,EAGvBwZ,EAAUxZ,WAAalE,EAAKiC,WAAWC,OAAOqZ,GAAa5e,OAAS,EAIzF6e,EAAgBJ,EAAYhS,KAAK,KAGjCoS,EAAcla,MACZma,iBAAkBvZ,EAAOoI,KACzBoR,UAAWpgB,EAASyE,UAAUmC,EAAOyB,QAIvC6X,EAAcja,UACZ+C,EAAQ+V,WAAWnY,OAClBA,EAAOrB,WAAayD,EAAQ+V,WAAWnY,OAAS,IAAM5G,EAASY,cAAcqf,IAC9ElS,KAAK,MAEPrJ,EAAKiC,WAAWC,OAAOqZ,GAAatc,QAAQ,SAAS5B,EAAOue,GAC1D,GAAImC,GACFC,EACAC,EACAC,CA+CF,IAzCEA,EAHC5Z,EAAQ4Y,mBAAqB5Y,EAAQ8Y,UAGhB7B,EACdjX,EAAQ4Y,kBAAoB5Y,EAAQ8Y,UAGtB,EAGAxB,EAKtBmC,EADCzZ,EAAQ6Y,gBAEPja,EAAGoF,EAAUpC,GAAKsX,EAAUzD,aAAa1c,GAASA,EAAM6F,EAAI7F,EAAM6F,EAAI,EAAG0Y,EAAY5b,EAAKiC,WAAWC,OAAOqZ,IAC5GtY,EAAGqF,EAAUC,GAAKmV,EAAU3D,aAAa1c,GAASA,EAAM4F,EAAI5F,EAAM4F,EAAI,EAAGib,EAAqBle,EAAKiC,WAAWC,OAAOqZ,MAIrHrY,EAAGoF,EAAUpC,GAAKwX,EAAU3D,aAAa1c,GAASA,EAAM6F,EAAI7F,EAAM6F,EAAI,EAAGgb,EAAqBle,EAAKiC,WAAWC,OAAOqZ,IACrHtY,EAAGqF,EAAUC,GAAKiV,EAAUzD,aAAa1c,GAASA,EAAM4F,EAAI5F,EAAM4F,EAAI,EAAG2Y,EAAY5b,EAAKiC,WAAWC,OAAOqZ,KAQ7GmC,YAAqBpiB,GAAS0f,WAE3B0C,EAAUpZ,QAAQ4W,UACpB6C,EAAUL,EAAU1U,MAAMC,MAAQ4U,GAAoBvZ,EAAQ6Y,kBAAsB,IAGtFY,EAAUL,EAAU1U,MAAMC,MAAS3E,EAAQ8Y,WAAa9Y,EAAQ4Y,iBAAoB,EAAIY,EAAQxZ,EAAQ6Z,mBAAqB7Z,EAAQ6Y,kBAAsB,IAI7Jc,EAAgBL,EAAiBhC,IAAe+B,EAChDC,EAAiBhC,GAAcqC,GAAiBN,EAAYI,EAAUL,EAAUxU,aAAaD,MAGhFtL,SAAVN,EAAH,CAIA,GAAI+gB,KACJA,GAAUV,EAAU1U,MAAMC,IAAM,KAAO8U,EAAUL,EAAU1U,MAAMC,KACjEmV,EAAUV,EAAU1U,MAAMC,IAAM,KAAO8U,EAAUL,EAAU1U,MAAMC,MAE9D3E,EAAQ8Y,WAAoC,eAAtB9Y,EAAQ+Z,WAA+B/Z,EAAQ+Z,WAUtED,EAAUV,EAAUxU,aAAaD,IAAM,KAAO0U,EAC9CS,EAAUV,EAAUxU,aAAaD,IAAM,KAAO8U,EAAUL,EAAUxU,aAAaD,OAN/EmV,EAAUV,EAAUxU,aAAaD,IAAM,KAAOgV,EAC9CG,EAAUV,EAAUxU,aAAaD,IAAM,KAAO2U,EAAiBhC,IASjEwC,EAAUlY,GAAKrH,KAAKyG,IAAIzG,KAAKC,IAAIsf,EAAUlY,GAAIoC,EAAUpC,IAAKoC,EAAUnC,IACxEiY,EAAUjY,GAAKtH,KAAKyG,IAAIzG,KAAKC,IAAIsf,EAAUjY,GAAImC,EAAUpC,IAAKoC,EAAUnC,IACxEiY,EAAU7V,GAAK1J,KAAKyG,IAAIzG,KAAKC,IAAIsf,EAAU7V,GAAID,EAAUE,IAAKF,EAAUC,IACxE6V,EAAU5V,GAAK3J,KAAKyG,IAAIzG,KAAKC,IAAIsf,EAAU5V,GAAIF,EAAUE,IAAKF,EAAUC,GAExE,IAAI+V,GAAWhjB,EAASoI,YAAYxB,EAAQ0Z,EAG5CoC,GAAMxC,EAAcpS,KAAK,OAAQgV,EAAW9Z,EAAQ+V,WAAW2D,KAAK1c,MAClE8a,YAAa/e,EAAM6F,EAAG7F,EAAM4F,GAAG/B,OAAO5F,EAASiK,WAAW8D,KAAK,KAC/DqS,UAAWpgB,EAASyE,UAAUue,KAGhCjjB,KAAKyN,aAAaQ,KAAK,OAAQhO,EAASe,QACtCkN,KAAM,MACNlM,MAAOA,EACP6B,MAAO0c,EACPjY,KAAM2a,EACNpc,OAAQA,EACRqZ,YAAaA,EACb/W,MAAOA,EACP0D,MAAOA,EACPI,UAAWA,EACXM,MAAO4S,EACPhS,QAASwU,GACRI,MACHhO,KAAK/U,QACP+U,KAAK/U,OAEPA,KAAKyN,aAAaQ,KAAK,WACrBnF,OAAQqZ,EAAUrZ,OAClBmE,UAAWA,EACX9D,MAAOA,EACP0D,MAAOA,EACPvM,IAAKN,KAAKM,IACV2I,QAASA,IAyCb,QAASia,GAAI1gB,EAAOmC,EAAMsE,EAASmG,GACjCnP,EAASijB,IAATjjB,SAAmB6T,YAAYnO,KAAK3F,KAClCwC,EACAmC,EACA0L,EACApQ,EAASe,UAAWqP,EAAgBpH,GACpCmG,GAnaJ,GAAIiB,IAEFlH,OAEEC,OAAQ,GAERgE,SAAU,MAEVqB,aACE5G,EAAG,EACHD,EAAG,GAGLuX,WAAW,EAEXJ,UAAU,EAEVH,sBAAuB3e,EAASU,KAEhCqK,cAAe,GAEfC,aAAa,GAGf4B,OAEEzD,OAAQ,GAERgE,SAAU,QAEVqB,aACE5G,EAAG,EACHD,EAAG,GAGLuX,WAAW,EAEXJ,UAAU,EAEVH,sBAAuB3e,EAASU,KAEhCqK,cAAe,GAEfC,aAAa,GAGf3F,MAAOhD,OAEPiD,OAAQjD,OAERoH,KAAMpH,OAENsH,IAAKtH,OAEL0H,eAAgB,EAEhBd,cACEjB,IAAK,GACLC,MAAO,GACPC,OAAQ,EACRC,KAAM,IAGR0a,kBAAmB,GAEnBf,WAAW,EAGXiB,UAAW,aAEXlB,gBAAgB,EAEhBD,kBAAkB,EAElB3a,aAAa,EAEb+Y,oBAAoB,EAEpBjB,YACEc,MAAO,eACPgC,eAAgB,qBAChB1C,MAAO,WACPd,WAAY,YACZzX,OAAQ,YACR8b,IAAK,SACL1D,KAAM,UACN5Q,UAAW,WACXC,eAAgB,qBAChBmT,SAAU,cACVC,WAAY,gBACZC,MAAO,WACPC,IAAK,UA4UT3hB,GAASijB,IAAMjjB,EAASiV,KAAKlU,QAC3B8S,YAAaoP,EACb5O,YAAaA,KAGfnU,OAAQC,SAAUH,GAOnB,SAASE,EAAQC,EAAUH,GAC1B,YAwDA,SAASkjB,GAAwBC,EAAQhE,EAAOiE,GAC9C,GAAIC,GAAalE,EAAMvX,EAAIub,EAAOvb,CAElC,OAAGyb,IAA4B,YAAdD,IACdC,GAA4B,YAAdD,EACR,QACCC,GAA4B,YAAdD,IACrBC,GAA4B,YAAdD,EACR,MAEA,SASX,QAAS/O,GAAYrL,GACnB,GAEEsa,GACAtW,EACAb,EACAoX,EACAC,EANE9e,EAAO1E,EAASqG,cAActG,KAAK2E,MACnC+e,KAMFC,EAAa1a,EAAQ0a,UAGvB3jB,MAAKM,IAAML,EAASmF,UAAUpF,KAAKqF,UAAW4D,EAAQ3D,MAAO2D,EAAQ1D,OAAO0D,EAAQ2a,MAAQ3a,EAAQ+V,WAAW6E,WAAa5a,EAAQ+V,WAAW8E,UAE/I7W,EAAYhN,EAASyM,gBAAgB1M,KAAKM,IAAK2I,EAASoH,EAAetI,SAEvEqE,EAAS5I,KAAKyG,IAAIgD,EAAU3H,QAAU,EAAG2H,EAAU1H,SAAW,GAE9Dke,EAAexa,EAAQ8a,OAASpf,EAAKiC,WAAWC,OAAO7B,OAAO,SAASgf,EAAeC,GACpF,MAAOD,GAAgBC,GACtB,EAEH,IAAIC,GAAajkB,EAASiC,SAAS+G,EAAQib,WACnB,OAApBA,EAAWjiB,OACbiiB,EAAWliB,OAASoK,EAAS,KAM/BA,GAAUnD,EAAQ2a,MAAQM,EAAWliB,MAAQ,EAAK,EAKhDwhB,EAD2B,YAA1Bva,EAAQkb,eAA+Blb,EAAQ2a,MAClCxX,EACoB,WAA1BnD,EAAQkb,cAEF,EAIA/X,EAAS,EAGzBoX,GAAeva,EAAQwF,WAGvB,IAAI2U,IACFvb,EAAGoF,EAAUpC,GAAKoC,EAAU3H,QAAU,EACtCsC,EAAGqF,EAAUE,GAAKF,EAAU1H,SAAW,GAIrC6e,EAEU,IAFazf,EAAKgC,IAAIE,OAAOhB,OAAO,SAASwe,GACzD,MAAOA,GAAI/c,eAAe,SAAyB,IAAd+c,EAAIriB,MAAsB,IAARqiB,IACtD/iB,MAGHqD,GAAKgC,IAAIE,OAAOjD,QAAQ,SAASiD,EAAQhD,GACvC6f,EAAa7f,GAAS7D,KAAKM,IAAIyN,KAAK,IAAK,KAAM,OAC/CgH,KAAK/U,OAEJiJ,EAAQkW,YACToE,EAAcvjB,KAAKM,IAAIyN,KAAK,IAAK,KAAM,OAKzCpJ,EAAKgC,IAAIE,OAAOjD,QAAQ,SAASiD,EAAQhD,GAEvC,GAAsC,IAAlCc,EAAKiC,WAAWC,OAAOhD,KAAgBoF,EAAQqb,kBAAnD,CAGAZ,EAAa7f,GAAOoC,MAClBma,iBAAkBvZ,EAAOoI,OAI3ByU,EAAa7f,GAAOqC,UAClB+C,EAAQ+V,WAAWnY,OAClBA,EAAOrB,WAAayD,EAAQ+V,WAAWnY,OAAS,IAAM5G,EAASY,cAAcgD,IAC9EmK,KAAK,KAGP,IAAIuW,GAAYd,EAAe,EAAIE,EAAahf,EAAKiC,WAAWC,OAAOhD,GAAS4f,EAAe,IAAM,EAGjGe,EAAuBhhB,KAAKC,IAAI,EAAGkgB,GAAwB,IAAV9f,GAAeugB,EAAuB,EAAI,IAI5FG,GAAWC,GAAwB,SACpCD,EAAWC,EAAuB,OAGpC,IAAI7C,GAAQ1hB,EAASgM,iBAAiBmX,EAAOvb,EAAGub,EAAOxb,EAAGwE,EAAQoY,GAChE5C,EAAM3hB,EAASgM,iBAAiBmX,EAAOvb,EAAGub,EAAOxb,EAAGwE,EAAQmY,GAG1D3T,EAAO,GAAI3Q,GAAS+F,IAAI6K,OAAM5H,EAAQ2a,QACvC3S,KAAK2Q,EAAI/Z,EAAG+Z,EAAIha,GAChBuU,IAAI/P,EAAQA,EAAQ,EAAGmY,EAAWZ,EAAa,IAAK,EAAGhC,EAAM9Z,EAAG8Z,EAAM/Z,EAGrEqB,GAAQ2a,OACVhT,EAAKM,KAAKkS,EAAOvb,EAAGub,EAAOxb,EAK7B,IAAI6T,GAAciI,EAAa7f,GAAOkK,KAAK,QACzCqD,EAAGR,EAAK/L,aACPoE,EAAQ2a,MAAQ3a,EAAQ+V,WAAWyF,WAAaxb,EAAQ+V,WAAW0F,SAiCtE,IA9BAjJ,EAAYxV,MACV8a,WAAYpc,EAAKiC,WAAWC,OAAOhD,GACnCwc,UAAWpgB,EAASyE,UAAUmC,EAAOyB,QAIpCW,EAAQ2a,OACTnI,EAAYxV,MACVE,MAAS,iBAAmB+d,EAAWliB,MAAQ,OAKnDhC,KAAKyN,aAAaQ,KAAK,QACrBC,KAAM,QACNlM,MAAO2C,EAAKiC,WAAWC,OAAOhD,GAC9B4f,aAAcA,EACd5f,MAAOA,EACPyE,KAAMzB,EAAOyB,KACbzB,OAAQA,EACR0G,MAAOmW,EAAa7f,GACpBsK,QAASsN,EACT7K,KAAMA,EAAK2M,QACX6F,OAAQA,EACRhX,OAAQA,EACRuX,WAAYA,EACZY,SAAUA,IAITtb,EAAQkW,UAAW,CACpB,GAAIgF,EAGFA,GAF4B,IAA3Bxf,EAAKgC,IAAIE,OAAOvF,QAGfuG,EAAGub,EAAOvb,EACVD,EAAGwb,EAAOxb,GAII3H,EAASgM,iBACvBmX,EAAOvb,EACPub,EAAOxb,EACP4b,EACAG,GAAcY,EAAWZ,GAAc,EAI3C,IAAIgB,EAEFA,GADChgB,EAAKiC,WAAWI,SAAW/G,EAASmK,gBAAgBzF,EAAKiC,WAAWI,OAAOnD,IACjEc,EAAKiC,WAAWI,OAAOnD,GAEvBc,EAAKiC,WAAWC,OAAOhD,EAGpC,IAAI+gB,GAAoB3b,EAAQ2V,sBAAsB+F,EAAU9gB,EAEhE,IAAG+gB,GAA2C,IAAtBA,EAAyB,CAC/C,GAAIjW,GAAe4U,EAAYxV,KAAK,QAClC8W,GAAIV,EAActc,EAClBid,GAAIX,EAAcvc,EAClBmd,cAAe5B,EAAwBC,EAAQe,EAAelb,EAAQ+b,iBACrE/b,EAAQ+V,WAAWI,OAAOrQ,KAAK,GAAK6V,EAGvC5kB,MAAKyN,aAAaQ,KAAK,QACrBC,KAAM,QACNrK,MAAOA,EACP0J,MAAOgW,EACPpV,QAASQ,EACTI,KAAM,GAAK6V,EACX/c,EAAGsc,EAActc,EACjBD,EAAGuc,EAAcvc,KAOvB+b,EAAaY,IACbxP,KAAK/U,OAEPA,KAAKyN,aAAaQ,KAAK,WACrBhB,UAAWA,EACX3M,IAAKN,KAAKM,IACV2I,QAASA,IAwEb,QAASgc,GAAIziB,EAAOmC,EAAMsE,EAASmG,GACjCnP,EAASglB,IAAThlB,SAAmB6T,YAAYnO,KAAK3F,KAClCwC,EACAmC,EACA0L,EACApQ,EAASe,UAAWqP,EAAgBpH,GACpCmG,GA1VJ,GAAIiB,IAEF/K,MAAOhD,OAEPiD,OAAQjD,OAER4G,aAAc,EAEd8V,YACE8E,SAAU,eACVD,WAAY,iBACZhd,OAAQ,YACR6d,SAAU,eACVD,WAAY,iBACZrF,MAAO,YAGTuE,WAAY,EAEZI,MAAOzhB,OAEPshB,OAAO,EAGPM,WAAY,GAEZ/E,WAAW,EAEX1Q,YAAa,EAEb0V,cAAe,SAEfvF,sBAAuB3e,EAASU,KAEhCqkB,eAAgB,UAEhB9d,aAAa,EAEbod,mBAAmB,EAwTrBrkB,GAASglB,IAAMhlB,EAASiV,KAAKlU,QAC3B8S,YAAamR,EACb3Q,YAAaA,EACb6O,wBAAyBA,KAG3BhjB,OAAQC,SAAUH,GAEbA","file":"chartist.min.js","sourcesContent":["(function (root, factory) {\n if (typeof define === 'function' && define.amd) {\n // AMD. Register as an anonymous module unless amdModuleId is set\n define('Chartist', [], function () {\n return (root['Chartist'] = factory());\n });\n } else if (typeof exports === 'object') {\n // Node. Does not work with strict CommonJS, but\n // only CommonJS-like environments that support module.exports,\n // like Node.\n module.exports = factory();\n } else {\n root['Chartist'] = factory();\n }\n}(this, function () {\n\n/\* Chartist.js 0.10.0\n \* Copyright © 2016 Gion Kunz\n \* Free to use under either the WTFPL license or the MIT license.\n \* https://raw.githubusercontent.com/gionkunz/chartist-js/master/LICENSE-WTFPL\n \* https://raw.githubusercontent.com/gionkunz/chartist-js/master/LICENSE-MIT\n \*/\n/\*\*\n \* The core module of Chartist that is mainly providing static functions and higher level functions for chart modules.\n \*\n \* @module Chartist.Core\n \*/\nvar Chartist = {\n version: '0.10.0'\n};\n\n(function (window, document, Chartist) {\n 'use strict';\n\n /\*\*\n \* This object contains all namespaces used within Chartist.\n \*\n \* @memberof Chartist.Core\n \* @type {{svg: string, xmlns: string, xhtml: string, xlink: string, ct: string}}\n \*/\n Chartist.namespaces = {\n svg: 'http://www.w3.org/2000/svg',\n xmlns: 'http://www.w3.org/2000/xmlns/',\n xhtml: 'http://www.w3.org/1999/xhtml',\n xlink: 'http://www.w3.org/1999/xlink',\n ct: 'http://gionkunz.github.com/chartist-js/ct'\n };\n\n /\*\*\n \* Helps to simplify functional style code\n \*\n \* @memberof Chartist.Core\n \* @param {\*} n This exact value will be returned by the noop function\n \* @return {\*} The same value that was provided to the n parameter\n \*/\n Chartist.noop = function (n) {\n return n;\n };\n\n /\*\*\n \* Generates a-z from a number 0 to 26\n \*\n \* @memberof Chartist.Core\n \* @param {Number} n A number from 0 to 26 that will result in a letter a-z\n \* @return {String} A character from a-z based on the input number n\n \*/\n Chartist.alphaNumerate = function (n) {\n // Limit to a-z\n return String.fromCharCode(97 + n % 26);\n };\n\n /\*\*\n \* Simple recursive object extend\n \*\n \* @memberof Chartist.Core\n \* @param {Object} target Target object where the source will be merged into\n \* @param {Object...} sources This object (objects) will be merged into target and then target is returned\n \* @return {Object} An object that has the same reference as target but is extended and merged with the properties of source\n \*/\n Chartist.extend = function (target) {\n var i, source, sourceProp;\n target = target || {};\n\n for (i = 1; i < arguments.length; i++) {\n source = arguments[i];\n for (var prop in source) {\n sourceProp = source[prop];\n if (typeof sourceProp === 'object' && sourceProp !== null && !(sourceProp instanceof Array)) {\n target[prop] = Chartist.extend(target[prop], sourceProp);\n } else {\n target[prop] = sourceProp;\n }\n }\n }\n\n return target;\n };\n\n /\*\*\n \* Replaces all occurrences of subStr in str with newSubStr and returns a new string.\n \*\n \* @memberof Chartist.Core\n \* @param {String} str\n \* @param {String} subStr\n \* @param {String} newSubStr\n \* @return {String}\n \*/\n Chartist.replaceAll = function(str, subStr, newSubStr) {\n return str.replace(new RegExp(subStr, 'g'), newSubStr);\n };\n\n /\*\*\n \* Converts a number to a string with a unit. If a string is passed then this will be returned unmodified.\n \*\n \* @memberof Chartist.Core\n \* @param {Number} value\n \* @param {String} unit\n \* @return {String} Returns the passed number value with unit.\n \*/\n Chartist.ensureUnit = function(value, unit) {\n if(typeof value === 'number') {\n value = value + unit;\n }\n\n return value;\n };\n\n /\*\*\n \* Converts a number or string to a quantity object.\n \*\n \* @memberof Chartist.Core\n \* @param {String|Number} input\n \* @return {Object} Returns an object containing the value as number and the unit as string.\n \*/\n Chartist.quantity = function(input) {\n if (typeof input === 'string') {\n var match = (/^(\\d+)\\s\*(.\*)$/g).exec(input);\n return {\n value : +match[1],\n unit: match[2] || undefined\n };\n }\n return { value: input };\n };\n\n /\*\*\n \* This is a wrapper around document.querySelector that will return the query if it's already of type Node\n \*\n \* @memberof Chartist.Core\n \* @param {String|Node} query The query to use for selecting a Node or a DOM node that will be returned directly\n \* @return {Node}\n \*/\n Chartist.querySelector = function(query) {\n return query instanceof Node ? query : document.querySelector(query);\n };\n\n /\*\*\n \* Functional style helper to produce array with given length initialized with undefined values\n \*\n \* @memberof Chartist.Core\n \* @param length\n \* @return {Array}\n \*/\n Chartist.times = function(length) {\n return Array.apply(null, new Array(length));\n };\n\n /\*\*\n \* Sum helper to be used in reduce functions\n \*\n \* @memberof Chartist.Core\n \* @param previous\n \* @param current\n \* @return {\*}\n \*/\n Chartist.sum = function(previous, current) {\n return previous + (current ? current : 0);\n };\n\n /\*\*\n \* Multiply helper to be used in `Array.map` for multiplying each value of an array with a factor.\n \*\n \* @memberof Chartist.Core\n \* @param {Number} factor\n \* @returns {Function} Function that can be used in `Array.map` to multiply each value in an array\n \*/\n Chartist.mapMultiply = function(factor) {\n return function(num) {\n return num \* factor;\n };\n };\n\n /\*\*\n \* Add helper to be used in `Array.map` for adding a addend to each value of an array.\n \*\n \* @memberof Chartist.Core\n \* @param {Number} addend\n \* @returns {Function} Function that can be used in `Array.map` to add a addend to each value in an array\n \*/\n Chartist.mapAdd = function(addend) {\n return function(num) {\n return num + addend;\n };\n };\n\n /\*\*\n \* Map for multi dimensional arrays where their nested arrays will be mapped in serial. The output array will have the length of the largest nested array. The callback function is called with variable arguments where each argument is the nested array value (or undefined if there are no more values).\n \*\n \* @memberof Chartist.Core\n \* @param arr\n \* @param cb\n \* @return {Array}\n \*/\n Chartist.serialMap = function(arr, cb) {\n var result = [],\n length = Math.max.apply(null, arr.map(function(e) {\n return e.length;\n }));\n\n Chartist.times(length).forEach(function(e, index) {\n var args = arr.map(function(e) {\n return e[index];\n });\n\n result[index] = cb.apply(null, args);\n });\n\n return result;\n };\n\n /\*\*\n \* This helper function can be used to round values with certain precision level after decimal. This is used to prevent rounding errors near float point precision limit.\n \*\n \* @memberof Chartist.Core\n \* @param {Number} value The value that should be rounded with precision\n \* @param {Number} [digits] The number of digits after decimal used to do the rounding\n \* @returns {number} Rounded value\n \*/\n Chartist.roundWithPrecision = function(value, digits) {\n var precision = Math.pow(10, digits || Chartist.precision);\n return Math.round(value \* precision) / precision;\n };\n\n /\*\*\n \* Precision level used internally in Chartist for rounding. If you require more decimal places you can increase this number.\n \*\n \* @memberof Chartist.Core\n \* @type {number}\n \*/\n Chartist.precision = 8;\n\n /\*\*\n \* A map with characters to escape for strings to be safely used as attribute values.\n \*\n \* @memberof Chartist.Core\n \* @type {Object}\n \*/\n Chartist.escapingMap = {\n '&': '&amp;',\n '<': '&lt;',\n '>': '&gt;',\n '\"': '&quot;',\n '\\'': '&#039;'\n };\n\n /\*\*\n \* This function serializes arbitrary data to a string. In case of data that can't be easily converted to a string, this function will create a wrapper object and serialize the data using JSON.stringify. The outcoming string will always be escaped using Chartist.escapingMap.\n \* If called with null or undefined the function will return immediately with null or undefined.\n \*\n \* @memberof Chartist.Core\n \* @param {Number|String|Object} data\n \* @return {String}\n \*/\n Chartist.serialize = function(data) {\n if(data === null || data === undefined) {\n return data;\n } else if(typeof data === 'number') {\n data = ''+data;\n } else if(typeof data === 'object') {\n data = JSON.stringify({data: data});\n }\n\n return Object.keys(Chartist.escapingMap).reduce(function(result, key) {\n return Chartist.replaceAll(result, key, Chartist.escapingMap[key]);\n }, data);\n };\n\n /\*\*\n \* This function de-serializes a string previously serialized with Chartist.serialize. The string will always be unescaped using Chartist.escapingMap before it's returned. Based on the input value the return type can be Number, String or Object. JSON.parse is used with try / catch to see if the unescaped string can be parsed into an Object and this Object will be returned on success.\n \*\n \* @memberof Chartist.Core\n \* @param {String} data\n \* @return {String|Number|Object}\n \*/\n Chartist.deserialize = function(data) {\n if(typeof data !== 'string') {\n return data;\n }\n\n data = Object.keys(Chartist.escapingMap).reduce(function(result, key) {\n return Chartist.replaceAll(result, Chartist.escapingMap[key], key);\n }, data);\n\n try {\n data = JSON.parse(data);\n data = data.data !== undefined ? data.data : data;\n } catch(e) {}\n\n return data;\n };\n\n /\*\*\n \* Create or reinitialize the SVG element for the chart\n \*\n \* @memberof Chartist.Core\n \* @param {Node} container The containing DOM Node object that will be used to plant the SVG element\n \* @param {String} width Set the width of the SVG element. Default is 100%\n \* @param {String} height Set the height of the SVG element. Default is 100%\n \* @param {String} className Specify a class to be added to the SVG element\n \* @return {Object} The created/reinitialized SVG element\n \*/\n Chartist.createSvg = function (container, width, height, className) {\n var svg;\n\n width = width || '100%';\n height = height || '100%';\n\n // Check if there is a previous SVG element in the container that contains the Chartist XML namespace and remove it\n // Since the DOM API does not support namespaces we need to manually search the returned list http://www.w3.org/TR/selectors-api/\n Array.prototype.slice.call(container.querySelectorAll('svg')).filter(function filterChartistSvgObjects(svg) {\n return svg.getAttributeNS(Chartist.namespaces.xmlns, 'ct');\n }).forEach(function removePreviousElement(svg) {\n container.removeChild(svg);\n });\n\n // Create svg object with width and height or use 100% as default\n svg = new Chartist.Svg('svg').attr({\n width: width,\n height: height\n }).addClass(className).attr({\n style: 'width: ' + width + '; height: ' + height + ';'\n });\n\n // Add the DOM node to our container\n container.appendChild(svg.\_node);\n\n return svg;\n };\n\n /\*\*\n \* Ensures that the data object passed as second argument to the charts is present and correctly initialized.\n \*\n \* @param {Object} data The data object that is passed as second argument to the charts\n \* @return {Object} The normalized data object\n \*/\n Chartist.normalizeData = function(data, reverse, multi) {\n var labelCount;\n var output = {\n raw: data,\n normalized: {}\n };\n\n // Check if we should generate some labels based on existing series data\n output.normalized.series = Chartist.getDataArray({\n series: data.series || []\n }, reverse, multi);\n\n // If all elements of the normalized data array are arrays we're dealing with\n // multi series data and we need to find the largest series if they are un-even\n if (output.normalized.series.every(function(value) {\n return value instanceof Array;\n })) {\n // Getting the series with the the most elements\n labelCount = Math.max.apply(null, output.normalized.series.map(function(series) {\n return series.length;\n }));\n } else {\n // We're dealing with Pie data so we just take the normalized array length\n labelCount = output.normalized.series.length;\n }\n\n output.normalized.labels = (data.labels || []).slice();\n // Padding the labels to labelCount with empty strings\n Array.prototype.push.apply(\n output.normalized.labels,\n Chartist.times(Math.max(0, labelCount - output.normalized.labels.length)).map(function() {\n return '';\n })\n );\n\n if(reverse) {\n Chartist.reverseData(output.normalized);\n }\n\n return output;\n };\n\n /\*\*\n \* This function safely checks if an objects has an owned property.\n \*\n \* @param {Object} object The object where to check for a property\n \* @param {string} property The property name\n \* @returns {boolean} Returns true if the object owns the specified property\n \*/\n Chartist.safeHasProperty = function(object, property) {\n return object !== null &&\n typeof object === 'object' &&\n object.hasOwnProperty(property);\n };\n\n /\*\*\n \* Checks if a value is considered a hole in the data series.\n \*\n \* @param {\*} value\n \* @returns {boolean} True if the value is considered a data hole\n \*/\n Chartist.isDataHoleValue = function(value) {\n return value === null ||\n value === undefined ||\n (typeof value === 'number' && isNaN(value));\n };\n\n /\*\*\n \* Reverses the series, labels and series data arrays.\n \*\n \* @memberof Chartist.Core\n \* @param data\n \*/\n Chartist.reverseData = function(data) {\n data.labels.reverse();\n data.series.reverse();\n for (var i = 0; i < data.series.length; i++) {\n if(typeof(data.series[i]) === 'object' && data.series[i].data !== undefined) {\n data.series[i].data.reverse();\n } else if(data.series[i] instanceof Array) {\n data.series[i].reverse();\n }\n }\n };\n\n /\*\*\n \* Convert data series into plain array\n \*\n \* @memberof Chartist.Core\n \* @param {Object} data The series object that contains the data to be visualized in the chart\n \* @param {Boolean} [reverse] If true the whole data is reversed by the getDataArray call. This will modify the data object passed as first parameter. The labels as well as the series order is reversed. The whole series data arrays are reversed too.\n \* @param {Boolean} [multi] Create a multi dimensional array from a series data array where a value object with `x` and `y` values will be created.