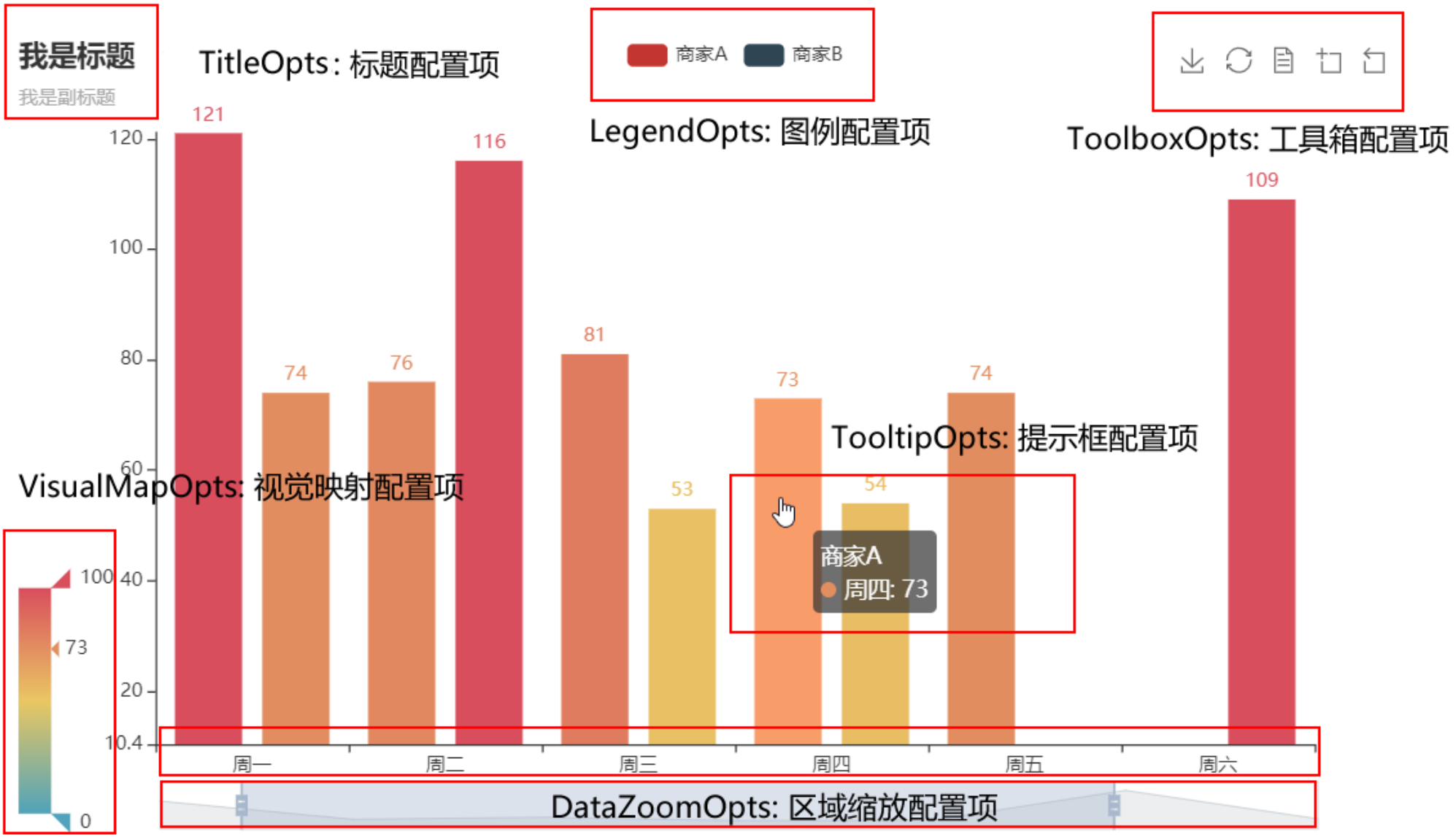
全局配置项

全局配置项可通过 `set_global_opts` 方法设置



AnimationOpts：Echarts 画图动画配置项

class pyecharts.options.Animation

```
class AnimationOpts(
    # 是否开启动画，默认为 True 开启。
    animation: bool = True,

    # 是否开启动画的阈值，当单个系列显示的图形数量大于这个阈值时会关闭动画。默认 2000。
    animation_threshold: Numeric = 2000,

    # 初始动画的时长，默认值为 1000。
    # 支持回调函数，可以通过每个数据返回不同的 delay 时间实现更戏剧的初始动画效果：
    animation_duration: Union[Numeric, JSFunc] = 1000,

    # 初始动画的缓动效果。
    # 不同的缓动效果可以参考，缓动示例 (https://www.echartsjs.com/gallery/editor.html?c=line-easing)。
    animation_easing: Union[str] = "cubicOut",

    # 初始动画的延迟，默认值为 0。
    # 支持回调函数，可以通过每个数据返回不同的 delay 时间实现更戏剧的初始动画效果。
    animation_delay: Union[Numeric, JSFunc] = 0,

    # 数据更新动画的时长，默认值为 300。
    # 支持回调函数，可以通过每个数据返回不同的 delay 时间实现更戏剧的更新动画效果：
    animation_duration_update: Union[Numeric, JSFunc] = 300,
```

```
    # 数据更新动画的缓动效果。
    # 不同的缓动效果可以参考，缓动示例 (https://www.echartsjs.com/gallery/editor.html?c=line-easing)。
    animation_easing_update: Union[Numeric] = "cubicOut",

    # 数据更新动画的延迟，默认值为 0。
    # 支持回调函数，可以通过每个数据返回不同的 delay 时间实现更戏剧的更新动画效果。
    animation_delay_update: Union[Numeric, JSFunc] = 0,
)
```

InitOpts：初始化配置项

class pyecharts.options.InitOpts

```
class InitOpts(
    # 图表画布宽度，css 长度单位。
    width: str = "900px",

    # 图表画布高度，css 长度单位。
    height: str = "500px",

    # 图表 ID，图表唯一标识，用于在多图表时区分。