\n \* @return {Array} A plain array that contains the data to be visualized in the chart\n \*/\n Chartist.getDataArray = function(data, reverse, multi) {\n // Recursively walks through nested arrays and convert string values to numbers and objects with value properties\n // to values. Check the tests in data core -> data normalization for a detailed specification of expected values\n function recursiveConvert(value) {\n if(Chartist.safeHasProperty(value, 'value')) {\n // We are dealing with value object notation so we need to recurse on value property\n return recursiveConvert(value.value);\n } else if(Chartist.safeHasProperty(value, 'data')) {\n // We are dealing with series object notation so we need to recurse on data property\n return recursiveConvert(value.data);\n } else if(value instanceof Array) {\n // Data is of type array so we need to recurse on the series\n return value.map(recursiveConvert);\n } else if(Chartist.isDataHoleValue(value)) {\n // We're dealing with a hole in the data and therefore need to return undefined\n // We're also returning undefined for multi value output\n return undefined;\n } else {\n // We need to prepare multi value output (x and y data)\n if(multi) {\n var multiValue = {};\n\n // Single series value arrays are assumed to specify the Y-Axis value\n // For example: [1, 2] => [{x: undefined, y: 1}, {x: undefined, y: 2}]\n // If multi is a string then it's assumed that it specified which dimension should be filled as default\n if(typeof multi === 'string') {\n multiValue[multi] = Chartist.getNumberOrUndefined(value);\n } else {\n multiValue.y = Chartist.getNumberOrUndefined(value);\n }\n\n multiValue.x = value.hasOwnProperty('x') ? Chartist.getNumberOrUndefined(value.x) : multiValue.x;\n multiValue.y = value.hasOwnProperty('y') ? Chartist.getNumberOrUndefined(value.y) : multiValue.y;\n\n return multiValue;\n\n } else {\n // We can return simple data\n return Chartist.getNumberOrUndefined(value);\n }\n }\n }\n\n return data.series.map(recursiveConvert);\n };\n\n /\*\*\n \* Converts a number into a padding object.\n \*\n \* @memberof Chartist.Core\n \* @param {Object|Number} padding\n \* @param {Number} [fallback] This value is used to fill missing values if a incomplete padding object was passed\n \* @returns {Object} Returns a padding object containing top, right, bottom, left properties filled with the padding number passed in as argument. If the argument is something else than a number (presumably already a correct padding object) then this argument is directly returned.\n \*/\n Chartist.normalizePadding = function(padding, fallback) {\n fallback = fallback || 0;\n\n return typeof padding === 'number' ? {\n top: padding,\n right: padding,\n bottom: padding,\n left: padding\n } : {\n top: typeof padding.top === 'number' ? padding.top : fallback,\n right: typeof padding.right === 'number' ? padding.right : fallback,\n bottom: typeof padding.bottom === 'number' ? padding.bottom : fallback,\n left: typeof padding.left === 'number' ? padding.left : fallback\n };\n };\n\n Chartist.getMetaData = function(series, index) {\n var value = series.data ? series.data[index] : series[index];\n return value ? value.meta : undefined;\n };\n\n /\*\*\n \* Calculate the order of magnitude for the chart scale\n \*\n \* @memberof Chartist.Core\n \* @param {Number} value The value Range of the chart\n \* @return {Number} The order of magnitude\n \*/\n Chartist.orderOfMagnitude = function (value) {\n return Math.floor(Math.log(Math.abs(value)) / Math.LN10);\n };\n\n /\*\*\n \* Project a data length into screen coordinates (pixels)\n \*\n \* @memberof Chartist.Core\n \* @param {Object} axisLength The svg element for the chart\n \* @param {Number} length Single data value from a series array\n \* @param {Object} bounds All the values to set the bounds of the chart\n \* @return {Number} The projected data length in pixels\n \*/\n Chartist.projectLength = function (axisLength, length, bounds) {\n return length / bounds.range \* axisLength;\n };\n\n /\*\*\n \* Get the height of the area in the chart for the data series\n \*\n \* @memberof Chartist.Core\n \* @param {Object} svg The svg element for the chart\n \* @param {Object} options The Object that contains all the optional values for the chart\n \* @return {Number} The height of the area in the chart for the data series\n \*/\n Chartist.getAvailableHeight = function (svg, options) {\n return Math.max((Chartist.quantity(options.height).value || svg.height()) - (options.chartPadding.top + options.chartPadding.bottom) - options.axisX.offset, 0);\n };\n\n /\*\*\n \* Get highest and lowest value of data array. This Array contains the data that will be visualized in the chart.\n \*\n \* @memberof Chartist.Core\n \* @param {Array} data The array that contains the data to be visualized in the chart\n \* @param {Object} options The Object that contains the chart options\n \* @param {String} dimension Axis dimension 'x' or 'y' used to access the correct value and high / low configuration\n \* @return {Object} An object that contains the highest and lowest value that will be visualized on the chart.\n \*/\n Chartist.getHighLow = function (data, options, dimension) {\n // TODO: Remove workaround for deprecated global high / low config. Axis high / low configuration is preferred\n options = Chartist.extend({}, options, dimension ? options['axis' + dimension.toUpperCase()] : {});\n\n var highLow = {\n high: options.high === undefined ? -Number.MAX\_VALUE : +options.high,\n low: options.low === undefined ? Number.MAX\_VALUE : +options.low\n };\n var findHigh = options.high === undefined;\n var findLow = options.low === undefined;\n\n // Function to recursively walk through arrays and find highest and lowest number\n function recursiveHighLow(data) {\n if(data === undefined) {\n return undefined;\n } else if(data instanceof Array) {\n for (var i = 0; i < data.length; i++) {\n recursiveHighLow(data[i]);\n }\n } else {\n var value = dimension ? +data[dimension] : +data;\n\n if (findHigh && value > highLow.high) {\n highLow.high = value;\n }\n\n if (findLow && value < highLow.low) {\n highLow.low = value;\n }\n }\n }\n\n // Start to find highest and lowest number recursively\n if(findHigh || findLow) {\n recursiveHighLow(data);\n }\n\n // Overrides of high / low based on reference value, it will make sure that the invisible reference value is\n // used to generate the chart. This is useful when the chart always needs to contain the position of the\n // invisible reference value in the view i.e. for bipolar scales.\n if (options.referenceValue || options.referenceValue === 0) {\n highLow.high = Math.max(options.referenceValue, highLow.high);\n highLow.low = Math.min(options.referenceValue, highLow.low);\n }\n\n // If high and low are the same because of misconfiguration or flat data (only the same value) we need\n // to set the high or low to 0 depending on the polarity\n if (highLow.high <= highLow.low) {\n // If both values are 0 we set high to 1\n if (highLow.low === 0) {\n highLow.high = 1;\n } else if (highLow.low < 0) {\n // If we have the same negative value for the bounds we set bounds.high to 0\n highLow.high = 0;\n } else if (highLow.high > 0) {\n // If we have the same positive value for the bounds we set bounds.low to 0\n highLow.low = 0;\n } else {\n // If data array was empty, values are Number.MAX\_VALUE and -Number.MAX\_VALUE. Set bounds to prevent errors\n highLow.high = 1;\n highLow.low = 0;\n }\n }\n\n return highLow;\n };\n\n /\*\*\n \* Checks if a value can be safely coerced to a number. This includes all values except null which result in finite numbers when coerced. This excludes NaN, since it's not finite.\n \*\n \* @memberof Chartist.Core\n \* @param value\n \* @returns {Boolean}\n \*/\n Chartist.isNumeric = function(value) {\n return value === null ? false : isFinite(value);\n };\n\n /\*\*\n \* Returns true on all falsey values except the numeric value 0.\n \*\n \* @memberof Chartist.Core\n \* @param value\n \* @returns {boolean}\n \*/\n Chartist.isFalseyButZero = function(value) {\n return !value && value !== 0;\n };\n\n /\*\*\n \* Returns a number if the passed parameter is a valid number or the function will return undefined. On all other values than a valid number, this function will return undefined.\n \*\n \* @memberof Chartist.Core\n \* @param value\n \* @returns {\*}\n \*/\n Chartist.getNumberOrUndefined = function(value) {\n return Chartist.isNumeric(value) ? +value : undefined;\n };\n\n /\*\*\n \* Checks if provided value object is multi value (contains x or y properties)\n \*\n \* @memberof Chartist.Core\n \* @param value\n \*/\n Chartist.isMultiValue = function(value) {\n return typeof value === 'object' && ('x' in value || 'y' in value);\n };\n\n /\*\*\n \* Gets a value from a dimension `value.x` or `value.y` while returning value directly if it's a valid numeric value. If the value is not numeric and it's falsey this function will return `defaultValue`.\n \*\n \* @memberof Chartist.Core\n \* @param value\n \* @param dimension\n \* @param defaultValue\n \* @returns {\*}\n \*/\n Chartist.getMultiValue = function(value, dimension) {\n if(Chartist.isMultiValue(value)) {\n return Chartist.getNumberOrUndefined(value[dimension || 'y']);\n } else {\n return Chartist.getNumberOrUndefined(value);\n }\n };\n\n /\*\*\n \* Pollard Rho Algorithm to find smallest factor of an integer value. There are more efficient algorithms for factorization, but this one is quite efficient and not so complex.\n \*\n \* @memberof Chartist.Core\n \* @param {Number} num An integer number where the smallest factor should be searched for\n \* @returns {Number} The smallest integer factor of the parameter num.\n \*/\n Chartist.rho = function(num) {\n if(num === 1) {\n return num;\n }\n\n function gcd(p, q) {\n if (p % q === 0) {\n return q;\n } else {\n return gcd(q, p % q);\n }\n }\n\n function f(x) {\n return x \* x + 1;\n }\n\n var x1 = 2, x2 = 2, divisor;\n if (num % 2 === 0) {\n return 2;\n }\n\n do {\n x1 = f(x1) % num;\n x2 = f(f(x2)) % num;\n divisor = gcd(Math.abs(x1 - x2), num);\n } while (divisor === 1);\n\n return divisor;\n };\n\n /\*\*\n \* Calculate and retrieve all the bounds for the chart and return them in one array\n \*\n \* @memberof Chartist.Core\n \* @param {Number} axisLength The length of the Axis used for\n \* @param {Object} highLow An object containing a high and low property indicating the value range of the chart.\n \* @param {Number} scaleMinSpace The minimum projected length a step should result in\n \* @param {Boolean} onlyInteger\n \* @return {Object} All the values to set the bounds of the chart\n \*/\n Chartist.getBounds = function (axisLength, highLow, scaleMinSpace, onlyInteger) {\n var i,\n optimizationCounter = 0,\n newMin,\n newMax,\n bounds = {\n high: highLow.high,\n low: highLow.low\n };\n\n bounds.valueRange = bounds.high - bounds.low;\n bounds.oom = Chartist.orderOfMagnitude(bounds.valueRange);\n bounds.step = Math.pow(10, bounds.oom);\n bounds.min = Math.floor(bounds.low / bounds.step) \* bounds.step;\n bounds.max = Math.ceil(bounds.high / bounds.step) \* bounds.step;\n bounds.range = bounds.max - bounds.min;\n bounds.numberOfSteps = Math.round(bounds.range / bounds.step);\n\n // Optimize scale step by checking if subdivision is possible based on horizontalGridMinSpace\n // If we are already below the scaleMinSpace value we will scale up\n var length = Chartist.projectLength(axisLength, bounds.step, bounds);\n var scaleUp = length < scaleMinSpace;\n var smallestFactor = onlyInteger ? Chartist.rho(bounds.range) : 0;\n\n // First check if we should only use integer steps and if step 1 is still larger than scaleMinSpace so we can use 1\n if(onlyInteger && Chartist.projectLength(axisLength, 1, bounds) >= scaleMinSpace) {\n bounds.step = 1;\n } else if(onlyInteger && smallestFactor < bounds.step && Chartist.projectLength(axisLength, smallestFactor, bounds) >= scaleMinSpace) {\n // If step 1 was too small, we can try the smallest factor of range\n // If the smallest factor is smaller than the current bounds.step and the projected length of smallest factor\n // is larger than the scaleMinSpace we should go for it.\n bounds.step = smallestFactor;\n } else {\n // Trying to divide or multiply by 2 and find the best step value\n while (true) {\n if (scaleUp && Chartist.projectLength(axisLength, bounds.step, bounds) <= scaleMinSpace) {\n bounds.step \*= 2;\n } else if (!scaleUp && Chartist.projectLength(axisLength, bounds.step / 2, bounds) >= scaleMinSpace) {\n bounds.step /= 2;\n if(onlyInteger && bounds.step % 1 !== 0) {\n bounds.step \*= 2;\n break;\n }\n } else {\n break;\n }\n\n if(optimizationCounter++ > 1000) {\n throw new Error('Exceeded maximum number of iterations while optimizing scale step!');\n }\n }\n }\n\n var EPSILON = 2.221E-16;\n bounds.step = Math.max(bounds.step, EPSILON);\n function safeIncrement(value, increment) {\n // If increment is too small use \*= (1+EPSILON) as a simple nextafter\n if (value === (value += increment)) {\n \tvalue \*= (1 + (increment > 0 ? EPSILON : -EPSILON));\n }\n return value;\n }\n\n // Narrow min and max based on new step\n newMin = bounds.min;\n newMax = bounds.max;\n while (newMin + bounds.step <= bounds.low) {\n \tnewMin = safeIncrement(newMin, bounds.step);\n }\n while (newMax - bounds.step >= bounds.high) {\n \tnewMax = safeIncrement(newMax, -bounds.step);\n }\n bounds.min = newMin;\n bounds.max = newMax;\n bounds.range = bounds.max - bounds.min;\n\n var values = [];\n for (i = bounds.min; i <= bounds.max; i = safeIncrement(i, bounds.step)) {\n var value = Chartist.roundWithPrecision(i);\n if (value !== values[values.length - 1]) {\n values.push(value);\n }\n }\n bounds.values = values;\n return bounds;\n };\n\n /\*\*\n \* Calculate cartesian coordinates of polar coordinates\n \*\n \* @memberof Chartist.Core\n \* @param {Number} centerX X-axis coordinates of center point of circle segment\n \* @param {Number} centerY X-axis coordinates of center point of circle segment\n \* @param {Number} radius Radius of circle segment\n \* @param {Number} angleInDegrees Angle of circle segment in degrees\n \* @return {{x:Number, y:Number}} Coordinates of point on circumference\n \*/\n Chartist.polarToCartesian = function (centerX, centerY, radius, angleInDegrees) {\n var angleInRadians = (angleInDegrees - 90) \* Math.PI / 180.0;\n\n return {\n x: centerX + (radius \* Math.cos(angleInRadians)),\n y: centerY + (radius \* Math.sin(angleInRadians))\n };\n };\n\n /\*\*\n \* Initialize chart drawing rectangle (area where chart is drawn) x1,y1 = bottom left / x2,y2 = top right\n \*\n \* @memberof Chartist.Core\n \* @param {Object} svg The svg element for the chart\n \* @param {Object} options The Object that contains all the optional values for the chart\n \* @param {Number} [fallbackPadding] The fallback padding if partial padding objects are used\n \* @return {Object} The chart rectangles coordinates inside the svg element plus the rectangles measurements\n \*/\n Chartist.createChartRect = function (svg, options, fallbackPadding) {\n var hasAxis = !!(options.axisX || options.axisY);\n var yAxisOffset = hasAxis ? options.axisY.offset : 0;\n var xAxisOffset = hasAxis ? options.axisX.offset : 0;\n // If width or height results in invalid value (including 0) we fallback to the unitless settings or even 0\n var width = svg.width() || Chartist.quantity(options.width).value || 0;\n var height = svg.height() || Chartist.quantity(options.height).value || 0;\n var normalizedPadding = Chartist.normalizePadding(options.chartPadding, fallbackPadding);\n\n // If settings were to small to cope with offset (legacy) and padding, we'll adjust\n width = Math.max(width, yAxisOffset + normalizedPadding.left + normalizedPadding.right);\n height = Math.max(height, xAxisOffset + normalizedPadding.top + normalizedPadding.bottom);\n\n var chartRect = {\n padding: normalizedPadding,\n width: function () {\n return this.x2 - this.x1;\n },\n height: function () {\n return this.y1 - this.y2;\n }\n };\n\n if(hasAxis) {\n if (options.axisX.position === 'start') {\n chartRect.y2 = normalizedPadding.top + xAxisOffset;\n chartRect.y1 = Math.max(height - normalizedPadding.bottom, chartRect.y2 + 1);\n } else {\n chartRect.y2 = normalizedPadding.top;\n chartRect.y1 = Math.max(height - normalizedPadding.bottom - xAxisOffset, chartRect.y2 + 1);\n }\n\n if (options.axisY.position === 'start') {\n chartRect.x1 = normalizedPadding.left + yAxisOffset;\n chartRect.x2 = Math.max(width - normalizedPadding.right, chartRect.x1 + 1);\n } else {\n chartRect.x1 = normalizedPadding.left;\n chartRect.x2 = Math.max(width - normalizedPadding.right - yAxisOffset, chartRect.x1 + 1);\n }\n } else {\n chartRect.x1 = normalizedPadding.left;\n chartRect.x2 = Math.max(width - normalizedPadding.right, chartRect.x1 + 1);\n chartRect.y2 = normalizedPadding.top;\n chartRect.y1 = Math.max(height - normalizedPadding.bottom, chartRect.y2 + 1);\n }\n\n return chartRect;\n };\n\n /\*\*\n \* Creates a grid line based on a projected value.\n \*\n \* @memberof Chartist.Core\n \* @param position\n \* @param index\n \* @param axis\n \* @param offset\n \* @param length\n \* @param group\n \* @param classes\n \* @param eventEmitter\n \*/\n Chartist.createGrid = function(position, index, axis, offset, length, group, classes, eventEmitter) {\n var positionalData = {};\n positionalData[axis.units.pos + '1'] = position;\n positionalData[axis.units.pos + '2'] = position;\n positionalData[axis.counterUnits.pos + '1'] = offset;\n positionalData[axis.counterUnits.pos + '2'] = offset + length;\n\n var gridElement = group.elem('line', positionalData, classes.join(' '));\n\n // Event for grid draw\n eventEmitter.emit('draw',\n Chartist.extend({\n type: 'grid',\n axis: axis,\n index: index,\n group: group,\n element: gridElement\n }, positionalData)\n );\n };\n\n /\*\*\n \* Creates a grid background rect and emits the draw event.\n \*\n \* @memberof Chartist.Core\n \* @param gridGroup\n \* @param chartRect\n \* @param className\n \* @param eventEmitter\n \*/\n Chartist.createGridBackground = function (gridGroup, chartRect, className, eventEmitter) {\n var gridBackground = gridGroup.elem('rect', {\n x: chartRect.x1,\n y: chartRect.y2,\n width: chartRect.width(),\n height: chartRect.height(),\n }, className, true);\n\n // Event for grid background draw\n eventEmitter.emit('draw', {\n type: 'gridBackground',\n group: gridGroup,\n element: gridBackground\n });\n };\n\n /\*\*\n \* Creates a label based on a projected value and an axis.\n \*\n \* @memberof Chartist.Core\n \* @param position\n \* @param length\n \* @param index\n \* @param labels\n \* @param axis\n \* @param axisOffset\n \* @param labelOffset\n \* @param group\n \* @param classes\n \* @param useForeignObject\n \* @param eventEmitter\n \*/\n Chartist.createLabel = function(position, length, index, labels, axis, axisOffset, labelOffset, group, classes, useForeignObject, eventEmitter) {\n var labelElement;\n var positionalData = {};\n\n positionalData[axis.units.pos] = position + labelOffset[axis.units.pos];\n positionalData[axis.counterUnits.pos] = labelOffset[axis.counterUnits.pos];\n positionalData[axis.units.len] = length;\n positionalData[axis.counterUnits.len] = Math.max(0, axisOffset - 10);\n\n if(useForeignObject) {\n // We need to set width and height explicitly to px as span will not expand with width and height being\n // 100% in all browsers\n var content = '<span class=\"' + classes.join(' ') + '\" style=\"' +\n axis.units.len + ': ' + Math.round(positionalData[axis.units.len]) + 'px; ' +\n axis.counterUnits.len + ': ' + Math.round(positionalData[axis.counterUnits.len]) + 'px\">' +\n labels[index] + '</span>';\n\n labelElement = group.foreignObject(content, Chartist.extend({\n style: 'overflow: visible;'\n }, positionalData));\n } else {\n labelElement = group.elem('text', positionalData, classes.join(' ')).text(labels[index]);\n }\n\n eventEmitter.emit('draw', Chartist.extend({\n type: 'label',\n axis: axis,\n index: index,\n group: group,\n element: labelElement,\n text: labels[index]\n }, positionalData));\n };\n\n /\*\*\n \* Helper to read series specific options from options object. It automatically falls back to the global option if\n \* there is no option in the series options.\n \*\n \* @param {Object} series Series object\n \* @param {Object} options Chartist options object\n \* @param {string} key The options key that should be used to obtain the options\n \* @returns {\*}\n \*/\n Chartist.getSeriesOption = function(series, options, key) {\n if(series.name && options.series && options.series[series.name]) {\n var seriesOptions = options.series[series.name];\n return seriesOptions.hasOwnProperty(key) ? seriesOptions[key] : options[key];\n } else {\n return options[key];\n }\n };\n\n /\*\*\n \* Provides options handling functionality with callback for options changes triggered by responsive options and media query matches\n \*\n \* @memberof Chartist.Core\n \* @param {Object} options Options set by user\n \* @param {Array} responsiveOptions Optional functions to add responsive behavior to chart\n \* @param {Object} eventEmitter The event emitter that will be used to emit the options changed events\n \* @return {Object} The consolidated options object from the defaults, base and matching responsive options\n \*/\n Chartist.optionsProvider = function (options, responsiveOptions, eventEmitter) {\n var baseOptions = Chartist.extend({}, options),\n currentOptions,\n mediaQueryListeners = [],\n i;\n\n function updateCurrentOptions(mediaEvent) {\n var previousOptions = currentOptions;\n currentOptions = Chartist.extend({}, baseOptions);\n\n if (responsiveOptions) {\n for (i = 0; i < responsiveOptions.length; i++) {\n var mql = window.matchMedia(responsiveOptions[i][0]);\n if (mql.matches) {\n currentOptions = Chartist.extend(currentOptions, responsiveOptions[i][1]);\n }\n }\n }\n\n if(eventEmitter && mediaEvent) {\n eventEmitter.emit('optionsChanged', {\n previousOptions: previousOptions,\n currentOptions: currentOptions\n });\n }\n }\n\n function removeMediaQueryListeners() {\n mediaQueryListeners.forEach(function(mql) {\n mql.removeListener(updateCurrentOptions);\n });\n }\n\n if (!window.matchMedia) {\n throw 'window.matchMedia not found! Make sure you\\'re using a polyfill.';\n } else if (responsiveOptions) {\n\n for (i = 0; i < responsiveOptions.length; i++) {\n var mql = window.matchMedia(responsiveOptions[i][0]);\n mql.addListener(updateCurrentOptions);\n mediaQueryListeners.push(mql);\n }\n }\n // Execute initially without an event argument so we get the correct options\n updateCurrentOptions();\n\n return {\n removeMediaQueryListeners: removeMediaQueryListeners,\n getCurrentOptions: function getCurrentOptions() {\n return Chartist.extend({}, currentOptions);\n }\n };\n };\n\n\n /\*\*\n \* Splits a list of coordinates and associated values into segments. Each returned segment contains a pathCoordinates\n \* valueData property describing the segment.\n \*\n \* With the default options, segments consist of contiguous sets of points that do not have an undefined value. Any\n \* points with undefined values are discarded.\n \*\n \* \*\*Options\*\*\n \* The following options are used to determine how segments are formed\n \* ```javascript\n \* var options = {\n \* // If fillHoles is true, undefined values are simply discarded without creating a new segment. Assuming other options are default, this returns single segment.\n \* fillHoles: false,\n \* // If increasingX is true, the coordinates in all segments have strictly increasing x-values.\n \* increasingX: false\n \* };\n \* ```\n \*\n \* @memberof Chartist.Core\n \* @param {Array} pathCoordinates List of point coordinates to be split in the form [x1, y1, x2, y2 ... xn, yn]\n \* @param {Array} values List of associated point values in the form [v1, v2 .. vn]\n \* @param {Object} options Options set by user\n \* @return {Array} List of segments, each containing a pathCoordinates and valueData property.\n \*/\n Chartist.splitIntoSegments = function(pathCoordinates, valueData, options) {\n var defaultOptions = {\n increasingX: false,\n fillHoles: false\n };\n\n options = Chartist.extend({}, defaultOptions, options);\n\n var segments = [];\n var hole = true;\n\n for(var i = 0; i < pathCoordinates.length; i += 2) {\n // If this value is a \"hole\" we set the hole flag\n if(Chartist.getMultiValue(valueData[i / 2].value) === undefined) {\n // if(valueData[i / 2].value === undefined) {\n if(!options.fillHoles) {\n hole = true;\n }\n } else {\n if(options.increasingX && i >= 2 && pathCoordinates[i] <= pathCoordinates[i-2]) {\n // X is not increasing, so we need to make sure we start a new segment\n hole = true;\n }\n\n\n // If it's a valid value we need to check if we're coming out of a hole and create a new empty segment\n if(hole) {\n segments.push({\n pathCoordinates: [],\n valueData: []\n });\n // As we have a valid value now, we are not in a \"hole\" anymore\n hole = false;\n }\n\n // Add to the segment pathCoordinates and valueData\n segments[segments.length - 1].pathCoordinates.push(pathCoordinates[i], pathCoordinates[i + 1]);\n segments[segments.length - 1].valueData.push(valueData[i / 2]);\n }\n }\n\n return segments;\n };\n}(window, document, Chartist));\n;/\*\*\n \* Chartist path interpolation functions.\n \*\n \* @module Chartist.Interpolation\n \*/\n/\* global Chartist \*/\n(function(window, document, Chartist) {\n 'use strict';\n\n Chartist.Interpolation = {};\n\n /\*\*\n \* This interpolation function does not smooth the path and the result is only containing lines and no curves.\n \*\n \* @example\n \* var chart = new Chartist.Line('.ct-chart', {\n \* labels: [1, 2, 3, 4, 5],\n \* series: [[1, 2, 8, 1, 7]]\n \* }, {\n \* lineSmooth: Chartist.Interpolation.none({\n \* fillHoles: false\n \* })\n \* });\n \*\n \*\n \* @memberof Chartist.Interpolation\n \* @return {Function}\n \*/\n Chartist.Interpolation.none = function(options) {\n var defaultOptions = {\n fillHoles: false\n };\n options = Chartist.extend({}, defaultOptions, options);\n return function none(pathCoordinates, valueData) {\n var path = new Chartist.Svg.Path();\n var hole = true;\n\n for(var i = 0; i < pathCoordinates.length; i += 2) {\n var currX = pathCoordinates[i];\n var currY = pathCoordinates[i + 1];\n var currData = valueData[i / 2];\n\n if(Chartist.getMultiValue(currData.value) !== undefined) {\n\n if(hole) {\n path.move(currX, currY, false, currData);\n } else {\n path.line(currX, currY, false, currData);\n }\n\n hole = false;\n } else if(!options.fillHoles) {\n hole = true;\n }\n }\n\n return path;\n };\n };\n\n /\*\*\n \* Simple smoothing creates horizontal handles that are positioned with a fraction of the length between two data points. You can use the divisor option to specify the amount of smoothing.\n \*\n \* Simple smoothing can be used instead of `Chartist.Smoothing.cardinal` if you'd like to get rid of the artifacts it produces sometimes. Simple smoothing produces less flowing lines but is accurate by hitting the points and it also doesn't swing below or above the given data point.\n \*\n \* All smoothing functions within Chartist are factory functions that accept an options parameter. The simple interpolation function accepts one configuration parameter `divisor`, between 1 and ∞, which controls the smoothing characteristics.\n \*\n \* @example\n \* var chart = new Chartist.Line('.ct-chart', {\n \* labels: [1, 2, 3, 4, 5],\n \* series: [[1, 2, 8, 1, 7]]\n \* }, {\n \* lineSmooth: Chartist.Interpolation.simple({\n \* divisor: 2,\n \* fillHoles: false\n \* })\n \* });\n \*\n \*\n \* @memberof Chartist.Interpolation\n \* @param {Object} options The options of the simple interpolation factory function.\n \* @return {Function}\n \*/\n Chartist.Interpolation.simple = function(options) {\n var defaultOptions = {\n divisor: 2,\n fillHoles: false\n };\n options = Chartist.extend({}, defaultOptions, options);\n\n var d = 1 / Math.max(1, options.divisor);\n\n return function simple(pathCoordinates, valueData) {\n var path = new Chartist.Svg.Path();\n var prevX, prevY, prevData;\n\n for(var i = 0; i < pathCoordinates.length; i += 2) {\n var currX = pathCoordinates[i];\n var currY = pathCoordinates[i + 1];\n var length = (currX - prevX) \* d;\n var currData = valueData[i / 2];\n\n if(currData.value !== undefined) {\n\n if(prevData === undefined) {\n path.move(currX, currY, false, currData);\n } else {\n path.curve(\n prevX + length,\n prevY,\n currX - length,\n currY,\n currX,\n currY,\n false,\n currData\n );\n }\n\n prevX = currX;\n prevY = currY;\n prevData = currData;\n } else if(!options.fillHoles) {\n prevX = currX = prevData = undefined;\n }\n }\n\n return path;\n };\n };\n\n /\*\*\n \* Cardinal / Catmull-Rome spline interpolation is the default smoothing function in Chartist. It produces nice results where the splines will always meet the points. It produces some artifacts though when data values are increased or decreased rapidly. The line may not follow a very accurate path and if the line should be accurate this smoothing function does not produce the best results.\n \*\n \* Cardinal splines can only be created if there are more than two data points. If this is not the case this smoothing will fallback to `Chartist.Smoothing.none`.\n \*\n \* All smoothing functions within Chartist are factory functions that accept an options parameter. The cardinal interpolation function accepts one configuration parameter `tension`, between 0 and 1, which controls the smoothing intensity.\n \*\n \* @example\n \* var chart = new Chartist.Line('.ct-chart', {\n \* labels: [1, 2, 3, 4, 5],\n \* series: [[1, 2, 8, 1, 7]]\n \* }, {\n \* lineSmooth: Chartist.Interpolation.cardinal({\n \* tension: 1,\n \* fillHoles: false\n \* })\n \* });\n \*\n \* @memberof Chartist.Interpolation\n \* @param {Object} options The options of the cardinal factory function.\n \* @return {Function}\n \*/\n Chartist.Interpolation.cardinal = function(options) {\n var defaultOptions = {\n tension: 1,\n fillHoles: false\n };\n\n options = Chartist.extend({}, defaultOptions, options);\n\n var t = Math.min(1, Math.max(0, options.tension)),\n c = 1 - t;\n\n return function cardinal(pathCoordinates, valueData) {\n // First we try to split the coordinates into segments\n // This is necessary to treat \"holes\" in line charts\n var segments = Chartist.splitIntoSegments(pathCoordinates, valueData, {\n fillHoles: options.fillHoles\n });\n\n if(!segments.length) {\n // If there were no segments return 'Chartist.Interpolation.none'\n return Chartist.Interpolation.none()([]);\n } else if(segments.length > 1) {\n // If the split resulted in more that one segment we need to interpolate each segment individually and join them\n // afterwards together into a single path.\n var paths = [];\n // For each segment we will recurse the cardinal function\n segments.forEach(function(segment) {\n paths.push(cardinal(segment.pathCoordinates, segment.valueData));\n });\n // Join the segment path data into a single path and return\n return Chartist.Svg.Path.join(paths);\n } else {\n // If there was only one segment we can proceed regularly by using pathCoordinates and valueData from the first\n // segment\n pathCoordinates = segments[0].pathCoordinates;\n valueData = segments[0].valueData;\n\n // If less than two points we need to fallback to no smoothing\n if(pathCoordinates.length <= 4) {\n return Chartist.Interpolation.none()(pathCoordinates, valueData);\n }\n\n var path = new Chartist.Svg.Path().move(pathCoordinates[0], pathCoordinates[1], false, valueData[0]),\n z;\n\n for (var i = 0, iLen = pathCoordinates.length; iLen - 2 \* !z > i; i += 2) {\n var p = [\n {x: +pathCoordinates[i - 2], y: +pathCoordinates[i - 1]},\n {x: +pathCoordinates[i], y: +pathCoordinates[i + 1]},\n {x: +pathCoordinates[i + 2], y: +pathCoordinates[i + 3]},\n {x: +pathCoordinates[i + 4], y: +pathCoordinates[i + 5]}\n ];\n if (z) {\n if (!i) {\n p[0] = {x: +pathCoordinates[iLen - 2], y: +pathCoordinates[iLen - 1]};\n } else if (iLen - 4 === i) {\n p[3] = {x: +pathCoordinates[0], y: +pathCoordinates[1]};\n } else if (iLen - 2 === i) {\n p[2] = {x: +pathCoordinates[0], y: +pathCoordinates[1]};\n p[3] = {x: +pathCoordinates[2], y: +pathCoordinates[3]};\n }\n } else {\n if (iLen - 4 === i) {\n p[3] = p[2];\n } else if (!i) {\n p[0] = {x: +pathCoordinates[i], y: +pathCoordinates[i + 1]};\n }\n }\n\n path.curve(\n (t \* (-p[0].x + 6 \* p[1].x + p[2].x) / 6) + (c \* p[2].x),\n (t \* (-p[0].y + 6 \* p[1].y + p[2].y) / 6) + (c \* p[2].y),\n (t \* (p[1].x + 6 \* p[2].x - p[3].x) / 6) + (c \* p[2].x),\n (t \* (p[1].y + 6 \* p[2].y - p[3].y) / 6) + (c \* p[2].y),\n p[2].x,\n p[2].y,\n false,\n valueData[(i + 2) / 2]\n );\n }\n\n return path;\n }\n };\n };\n\n /\*\*\n \* Monotone Cubic spline interpolation produces a smooth curve which preserves monotonicity. Unlike cardinal splines, the curve will not extend beyond the range of y-values of the original data points.\n \*\n \* Monotone Cubic splines can only be created if there are more than two data points. If this is not the case this smoothing will fallback to `Chartist.Smoothing.none`.\n \*\n \* The x-values of subsequent points must be increasing to fit a Monotone Cubic spline. If this condition is not met for a pair of adjacent points, then there will be a break in the curve between those data points.\n \*\n \* All smoothing functions within Chartist are factory functions that accept an options parameter.\n \*\n \* @example\n \* var chart = new Chartist.Line('.ct-chart', {\n \* labels: [1, 2, 3, 4, 5],\n \* series: [[1, 2, 8, 1, 7]]\n \* }, {\n \* lineSmooth: Chartist.Interpolation.monotoneCubic({\n \* fillHoles: false\n \* })\n \* });\n \*\n \* @memberof Chartist.Interpolation\n \* @param {Object} options The options of the monotoneCubic factory function.\n \* @return {Function}\n \*/\n Chartist.Interpolation.monotoneCubic = function(options) {\n var defaultOptions = {\n fillHoles: false\n };\n\n options = Chartist.extend({}, defaultOptions, options);\n\n return function monotoneCubic(pathCoordinates, valueData) {\n // First we try to split the coordinates into segments\n // This is necessary to treat \"holes\" in line charts\n var segments = Chartist.splitIntoSegments(pathCoordinates, valueData, {\n fillHoles: options.fillHoles,\n increasingX: true\n });\n\n if(!segments.length) {\n // If there were no segments return 'Chartist.Interpolation.none'\n return Chartist.Interpolation.none()([]);\n } else if(segments.length > 1) {\n // If the split resulted in more that one segment we need to interpolate each segment individually and join them\n // afterwards together into a single path.\n var paths = [];\n // For each segment we will recurse the monotoneCubic fn function\n segments.forEach(function(segment) {\n paths.push(monotoneCubic(segment.pathCoordinates, segment.valueData));\n });\n // Join the segment path data into a single path and return\n return Chartist.Svg.Path.join(paths);\n } else {\n // If there was only one segment we can proceed regularly by using pathCoordinates and valueData from the first\n // segment\n pathCoordinates = segments[0].pathCoordinates;\n valueData = segments[0].valueData;\n\n // If less than three points we need to fallback to no smoothing\n if(pathCoordinates.length <= 4) {\n return Chartist.Interpolation.none()(pathCoordinates, valueData);\n }\n\n var xs = [],\n ys = [],\n i,\n n = pathCoordinates.length / 2,\n ms = [],\n ds = [], dys = [], dxs = [],\n path;\n\n // Populate x and y coordinates into separate arrays, for readability\n\n for(i = 0; i < n; i++) {\n xs[i] = pathCoordinates[i \* 2];\n ys[i] = pathCoordinates[i \* 2 + 1];\n }\n\n // Calculate deltas and derivative\n\n for(i = 0; i < n - 1; i++) {\n dys[i] = ys[i + 1] - ys[i];\n dxs[i] = xs[i + 1] - xs[i];\n ds[i] = dys[i] / dxs[i];\n }\n\n // Determine desired slope (m) at each point using Fritsch-Carlson method\n // See: http://math.stackexchange.com/questions/45218/implementation-of-monotone-cubic-interpolation\n\n ms[0] = ds[0];\n ms[n - 1] = ds[n - 2];\n\n for(i = 1; i < n - 1; i++) {\n if(ds[i] === 0 || ds[i - 1] === 0 || (ds[i - 1] > 0) !== (ds[i] > 0)) {\n ms[i] = 0;\n } else {\n ms[i] = 3 \* (dxs[i - 1] + dxs[i]) / (\n (2 \* dxs[i] + dxs[i - 1]) / ds[i - 1] +\n (dxs[i] + 2 \* dxs[i - 1]) / ds[i]);\n\n if(!isFinite(ms[i])) {\n ms[i] = 0;\n }\n }\n }\n\n // Now build a path from the slopes\n\n path = new Chartist.Svg.Path().move(xs[0], ys[0], false, valueData[0]);\n\n for(i = 0; i < n - 1; i++) {\n path.curve(\n // First control point\n xs[i] + dxs[i] / 3,\n ys[i] + ms[i] \* dxs[i] / 3,\n // Second control point\n xs[i + 1] - dxs[i] / 3,\n ys[i + 1] - ms[i + 1] \* dxs[i] / 3,\n // End point\n xs[i + 1],\n ys[i + 1],\n\n false,\n valueData[i + 1]\n );\n }\n\n return path;\n }\n };\n };\n\n /\*\*\n \* Step interpolation will cause the line chart to move in steps rather than diagonal or smoothed lines. This interpolation will create additional points that will also be drawn when the `showPoint` option is enabled.\n \*\n \* All smoothing functions within Chartist are factory functions that accept an options parameter. The step interpolation function accepts one configuration parameter `postpone`, that can be `true` or `false`. The default value is `true` and will cause the step to occur where the value actually changes. If a different behaviour is needed where the step is shifted to the left and happens before the actual value, this option can be set to `false`.\n \*\n \* @example\n \* var chart = new Chartist.Line('.ct-chart', {\n \* labels: [1, 2, 3, 4, 5],\n \* series: [[1, 2, 8, 1, 7]]\n \* }, {\n \* lineSmooth: Chartist.Interpolation.step({\n \* postpone: true,\n \* fillHoles: false\n \* })\n \* });\n \*\n \* @memberof Chartist.Interpolation\n \* @param options\n \* @returns {Function}\n \*/\n Chartist.Interpolation.step = function(options) {\n var defaultOptions = {\n postpone: true,\n fillHoles: false\n };\n\n options = Chartist.extend({}, defaultOptions, options);\n\n return function step(pathCoordinates, valueData) {\n var path = new Chartist.Svg.Path();\n\n var prevX, prevY, prevData;\n\n for (var i = 0; i < pathCoordinates.length; i += 2) {\n var currX = pathCoordinates[i];\n var currY = pathCoordinates[i + 1];\n var currData = valueData[i / 2];\n\n // If the current point is also not a hole we can draw the step lines\n if(currData.value !== undefined) {\n if(prevData === undefined) {\n path.move(currX, currY, false, currData);\n } else {\n if(options.postpone) {\n // If postponed we should draw the step line with the value of the previous value\n path.line(currX, prevY, false, prevData);\n } else {\n // If not postponed we should draw the step line with the value of the current value\n path.line(prevX, currY, false, currData);\n }\n // Line to the actual point (this should only be a Y-Axis movement\n path.line(currX, currY, false, currData);\n }\n\n prevX = currX;\n prevY = currY;\n prevData = currData;\n } else if(!options.fillHoles) {\n prevX = prevY = prevData = undefined;\n }\n }\n\n return path;\n };\n };\n\n}(window, document, Chartist));\n;/\*\*\n \* A very basic event module that helps to generate and catch events.\n \*\n \* @module Chartist.Event\n \*/\n/\* global Chartist \*/\n(function (window, document, Chartist) {\n 'use strict';\n\n Chartist.EventEmitter = function () {\n var handlers = [];\n\n /\*\*\n \* Add an event handler for a specific event\n \*\n \* @memberof Chartist.Event\n \* @param {String} event The event name\n \* @param {Function} handler A event handler function\n \*/\n function addEventHandler(event, handler) {\n handlers[event] = handlers[event] || [];\n handlers[event].push(handler);\n }\n\n /\*\*\n \* Remove an event handler of a specific event name or remove all event handlers for a specific event.\n \*\n \* @memberof Chartist.Event\n \* @param {String} event The event name where a specific or all handlers should be removed\n \* @param {Function} [handler] An optional event handler function. If specified only this specific handler will be removed and otherwise all handlers are removed.\n \*/\n function removeEventHandler(event, handler) {\n // Only do something if there are event handlers with this name existing\n if(handlers[event]) {\n // If handler is set we will look for a specific handler and only remove this\n if(handler) {\n handlers[event].splice(handlers[event].indexOf(handler), 1);\n if(handlers[event].length === 0) {\n delete handlers[event];\n }\n } else {\n // If no handler is specified we remove all handlers for this event\n delete handlers[event];\n }\n }\n }\n\n /\*\*\n \* Use this function to emit an event. All handlers that are listening for this event will be triggered with the data parameter.\n \*\n \* @memberof Chartist.Event\n \* @param {String} event The event name that should be triggered\n \* @param {\*} data Arbitrary data that will be passed to the event handler callback functions\n \*/\n function emit(event, data) {\n // Only do something if there are event handlers with this name existing\n if(handlers[event]) {\n handlers[event].forEach(function(handler) {\n handler(data);\n });\n }\n\n // Emit event to star event handlers\n if(handlers['\*']) {\n handlers['\*'].forEach(function(starHandler) {\n starHandler(event, data);\n });\n }\n }\n\n return {\n addEventHandler: addEventHandler,\n removeEventHandler: removeEventHandler,\n emit: emit\n };\n };\n\n}(window, document, Chartist));\n;/\*\*\n \* This module provides some basic prototype inheritance utilities.\n \*\n \* @module Chartist.Class\n \*/\n/\* global Chartist \*/\n(function(window, document, Chartist) {\n 'use strict';\n\n function listToArray(list) {\n var arr = [];\n if (list.length) {\n for (var i = 0; i < list.length; i++) {\n arr.push(list[i]);\n }\n }\n return arr;\n }\n\n /\*\*\n \* Method to extend from current prototype.\n \*\n \* @memberof Chartist.Class\n \* @param {Object} properties The object that serves as definition for the prototype that gets created for the new class. This object should always contain a constructor property that is the desired constructor for the newly created class.\n \* @param {Object} [superProtoOverride] By default extens will use the current class prototype or Chartist.class. With this parameter you can specify any super prototype that will be used.