    chart_id: Optional[str] = None,

    # 渲染风格，可选 "canvas", "svg"
    # # 参考 `全局变量` 章节
    renderer: str = RenderType.CANVAS,
```

```
    # 网页标题
    page_title: str = "Awesome-pyecharts",

    # 图表主题
    theme: str = "white",

    # 图表背景颜色
    bg_color: Optional[str] = None,

    # 远程 js host，如不设置默认为 https://assets.pyecharts.org/assets/"
    # 参考 `全局变量` 章节
    js_host: str = "",

    # 画图动画初始化配置，参考 `global_options.AnimationOpts`
    animation_opts: Union[AnimationOpts, dict] = AnimationOpts(),
)
```

ToolBoxFeatureSaveAsImagesOpts：工具箱保存图片配置项

class pyecharts.options.ToolBoxFeatureSaveAsImagesOpts

```
class ToolBoxFeatureSaveAsImageOpts(
    # 保存的图片格式。支持 'png' 和 'jpeg'。
    type_: str = "png",

    # 保存的文件名称，默认使用 title.text 作为名称。
    name: Optional[str] = None,
```

```
    # 保存的图片背景色，默认使用 backgroundColor，如果backgroundColor不存在的话会取白色。
    background_color: str = "auto",

    # 如果图表使用了 echarts.connect 对多个图表进行联动，则在导出图片时会导出这些联动的图表。该配置项决定了图表与图表之间间隙处的填充色。
    connected_background_color: str = "#fff",

    # 保存为图片时忽略的组件列表，默认忽略工具栏。
    exclude_components: Optional[Sequence[str]] = None,

    # 是否显示该工具。
    is_show: bool = True,

    # 提示语
    title: str = "保存为图片",

    # 可以通过 'image://url' 设置为图片，其中 URL 为图片的链接，或者 dataURI。
    icon: Optional[JSFunc] = None,

    # 保存图片的分辨率比例，默认跟容器相同大小，如果需要保存更高分辨率的，可以设置为大于 1 的值，例如 2。
    pixel_ratio: Numeric = 1,
):
```

ToolBoxFeatureRestoreOpts：工具箱还原配置项

class pyecharts.options.ToolBoxFeatureRestoreOpts

```
class ToolBoxFeatureRestoreOpts(
    # 是否显示该工具。
    is_show: bool = True,

    # 提示语
    title: str = "还原",

    # 可以通过 'image://url' 设置为图片，其中 URL 为图片的链接，或者 dataURI。
    icon: Optional[JSFunc] = None
):
```

ToolBoxFeatureDataViewOpts：工具箱数据视图工具

class pyecharts.options.ToolBoxFeatureDataViewOpts

```
class ToolBoxFeatureDataViewOpts(
    # 是否显示该工具。
    is_show: bool = True,

    # 提示语
    title: str = "还原",

    # 可以通过 'image://url' 设置为图片，其中 URL 为图片的链接，或者 dataURI。
    icon: Optional[JSFunc] = None
```

```
# 是否不可编辑（只读）。默认为 False
is_read_only: bool = False,

# 自定义 dataView 展现函数，用以取代默认的 textarea 使用更丰富的数据编辑。可以返回 dom 对象或者 html 字符串。
option_to_content: Optional[JSFunc] = None,

# 在使用 optionToContent 的情况下，如果支持数据编辑后的刷新，需要自行通过该函数实现组装 option 的逻辑。
content_to_option: Optional[JSFunc] = None,

# 数据视图上有三个话术，默认是['数据视图', '关闭', '刷新']。
lang: Optional[Sequence[str]] = None,

# 数据视图浮层背景色。
background_color: str = "#fff",

# 数据视图浮层文本输入区背景色。
text_area_color: str = "#fff",

# 数据视图浮层文本输入区边框颜色。
text_area_border_color: str = "#333",

# 文本颜色。
text_color: str = "#000",

# 按钮颜色。
button_color: str = "#c23531",

# 按钮文本颜色。
```

```
    button_text_color: str = "#fff",
):
```

ToolBoxFeatureDataZoomOpts：工具箱区域缩放配置项

class pyecharts.options.ToolBoxFeatureDataZoomOpts

```
class ToolBoxFeatureDataZoomOpts(
    # 是否显示该工具。
    is_show: bool = True,

    # 提示语
    zoom_title: str = "区域缩放",

    # 提示语
    back_title: str = "区域缩放还原",

    # 可以通过 'image://url' 设置为图片，其中 URL 为图片的链接，或者 dataURI。
    zoom_icon: Optional[JSFunc] = None,

    # 可以通过 'image://url' 设置为图片，其中 URL 为图片的链接，或者 dataURI。
    back_icon: Optional[JSFunc] = None,

    # 指定哪些 xAxis 被控制。如果缺省则控制所有的 x 轴。
    # 如果设置为 false 则不控制任何x轴。如果设置成 3 则控制 axisIndex 为 3 的 x 轴。
    # 如果设置为 [0, 3] 则控制 axisIndex 为 0 和 3 的 x 轴。
    xaxis_index: Union[Numeric, Sequence, bool] = None,
```

```
    # 指定哪些 yAxis 被控制。如果缺省则控制所有的 y 轴。
    # 如果设置为 false 则不控制任何y轴。如果设置成 3 则控制 axisIndex 为 3 的 y 轴。
    # 如果设置为 [0, 3] 则控制 axisIndex 为 0 和 3 的 y 轴。
    yaxis_index: Union[Numeric, Sequence, bool] = None,

    # dataZoom 的运行原理是通过数据过滤以及在内部设置轴的显示窗口来达到数据窗口缩放的效果。
    # 'filter': 当前数据窗口外的数据，被过滤掉。即会影响其他轴的数据范围。
    #  每个数据项，只要有一个维度在数据窗口外，整个数据项就会被过滤掉。
    # 'weakFilter': 当前数据窗口外的数据，被过滤掉。即会影响其他轴的数据范围。
    #  每个数据项，只有当全部维度都在数据窗口同侧外部，整个数据项才会被过滤掉。
    # 'empty': 当前数据窗口外的数据，被设置为空。即不会影响其他轴的数据范围。
    # 'none': 不过滤数据，只改变数轴范围。
    filter_mode: str = "filter",
):
```