\n \* @return {Function} Constructor function of the new class\n \*\n \* @example\n \* var Fruit = Class.extend({\n \* color: undefined,\n \* sugar: undefined,\n \*\n \* constructor: function(color, sugar) {\n \* this.color = color;\n \* this.sugar = sugar;\n \* },\n \*\n \* eat: function() {\n \* this.sugar = 0;\n \* return this;\n \* }\n \* });\n \*\n \* var Banana = Fruit.extend({\n \* length: undefined,\n \*\n \* constructor: function(length, sugar) {\n \* Banana.super.constructor.call(this, 'Yellow', sugar);\n \* this.length = length;\n \* }\n \* });\n \*\n \* var banana = new Banana(20, 40);\n \* console.log('banana instanceof Fruit', banana instanceof Fruit);\n \* console.log('Fruit is prototype of banana', Fruit.prototype.isPrototypeOf(banana));\n \* console.log('bananas prototype is Fruit', Object.getPrototypeOf(banana) === Fruit.prototype);\n \* console.log(banana.sugar);\n \* console.log(banana.eat().sugar);\n \* console.log(banana.color);\n \*/\n function extend(properties, superProtoOverride) {\n var superProto = superProtoOverride || this.prototype || Chartist.Class;\n var proto = Object.create(superProto);\n\n Chartist.Class.cloneDefinitions(proto, properties);\n\n var constr = function() {\n var fn = proto.constructor || function () {},\n instance;\n\n // If this is linked to the Chartist namespace the constructor was not called with new\n // To provide a fallback we will instantiate here and return the instance\n instance = this === Chartist ? Object.create(proto) : this;\n fn.apply(instance, Array.prototype.slice.call(arguments, 0));\n\n // If this constructor was not called with new we need to return the instance\n // This will not harm when the constructor has been called with new as the returned value is ignored\n return instance;\n };\n\n constr.prototype = proto;\n constr.super = superProto;\n constr.extend = this.extend;\n\n return constr;\n }\n\n // Variable argument list clones args > 0 into args[0] and retruns modified args[0]\n function cloneDefinitions() {\n var args = listToArray(arguments);\n var target = args[0];\n\n args.splice(1, args.length - 1).forEach(function (source) {\n Object.getOwnPropertyNames(source).forEach(function (propName) {\n // If this property already exist in target we delete it first\n delete target[propName];\n // Define the property with the descriptor from source\n Object.defineProperty(target, propName,\n Object.getOwnPropertyDescriptor(source, propName));\n });\n });\n\n return target;\n }\n\n Chartist.Class = {\n extend: extend,\n cloneDefinitions: cloneDefinitions\n };\n\n}(window, document, Chartist));\n;/\*\*\n \* Base for all chart types. The methods in Chartist.Base are inherited to all chart types.\n \*\n \* @module Chartist.Base\n \*/\n/\* global Chartist \*/\n(function(window, document, Chartist) {\n 'use strict';\n\n // TODO: Currently we need to re-draw the chart on window resize. This is usually very bad and will affect performance.\n // This is done because we can't work with relative coordinates when drawing the chart because SVG Path does not\n // work with relative positions yet. We need to check if we can do a viewBox hack to switch to percentage.\n // See http://mozilla.6506.n7.nabble.com/Specyfing-paths-with-percentages-unit-td247474.html\n // Update: can be done using the above method tested here: http://codepen.io/gionkunz/pen/KDvLj\n // The problem is with the label offsets that can't be converted into percentage and affecting the chart container\n /\*\*\n \* Updates the chart which currently does a full reconstruction of the SVG DOM\n \*\n \* @param {Object} [data] Optional data you'd like to set for the chart before it will update. If not specified the update method will use the data that is already configured with the chart.\n \* @param {Object} [options] Optional options you'd like to add to the previous options for the chart before it will update. If not specified the update method will use the options that have been already configured with the chart.\n \* @param {Boolean} [override] If set to true, the passed options will be used to extend the options that have been configured already. Otherwise the chart default options will be used as the base\n \* @memberof Chartist.Base\n \*/\n function update(data, options, override) {\n if(data) {\n this.data = data || {};\n this.data.labels = this.data.labels || [];\n this.data.series = this.data.series || [];\n // Event for data transformation that allows to manipulate the data before it gets rendered in the charts\n this.eventEmitter.emit('data', {\n type: 'update',\n data: this.data\n });\n }\n\n if(options) {\n this.options = Chartist.extend({}, override ? this.options : this.defaultOptions, options);\n\n // If chartist was not initialized yet, we just set the options and leave the rest to the initialization\n // Otherwise we re-create the optionsProvider at this point\n if(!this.initializeTimeoutId) {\n this.optionsProvider.removeMediaQueryListeners();\n this.optionsProvider = Chartist.optionsProvider(this.options, this.responsiveOptions, this.eventEmitter);\n }\n }\n\n // Only re-created the chart if it has been initialized yet\n if(!this.initializeTimeoutId) {\n this.createChart(this.optionsProvider.getCurrentOptions());\n }\n\n // Return a reference to the chart object to chain up calls\n return this;\n }\n\n /\*\*\n \* This method can be called on the API object of each chart and will un-register all event listeners that were added to other components. This currently includes a window.resize listener as well as media query listeners if any responsive options have been provided. Use this function if you need to destroy and recreate Chartist charts dynamically.\n \*\n \* @memberof Chartist.Base\n \*/\n function detach() {\n // Only detach if initialization already occurred on this chart. If this chart still hasn't initialized (therefore\n // the initializationTimeoutId is still a valid timeout reference, we will clear the timeout\n if(!this.initializeTimeoutId) {\n window.removeEventListener('resize', this.resizeListener);\n this.optionsProvider.removeMediaQueryListeners();\n } else {\n window.clearTimeout(this.initializeTimeoutId);\n }\n\n return this;\n }\n\n /\*\*\n \* Use this function to register event handlers. The handler callbacks are synchronous and will run in the main thread rather than the event loop.\n \*\n \* @memberof Chartist.Base\n \* @param {String} event Name of the event. Check the examples for supported events.\n \* @param {Function} handler The handler function that will be called when an event with the given name was emitted. This function will receive a data argument which contains event data. See the example for more details.\n \*/\n function on(event, handler) {\n this.eventEmitter.addEventHandler(event, handler);\n return this;\n }\n\n /\*\*\n \* Use this function to un-register event handlers. If the handler function parameter is omitted all handlers for the given event will be un-registered.\n \*\n \* @memberof Chartist.Base\n \* @param {String} event Name of the event for which a handler should be removed\n \* @param {Function} [handler] The handler function that that was previously used to register a new event handler. This handler will be removed from the event handler list. If this parameter is omitted then all event handlers for the given event are removed from the list.\n \*/\n function off(event, handler) {\n this.eventEmitter.removeEventHandler(event, handler);\n return this;\n }\n\n function initialize() {\n // Add window resize listener that re-creates the chart\n window.addEventListener('resize', this.resizeListener);\n\n // Obtain current options based on matching media queries (if responsive options are given)\n // This will also register a listener that is re-creating the chart based on media changes\n this.optionsProvider = Chartist.optionsProvider(this.options, this.responsiveOptions, this.eventEmitter);\n // Register options change listener that will trigger a chart update\n this.eventEmitter.addEventHandler('optionsChanged', function() {\n this.update();\n }.bind(this));\n\n // Before the first chart creation we need to register us with all plugins that are configured\n // Initialize all relevant plugins with our chart object and the plugin options specified in the config\n if(this.options.plugins) {\n this.options.plugins.forEach(function(plugin) {\n if(plugin instanceof Array) {\n plugin[0](this, plugin[1]);\n } else {\n plugin(this);\n }\n }.bind(this));\n }\n\n // Event for data transformation that allows to manipulate the data before it gets rendered in the charts\n this.eventEmitter.emit('data', {\n type: 'initial',\n data: this.data\n });\n\n // Create the first chart\n this.createChart(this.optionsProvider.getCurrentOptions());\n\n // As chart is initialized from the event loop now we can reset our timeout reference\n // This is important if the chart gets initialized on the same element twice\n this.initializeTimeoutId = undefined;\n }\n\n /\*\*\n \* Constructor of chart base class.\n \*\n \* @param query\n \* @param data\n \* @param defaultOptions\n \* @param options\n \* @param responsiveOptions\n \* @constructor\n \*/\n function Base(query, data, defaultOptions, options, responsiveOptions) {\n this.container = Chartist.querySelector(query);\n this.data = data || {};\n this.data.labels = this.data.labels || [];\n this.data.series = this.data.series || [];\n this.defaultOptions = defaultOptions;\n this.options = options;\n this.responsiveOptions = responsiveOptions;\n this.eventEmitter = Chartist.EventEmitter();\n this.supportsForeignObject = Chartist.Svg.isSupported('Extensibility');\n this.supportsAnimations = Chartist.Svg.isSupported('AnimationEventsAttribute');\n this.resizeListener = function resizeListener(){\n this.update();\n }.bind(this);\n\n if(this.container) {\n // If chartist was already initialized in this container we are detaching all event listeners first\n if(this.container.\_\_chartist\_\_) {\n this.container.\_\_chartist\_\_.detach();\n }\n\n this.container.\_\_chartist\_\_ = this;\n }\n\n // Using event loop for first draw to make it possible to register event listeners in the same call stack where\n // the chart was created.\n this.initializeTimeoutId = setTimeout(initialize.bind(this), 0);\n }\n\n // Creating the chart base class\n Chartist.Base = Chartist.Class.extend({\n constructor: Base,\n optionsProvider: undefined,\n container: undefined,\n svg: undefined,\n eventEmitter: undefined,\n createChart: function() {\n throw new Error('Base chart type can\\'t be instantiated!');\n },\n update: update,\n detach: detach,\n on: on,\n off: off,\n version: Chartist.version,\n supportsForeignObject: false\n });\n\n}(window, document, Chartist));\n;/\*\*\n \* Chartist SVG module for simple SVG DOM abstraction\n \*\n \* @module Chartist.Svg\n \*/\n/\* global Chartist \*/\n(function(window, document, Chartist) {\n 'use strict';\n\n /\*\*\n \* Chartist.Svg creates a new SVG object wrapper with a starting element. You can use the wrapper to fluently create sub-elements and modify them.\n \*\n \* @memberof Chartist.Svg\n \* @constructor\n \* @param {String|Element} name The name of the SVG element to create or an SVG dom element which should be wrapped into Chartist.Svg\n \* @param {Object} attributes An object with properties that will be added as attributes to the SVG element that is created. Attributes with undefined values will not be added.\n \* @param {String} className This class or class list will be added to the SVG element\n \* @param {Object} parent The parent SVG wrapper object where this newly created wrapper and it's element will be attached to as child\n \* @param {Boolean} insertFirst If this param is set to true in conjunction with a parent element the newly created element will be added as first child element in the parent element\n \*/\n function Svg(name, attributes, className, parent, insertFirst) {\n // If Svg is getting called with an SVG element we just return the wrapper\n if(name instanceof Element) {\n this.\_node = name;\n } else {\n this.\_node = document.createElementNS(Chartist.namespaces.svg, name);\n\n // If this is an SVG element created then custom namespace\n if(name === 'svg') {\n this.attr({\n 'xmlns:ct': Chartist.namespaces.ct\n });\n }\n }\n\n if(attributes) {\n this.attr(attributes);\n }\n\n if(className) {\n this.addClass(className);\n }\n\n if(parent) {\n if (insertFirst && parent.\_node.firstChild) {\n parent.\_node.insertBefore(this.\_node, parent.\_node.firstChild);\n } else {\n parent.\_node.appendChild(this.\_node);\n }\n }\n }\n\n /\*\*\n \* Set attributes on the current SVG element of the wrapper you're currently working on.\n \*\n \* @memberof Chartist.Svg\n \* @param {Object|String} attributes An object with properties that will be added as attributes to the SVG element that is created. Attributes with undefined values will not be added. If this parameter is a String then the function is used as a getter and will return the attribute value.\n \* @param {String} [ns] If specified, the attribute will be obtained using getAttributeNs. In order to write namepsaced attributes you can use the namespace:attribute notation within the attributes object.\n \* @return {Object|String} The current wrapper object will be returned so it can be used for chaining or the attribute value if used as getter function.\n \*/\n function attr(attributes, ns) {\n if(typeof attributes === 'string') {\n if(ns) {\n return this.\_node.getAttributeNS(ns, attributes);\n } else {\n return this.\_node.getAttribute(attributes);\n }\n }\n\n Object.keys(attributes).forEach(function(key) {\n // If the attribute value is undefined we can skip this one\n if(attributes[key] === undefined) {\n return;\n }\n\n if (key.indexOf(':') !== -1) {\n var namespacedAttribute = key.split(':');\n this.\_node.setAttributeNS(Chartist.namespaces[namespacedAttribute[0]], key, attributes[key]);\n } else {\n this.\_node.setAttribute(key, attributes[key]);\n }\n }.bind(this));\n\n return this;\n }\n\n /\*\*\n \* Create a new SVG element whose wrapper object will be selected for further operations. This way you can also create nested groups easily.\n \*\n \* @memberof Chartist.Svg\n \* @param {String} name The name of the SVG element that should be created as child element of the currently selected element wrapper\n \* @param {Object} [attributes] An object with properties that will be added as attributes to the SVG element that is created. Attributes with undefined values will not be added.\n \* @param {String} [className] This class or class list will be added to the SVG element\n \* @param {Boolean} [insertFirst] If this param is set to true in conjunction with a parent element the newly created element will be added as first child element in the parent element\n \* @return {Chartist.Svg} Returns a Chartist.Svg wrapper object that can be used to modify the containing SVG data\n \*/\n function elem(name, attributes, className, insertFirst) {\n return new Chartist.Svg(name, attributes, className, this, insertFirst);\n }\n\n /\*\*\n \* Returns the parent Chartist.SVG wrapper object\n \*\n \* @memberof Chartist.Svg\n \* @return {Chartist.Svg} Returns a Chartist.Svg wrapper around the parent node of the current node. If the parent node is not existing or it's not an SVG node then this function will return null.\n \*/\n function parent() {\n return this.\_node.parentNode instanceof SVGElement ? new Chartist.Svg(this.\_node.parentNode) : null;\n }\n\n /\*\*\n \* This method returns a Chartist.Svg wrapper around the root SVG element of the current tree.\n \*\n \* @memberof Chartist.Svg\n \* @return {Chartist.Svg} The root SVG element wrapped in a Chartist.Svg element\n \*/\n function root() {\n var node = this.\_node;\n while(node.nodeName !== 'svg') {\n node = node.parentNode;\n }\n return new Chartist.Svg(node);\n }\n\n /\*\*\n \* Find the first child SVG element of the current element that matches a CSS selector. The returned object is a Chartist.Svg wrapper.\n \*\n \* @memberof Chartist.Svg\n \* @param {String} selector A CSS selector that is used to query for child SVG elements\n \* @return {Chartist.Svg} The SVG wrapper for the element found or null if no element was found\n \*/\n function querySelector(selector) {\n var foundNode = this.\_node.querySelector(selector);\n return foundNode ? new Chartist.Svg(foundNode) : null;\n }\n\n /\*\*\n \* Find the all child SVG elements of the current element that match a CSS selector. The returned object is a Chartist.Svg.List wrapper.\n \*\n \* @memberof Chartist.Svg\n \* @param {String} selector A CSS selector that is used to query for child SVG elements\n \* @return {Chartist.Svg.List} The SVG wrapper list for the element found or null if no element was found\n \*/\n function querySelectorAll(selector) {\n var foundNodes = this.\_node.querySelectorAll(selector);\n return foundNodes.length ? new Chartist.Svg.List(foundNodes) : null;\n }\n\n /\*\*\n \* Returns the underlying SVG node for the current element.\n \*\n \* @memberof Chartist.Svg\n \* @returns {Node}\n \*/\n function getNode() {\n return this.\_node;\n }\n\n /\*\*\n \* This method creates a foreignObject (see https://developer.mozilla.org/en-US/docs/Web/SVG/Element/foreignObject) that allows to embed HTML content into a SVG graphic. With the help of foreignObjects you can enable the usage of regular HTML elements inside of SVG where they are subject for SVG positioning and transformation but the Browser will use the HTML rendering capabilities for the containing DOM.\n \*\n \* @memberof Chartist.Svg\n \* @param {Node|String} content The DOM Node, or HTML string that will be converted to a DOM Node, that is then placed into and wrapped by the foreignObject\n \* @param {String} [attributes] An object with properties that will be added as attributes to the foreignObject element that is created. Attributes with undefined values will not be added.\n \* @param {String} [className] This class or class list will be added to the SVG element\n \* @param {Boolean} [insertFirst] Specifies if the foreignObject should be inserted as first child\n \* @return {Chartist.Svg} New wrapper object that wraps the foreignObject element\n \*/\n function foreignObject(content, attributes, className, insertFirst) {\n // If content is string then we convert it to DOM\n // TODO: Handle case where content is not a string nor a DOM Node\n if(typeof content === 'string') {\n var container = document.createElement('div');\n container.innerHTML = content;\n content = container.firstChild;\n }\n\n // Adding namespace to content element\n content.setAttribute('xmlns', Chartist.namespaces.xmlns);\n\n // Creating the foreignObject without required extension attribute (as described here\n // http://www.w3.org/TR/SVG/extend.html#ForeignObjectElement)\n var fnObj = this.elem('foreignObject', attributes, className, insertFirst);\n\n // Add content to foreignObjectElement\n fnObj.\_node.appendChild(content);\n\n return fnObj;\n }\n\n /\*\*\n \* This method adds a new text element to the current Chartist.Svg wrapper.\n \*\n \* @memberof Chartist.Svg\n \* @param {String} t The text that should be added to the text element that is created\n \* @return {Chartist.Svg} The same wrapper object that was used to add the newly created element\n \*/\n function text(t) {\n this.\_node.appendChild(document.createTextNode(t));\n return this;\n }\n\n /\*\*\n \* This method will clear all child nodes of the current wrapper object.\n \*\n \* @memberof Chartist.Svg\n \* @return {Chartist.Svg} The same wrapper object that got emptied\n \*/\n function empty() {\n while (this.\_node.firstChild) {\n this.\_node.removeChild(this.\_node.firstChild);\n }\n\n return this;\n }\n\n /\*\*\n \* This method will cause the current wrapper to remove itself from its parent wrapper. Use this method if you'd like to get rid of an element in a given DOM structure.\n \*\n \* @memberof Chartist.Svg\n \* @return {Chartist.Svg} The parent wrapper object of the element that got removed\n \*/\n function remove() {\n this.\_node.parentNode.removeChild(this.\_node);\n return this.parent();\n }\n\n /\*\*\n \* This method will replace the element with a new element that can be created outside of the current DOM.\n \*\n \* @memberof Chartist.Svg\n \* @param {Chartist.Svg} newElement The new Chartist.Svg object that will be used to replace the current wrapper object\n \* @return {Chartist.Svg} The wrapper of the new element\n \*/\n function replace(newElement) {\n this.\_node.parentNode.replaceChild(newElement.\_node, this.\_node);\n return newElement;\n }\n\n /\*\*\n \* This method will append an element to the current element as a child.\n \*\n \* @memberof Chartist.Svg\n \* @param {Chartist.Svg} element The Chartist.Svg element that should be added as a child\n \* @param {Boolean} [insertFirst] Specifies if the element should be inserted as first child\n \* @return {Chartist.Svg} The wrapper of the appended object\n \*/\n function append(element, insertFirst) {\n if(insertFirst && this.\_node.firstChild) {\n this.\_node.insertBefore(element.\_node, this.\_node.firstChild);\n } else {\n this.\_node.appendChild(element.\_node);\n }\n\n return this;\n }\n\n /\*\*\n \* Returns an array of class names that are attached to the current wrapper element. This method can not be chained further.\n \*\n \* @memberof Chartist.Svg\n \* @return {Array} A list of classes or an empty array if there are no classes on the current element\n \*/\n function classes() {\n return this.\_node.getAttribute('class') ? this.\_node.getAttribute('class').trim().split(/\\s+/) : [];\n }\n\n /\*\*\n \* Adds one or a space separated list of classes to the current element and ensures the classes are only existing once.\n \*\n \* @memberof Chartist.Svg\n \* @param {String} names A white space separated list of class names\n \* @return {Chartist.Svg} The wrapper of the current element\n \*/\n function addClass(names) {\n this.\_node.setAttribute('class',\n this.classes(this.\_node)\n .concat(names.trim().split(/\\s+/))\n .filter(function(elem, pos, self) {\n return self.indexOf(elem) === pos;\n }).join(' ')\n );\n\n return this;\n }\n\n /\*\*\n \* Removes one or a space separated list of classes from the current element.\n \*\n \* @memberof Chartist.Svg\n \* @param {String} names A white space separated list of class names\n \* @return {Chartist.Svg} The wrapper of the current element\n \*/\n function removeClass(names) {\n var removedClasses = names.trim().split(/\\s+/);\n\n this.\_node.setAttribute('class', this.classes(this.\_node).filter(function(name) {\n return removedClasses.indexOf(name) === -1;\n }).join(' '));\n\n return this;\n }\n\n /\*\*\n \* Removes all classes from the current element.