ToolBoxFeatureMagicTypeOpts：工具箱动态类型切换配置项

class pyecharts.options.ToolBoxFeatureMagicTypeOpts

```
class ToolBoxFeatureMagicTypeOpts(
    # 是否显示该工具。
    is_show: bool = True,

    # 启用的动态类型
    # 包括'line'（切换为折线图），'bar'（切换为柱状图），
    # 'stack'（切换为堆叠模式），'tiled'（切换为平铺模式）。
```

```
    type_: Optional[Sequence] = None,

    # 各个类型的标题文本，可以分别配置。
    line_title: str = "切换为折线图",

    # 各个类型的标题文本，可以分别配置。
    bar_title: str = "切换为柱状图",

    # 各个类型的标题文本，可以分别配置。
    stack_title: str = "切换为堆叠",

    # 各个类型的标题文本，可以分别配置。
    tiled_title: str = "切换为平铺",

    # 各个类型的 icon path，可以分别配置。
    line_icon: Optional[JSFunc] = None,

    # 各个类型的 icon path，可以分别配置。
    bar_icon: Optional[JSFunc] = None,

    # 各个类型的 icon path，可以分别配置。
    stack_icon: Optional[JSFunc] = None,

    # 各个类型的 icon path，可以分别配置。
    tiled_icon: Optional[JSFunc] = None,
):
```

ToolBoxFeatureBrushOpts：工具箱选框组件配置项

class pyecharts.options.ToolBoxFeatureBrushOpts

```
class ToolBoxFeatureBrushOpts(
    # 使用的按钮，取值：
    # 'rect'：开启矩形选框选择功能。
    # 'polygon'：开启任意形状选框选择功能。
    # 'lineX'：开启横向选择功能。
    # 'lineY'：开启纵向选择功能。
    # 'keep'：切换『单选』和『多选』模式。后者可支持同时画多个选框。前者支持单击清除所有选框。
    # 'clear'：清空所有选框。
    type_: Optional[str] = None,

    # 每个按钮的 icon path。
    rect_icon: Optional[JSFunc] = None,

    # 每个按钮的 icon path。
    polygon_icon: Optional[JSFunc] = None,

    # 每个按钮的 icon path。
    line_x_icon: Optional[JSFunc] = None,

    # 每个按钮的 icon path。
    line_y_icon: Optional[JSFunc] = None,

    # 每个按钮的 icon path。
    keep_icon: Optional[JSFunc] = None,

    # 每个按钮的 icon path。
```

```
    clear_icon: Optional[JSFunc] = None,

    # 标题文本。
    rect_title: str = "矩形选择",

    # 标题文本。
    polygon_title: str = "圈选",

    # 标题文本。
    line_x_title: str = "横向选择",

    # 标题文本。
    line_y_title: str = "纵向选择",

    # 标题文本。
    keep_title: str = "保持选择",

    # 标题文本。
    clear_title: str = "清除选择",
):
```

ToolBoxFeatureOpts：工具箱工具配置项

class pyecharts.options.ToolBoxFeatureOpts

```
class ToolBoxFeatureOpts(
    # 保存为图片
```

```
    save_as_image: Union[ToolBoxFeatureSaveAsImageOpts, dict] = ToolBoxFeatureSaveAsImageOpts(),

    # 配置项还原
    restore: Union[ToolBoxFeatureRestoreOpts, dict] = ToolBoxFeatureRestoreOpts(),

    # 数据视图工具，可以展现当前图表所用的数据，编辑后可以动态更新
    data_view: Union[ToolBoxFeatureDataViewOpts, dict] = ToolBoxFeatureDataViewOpts(),

    # 数据区域缩放。（目前只支持直角坐标系的缩放）
    data_zoom: Union[ToolBoxFeatureDataZoomOpts, dict] = ToolBoxFeatureDataZoomOpts(),

    # 动态类型切换。
    magic_type: Union[ToolBoxFeatureMagicTypeOpts, dict] = ToolBoxFeatureMagicTypeOpts(),

    # 选框组件的控制按钮。
    brush: Union[ToolBoxFeatureBrushOpts, dict] = ToolBoxFeatureBrushOpts(),
)
```

ToolboxOpts：工具箱配置项

class pyecharts.options.ToolboxOpts

```
class ToolboxOpts(
    # 是否显示工具栏组件
    is_show: bool = True,

    # 工具栏 icon 的布局朝向。
```

```
    # 可选：'horizontal', 'vertical'
    orient: str = "horizontal",

    # 工具栏组件离容器左侧的距离。
    # left 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比，
    # 也可以是 'left', 'center', 'right'。
    # 如果 left 的值为'left', 'center', 'right'，组件会根据相应的位置自动对齐
    pos_left: str = "80%",

    # 工具栏组件离容器右侧的距离。
    # right 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比。
    pos_right: Optional[str] = None,

    # 工具栏组件离容器上侧的距离。
    # top 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比，
    # 也可以是 'top', 'middle', 'bottom'。
    # 如果 top 的值为'top', 'middle', 'bottom'，组件会根据相应的位置自动对齐。
    pos_top: Optional[str] = None,

    # 工具栏组件离容器下侧的距离。
    # bottom 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比。
    pos_bottom: Optional[str] = None,

    # 各工具配置项，参考 `global_options.ToolBoxFeatureOpts`
    feature: Union[ToolBoxFeatureOpts, dict] = ToolBoxFeatureOpts(),
)
```

BrushOpts：区域选择组件配置项

class pyecharts.options.BrushOpts

```
class BrushOpts(
    # 使用在 toolbox 中的按钮。默认值为 ["rect", "polygon", "keep", "clear"]
    # brush 相关的 toolbox 按钮有：
    # "rect"：开启矩形选框选择功能。
    # "polygon"：开启任意形状选框选择功能。
    # "lineX"：开启横向选择功能。
    # "lineY"'：开启纵向选择功能。
    # "keep"：切换『单选』和『多选』模式。后者可支持同时画多个选框。前者支持单击清除所有选框。
    # "clear"：清空所有选框。
    tool_box: Optional[Sequence] = None,

    # 不同系列间，选中的项可以联动。
    # brush_link 配置项是一个列表，内容是 seriesIndex，指定了哪些 series 可以被联动。
    # 例如可以是：
    # [3, 4, 5] 表示 seriesIndex 为 3, 4, 5 的 series 可以被联动。
    # "all" 表示所有 series 都进行 brushLink。
    # None 表示不启用 brush_link 功能。
    brush_link: Union[Sequence, str] = None,

    # 指定哪些 series 可以被刷选，可取值为：
    # "all": 所有 series
    # series index 列表，如 [0, 4, 2]，表示指定这些 index 所对应的坐标系。
    # 某个 series index，如 0，表示这个 index 所对应的坐标系。
    series_index: Union[Sequence, Numeric, str] = None,

    # 指定哪些 geo 可以被刷选。可以设置 brush 是全局的还是属于坐标系的。
```

```
# 全局 brush
# 在 echarts 实例中任意地方刷选。这是默认情况。如果没有指定为坐标系 brush，就是全局 brush。
# 坐标系 brush
# 在指定的坐标系中刷选。选框可以跟随坐标系的缩放和平移（ roam 和 dataZoom ）而移动。
# 坐标系 brush 实际更为常用，尤其是在 geo 中。
# 通过指定 brush.geoIndex 或 brush.xAxisIndex 或 brush.yAxisIndex 来规定可以在哪些坐标系中进行刷选。
# 指定哪些 series 可以被刷选，可取值为：
# "all"：表示所有 series
# series index 列表，如 [0, 4, 2]，表示指定这些 index 所对应的坐标系。
# 某个 series index，如 0，表示这个 index 所对应的坐标系。
geo_index: Union[Sequence, Numeric, str] = None,