\n \*\n \* @memberof Chartist.Svg\n \* @return {Chartist.Svg} The wrapper of the current element\n \*/\n function removeAllClasses() {\n this.\_node.setAttribute('class', '');\n\n return this;\n }\n\n /\*\*\n \* Get element height using `getBoundingClientRect`\n \*\n \* @memberof Chartist.Svg\n \* @return {Number} The elements height in pixels\n \*/\n function height() {\n return this.\_node.getBoundingClientRect().height;\n }\n\n /\*\*\n \* Get element width using `getBoundingClientRect`\n \*\n \* @memberof Chartist.Core\n \* @return {Number} The elements width in pixels\n \*/\n function width() {\n return this.\_node.getBoundingClientRect().width;\n }\n\n /\*\*\n \* The animate function lets you animate the current element with SMIL animations. You can add animations for multiple attributes at the same time by using an animation definition object. This object should contain SMIL animation attributes. Please refer to http://www.w3.org/TR/SVG/animate.html for a detailed specification about the available animation attributes. Additionally an easing property can be passed in the animation definition object. This can be a string with a name of an easing function in `Chartist.Svg.Easing` or an array with four numbers specifying a cubic Bézier curve.\n \* \*\*An animations object could look like this:\*\*\n \* ```javascript\n \* element.animate({\n \* opacity: {\n \* dur: 1000,\n \* from: 0,\n \* to: 1\n \* },\n \* x1: {\n \* dur: '1000ms',\n \* from: 100,\n \* to: 200,\n \* easing: 'easeOutQuart'\n \* },\n \* y1: {\n \* dur: '2s',\n \* from: 0,\n \* to: 100\n \* }\n \* });\n \* ```\n \* \*\*Automatic unit conversion\*\*\n \* For the `dur` and the `begin` animate attribute you can also omit a unit by passing a number. The number will automatically be converted to milli seconds.\n \* \*\*Guided mode\*\*\n \* The default behavior of SMIL animations with offset using the `begin` attribute is that the attribute will keep it's original value until the animation starts. Mostly this behavior is not desired as you'd like to have your element attributes already initialized with the animation `from` value even before the animation starts. Also if you don't specify `fill=\"freeze\"` on an animate element or if you delete the animation after it's done (which is done in guided mode) the attribute will switch back to the initial value. This behavior is also not desired when performing simple one-time animations. For one-time animations you'd want to trigger animations immediately instead of relative to the document begin time. That's why in guided mode Chartist.Svg will also use the `begin` property to schedule a timeout and manually start the animation after the timeout. If you're using multiple SMIL definition objects for an attribute (in an array), guided mode will be disabled for this attribute, even if you explicitly enabled it.\n \* If guided mode is enabled the following behavior is added:\n \* - Before the animation starts (even when delayed with `begin`) the animated attribute will be set already to the `from` value of the animation\n \* - `begin` is explicitly set to `indefinite` so it can be started manually without relying on document begin time (creation)\n \* - The animate element will be forced to use `fill=\"freeze\"`\n \* - The animation will be triggered with `beginElement()` in a timeout where `begin` of the definition object is interpreted in milli seconds. If no `begin` was specified the timeout is triggered immediately.\n \* - After the animation the element attribute value will be set to the `to` value of the animation\n \* - The animate element is deleted from the DOM\n \*\n \* @memberof Chartist.Svg\n \* @param {Object} animations An animations object where the property keys are the attributes you'd like to animate. The properties should be objects again that contain the SMIL animation attributes (usually begin, dur, from, and to). The property begin and dur is auto converted (see Automatic unit conversion). You can also schedule multiple animations for the same attribute by passing an Array of SMIL definition objects. Attributes that contain an array of SMIL definition objects will not be executed in guided mode.\n \* @param {Boolean} guided Specify if guided mode should be activated for this animation (see Guided mode). If not otherwise specified, guided mode will be activated.\n \* @param {Object} eventEmitter If specified, this event emitter will be notified when an animation starts or ends.\n \* @return {Chartist.Svg} The current element where the animation was added\n \*/\n function animate(animations, guided, eventEmitter) {\n if(guided === undefined) {\n guided = true;\n }\n\n Object.keys(animations).forEach(function createAnimateForAttributes(attribute) {\n\n function createAnimate(animationDefinition, guided) {\n var attributeProperties = {},\n animate,\n timeout,\n easing;\n\n // Check if an easing is specified in the definition object and delete it from the object as it will not\n // be part of the animate element attributes.\n if(animationDefinition.easing) {\n // If already an easing Bézier curve array we take it or we lookup a easing array in the Easing object\n easing = animationDefinition.easing instanceof Array ?\n animationDefinition.easing :\n Chartist.Svg.Easing[animationDefinition.easing];\n delete animationDefinition.easing;\n }\n\n // If numeric dur or begin was provided we assume milli seconds\n animationDefinition.begin = Chartist.ensureUnit(animationDefinition.begin, 'ms');\n animationDefinition.dur = Chartist.ensureUnit(animationDefinition.dur, 'ms');\n\n if(easing) {\n animationDefinition.calcMode = 'spline';\n animationDefinition.keySplines = easing.join(' ');\n animationDefinition.keyTimes = '0;1';\n }\n\n // Adding \"fill: freeze\" if we are in guided mode and set initial attribute values\n if(guided) {\n animationDefinition.fill = 'freeze';\n // Animated property on our element should already be set to the animation from value in guided mode\n attributeProperties[attribute] = animationDefinition.from;\n this.attr(attributeProperties);\n\n // In guided mode we also set begin to indefinite so we can trigger the start manually and put the begin\n // which needs to be in ms aside\n timeout = Chartist.quantity(animationDefinition.begin || 0).value;\n animationDefinition.begin = 'indefinite';\n }\n\n animate = this.elem('animate', Chartist.extend({\n attributeName: attribute\n }, animationDefinition));\n\n if(guided) {\n // If guided we take the value that was put aside in timeout and trigger the animation manually with a timeout\n setTimeout(function() {\n // If beginElement fails we set the animated attribute to the end position and remove the animate element\n // This happens if the SMIL ElementTimeControl interface is not supported or any other problems occured in\n // the browser. (Currently FF 34 does not support animate elements in foreignObjects)\n try {\n animate.\_node.beginElement();\n } catch(err) {\n // Set animated attribute to current animated value\n attributeProperties[attribute] = animationDefinition.to;\n this.attr(attributeProperties);\n // Remove the animate element as it's no longer required\n animate.remove();\n }\n }.bind(this), timeout);\n }\n\n if(eventEmitter) {\n animate.\_node.addEventListener('beginEvent', function handleBeginEvent() {\n eventEmitter.emit('animationBegin', {\n element: this,\n animate: animate.\_node,\n params: animationDefinition\n });\n }.bind(this));\n }\n\n animate.\_node.addEventListener('endEvent', function handleEndEvent() {\n if(eventEmitter) {\n eventEmitter.emit('animationEnd', {\n element: this,\n animate: animate.\_node,\n params: animationDefinition\n });\n }\n\n if(guided) {\n // Set animated attribute to current animated value\n attributeProperties[attribute] = animationDefinition.to;\n this.attr(attributeProperties);\n // Remove the animate element as it's no longer required\n animate.remove();\n }\n }.bind(this));\n }\n\n // If current attribute is an array of definition objects we create an animate for each and disable guided mode\n if(animations[attribute] instanceof Array) {\n animations[attribute].forEach(function(animationDefinition) {\n createAnimate.bind(this)(animationDefinition, false);\n }.bind(this));\n } else {\n createAnimate.bind(this)(animations[attribute], guided);\n }\n\n }.bind(this));\n\n return this;\n }\n\n Chartist.Svg = Chartist.Class.extend({\n constructor: Svg,\n attr: attr,\n elem: elem,\n parent: parent,\n root: root,\n querySelector: querySelector,\n querySelectorAll: querySelectorAll,\n getNode: getNode,\n foreignObject: foreignObject,\n text: text,\n empty: empty,\n remove: remove,\n replace: replace,\n append: append,\n classes: classes,\n addClass: addClass,\n removeClass: removeClass,\n removeAllClasses: removeAllClasses,\n height: height,\n width: width,\n animate: animate\n });\n\n /\*\*\n \* This method checks for support of a given SVG feature like Extensibility, SVG-animation or the like. Check http://www.w3.org/TR/SVG11/feature for a detailed list.\n \*\n \* @memberof Chartist.Svg\n \* @param {String} feature The SVG 1.1 feature that should be checked for support.\n \* @return {Boolean} True of false if the feature is supported or not\n \*/\n Chartist.Svg.isSupported = function(feature) {\n return document.implementation.hasFeature('http://www.w3.org/TR/SVG11/feature#' + feature, '1.1');\n };\n\n /\*\*\n \* This Object contains some standard easing cubic bezier curves. Then can be used with their name in the `Chartist.Svg.animate`. You can also extend the list and use your own name in the `animate` function. Click the show code button to see the available bezier functions.\n \*\n \* @memberof Chartist.Svg\n \*/\n var easingCubicBeziers = {\n easeInSine: [0.47, 0, 0.745, 0.715],\n easeOutSine: [0.39, 0.575, 0.565, 1],\n easeInOutSine: [0.445, 0.05, 0.55, 0.95],\n easeInQuad: [0.55, 0.085, 0.68, 0.53],\n easeOutQuad: [0.25, 0.46, 0.45, 0.94],\n easeInOutQuad: [0.455, 0.03, 0.515, 0.955],\n easeInCubic: [0.55, 0.055, 0.675, 0.19],\n easeOutCubic: [0.215, 0.61, 0.355, 1],\n easeInOutCubic: [0.645, 0.045, 0.355, 1],\n easeInQuart: [0.895, 0.03, 0.685, 0.22],\n easeOutQuart: [0.165, 0.84, 0.44, 1],\n easeInOutQuart: [0.77, 0, 0.175, 1],\n easeInQuint: [0.755, 0.05, 0.855, 0.06],\n easeOutQuint: [0.23, 1, 0.32, 1],\n easeInOutQuint: [0.86, 0, 0.07, 1],\n easeInExpo: [0.95, 0.05, 0.795, 0.035],\n easeOutExpo: [0.19, 1, 0.22, 1],\n easeInOutExpo: [1, 0, 0, 1],\n easeInCirc: [0.6, 0.04, 0.98, 0.335],\n easeOutCirc: [0.075, 0.82, 0.165, 1],\n easeInOutCirc: [0.785, 0.135, 0.15, 0.86],\n easeInBack: [0.6, -0.28, 0.735, 0.045],\n easeOutBack: [0.175, 0.885, 0.32, 1.275],\n easeInOutBack: [0.68, -0.55, 0.265, 1.55]\n };\n\n Chartist.Svg.Easing = easingCubicBeziers;\n\n /\*\*\n \* This helper class is to wrap multiple `Chartist.Svg` elements into a list where you can call the `Chartist.Svg` functions on all elements in the list with one call. This is helpful when you'd like to perform calls with `Chartist.Svg` on multiple elements.\n \* An instance of this class is also returned by `Chartist.Svg.querySelectorAll`.\n \*\n \* @memberof Chartist.Svg\n \* @param {Array<Node>|NodeList} nodeList An Array of SVG DOM nodes or a SVG DOM NodeList (as returned by document.querySelectorAll)\n \* @constructor\n \*/\n function SvgList(nodeList) {\n var list = this;\n\n this.svgElements = [];\n for(var i = 0; i < nodeList.length; i++) {\n this.svgElements.push(new Chartist.Svg(nodeList[i]));\n }\n\n // Add delegation methods for Chartist.Svg\n Object.keys(Chartist.Svg.prototype).filter(function(prototypeProperty) {\n return ['constructor',\n 'parent',\n 'querySelector',\n 'querySelectorAll',\n 'replace',\n 'append',\n 'classes',\n 'height',\n 'width'].indexOf(prototypeProperty) === -1;\n }).forEach(function(prototypeProperty) {\n list[prototypeProperty] = function() {\n var args = Array.prototype.slice.call(arguments, 0);\n list.svgElements.forEach(function(element) {\n Chartist.Svg.prototype[prototypeProperty].apply(element, args);\n });\n return list;\n };\n });\n }\n\n Chartist.Svg.List = Chartist.Class.extend({\n constructor: SvgList\n });\n}(window, document, Chartist));\n;/\*\*\n \* Chartist SVG path module for SVG path description creation and modification.\n \*\n \* @module Chartist.Svg.Path\n \*/\n/\* global Chartist \*/\n(function(window, document, Chartist) {\n 'use strict';\n\n /\*\*\n \* Contains the descriptors of supported element types in a SVG path. Currently only move, line and curve are supported.\n \*\n \* @memberof Chartist.Svg.Path\n \* @type {Object}\n \*/\n var elementDescriptions = {\n m: ['x', 'y'],\n l: ['x', 'y'],\n c: ['x1', 'y1', 'x2', 'y2', 'x', 'y'],\n a: ['rx', 'ry', 'xAr', 'lAf', 'sf', 'x', 'y']\n };\n\n /\*\*\n \* Default options for newly created SVG path objects.\n \*\n \* @memberof Chartist.Svg.Path\n \* @type {Object}\n \*/\n var defaultOptions = {\n // The accuracy in digit count after the decimal point. This will be used to round numbers in the SVG path. If this option is set to false then no rounding will be performed.\n accuracy: 3\n };\n\n function element(command, params, pathElements, pos, relative, data) {\n var pathElement = Chartist.extend({\n command: relative ? command.toLowerCase() : command.toUpperCase()\n }, params, data ? { data: data } : {} );\n\n pathElements.splice(pos, 0, pathElement);\n }\n\n function forEachParam(pathElements, cb) {\n pathElements.forEach(function(pathElement, pathElementIndex) {\n elementDescriptions[pathElement.command.toLowerCase()].forEach(function(paramName, paramIndex) {\n cb(pathElement, paramName, pathElementIndex, paramIndex, pathElements);\n });\n });\n }\n\n /\*\*\n \* Used to construct a new path object.\n \*\n \* @memberof Chartist.Svg.Path\n \* @param {Boolean} close If set to true then this path will be closed when stringified (with a Z at the end)\n \* @param {Object} options Options object that overrides the default objects. See default options for more details.\n \* @constructor\n \*/\n function SvgPath(close, options) {\n this.pathElements = [];\n this.pos = 0;\n this.close = close;\n this.options = Chartist.extend({}, defaultOptions, options);\n }\n\n /\*\*\n \* Gets or sets the current position (cursor) inside of the path. You can move around the cursor freely but limited to 0 or the count of existing elements. All modifications with element functions will insert new elements at the position of this cursor.\n \*\n \* @memberof Chartist.Svg.Path\n \* @param {Number} [pos] If a number is passed then the cursor is set to this position in the path element array.\n \* @return {Chartist.Svg.Path|Number} If the position parameter was passed then the return value will be the path object for easy call chaining. If no position parameter was passed then the current position is returned.\n \*/\n function position(pos) {\n if(pos !== undefined) {\n this.pos = Math.max(0, Math.min(this.pathElements.length, pos));\n return this;\n } else {\n return this.pos;\n }\n }\n\n /\*\*\n \* Removes elements from the path starting at the current position.\n \*\n \* @memberof Chartist.Svg.Path\n \* @param {Number} count Number of path elements that should be removed from the current position.\n \* @return {Chartist.Svg.Path} The current path object for easy call chaining.\n \*/\n function remove(count) {\n this.pathElements.splice(this.pos, count);\n return this;\n }\n\n /\*\*\n \* Use this function to add a new move SVG path element.\n \*\n \* @memberof Chartist.Svg.Path\n \* @param {Number} x The x coordinate for the move element.\n \* @param {Number} y The y coordinate for the move element.\n \* @param {Boolean} [relative] If set to true the move element will be created with relative coordinates (lowercase letter)\n \* @param {\*} [data] Any data that should be stored with the element object that will be accessible in pathElement\n \* @return {Chartist.Svg.Path} The current path object for easy call chaining.\n \*/\n function move(x, y, relative, data) {\n element('M', {\n x: +x,\n y: +y\n }, this.pathElements, this.pos++, relative, data);\n return this;\n }\n\n /\*\*\n \* Use this function to add a new line SVG path element.\n \*\n \* @memberof Chartist.Svg.Path\n \* @param {Number} x The x coordinate for the line element.\n \* @param {Number} y The y coordinate for the line element.\n \* @param {Boolean} [relative] If set to true the line element will be created with relative coordinates (lowercase letter)\n \* @param {\*} [data] Any data that should be stored with the element object that will be accessible in pathElement\n \* @return {Chartist.Svg.Path} The current path object for easy call chaining.\n \*/\n function line(x, y, relative, data) {\n element('L', {\n x: +x,\n y: +y\n }, this.pathElements, this.pos++, relative, data);\n return this;\n }\n\n /\*\*\n \* Use this function to add a new curve SVG path element.\n \*\n \* @memberof Chartist.Svg.Path\n \* @param {Number} x1 The x coordinate for the first control point of the bezier curve.\n \* @param {Number} y1 The y coordinate for the first control point of the bezier curve.\n \* @param {Number} x2 The x coordinate for the second control point of the bezier curve.\n \* @param {Number} y2 The y coordinate for the second control point of the bezier curve.\n \* @param {Number} x The x coordinate for the target point of the curve element.\n \* @param {Number} y The y coordinate for the target point of the curve element.\n \* @param {Boolean} [relative] If set to true the curve element will be created with relative coordinates (lowercase letter)\n \* @param {\*} [data] Any data that should be stored with the element object that will be accessible in pathElement\n \* @return {Chartist.Svg.Path} The current path object for easy call chaining.\n \*/\n function curve(x1, y1, x2, y2, x, y, relative, data) {\n element('C', {\n x1: +x1,\n y1: +y1,\n x2: +x2,\n y2: +y2,\n x: +x,\n y: +y\n }, this.pathElements, this.pos++, relative, data);\n return this;\n }\n\n /\*\*\n \* Use this function to add a new non-bezier curve SVG path element.\n \*\n \* @memberof Chartist.Svg.Path\n \* @param {Number} rx The radius to be used for the x-axis of the arc.\n \* @param {Number} ry The radius to be used for the y-axis of the arc.\n \* @param {Number} xAr Defines the orientation of the arc\n \* @param {Number} lAf Large arc flag\n \* @param {Number} sf Sweep flag\n \* @param {Number} x The x coordinate for the target point of the curve element.\n \* @param {Number} y The y coordinate for the target point of the curve element.\n \* @param {Boolean} [relative] If set to true the curve element will be created with relative coordinates (lowercase letter)\n \* @param {\*} [data] Any data that should be stored with the element object that will be accessible in pathElement\n \* @return {Chartist.Svg.Path} The current path object for easy call chaining.\n \*/\n function arc(rx, ry, xAr, lAf, sf, x, y, relative, data) {\n element('A', {\n rx: +rx,\n ry: +ry,\n xAr: +xAr,\n lAf: +lAf,\n sf: +sf,\n x: +x,\n y: +y\n }, this.pathElements, this.pos++, relative, data);\n return this;\n }\n\n /\*\*\n \* Parses an SVG path seen in the d attribute of path elements, and inserts the parsed elements into the existing path object at the current cursor position. Any closing path indicators (Z at the end of the path) will be ignored by the parser as this is provided by the close option in the options of the path object.\n \*\n \* @memberof Chartist.Svg.Path\n \* @param {String} path Any SVG path that contains move (m), line (l) or curve (c) components.\n \* @return {Chartist.Svg.Path} The current path object for easy call chaining.\n \*/\n function parse(path) {\n // Parsing the SVG path string into an array of arrays [['M', '10', '10'], ['L', '100', '100']]\n var chunks = path.replace(/([A-Za-z])([0-9])/g, '$1 $2')\n .replace(/([0-9])([A-Za-z])/g, '$1 $2')\n .split(/[\\s,]+/)\n .reduce(function(result, element) {\n if(element.match(/[A-Za-z]/)) {\n result.push([]);\n }\n\n result[result.length - 1].push(element);\n return result;\n }, []);\n\n // If this is a closed path we remove the Z at the end because this is determined by the close option\n if(chunks[chunks.length - 1][0].toUpperCase() === 'Z') {\n chunks.pop();\n }\n\n // Using svgPathElementDescriptions to map raw path arrays into objects that contain the command and the parameters\n // For example {command: 'M', x: '10', y: '10'}\n var elements = chunks.map(function(chunk) {\n var command = chunk.shift(),\n description = elementDescriptions[command.toLowerCase()];\n\n return Chartist.extend({\n command: command\n }, description.reduce(function(result, paramName, index) {\n result[paramName] = +chunk[index];\n return result;\n }, {}));\n });\n\n // Preparing a splice call with the elements array as var arg params and insert the parsed elements at the current position\n var spliceArgs = [this.pos, 0];\n Array.prototype.push.apply(spliceArgs, elements);\n Array.prototype.splice.apply(this.pathElements, spliceArgs);\n // Increase the internal position by the element count\n this.pos += elements.length;\n\n return this;\n }\n\n /\*\*\n \* This function renders to current SVG path object into a final SVG string that can be used in the d attribute of SVG path elements. It uses the accuracy option to round big decimals. If the close parameter was set in the constructor of this path object then a path closing Z will be appended to the output string.\n \*\n \* @memberof Chartist.Svg.Path\n \* @return {String}\n \*/\n function stringify() {\n var accuracyMultiplier = Math.pow(10, this.options.accuracy);\n\n return this.pathElements.reduce(function(path, pathElement) {\n var params = elementDescriptions[pathElement.command.toLowerCase()].map(function(paramName) {\n return this.options.accuracy ?