# 指定哪些 xAxisIndex 可以被刷选。可以设置 brush 是全局的还是属于坐标系的。
# 全局 brush
# 在 echarts 实例中任意地方刷选。这是默认情况。如果没有指定为坐标系 brush，就是全局 brush。
# 坐标系 brush
# 在指定的坐标系中刷选。选框可以跟随坐标系的缩放和平移（ roam 和 dataZoom ）而移动。
# 坐标系 brush 实际更为常用，尤其是在 geo 中。
# 通过指定 brush.geoIndex 或 brush.xAxisIndex 或 brush.yAxisIndex 来规定可以在哪些坐标系中进行刷选。
# 指定哪些 series 可以被刷选，可取值为：
# "all"：表示所有 series
# series index 列表，如 [0, 4, 2]，表示指定这些 index 所对应的坐标系。
# 某个 series index，如 0，表示这个 index 所对应的坐标系。
x_axis_index: Union[Sequence, Numeric, str] = None,

# 指定哪些 yAxisIndex 可以被刷选。可以设置 brush 是全局的还是属于坐标系的。
# 全局 brush
# 在 echarts 实例中任意地方刷选。这是默认情况。如果没有指定为坐标系 brush，就是全局 brush。
# 坐标系 brush
```

```
# 在指定的坐标系中刷选。选框可以跟随坐标系的缩放和平移（ roam 和 dataZoom ）而移动。
# 坐标系 brush 实际更为常用，尤其是在 geo 中。
# 通过指定 brush.geoIndex 或 brush.xAxisIndex 或 brush.yAxisIndex 来规定可以在哪些坐标系中进行刷选。
# 指定哪些 series 可以被刷选，可取值为：
# "all"：表示所有 series
# series index 列表，如 [0, 4, 2]，表示指定这些 index 所对应的坐标系。
# 某个 series index，如 0，表示这个 index 所对应的坐标系。
y_axis_index: Union[Sequence, Numeric, str] = None,

# 默认的刷子类型。默认值为 rect。
# 可选参数如下：
# "rect"：矩形选框。
# "polygon"：任意形状选框。
# "lineX"：横向选择。
# "lineY"：纵向选择。
brush_type: str = "rect",

# 默认的刷子的模式。可取值为：
# 默认为 single
# "single"：单选。
# "multiple"：多选。
brush_mode: str = "single",

# 已经选好的选框是否可以被调整形状或平移。默认值为 True
transformable: bool = True,

# 选框的默认样式，值为
# {
#       "borderWidth": 1,
```

```
#       "color": "rgba(120,140,180,0.3)",
#       "borderColor": "rgba(120,140,180,0.8)"
# },
brush_style: Optional[dict] = None,

# 默认情况，刷选或者移动选区的时候，会不断得发 brushSelected 事件，从而告诉外界选中的内容。
# 但是频繁的事件可能导致性能问题，或者动画效果很差。
# 所以 brush 组件提供了 brush.throttleType，brush.throttleDelay 来解决这个问题。
# throttleType 取值可以是：
# "debounce"：表示只有停止动作了（即一段时间没有操作了），才会触发事件。时间阈值由 brush.throttleDelay 指定。
# "fixRate"：表示按照一定的频率触发事件，时间间隔由 brush.throttleDelay 指定。
throttle_type: str = "fixRate",

# 默认为 0 表示不开启 throttle。
throttle_delay: Numeric = 0,

# 在 brush_mode 为 "single" 的情况下，是否支持单击清除所有选框。
remove_on_click: bool = True,

# 定义在选中范围外的视觉元素。最终参数以字典的形式进行配置
# 可选的视觉元素有：
# symbol: 图元的图形类别。
# symbolSize: 图元的大小。
# color: 图元的颜色。
# colorAlpha: 图元的颜色的透明度。
# opacity: 图元以及其附属物（如文字标签）的透明度。
# colorLightness: 颜色的明暗度，参见 https://en.wikipedia.org/wiki/HSL_and_HSV。
# colorSaturation: 颜色的饱和度，参见 https://en.wikipedia.org/wiki/HSL_and_HSV。
# colorHue: 颜色的色调，参见 https://en.wikipedia.org/wiki/HSL_and_HSV。
```

```
        out_of_brush: dict = None,
    )
```