\n (Math.round(pathElement[paramName] \* accuracyMultiplier) / accuracyMultiplier) :\n pathElement[paramName];\n }.bind(this));\n\n return path + pathElement.command + params.join(',');\n }.bind(this), '') + (this.close ? 'Z' : '');\n }\n\n /\*\*\n \* Scales all elements in the current SVG path object. There is an individual parameter for each coordinate. Scaling will also be done for control points of curves, affecting the given coordinate.\n \*\n \* @memberof Chartist.Svg.Path\n \* @param {Number} x The number which will be used to scale the x, x1 and x2 of all path elements.\n \* @param {Number} y The number which will be used to scale the y, y1 and y2 of all path elements.\n \* @return {Chartist.Svg.Path} The current path object for easy call chaining.\n \*/\n function scale(x, y) {\n forEachParam(this.pathElements, function(pathElement, paramName) {\n pathElement[paramName] \*= paramName[0] === 'x' ? x : y;\n });\n return this;\n }\n\n /\*\*\n \* Translates all elements in the current SVG path object. The translation is relative and there is an individual parameter for each coordinate. Translation will also be done for control points of curves, affecting the given coordinate.\n \*\n \* @memberof Chartist.Svg.Path\n \* @param {Number} x The number which will be used to translate the x, x1 and x2 of all path elements.\n \* @param {Number} y The number which will be used to translate the y, y1 and y2 of all path elements.\n \* @return {Chartist.Svg.Path} The current path object for easy call chaining.\n \*/\n function translate(x, y) {\n forEachParam(this.pathElements, function(pathElement, paramName) {\n pathElement[paramName] += paramName[0] === 'x' ? x : y;\n });\n return this;\n }\n\n /\*\*\n \* This function will run over all existing path elements and then loop over their attributes. The callback function will be called for every path element attribute that exists in the current path.\n \* The method signature of the callback function looks like this:\n \* ```javascript\n \* function(pathElement, paramName, pathElementIndex, paramIndex, pathElements)\n \* ```\n \* If something else than undefined is returned by the callback function, this value will be used to replace the old value. This allows you to build custom transformations of path objects that can't be achieved using the basic transformation functions scale and translate.\n \*\n \* @memberof Chartist.Svg.Path\n \* @param {Function} transformFnc The callback function for the transformation. Check the signature in the function description.\n \* @return {Chartist.Svg.Path} The current path object for easy call chaining.\n \*/\n function transform(transformFnc) {\n forEachParam(this.pathElements, function(pathElement, paramName, pathElementIndex, paramIndex, pathElements) {\n var transformed = transformFnc(pathElement, paramName, pathElementIndex, paramIndex, pathElements);\n if(transformed || transformed === 0) {\n pathElement[paramName] = transformed;\n }\n });\n return this;\n }\n\n /\*\*\n \* This function clones a whole path object with all its properties. This is a deep clone and path element objects will also be cloned.\n \*\n \* @memberof Chartist.Svg.Path\n \* @param {Boolean} [close] Optional option to set the new cloned path to closed. If not specified or false, the original path close option will be used.\n \* @return {Chartist.Svg.Path}\n \*/\n function clone(close) {\n var c = new Chartist.Svg.Path(close || this.close);\n c.pos = this.pos;\n c.pathElements = this.pathElements.slice().map(function cloneElements(pathElement) {\n return Chartist.extend({}, pathElement);\n });\n c.options = Chartist.extend({}, this.options);\n return c;\n }\n\n /\*\*\n \* Split a Svg.Path object by a specific command in the path chain. The path chain will be split and an array of newly created paths objects will be returned. This is useful if you'd like to split an SVG path by it's move commands, for example, in order to isolate chunks of drawings.\n \*\n \* @memberof Chartist.Svg.Path\n \* @param {String} command The command you'd like to use to split the path\n \* @return {Array<Chartist.Svg.Path>}\n \*/\n function splitByCommand(command) {\n var split = [\n new Chartist.Svg.Path()\n ];\n\n this.pathElements.forEach(function(pathElement) {\n if(pathElement.command === command.toUpperCase() && split[split.length - 1].pathElements.length !== 0) {\n split.push(new Chartist.Svg.Path());\n }\n\n split[split.length - 1].pathElements.push(pathElement);\n });\n\n return split;\n }\n\n /\*\*\n \* This static function on `Chartist.Svg.Path` is joining multiple paths together into one paths.\n \*\n \* @memberof Chartist.Svg.Path\n \* @param {Array<Chartist.Svg.Path>} paths A list of paths to be joined together. The order is important.\n \* @param {boolean} close If the newly created path should be a closed path\n \* @param {Object} options Path options for the newly created path.\n \* @return {Chartist.Svg.Path}\n \*/\n\n function join(paths, close, options) {\n var joinedPath = new Chartist.Svg.Path(close, options);\n for(var i = 0; i < paths.length; i++) {\n var path = paths[i];\n for(var j = 0; j < path.pathElements.length; j++) {\n joinedPath.pathElements.push(path.pathElements[j]);\n }\n }\n return joinedPath;\n }\n\n Chartist.Svg.Path = Chartist.Class.extend({\n constructor: SvgPath,\n position: position,\n remove: remove,\n move: move,\n line: line,\n curve: curve,\n arc: arc,\n scale: scale,\n translate: translate,\n transform: transform,\n parse: parse,\n stringify: stringify,\n clone: clone,\n splitByCommand: splitByCommand\n });\n\n Chartist.Svg.Path.elementDescriptions = elementDescriptions;\n Chartist.Svg.Path.join = join;\n}(window, document, Chartist));\n;/\* global Chartist \*/\n(function (window, document, Chartist) {\n 'use strict';\n\n var axisUnits = {\n x: {\n pos: 'x',\n len: 'width',\n dir: 'horizontal',\n rectStart: 'x1',\n rectEnd: 'x2',\n rectOffset: 'y2'\n },\n y: {\n pos: 'y',\n len: 'height',\n dir: 'vertical',\n rectStart: 'y2',\n rectEnd: 'y1',\n rectOffset: 'x1'\n }\n };\n\n function Axis(units, chartRect, ticks, options) {\n this.units = units;\n this.counterUnits = units === axisUnits.x ? axisUnits.y : axisUnits.x;\n this.chartRect = chartRect;\n this.axisLength = chartRect[units.rectEnd] - chartRect[units.rectStart];\n this.gridOffset = chartRect[units.rectOffset];\n this.ticks = ticks;\n this.options = options;\n }\n\n function createGridAndLabels(gridGroup, labelGroup, useForeignObject, chartOptions, eventEmitter) {\n var axisOptions = chartOptions['axis' + this.units.pos.toUpperCase()];\n var projectedValues = this.ticks.map(this.projectValue.bind(this));\n var labelValues = this.ticks.map(axisOptions.labelInterpolationFnc);\n\n projectedValues.forEach(function(projectedValue, index) {\n var labelOffset = {\n x: 0,\n y: 0\n };\n\n // TODO: Find better solution for solving this problem\n // Calculate how much space we have available for the label\n var labelLength;\n if(projectedValues[index + 1]) {\n // If we still have one label ahead, we can calculate the distance to the next tick / label\n labelLength = projectedValues[index + 1] - projectedValue;\n } else {\n // If we don't have a label ahead and we have only two labels in total, we just take the remaining distance to\n // on the whole axis length. We limit that to a minimum of 30 pixel, so that labels close to the border will\n // still be visible inside of the chart padding.\n labelLength = Math.max(this.axisLength - projectedValue, 30);\n }\n\n // Skip grid lines and labels where interpolated label values are falsey (execpt for 0)\n if(Chartist.isFalseyButZero(labelValues[index]) && labelValues[index] !== '') {\n return;\n }\n\n // Transform to global coordinates using the chartRect\n // We also need to set the label offset for the createLabel function\n if(this.units.pos === 'x') {\n projectedValue = this.chartRect.x1 + projectedValue;\n labelOffset.x = chartOptions.axisX.labelOffset.x;\n\n // If the labels should be positioned in start position (top side for vertical axis) we need to set a\n // different offset as for positioned with end (bottom)\n if(chartOptions.axisX.position === 'start') {\n labelOffset.y = this.chartRect.padding.top + chartOptions.axisX.labelOffset.y + (useForeignObject ? 5 : 20);\n } else {\n labelOffset.y = this.chartRect.y1 + chartOptions.axisX.labelOffset.y + (useForeignObject ? 5 : 20);\n }\n } else {\n projectedValue = this.chartRect.y1 - projectedValue;\n labelOffset.y = chartOptions.axisY.labelOffset.y - (useForeignObject ? labelLength : 0);\n\n // If the labels should be positioned in start position (left side for horizontal axis) we need to set a\n // different offset as for positioned with end (right side)\n if(chartOptions.axisY.position === 'start') {\n labelOffset.x = useForeignObject ? this.chartRect.padding.left + chartOptions.axisY.labelOffset.x : this.chartRect.x1 - 10;\n } else {\n labelOffset.x = this.chartRect.x2 + chartOptions.axisY.labelOffset.x + 10;\n }\n }\n\n if(axisOptions.showGrid) {\n Chartist.createGrid(projectedValue, index, this, this.gridOffset, this.chartRect[this.counterUnits.len](), gridGroup, [\n chartOptions.classNames.grid,\n chartOptions.classNames[this.units.dir]\n ], eventEmitter);\n }\n\n if(axisOptions.showLabel) {\n Chartist.createLabel(projectedValue, labelLength, index, labelValues, this, axisOptions.offset, labelOffset, labelGroup, [\n chartOptions.classNames.label,\n chartOptions.classNames[this.units.dir],\n (axisOptions.position === 'start' ? chartOptions.classNames[axisOptions.position] : chartOptions.classNames['end'])\n ], useForeignObject, eventEmitter);\n }\n }.bind(this));\n }\n\n Chartist.Axis = Chartist.Class.extend({\n constructor: Axis,\n createGridAndLabels: createGridAndLabels,\n projectValue: function(value, index, data) {\n throw new Error('Base axis can\\'t be instantiated!');\n }\n });\n\n Chartist.Axis.units = axisUnits;\n\n}(window, document, Chartist));\n;/\*\*\n \* The auto scale axis uses standard linear scale projection of values along an axis. It uses order of magnitude to find a scale automatically and evaluates the available space in order to find the perfect amount of ticks for your chart.\n \* \*\*Options\*\*\n \* The following options are used by this axis in addition to the default axis options outlined in the axis configuration of the chart default settings.\n \* ```javascript\n \* var options = {\n \* // If high is specified then the axis will display values explicitly up to this value and the computed maximum from the data is ignored\n \* high: 100,\n \* // If low is specified then the axis will display values explicitly down to this value and the computed minimum from the data is ignored\n \* low: 0,\n \* // This option will be used when finding the right scale division settings. The amount of ticks on the scale will be determined so that as many ticks as possible will be displayed, while not violating this minimum required space (in pixel).\n \* scaleMinSpace: 20,\n \* // Can be set to true or false. If set to true, the scale will be generated with whole numbers only.\n \* onlyInteger: true,\n \* // The reference value can be used to make sure that this value will always be on the chart. This is especially useful on bipolar charts where the bipolar center always needs to be part of the chart.\n \* referenceValue: 5\n \* };\n \* ```\n \*\n \* @module Chartist.AutoScaleAxis\n \*/\n/\* global Chartist \*/\n(function (window, document, Chartist) {\n 'use strict';\n\n function AutoScaleAxis(axisUnit, data, chartRect, options) {\n // Usually we calculate highLow based on the data but this can be overriden by a highLow object in the options\n var highLow = options.highLow || Chartist.getHighLow(data, options, axisUnit.pos);\n this.bounds = Chartist.getBounds(chartRect[axisUnit.rectEnd] - chartRect[axisUnit.rectStart], highLow, options.scaleMinSpace || 20, options.onlyInteger);\n this.range = {\n min: this.bounds.min,\n max: this.bounds.max\n };\n\n Chartist.AutoScaleAxis.super.constructor.call(this,\n axisUnit,\n chartRect,\n this.bounds.values,\n options);\n }\n\n function projectValue(value) {\n return this.axisLength \* (+Chartist.getMultiValue(value, this.units.pos) - this.bounds.min) / this.bounds.range;\n }\n\n Chartist.AutoScaleAxis = Chartist.Axis.extend({\n constructor: AutoScaleAxis,\n projectValue: projectValue\n });\n\n}(window, document, Chartist));\n;/\*\*\n \* The fixed scale axis uses standard linear projection of values along an axis. It makes use of a divisor option to divide the range provided from the minimum and maximum value or the options high and low that will override the computed minimum and maximum.\n \* \*\*Options\*\*\n \* The following options are used by this axis in addition to the default axis options outlined in the axis configuration of the chart default settings.\n \* ```javascript\n \* var options = {\n \* // If high is specified then the axis will display values explicitly up to this value and the computed maximum from the data is ignored\n \* high: 100,\n \* // If low is specified then the axis will display values explicitly down to this value and the computed minimum from the data is ignored\n \* low: 0,\n \* // If specified then the value range determined from minimum to maximum (or low and high) will be divided by this number and ticks will be generated at those division points. The default divisor is 1.\n \* divisor: 4,\n \* // If ticks is explicitly set, then the axis will not compute the ticks with the divisor, but directly use the data in ticks to determine at what points on the axis a tick need to be generated.\n \* ticks: [1, 10, 20, 30]\n \* };\n \* ```\n \*\n \* @module Chartist.FixedScaleAxis\n \*/\n/\* global Chartist \*/\n(function (window, document, Chartist) {\n 'use strict';\n\n function FixedScaleAxis(axisUnit, data, chartRect, options) {\n var highLow = options.highLow || Chartist.getHighLow(data, options, axisUnit.pos);\n this.divisor = options.divisor || 1;\n this.ticks = options.ticks || Chartist.times(this.divisor).map(function(value, index) {\n return highLow.low + (highLow.high - highLow.low) / this.divisor \* index;\n }.bind(this));\n this.ticks.sort(function(a, b) {\n return a - b;\n });\n this.range = {\n min: highLow.low,\n max: highLow.high\n };\n\n Chartist.FixedScaleAxis.super.constructor.call(this,\n axisUnit,\n chartRect,\n this.ticks,\n options);\n\n this.stepLength = this.axisLength / this.divisor;\n }\n\n function projectValue(value) {\n return this.axisLength \* (+Chartist.getMultiValue(value, this.units.pos) - this.range.min) / (this.range.max - this.range.min);\n }\n\n Chartist.FixedScaleAxis = Chartist.Axis.extend({\n constructor: FixedScaleAxis,\n projectValue: projectValue\n });\n\n}(window, document, Chartist));\n;/\*\*\n \* The step axis for step based charts like bar chart or step based line charts. It uses a fixed amount of ticks that will be equally distributed across the whole axis length. The projection is done using the index of the data value rather than the value itself and therefore it's only useful for distribution purpose.\n \* \*\*Options\*\*\n \* The following options are used by this axis in addition to the default axis options outlined in the axis configuration of the chart default settings.\n \* ```javascript\n \* var options = {\n \* // Ticks to be used to distribute across the axis length. As this axis type relies on the index of the value rather than the value, arbitrary data that can be converted to a string can be used as ticks.\n \* ticks: ['One', 'Two', 'Three'],\n \* // If set to true the full width will be used to distribute the values where the last value will be at the maximum of the axis length. If false the spaces between the ticks will be evenly distributed instead.\n \* stretch: true\n \* };\n \* ```\n \*\n \* @module Chartist.StepAxis\n \*/\n/\* global Chartist \*/\n(function (window, document, Chartist) {\n 'use strict';\n\n function StepAxis(axisUnit, data, chartRect, options) {\n Chartist.StepAxis.super.constructor.call(this,\n axisUnit,\n chartRect,\n options.ticks,\n options);\n\n var calc = Math.max(1, options.ticks.length - (options.stretch ? 1 : 0));\n this.stepLength = this.axisLength / calc;\n }\n\n function projectValue(value, index) {\n return this.stepLength \* index;\n }\n\n Chartist.StepAxis = Chartist.Axis.extend({\n constructor: StepAxis,\n projectValue: projectValue\n });\n\n}(window, document, Chartist));\n;/\*\*\n \* The Chartist line chart can be used to draw Line or Scatter charts. If used in the browser you can access the global `Chartist` namespace where you find the `Line` function as a main entry point.\n \*\n \* For examples on how to use the line chart please check the examples of the `Chartist.Line` method.\n \*\n \* @module Chartist.Line\n \*/\n/\* global Chartist \*/\n(function(window, document, Chartist){\n 'use strict';\n\n /\*\*\n \* Default options in line charts. Expand the code view to see a detailed list of options with comments.\n \*\n \* @memberof Chartist.Line\n \*/\n var defaultOptions = {\n // Options for X-Axis\n axisX: {\n // The offset of the labels to the chart area\n offset: 30,\n // Position where labels are placed. Can be set to `start` or `end` where `start` is equivalent to left or top on vertical axis and `end` is equivalent to right or bottom on horizontal axis.\n position: 'end',\n // Allows you to correct label positioning on this axis by positive or negative x and y offset.\n labelOffset: {\n x: 0,\n y: 0\n },\n // If labels should be shown or not\n showLabel: true,\n // If the axis grid should be drawn or not\n showGrid: true,\n // Interpolation function that allows you to intercept the value from the axis label\n labelInterpolationFnc: Chartist.noop,\n // Set the axis type to be used to project values on this axis. If not defined, Chartist.StepAxis will be used for the X-Axis, where the ticks option will be set to the labels in the data and the stretch option will be set to the global fullWidth option. This type can be changed to any axis constructor available (e.g. Chartist.FixedScaleAxis), where all axis options should be present here.\n type: undefined\n },\n // Options for Y-Axis\n axisY: {\n // The offset of the labels to the chart area\n offset: 40,\n // Position where labels are placed. Can be set to `start` or `end` where `start` is equivalent to left or top on vertical axis and `end` is equivalent to right or bottom on horizontal axis.\n position: 'start',\n // Allows you to correct label positioning on this axis by positive or negative x and y offset.\n labelOffset: {\n x: 0,\n y: 0\n },\n // If labels should be shown or not\n showLabel: true,\n // If the axis grid should be drawn or not\n showGrid: true,\n // Interpolation function that allows you to intercept the value from the axis label\n labelInterpolationFnc: Chartist.noop,\n // Set the axis type to be used to project values on this axis. If not defined, Chartist.AutoScaleAxis will be used for the Y-Axis, where the high and low options will be set to the global high and low options. This type can be changed to any axis constructor available (e.g. Chartist.FixedScaleAxis), where all axis options should be present here.\n type: undefined,\n // This value specifies the minimum height in pixel of the scale steps\n scaleMinSpace: 20,\n // Use only integer values (whole numbers) for the scale steps\n onlyInteger: false\n },\n // Specify a fixed width for the chart as a string (i.e. '100px' or '50%')\n width: undefined,\n // Specify a fixed height for the chart as a string (i.e. '100px' or '50%')\n height: undefined,\n // If the line should be drawn or not\n showLine: true,\n // If dots should be drawn or not\n showPoint: true,\n // If the line chart should draw an area\n showArea: false,\n // The base for the area chart that will be used to close the area shape (is normally 0)\n areaBase: 0,\n // Specify if the lines should be smoothed. This value can be true or false where true will result in smoothing using the default smoothing interpolation function Chartist.Interpolation.cardinal and false results in Chartist.Interpolation.none. You can also choose other smoothing / interpolation functions available in the Chartist.Interpolation module, or write your own interpolation function. Check the examples for a brief description.\n lineSmooth: true,\n // If the line chart should add a background fill to the .ct-grids group.\n showGridBackground: false,\n // Overriding the natural low of the chart allows you to zoom in or limit the charts lowest displayed value\n low: undefined,\n // Overriding the natural high of the chart allows you to zoom in or limit the charts highest displayed value\n high: undefined,\n // Padding of the chart drawing area to the container element and labels as a number or padding object {top: 5, right: 5, bottom: 5, left: 5}\n chartPadding: {\n top: 15,\n right: 15,\n bottom: 5,\n left: 10\n },\n // When set to true, the last grid line on the x-axis is not drawn and the chart elements will expand to the full available width of the chart. For the last label to be drawn correctly you might need to add chart padding or offset the last label with a draw event handler.\n fullWidth: false,\n // If true the whole data is reversed including labels, the series order as well as the whole series data arrays.\n reverseData: false,\n // Override the class names that get used to generate the SVG structure of the chart\n classNames: {\n chart: 'ct-chart-line',\n label: 'ct-label',\n labelGroup: 'ct-labels',\n series: 'ct-series',\n line: 'ct-line',\n point: 'ct-point',\n area: 'ct-area',\n grid: 'ct-grid',\n gridGroup: 'ct-grids',\n gridBackground: 'ct-grid-background',\n vertical: 'ct-vertical',\n horizontal: 'ct-horizontal',\n start: 'ct-start',\n end: 'ct-end'\n }\n };\n\n /\*\*\n \* Creates a new chart\n \*\n \*/\n function createChart(options) {\n var data = Chartist.normalizeData(this.data, options.reverseData, true);\n\n // Create new svg object\n this.svg = Chartist.createSvg(this.container, options.width, options.height, options.classNames.chart);\n // Create groups for labels, grid and series\n var gridGroup = this.svg.elem('g').addClass(options.classNames.gridGroup);\n var seriesGroup = this.svg.elem('g');\n var labelGroup = this.svg.elem('g').addClass(options.classNames.labelGroup);\n\n var chartRect = Chartist.createChartRect(this.svg, options, defaultOptions.padding);\n var axisX, axisY;\n\n if(options.axisX.type === undefined) {\n axisX = new Chartist.StepAxis(Chartist.Axis.units.x, data.normalized.series, chartRect, Chartist.extend({}, options.axisX, {\n ticks: data.normalized.labels,\n stretch: options.fullWidth\n }));\n } else {\n axisX = options.axisX.type.call(Chartist, Chartist.Axis.units.x, data.normalized.series, chartRect, options.axisX);\n }\n\n if(options.axisY.type === undefined) {\n axisY = new Chartist.AutoScaleAxis(Chartist.Axis.units.y, data.normalized.series, chartRect, Chartist.extend({}, options.axisY, {\n high: Chartist.isNumeric(options.high) ? options.high : options.axisY.high,\n low: Chartist.isNumeric(options.low) ? options.low : options.axisY.low\n }));\n } else {\n axisY = options.axisY.type.call(Chartist, Chartist.Axis.units.y, data.normalized.series, chartRect, options.axisY);\n }\n\n axisX.createGridAndLabels(gridGroup, labelGroup, this.supportsForeignObject, options, this.eventEmitter);\n axisY.createGridAndLabels(gridGroup, labelGroup, this.supportsForeignObject, options, this.eventEmitter);\n\n if (options.showGridBackground) {\n Chartist.createGridBackground(gridGroup, chartRect, options.classNames.gridBackground, this.eventEmitter);\n }\n\n // Draw the series\n data.raw.series.forEach(function(series, seriesIndex) {\n var seriesElement = seriesGroup.elem('g');\n\n // Write attributes to series group element. If series name or meta is undefined the attributes will not be written\n seriesElement.attr({\n 'ct:series-name': series.name,\n 'ct:meta': Chartist.serialize(series.meta)\n });\n\n // Use series class from series data or if not set generate one\n seriesElement.addClass([\n options.classNames.series,\n (series.className || options.classNames.series + '-' + Chartist.alphaNumerate(seriesIndex))\n ].join(' '));\n\n var pathCoordinates = [],\n pathData = [];\n\n data.normalized.series[seriesIndex].forEach(function(value, valueIndex) {\n var p = {\n x: chartRect.x1 + axisX.projectValue(value, valueIndex, data.normalized.series[seriesIndex]),\n y: chartRect.y1 - axisY.projectValue(value, valueIndex, data.normalized.series[seriesIndex])\n };\n pathCoordinates.push(p.x, p.y);\n pathData.push({\n value: value,\n valueIndex: valueIndex,\n meta: Chartist.getMetaData(series, valueIndex)\n });\n }.bind(this));\n\n var seriesOptions = {\n lineSmooth: Chartist.getSeriesOption(series, options, 'lineSmooth'),\n showPoint: Chartist.getSeriesOption(series, options, 'showPoint'),\n showLine: Chartist.getSeriesOption(series, options, 'showLine'),\n showArea: Chartist.getSeriesOption(series, options, 'showArea'),\n areaBase: Chartist.getSeriesOption(series, options, 'areaBase')\n };\n\n var smoothing = typeof seriesOptions.lineSmooth === 'function' ?\n seriesOptions.lineSmooth : (seriesOptions.lineSmooth ? Chartist.Interpolation.monotoneCubic() : Chartist.Interpolation.none());\n // Interpolating path where pathData will be used to annotate each path element so we can trace back the original\n // index, value and meta data\n var path = smoothing(pathCoordinates, pathData);\n\n // If we should show points we need to create them now to avoid secondary loop\n // Points are drawn from the pathElements returned by the interpolation function\n // Small offset for Firefox to render squares correctly\n if (seriesOptions.showPoint) {\n\n path.pathElements.forEach(function(pathElement) {\n var point = seriesElement.elem('line', {\n x1: pathElement.x,\n y1: pathElement.y,\n x2: pathElement.x + 0.01,\n y2: pathElement.y\n }, options.classNames.point).attr({\n 'ct:value': [pathElement.data.value.x, pathElement.data.value.y].filter(Chartist.isNumeric).join(','),\n 'ct:meta': Chartist.serialize(pathElement.data.meta)\n });\n\n this.eventEmitter.emit('draw', {\n type: 'point',\n value: pathElement.data.value,\n index: pathElement.data.valueIndex,\n meta: pathElement.data.meta,\n series: series,\n seriesIndex: seriesIndex,\n axisX: axisX,\n axisY: axisY,\n group: seriesElement,\n element: point,\n x: pathElement.x,\n y: pathElement.y\n });\n }.bind(this));\n }\n\n if(seriesOptions.showLine) {\n var line = seriesElement.elem('path', {\n d: path.stringify()\n }, options.classNames.line, true);\n\n this.eventEmitter.emit('draw', {\n type: 'line',\n values: data.normalized.series[seriesIndex],\n path: path.clone(),\n chartRect: chartRect,\n index: seriesIndex,\n series: series,\n seriesIndex: seriesIndex,\n seriesMeta: series.meta,\n axisX: axisX,\n axisY: axisY,\n group: seriesElement,\n element: line\n });\n }\n\n // Area currently only works with axes that support a range!\n if(seriesOptions.showArea && axisY.range) {\n // If areaBase is outside the chart area (< min or > max) we need to set it respectively so that\n // the area is not drawn outside the chart area.\n var areaBase = Math.max(Math.min(seriesOptions.areaBase, axisY.range.max), axisY.range.min);\n\n // We project the areaBase value into screen coordinates\n var areaBaseProjected = chartRect.y1 - axisY.projectValue(areaBase);\n\n // In order to form the area we'll first split the path by move commands so we can chunk it up into segments\n path.splitByCommand('M').filter(function onlySolidSegments(pathSegment) {\n // We filter only \"solid\" segments that contain more than one point. Otherwise there's no need for an area\n return pathSegment.pathElements.length > 1;\n }).map(function convertToArea(solidPathSegments) {\n // Receiving the filtered solid path segments we can now convert those segments into fill areas\n var firstElement = solidPathSegments.pathElements[0];\n var lastElement = solidPathSegments.pathElements[solidPathSegments.pathElements.length - 1];\n\n // Cloning the solid path segment with closing option and removing the first move command from the clone\n // We then insert a new move that should start at the area base and draw a straight line up or down\n // at the end of the path we add an additional straight line to the projected area base value\n // As the closing option is set our path will be automatically closed\n return solidPathSegments.clone(true)\n .position(0)\n .remove(1)\n .move(firstElement.x, areaBaseProjected)\n .line(firstElement.x, firstElement.y)\n .position(solidPathSegments.pathElements.length + 1)\n .line(lastElement.x, areaBaseProjected);\n\n }).forEach(function createArea(areaPath) {\n // For each of our newly created area paths, we'll now create path elements by stringifying our path objects\n // and adding the created DOM elements to the correct series group\n var area = seriesElement.elem('path', {\n d: areaPath.stringify()\n }, options.classNames.area, true);\n\n // Emit an event for each area that was drawn\n this.eventEmitter.emit('draw', {\n type: 'area',\n values: data.normalized.series[seriesIndex],\n path: areaPath.clone(),\n series: series,\n seriesIndex: seriesIndex,\n axisX: axisX,\n axisY: axisY,\n chartRect: chartRect,\n index: seriesIndex,\n group: seriesElement,\n element: area\n });\n }.bind(this));\n }\n }.bind(this));\n\n this.eventEmitter.emit('created', {\n bounds: axisY.bounds,\n chartRect: chartRect,\n axisX: axisX,\n axisY: axisY,\n svg: this.svg,\n options: options\n });\n }\n\n /\*\*\n \* This method creates a new line chart.\n \*\n \* @memberof Chartist.Line\n \* @param {String|Node} query A selector query string or directly a DOM element\n \* @param {Object} data The data object that needs to consist of a labels and a series array\n \* @param {Object} [options] The options object with options that override the default options. Check the examples for a detailed list.\n \* @param {Array} [responsiveOptions] Specify an array of responsive option arrays which are a media query and options object pair => [[mediaQueryString, optionsObject],[more...]]\n \* @return {Object} An object which exposes the API for the created chart\n \*\n \* @example\n \* // Create a simple line chart\n \* var data = {\n \* // A labels array that can contain any sort of values\n \* labels: ['Mon', 'Tue', 'Wed', 'Thu', 'Fri'],\n \* // Our series array that contains series objects or in this case series data arrays\n \* series: [\n \* [5, 2, 4, 2, 0]\n \* ]\n \* };\n \*\n \* // As options we currently only set a static size of 300x200 px\n \* var options = {\n \* width: '300px',\n \* height: '200px'\n \* };\n \*\n \* // In the global name space Chartist we call the Line function to initialize a line chart. As a first parameter we pass in a selector where we would like to get our chart created. Second parameter is the actual data object and as a third parameter we pass in our options\n \* new Chartist.Line('.ct-chart', data, options);\n \*\n \* @example\n \* // Use specific interpolation function with configuration from the Chartist.Interpolation module\n \*\n \* var chart = new Chartist.Line('.ct-chart', {\n \* labels: [1, 2, 3, 4, 5],\n \* series: [\n \* [1, 1, 8, 1, 7]\n \* ]\n \* }, {\n \* lineSmooth: Chartist.Interpolation.cardinal({\n \* tension: 0.2\n \* })\n \* });\n \*\n \* @example\n \* // Create a line chart with responsive options\n \*\n \* var data = {\n \* // A labels array that can contain any sort of values\n \* labels: ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'],\n \* // Our series array that contains series objects or in this case series data arrays\n \* series: [\n \* [5, 2, 4, 2, 0]\n \* ]\n \* };\n \*\n \* // In addition to the regular options we specify responsive option overrides that will override the default configutation based on the matching media queries.\n \* var responsiveOptions = [\n \* ['screen and (min-width: 641px) and (max-width: 1024px)', {\n \* showPoint: false,\n \* axisX: {\n \* labelInterpolationFnc: function(value) {\n \* // Will return Mon, Tue, Wed etc. on medium screens\n \* return value.slice(0, 3);\n \* }\n \* }\n \* }],\n \* ['screen and (max-width: 640px)', {\n \* showLine: false,\n \* axisX: {\n \* labelInterpolationFnc: function(value) {\n \* // Will return M, T, W etc. on small screens\n \* return value[0];\n \* }\n \* }\n \* }]\n \* ];\n \*\n \* new Chartist.Line('.ct-chart', data, null, responsiveOptions);\n \*\n \*/\n function Line(query, data, options, responsiveOptions) {\n Chartist.Line.super.constructor.call(this,\n query,\n data,\n defaultOptions,\n Chartist.extend({}, defaultOptions, options),\n responsiveOptions);\n }\n\n // Creating line chart type in Chartist namespace\n Chartist.Line = Chartist.Base.extend({\n constructor: Line,\n createChart: createChart\n });\n\n}(window, document, Chartist));\n;/\*\*\n \* The bar chart module of Chartist that can be used to draw unipolar or bipolar bar and grouped bar charts.\n \*\n \* @module Chartist.Bar\n \*/\n/\* global Chartist \*/\n(function(window, document, Chartist){\n 'use strict';\n\n /\*\*\n \* Default options in bar charts. Expand the code view to see a detailed list of options with comments.\n \*\n \* @memberof Chartist.Bar\n \*/\n var defaultOptions = {\n // Options for X-Axis\n axisX: {\n // The offset of the chart drawing area to the border of the container\n offset: 30,\n // Position where labels are placed. Can be set to `start` or `end` where `start` is equivalent to left or top on vertical axis and `end` is equivalent to right or bottom on horizontal axis.\n position: 'end',\n // Allows you to correct label positioning on this axis by positive or negative x and y offset.\n labelOffset: {\n x: 0,\n y: 0\n },\n // If labels should be shown or not\n showLabel: true,\n // If the axis grid should be drawn or not\n showGrid: true,\n // Interpolation function that allows you to intercept the value from the axis label\n labelInterpolationFnc: Chartist.noop,\n // This value specifies the minimum width in pixel of the scale steps\n scaleMinSpace: 30,\n // Use only integer values (whole numbers) for the scale steps\n onlyInteger: false\n },\n // Options for Y-Axis\n axisY: {\n // The offset of the chart drawing area to the border of the container\n offset: 40,\n // Position where labels are placed. Can be set to `start` or `end` where `start` is equivalent to left or top on vertical axis and `end` is equivalent to right or bottom on horizontal axis.\n position: 'start',\n // Allows you to correct label positioning on this axis by positive or negative x and y offset.\n labelOffset: {\n x: 0,\n y: 0\n },\n // If labels should be shown or not\n showLabel: true,\n // If the axis grid should be drawn or not\n showGrid: true,\n // Interpolation function that allows you to intercept the value from the axis label\n labelInterpolationFnc: Chartist.noop,\n // This value specifies the minimum height in pixel of the scale steps\n scaleMinSpace: 20,\n // Use only integer values (whole numbers) for the scale steps\n onlyInteger: false\n },\n // Specify a fixed width for the chart as a string (i.e. '100px' or '50%')\n width: undefined,\n // Specify a fixed height for the chart as a string (i.e. '100px' or '50%')\n height: undefined,\n // Overriding the natural high of the chart allows you to zoom in or limit the charts highest displayed value\n high: undefined,\n // Overriding the natural low of the chart allows you to zoom in or limit the charts lowest displayed value\n low: undefined,\n // Unless low/high are explicitly set, bar chart will be centered at zero by default. Set referenceValue to null to auto scale.\n referenceValue: 0,\n // Padding of the chart drawing area to the container element and labels as a number or padding object {top: 5, right: 5, bottom: 5, left: 5}\n chartPadding: {\n top: 15,\n right: 15,\n bottom: 5,\n left: 10\n },\n // Specify the distance in pixel of bars in a group\n seriesBarDistance: 15,\n // If set to true this property will cause the series bars to be stacked. Check the `stackMode` option for further stacking options.\n stackBars: false,\n // If set to 'overlap' this property will force the stacked bars to draw from the zero line.\n // If set to 'accumulate' this property will form a total for each series point. This will also influence the y-axis and the overall bounds of the chart. In stacked mode the seriesBarDistance property will have no effect.\n stackMode: 'accumulate',\n // Inverts the axes of the bar chart in order to draw a horizontal bar chart. Be aware that you also need to invert your axis settings as the Y Axis will now display the labels and the X Axis the values.\n horizontalBars: false,\n // If set to true then each bar will represent a series and the data array is expected to be a one dimensional array of data values rather than a series array of series. This is useful if the bar chart should represent a profile rather than some data over time.\n distributeSeries: false,\n // If true the whole data is reversed including labels, the series order as well as the whole series data arrays.\n reverseData: false,\n // If the bar chart should add a background fill to the .ct-grids group.\n showGridBackground: false,\n // Override the class names that get used to generate the SVG structure of the chart\n classNames: {\n chart: 'ct-chart-bar',\n horizontalBars: 'ct-horizontal-bars',\n label: 'ct-label',\n labelGroup: 'ct-labels',\n series: 'ct-series',\n bar: 'ct-bar',\n grid: 'ct-grid',\n gridGroup: 'ct-grids',\n gridBackground: 'ct-grid-background',\n vertical: 'ct-vertical',\n horizontal: 'ct-horizontal',\n start: 'ct-start',\n end: 'ct-end'\n }\n };\n\n /\*\*\n \* Creates a new chart\n \*\n \*/\n function createChart(options) {\n var data;\n var highLow;\n\n if(options.distributeSeries) {\n data = Chartist.normalizeData(this.data, options.reverseData, options.horizontalBars ? 'x' : 'y');\n data.normalized.series = data.normalized.series.map(function(value) {\n return [value];\n });\n } else {\n data = Chartist.normalizeData(this.data, options.reverseData, options.horizontalBars ? 'x' : 'y');\n }\n\n // Create new svg element\n this.svg = Chartist.createSvg(\n this.container,\n options.width,\n options.height,\n options.classNames.chart + (options.horizontalBars ? ' ' + options.classNames.horizontalBars : '')\n );\n\n // Drawing groups in correct order\n var gridGroup = this.svg.elem('g').addClass(options.classNames.gridGroup);\n var seriesGroup = this.svg.elem('g');\n var labelGroup = this.svg.elem('g').addClass(options.classNames.labelGroup);\n\n if(options.stackBars && data.normalized.series.length !== 0) {\n\n // If stacked bars we need to calculate the high low from stacked values from each series\n var serialSums = Chartist.serialMap(data.normalized.series, function serialSums() {\n return Array.prototype.slice.call(arguments).map(function(value) {\n return value;\n }).reduce(function(prev, curr) {\n return {\n x: prev.x + (curr && curr.x) || 0,\n y: prev.y + (curr && curr.y) || 0\n };\n }, {x: 0, y: 0});\n });\n\n highLow = Chartist.getHighLow([serialSums], options, options.horizontalBars ? 'x' : 'y');\n\n } else {\n\n highLow = Chartist.getHighLow(data.normalized.series, options, options.horizontalBars ? 'x' : 'y');\n }\n\n // Overrides of high / low from settings\n highLow.high = +options.high || (options.high === 0 ? 0 : highLow.high);\n highLow.low = +options.low || (options.low === 0 ? 0 : highLow.low);\n\n var chartRect = Chartist.createChartRect(this.svg, options, defaultOptions.padding);\n\n var valueAxis,\n labelAxisTicks,\n labelAxis,\n axisX,\n axisY;\n\n // We need to set step count based on some options combinations\n if(options.distributeSeries && options.stackBars) {\n // If distributed series are enabled and bars need to be stacked, we'll only have one bar and therefore should\n // use only the first label for the step axis\n labelAxisTicks = data.normalized.labels.slice(0, 1);\n } else {\n // If distributed series are enabled but stacked bars aren't, we should use the series labels\n // If we are drawing a regular bar chart with two dimensional series data, we just use the labels array\n // as the bars are normalized\n labelAxisTicks = data.normalized.labels;\n }\n\n // Set labelAxis and valueAxis based on the horizontalBars setting. This setting will flip the axes if necessary.\n if(options.horizontalBars) {\n if(options.axisX.type === undefined) {\n valueAxis = axisX = new Chartist.AutoScaleAxis(Chartist.Axis.units.x, data.normalized.series, chartRect, Chartist.extend({}, options.axisX, {\n highLow: highLow,\n referenceValue: 0\n }));\n } else {\n valueAxis = axisX = options.axisX.type.call(Chartist, Chartist.Axis.units.x, data.normalized.series, chartRect, Chartist.extend({}, options.axisX, {\n highLow: highLow,\n referenceValue: 0\n }));\n }\n\n if(options.axisY.type === undefined) {\n labelAxis = axisY = new Chartist.StepAxis(Chartist.Axis.units.y, data.normalized.series, chartRect, {\n ticks: labelAxisTicks\n });\n } else {\n labelAxis = axisY = options.axisY.type.call(Chartist, Chartist.Axis.units.y, data.normalized.series, chartRect, options.axisY);\n }\n } else {\n if(options.axisX.type === undefined) {\n labelAxis = axisX = new Chartist.StepAxis(Chartist.Axis.units.x, data.normalized.series, chartRect, {\n ticks: labelAxisTicks\n });\n } else {\n labelAxis = axisX = options.axisX.type.call(Chartist, Chartist.Axis.units.x, data.normalized.series, chartRect, options.axisX);\n }\n\n if(options.axisY.type === undefined) {\n valueAxis = axisY = new Chartist.AutoScaleAxis(Chartist.Axis.units.y, data.normalized.series, chartRect, Chartist.extend({}, options.axisY, {\n highLow: highLow,\n referenceValue: 0\n }));\n } else {\n valueAxis = axisY = options.axisY.type.call(Chartist, Chartist.Axis.units.y, data.normalized.series, chartRect, Chartist.extend({}, options.axisY, {\n highLow: highLow,\n referenceValue: 0\n }));\n }\n }\n\n // Projected 0 point\n var zeroPoint = options.horizontalBars ? (chartRect.x1 + valueAxis.projectValue(0)) : (chartRect.y1 - valueAxis.projectValue(0));\n // Used to track the screen coordinates of stacked bars\n var stackedBarValues = [];\n\n labelAxis.createGridAndLabels(gridGroup, labelGroup, this.supportsForeignObject, options, this.eventEmitter);\n valueAxis.createGridAndLabels(gridGroup, labelGroup, this.supportsForeignObject, options, this.eventEmitter);\n\n if (options.showGridBackground) {\n Chartist.createGridBackground(gridGroup, chartRect, options.classNames.gridBackground, this.eventEmitter);\n }\n\n // Draw the series\n data.raw.series.forEach(function(series, seriesIndex) {\n // Calculating bi-polar value of index for seriesOffset. For i = 0..4 biPol will be -1.5, -0.5, 0.5, 1.5 etc.