TitleOpts：标题配置项

class pyecharts.options.TitleOpts

```
class TitleOpts(
    # 主标题文本，支持使用 \n 换行。
    title: Optional[str] = None,

    # 主标题跳转 URL 链接
    title_link: Optional[str] = None,

    # 主标题跳转链接方式
    # 默认值是: blank
    # 可选参数: 'self', 'blank'
    # 'self' 当前窗口打开; 'blank' 新窗口打开
    title_target: Optional[str] = None,

    # 副标题文本，支持使用 \n 换行。
    subtitle: Optional[str] = None,

    # 副标题跳转 URL 链接
    subtitle_link: Optional[str] = None,

    # 副标题跳转链接方式
```

```
# 默认值是: blank
# 可选参数: 'self', 'blank'
# 'self' 当前窗口打开; 'blank' 新窗口打开
subtitle_target: Optional[str] = None,

# title 组件离容器左侧的距离。
# left 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比，
# 也可以是 'left', 'center', 'right'。
# 如果 left 的值为'left', 'center', 'right'，组件会根据相应的位置自动对齐。
pos_left: Optional[str] = None,

# title 组件离容器右侧的距离。
# right 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比。
pos_right: Optional[str] = None,

# title 组件离容器上侧的距离。
# top 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比，
# 也可以是 'top', 'middle', 'bottom'。
# 如果 top 的值为'top', 'middle', 'bottom'，组件会根据相应的位置自动对齐。
pos_top: Optional[str] = None,

# title 组件离容器下侧的距离。
# bottom 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比。
pos_bottom: Optional[str] = None,

# 标题内边距，单位px，默认各方向内边距为5，接受数组分别设定上右下左边距。
# // 设置内边距为 5
# padding: 5
# // 设置上下的内边距为 5，左右的内边距为 10
```

```
    # padding: [5, 10]
    # // 分别设置四个方向的内边距
    # padding: [
    #     5,  // 上
    #     10, // 右
    #     5,  // 下
    #     10, // 左
    # ]
    padding: Union[Sequence, Numeric] = 5,

    # 主副标题之间的间距。
    item_gap: Numeric = 10,

    # 主标题字体样式配置项，参考 `series_options.TextStyleOpts`
    title_textstyle_opts: Union[TextStyleOpts, dict, None] = None,

    # 副标题字体样式配置项，参考 `series_options.TextStyleOpts`
    subtitle_textstyle_opts: Union[TextStyleOpts, dict, None] = None,
)
```