\n var biPol = seriesIndex - (data.raw.series.length - 1) / 2;\n // Half of the period width between vertical grid lines used to position bars\n var periodHalfLength;\n // Current series SVG element\n var seriesElement;\n\n // We need to set periodHalfLength based on some options combinations\n if(options.distributeSeries && !options.stackBars) {\n // If distributed series are enabled but stacked bars aren't, we need to use the length of the normaizedData array\n // which is the series count and divide by 2\n periodHalfLength = labelAxis.axisLength / data.normalized.series.length / 2;\n } else if(options.distributeSeries && options.stackBars) {\n // If distributed series and stacked bars are enabled we'll only get one bar so we should just divide the axis\n // length by 2\n periodHalfLength = labelAxis.axisLength / 2;\n } else {\n // On regular bar charts we should just use the series length\n periodHalfLength = labelAxis.axisLength / data.normalized.series[seriesIndex].length / 2;\n }\n\n // Adding the series group to the series element\n seriesElement = seriesGroup.elem('g');\n\n // Write attributes to series group element. If series name or meta is undefined the attributes will not be written\n seriesElement.attr({\n 'ct:series-name': series.name,\n 'ct:meta': Chartist.serialize(series.meta)\n });\n\n // Use series class from series data or if not set generate one\n seriesElement.addClass([\n options.classNames.series,\n (series.className || options.classNames.series + '-' + Chartist.alphaNumerate(seriesIndex))\n ].join(' '));\n\n data.normalized.series[seriesIndex].forEach(function(value, valueIndex) {\n var projected,\n bar,\n previousStack,\n labelAxisValueIndex;\n\n // We need to set labelAxisValueIndex based on some options combinations\n if(options.distributeSeries && !options.stackBars) {\n // If distributed series are enabled but stacked bars aren't, we can use the seriesIndex for later projection\n // on the step axis for label positioning\n labelAxisValueIndex = seriesIndex;\n } else if(options.distributeSeries && options.stackBars) {\n // If distributed series and stacked bars are enabled, we will only get one bar and therefore always use\n // 0 for projection on the label step axis\n labelAxisValueIndex = 0;\n } else {\n // On regular bar charts we just use the value index to project on the label step axis\n labelAxisValueIndex = valueIndex;\n }\n\n // We need to transform coordinates differently based on the chart layout\n if(options.horizontalBars) {\n projected = {\n x: chartRect.x1 + valueAxis.projectValue(value && value.x ? value.x : 0, valueIndex, data.normalized.series[seriesIndex]),\n y: chartRect.y1 - labelAxis.projectValue(value && value.y ? value.y : 0, labelAxisValueIndex, data.normalized.series[seriesIndex])\n };\n } else {\n projected = {\n x: chartRect.x1 + labelAxis.projectValue(value && value.x ? value.x : 0, labelAxisValueIndex, data.normalized.series[seriesIndex]),\n y: chartRect.y1 - valueAxis.projectValue(value && value.y ? value.y : 0, valueIndex, data.normalized.series[seriesIndex])\n }\n }\n\n // If the label axis is a step based axis we will offset the bar into the middle of between two steps using\n // the periodHalfLength value. Also we do arrange the different series so that they align up to each other using\n // the seriesBarDistance. If we don't have a step axis, the bar positions can be chosen freely so we should not\n // add any automated positioning.\n if(labelAxis instanceof Chartist.StepAxis) {\n // Offset to center bar between grid lines, but only if the step axis is not stretched\n if(!labelAxis.options.stretch) {\n projected[labelAxis.units.pos] += periodHalfLength \* (options.horizontalBars ? -1 : 1);\n }\n // Using bi-polar offset for multiple series if no stacked bars or series distribution is used\n projected[labelAxis.units.pos] += (options.stackBars || options.distributeSeries) ? 0 : biPol \* options.seriesBarDistance \* (options.horizontalBars ? -1 : 1);\n }\n\n // Enter value in stacked bar values used to remember previous screen value for stacking up bars\n previousStack = stackedBarValues[valueIndex] || zeroPoint;\n stackedBarValues[valueIndex] = previousStack - (zeroPoint - projected[labelAxis.counterUnits.pos]);\n\n // Skip if value is undefined\n if(value === undefined) {\n return;\n }\n\n var positions = {};\n positions[labelAxis.units.pos + '1'] = projected[labelAxis.units.pos];\n positions[labelAxis.units.pos + '2'] = projected[labelAxis.units.pos];\n\n if(options.stackBars && (options.stackMode === 'accumulate' || !options.stackMode)) {\n // Stack mode: accumulate (default)\n // If bars are stacked we use the stackedBarValues reference and otherwise base all bars off the zero line\n // We want backwards compatibility, so the expected fallback without the 'stackMode' option\n // to be the original behaviour (accumulate)\n positions[labelAxis.counterUnits.pos + '1'] = previousStack;\n positions[labelAxis.counterUnits.pos + '2'] = stackedBarValues[valueIndex];\n } else {\n // Draw from the zero line normally\n // This is also the same code for Stack mode: overlap\n positions[labelAxis.counterUnits.pos + '1'] = zeroPoint;\n positions[labelAxis.counterUnits.pos + '2'] = projected[labelAxis.counterUnits.pos];\n }\n\n // Limit x and y so that they are within the chart rect\n positions.x1 = Math.min(Math.max(positions.x1, chartRect.x1), chartRect.x2);\n positions.x2 = Math.min(Math.max(positions.x2, chartRect.x1), chartRect.x2);\n positions.y1 = Math.min(Math.max(positions.y1, chartRect.y2), chartRect.y1);\n positions.y2 = Math.min(Math.max(positions.y2, chartRect.y2), chartRect.y1);\n\n var metaData = Chartist.getMetaData(series, valueIndex);\n\n // Create bar element\n bar = seriesElement.elem('line', positions, options.classNames.bar).attr({\n 'ct:value': [value.x, value.y].filter(Chartist.isNumeric).join(','),\n 'ct:meta': Chartist.serialize(metaData)\n });\n\n this.eventEmitter.emit('draw', Chartist.extend({\n type: 'bar',\n value: value,\n index: valueIndex,\n meta: metaData,\n series: series,\n seriesIndex: seriesIndex,\n axisX: axisX,\n axisY: axisY,\n chartRect: chartRect,\n group: seriesElement,\n element: bar\n }, positions));\n }.bind(this));\n }.bind(this));\n\n this.eventEmitter.emit('created', {\n bounds: valueAxis.bounds,\n chartRect: chartRect,\n axisX: axisX,\n axisY: axisY,\n svg: this.svg,\n options: options\n });\n }\n\n /\*\*\n \* This method creates a new bar chart and returns API object that you can use for later changes.\n \*\n \* @memberof Chartist.Bar\n \* @param {String|Node} query A selector query string or directly a DOM element\n \* @param {Object} data The data object that needs to consist of a labels and a series array\n \* @param {Object} [options] The options object with options that override the default options. Check the examples for a detailed list.\n \* @param {Array} [responsiveOptions] Specify an array of responsive option arrays which are a media query and options object pair => [[mediaQueryString, optionsObject],[more...]]\n \* @return {Object} An object which exposes the API for the created chart\n \*\n \* @example\n \* // Create a simple bar chart\n \* var data = {\n \* labels: ['Mon', 'Tue', 'Wed', 'Thu', 'Fri'],\n \* series: [\n \* [5, 2, 4, 2, 0]\n \* ]\n \* };\n \*\n \* // In the global name space Chartist we call the Bar function to initialize a bar chart. As a first parameter we pass in a selector where we would like to get our chart created and as a second parameter we pass our data object.\n \* new Chartist.Bar('.ct-chart', data);\n \*\n \* @example\n \* // This example creates a bipolar grouped bar chart where the boundaries are limitted to -10 and 10\n \* new Chartist.Bar('.ct-chart', {\n \* labels: [1, 2, 3, 4, 5, 6, 7],\n \* series: [\n \* [1, 3, 2, -5, -3, 1, -6],\n \* [-5, -2, -4, -1, 2, -3, 1]\n \* ]\n \* }, {\n \* seriesBarDistance: 12,\n \* low: -10,\n \* high: 10\n \* });\n \*\n \*/\n function Bar(query, data, options, responsiveOptions) {\n Chartist.Bar.super.constructor.call(this,\n query,\n data,\n defaultOptions,\n Chartist.extend({}, defaultOptions, options),\n responsiveOptions);\n }\n\n // Creating bar chart type in Chartist namespace\n Chartist.Bar = Chartist.Base.extend({\n constructor: Bar,\n createChart: createChart\n });\n\n}(window, document, Chartist));\n;/\*\*\n \* The pie chart module of Chartist that can be used to draw pie, donut or gauge charts\n \*\n \* @module Chartist.Pie\n \*/\n/\* global Chartist \*/\n(function(window, document, Chartist) {\n 'use strict';\n\n /\*\*\n \* Default options in line charts. Expand the code view to see a detailed list of options with comments.\n \*\n \* @memberof Chartist.Pie\n \*/\n var defaultOptions = {\n // Specify a fixed width for the chart as a string (i.e. '100px' or '50%')\n width: undefined,\n // Specify a fixed height for the chart as a string (i.e. '100px' or '50%')\n height: undefined,\n // Padding of the chart drawing area to the container element and labels as a number or padding object {top: 5, right: 5, bottom: 5, left: 5}\n chartPadding: 5,\n // Override the class names that are used to generate the SVG structure of the chart\n classNames: {\n chartPie: 'ct-chart-pie',\n chartDonut: 'ct-chart-donut',\n series: 'ct-series',\n slicePie: 'ct-slice-pie',\n sliceDonut: 'ct-slice-donut',\n label: 'ct-label'\n },\n // The start angle of the pie chart in degrees where 0 points north. A higher value offsets the start angle clockwise.\n startAngle: 0,\n // An optional total you can specify. By specifying a total value, the sum of the values in the series must be this total in order to draw a full pie. You can use this parameter to draw only parts of a pie or gauge charts.\n total: undefined,\n // If specified the donut CSS classes will be used and strokes will be drawn instead of pie slices.\n donut: false,\n // Specify the donut stroke width, currently done in javascript for convenience. May move to CSS styles in the future.\n // This option can be set as number or string to specify a relative width (i.e. 100 or '30%').\n donutWidth: 60,\n // If a label should be shown or not\n showLabel: true,\n // Label position offset from the standard position which is half distance of the radius. This value can be either positive or negative. Positive values will position the label away from the center.\n labelOffset: 0,\n // This option can be set to 'inside', 'outside' or 'center'. Positioned with 'inside' the labels will be placed on half the distance of the radius to the border of the Pie by respecting the 'labelOffset'. The 'outside' option will place the labels at the border of the pie and 'center' will place the labels in the absolute center point of the chart. The 'center' option only makes sense in conjunction with the 'labelOffset' option.\n labelPosition: 'inside',\n // An interpolation function for the label value\n labelInterpolationFnc: Chartist.noop,\n // Label direction can be 'neutral', 'explode' or 'implode'. The labels anchor will be positioned based on those settings as well as the fact if the labels are on the right or left side of the center of the chart. Usually explode is useful when labels are positioned far away from the center.\n labelDirection: 'neutral',\n // If true the whole data is reversed including labels, the series order as well as the whole series data arrays.\n reverseData: false,\n // If true empty values will be ignored to avoid drawing unncessary slices and labels\n ignoreEmptyValues: false\n };\n\n /\*\*\n \* Determines SVG anchor position based on direction and center parameter\n \*\n \* @param center\n \* @param label\n \* @param direction\n \* @return {string}\n \*/\n function determineAnchorPosition(center, label, direction) {\n var toTheRight = label.x > center.x;\n\n if(toTheRight && direction === 'explode' ||\n !toTheRight && direction === 'implode') {\n return 'start';\n } else if(toTheRight && direction === 'implode' ||\n !toTheRight && direction === 'explode') {\n return 'end';\n } else {\n return 'middle';\n }\n }\n\n /\*\*\n \* Creates the pie chart\n \*\n \* @param options\n \*/\n function createChart(options) {\n var data = Chartist.normalizeData(this.data);\n var seriesGroups = [],\n labelsGroup,\n chartRect,\n radius,\n labelRadius,\n totalDataSum,\n startAngle = options.startAngle;\n\n // Create SVG.js draw\n this.svg = Chartist.createSvg(this.container, options.width, options.height,options.donut ? options.classNames.chartDonut : options.classNames.chartPie);\n // Calculate charting rect\n chartRect = Chartist.createChartRect(this.svg, options, defaultOptions.padding);\n // Get biggest circle radius possible within chartRect\n radius = Math.min(chartRect.width() / 2, chartRect.height() / 2);\n // Calculate total of all series to get reference value or use total reference from optional options\n totalDataSum = options.total || data.normalized.series.reduce(function(previousValue, currentValue) {\n return previousValue + currentValue;\n }, 0);\n\n var donutWidth = Chartist.quantity(options.donutWidth);\n if (donutWidth.unit === '%') {\n donutWidth.value \*= radius / 100;\n }\n\n // If this is a donut chart we need to adjust our radius to enable strokes to be drawn inside\n // Unfortunately this is not possible with the current SVG Spec\n // See this proposal for more details: http://lists.w3.org/Archives/Public/www-svg/2003Oct/0000.html\n radius -= options.donut ? donutWidth.value / 2 : 0;\n\n // If labelPosition is set to `outside` or a donut chart is drawn then the label position is at the radius,\n // if regular pie chart it's half of the radius\n if(options.labelPosition === 'outside' || options.donut) {\n labelRadius = radius;\n } else if(options.labelPosition === 'center') {\n // If labelPosition is center we start with 0 and will later wait for the labelOffset\n labelRadius = 0;\n } else {\n // Default option is 'inside' where we use half the radius so the label will be placed in the center of the pie\n // slice\n labelRadius = radius / 2;\n }\n // Add the offset to the labelRadius where a negative offset means closed to the center of the chart\n labelRadius += options.labelOffset;\n\n // Calculate end angle based on total sum and current data value and offset with padding\n var center = {\n x: chartRect.x1 + chartRect.width() / 2,\n y: chartRect.y2 + chartRect.height() / 2\n };\n\n // Check if there is only one non-zero value in the series array.\n var hasSingleValInSeries = data.raw.series.filter(function(val) {\n return val.hasOwnProperty('value') ? val.value !== 0 : val !== 0;\n }).length === 1;\n\n // Creating the series groups\n data.raw.series.forEach(function(series, index) {\n seriesGroups[index] = this.svg.elem('g', null, null);\n }.bind(this));\n //if we need to show labels we create the label group now\n if(options.showLabel) {\n labelsGroup = this.svg.elem('g', null, null);\n }\n\n // Draw the series\n // initialize series groups\n data.raw.series.forEach(function(series, index) {\n // If current value is zero and we are ignoring empty values then skip to next value\n if (data.normalized.series[index] === 0 && options.ignoreEmptyValues) return;\n\n // If the series is an object and contains a name or meta data we add a custom attribute\n seriesGroups[index].attr({\n 'ct:series-name': series.name\n });\n\n // Use series class from series data or if not set generate one\n seriesGroups[index].addClass([\n options.classNames.series,\n (series.className || options.classNames.series + '-' + Chartist.alphaNumerate(index))\n ].join(' '));\n\n // If the whole dataset is 0 endAngle should be zero. Can't divide by 0.\n var endAngle = (totalDataSum > 0 ? startAngle + data.normalized.series[index] / totalDataSum \* 360 : 0);\n\n // Use slight offset so there are no transparent hairline issues\n var overlappigStartAngle = Math.max(0, startAngle - (index === 0 || hasSingleValInSeries ? 0 : 0.2));\n\n // If we need to draw the arc for all 360 degrees we need to add a hack where we close the circle\n // with Z and use 359.99 degrees\n if(endAngle - overlappigStartAngle >= 359.99) {\n endAngle = overlappigStartAngle + 359.99;\n }\n\n var start = Chartist.polarToCartesian(center.x, center.y, radius, overlappigStartAngle),\n end = Chartist.polarToCartesian(center.x, center.y, radius, endAngle);\n\n // Create a new path element for the pie chart. If this isn't a donut chart we should close the path for a correct stroke\n var path = new Chartist.Svg.Path(!options.donut)\n .move(end.x, end.y)\n .arc(radius, radius, 0, endAngle - startAngle > 180, 0, start.x, start.y);\n\n // If regular pie chart (no donut) we add a line to the center of the circle for completing the pie\n if(!options.donut) {\n path.line(center.x, center.y);\n }\n\n // Create the SVG path\n // If this is a donut chart we add the donut class, otherwise just a regular slice\n var pathElement = seriesGroups[index].elem('path', {\n d: path.stringify()\n }, options.donut ? options.classNames.sliceDonut : options.classNames.slicePie);\n\n // Adding the pie series value to the path\n pathElement.attr({\n 'ct:value': data.normalized.series[index],\n 'ct:meta': Chartist.serialize(series.meta)\n });\n\n // If this is a donut, we add the stroke-width as style attribute\n if(options.donut) {\n pathElement.attr({\n 'style': 'stroke-width: ' + donutWidth.value + 'px'\n });\n }\n\n // Fire off draw event\n this.eventEmitter.emit('draw', {\n type: 'slice',\n value: data.normalized.series[index],\n totalDataSum: totalDataSum,\n index: index,\n meta: series.meta,\n series: series,\n group: seriesGroups[index],\n element: pathElement,\n path: path.clone(),\n center: center,\n radius: radius,\n startAngle: startAngle,\n endAngle: endAngle\n });\n\n // If we need to show labels we need to add the label for this slice now\n if(options.showLabel) {\n var labelPosition;\n if(data.raw.series.length === 1) {\n // If we have only 1 series, we can position the label in the center of the pie\n labelPosition = {\n x: center.x,\n y: center.y\n };\n } else {\n // Position at the labelRadius distance from center and between start and end angle\n labelPosition = Chartist.polarToCartesian(\n center.x,\n center.y,\n labelRadius,\n startAngle + (endAngle - startAngle) / 2\n );\n }\n\n var rawValue;\n if(data.normalized.labels && !Chartist.isFalseyButZero(data.normalized.labels[index])) {\n rawValue = data.normalized.labels[index];\n } else {\n rawValue = data.normalized.series[index];\n }\n\n var interpolatedValue = options.labelInterpolationFnc(rawValue, index);\n\n if(interpolatedValue || interpolatedValue === 0) {\n var labelElement = labelsGroup.elem('text', {\n dx: labelPosition.x,\n dy: labelPosition.y,\n 'text-anchor': determineAnchorPosition(center, labelPosition, options.labelDirection)\n }, options.classNames.label).text('' + interpolatedValue);\n\n // Fire off draw event\n this.eventEmitter.emit('draw', {\n type: 'label',\n index: index,\n group: labelsGroup,\n element: labelElement,\n text: '' + interpolatedValue,\n x: labelPosition.x,\n y: labelPosition.y\n });\n }\n }\n\n // Set next startAngle to current endAngle.\n // (except for last slice)\n startAngle = endAngle;\n }.bind(this));\n\n this.eventEmitter.emit('created', {\n chartRect: chartRect,\n svg: this.svg,\n options: options\n });\n }\n\n /\*\*\n \* This method creates a new pie chart and returns an object that can be used to redraw the chart.\n \*\n \* @memberof Chartist.Pie\n \* @param {String|Node} query A selector query string or directly a DOM element\n \* @param {Object} data The data object in the pie chart needs to have a series property with a one dimensional data array. The values will be normalized against each other and don't necessarily need to be in percentage. The series property can also be an array of value objects that contain a value property and a className property to override the CSS class name for the series group.\n \* @param {Object} [options] The options object with options that override the default options. Check the examples for a detailed list.\n \* @param {Array} [responsiveOptions] Specify an array of responsive option arrays which are a media query and options object pair => [[mediaQueryString, optionsObject],[more...]]\n \* @return {Object} An object with a version and an update method to manually redraw the chart\n \*\n \* @example\n \* // Simple pie chart example with four series\n \* new Chartist.Pie('.ct-chart', {\n \* series: [10, 2, 4, 3]\n \* });\n \*\n \* @example\n \* // Drawing a donut chart\n \* new Chartist.Pie('.ct-chart', {\n \* series: [10, 2, 4, 3]\n \* }, {\n \* donut: true\n \* });\n \*\n \* @example\n \* // Using donut, startAngle and total to draw a gauge chart\n \* new Chartist.Pie('.ct-chart', {\n \* series: [20, 10, 30, 40]\n \* }, {\n \* donut: true,\n \* donutWidth: 20,\n \* startAngle: 270,\n \* total: 200\n \* });\n \*\n \* @example\n \* // Drawing a pie chart with padding and labels that are outside the pie\n \* new Chartist.Pie('.ct-chart', {\n \* series: [20, 10, 30, 40]\n \* }, {\n \* chartPadding: 30,\n \* labelOffset: 50,\n \* labelDirection: 'explode'\n \* });\n \*\n \* @example\n \* // Overriding the class names for individual series as well as a name and meta data.\n \* // The name will be written as ct:series-name attribute and the meta data will be serialized and written\n \* // to a ct:meta attribute.\n \* new Chartist.Pie('.ct-chart', {\n \* series: [{\n \* value: 20,\n \* name: 'Series 1',\n \* className: 'my-custom-class-one',\n \* meta: 'Meta One'\n \* }, {\n \* value: 10,\n \* name: 'Series 2',\n \* className: 'my-custom-class-two',\n \* meta: 'Meta Two'\n \* }, {\n \* value: 70,\n \* name: 'Series 3',\n \* className: 'my-custom-class-three',\n \* meta: 'Meta Three'\n \* }]\n \* });\n \*/\n function Pie(query, data, options, responsiveOptions) {\n Chartist.Pie.super.constructor.call(this,\n query,\n data,\n defaultOptions,\n Chartist.extend({}, defaultOptions, options),\n responsiveOptions);\n }\n\n // Creating pie chart type in Chartist namespace\n Chartist.Pie = Chartist.Base.extend({\n constructor: Pie,\n createChart: createChart,\n determineAnchorPosition: determineAnchorPosition\n });\n\n}(window, document, Chartist));\n\nreturn Chartist;\n\n}));\n"]}