DataZoomOpts：区域缩放配置项

class pyecharts.options.DataZoomOpts

```
class DataZoomOpts(
    # 是否显示 组件。如果设置为 false，不会显示，但是数据过滤的功能还存在。
    is_show: bool = True,
```

```
# 组件类型，可选 "slider", "inside"
type_: str = "slider",

# 拖动时，是否实时更新系列的视图。如果设置为 false，则只在拖拽结束的时候更新。
is_realtime: bool = True,

# 数据窗口范围的起始百分比。范围是：0 ~ 100。表示 0% ~ 100%。
range_start: Union[Numeric, None] = 20,

# 数据窗口范围的结束百分比。范围是：0 ~ 100
range_end: Union[Numeric, None] = 80,

# 数据窗口范围的起始数值。如果设置了 start 则 startValue 失效。
start_value: Union[int, str, None] = None,

# 数据窗口范围的结束数值。如果设置了 end 则 endValue 失效。
end_value: Union[int, str, None] = None,

# 布局方式是横还是竖。不仅是布局方式，对于直角坐标系而言，也决定了，缺省情况控制横向数轴还是纵向数轴
# 可选值为：'horizontal', 'vertical'
orient: str = "horizontal",

# 设置 dataZoom-inside 组件控制的 x 轴（即 xAxis，是直角坐标系中的概念，参见 grid）。
# 不指定时，当 dataZoom-inside.orient 为 'horizontal'时，默认控制和 dataZoom 平行的第一个 xAxis
# 如果是 number 表示控制一个轴，如果是 Array 表示控制多个轴。
xaxis_index: Union[int, Sequence[int], None] = None,

# 设置 dataZoom-inside 组件控制的 y 轴（即 yAxis，是直角坐标系中的概念，参见 grid）。
```

```
# 不指定时，当 dataZoom-inside.orient 为 'horizontal'时，默认控制和 dataZoom 平行的第一个 yAxis
# 如果是 number 表示控制一个轴，如果是 Array 表示控制多个轴。
yaxis_index: Union[int, Sequence[int], None] = None,

# 是否锁定选择区域（或叫做数据窗口）的大小。
# 如果设置为 true 则锁定选择区域的大小，也就是说，只能平移，不能缩放。
is_zoom_lock: bool = False,

# dataZoom-slider 组件离容器左侧的距离。
# left 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比，
# 也可以是 'left', 'center', 'right'。
# 如果 left 的值为 'left', 'center', 'right'，组件会根据相应的位置自动对齐。
pos_left: Optional[str] = None,

# dataZoom-slider 组件离容器上侧的距离。
# top 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比，
# 也可以是 'top', 'middle', 'bottom'。
# 如果 top 的值为 'top', 'middle', 'bottom'，组件会根据相应的位置自动对齐。
pos_top: Optional[str] = None,

# dataZoom-slider 组件离容器右侧的距离。
# right 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比。
# 默认自适应。
pos_right: Optional[str] = None,

# dataZoom-slider组件离容器下侧的距离。
# bottom 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比。
# 默认自适应。
pos_bottom: Optional[str] = None,
```

```
    # dataZoom 的运行原理是通过数据过滤以及在内部设置轴的显示窗口来达到数据窗口缩放的效果。
    # 'filter'：当前数据窗口外的数据，被过滤掉。即会影响其他轴的数据范围。
    #  每个数据项，只要有一个维度在数据窗口外，整个数据项就会被过滤掉。
    # 'weakFilter'：当前数据窗口外的数据，被过滤掉。即会影响其他轴的数据范围。
    #  每个数据项，只有当全部维度都在数据窗口同侧外部，整个数据项才会被过滤掉。
    # 'empty'：当前数据窗口外的数据，被设置为空。即不会影响其他轴的数据范围。
    # 'none'：不过滤数据，只改变数轴范围。
    filter_mode: str = "filter"
)
```

LegendOpts：图例配置项

class pyecharts.options.LegendOpts

```
class LegendOpts(
    # 图例的类型。可选值：
    # 'plain'：普通图例。缺省就是普通图例。
    # 'scroll'：可滚动翻页的图例。当图例数量较多时可以使用。
    type_: Optional[str] = None,

    # 图例选择的模式，控制是否可以通过点击图例改变系列的显示状态。默认开启图例选择，可以设成 false 关闭
    # 除此之外也可以设成 'single' 或者 'multiple' 使用单选或者多选模式。
    selected_mode: Union[str, bool, None] = None,

    # 是否显示图例组件
    is_show: bool = True,
```

```
# 图例组件离容器左侧的距离。
# left 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比，
# 也可以是 'left', 'center', 'right'。
# 如果 left 的值为'left', 'center', 'right'，组件会根据相应的位置自动对齐。
pos_left: Union[str, Numeric, None] = None,

# 图例组件离容器右侧的距离。
# right 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比。
pos_right: Union[str, Numeric, None] = None,

# 图例组件离容器上侧的距离。
# top 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比，
# 也可以是 'top', 'middle', 'bottom'。
# 如果 top 的值为'top', 'middle', 'bottom'，组件会根据相应的位置自动对齐。
pos_top: Union[str, Numeric, None] = None,

# 图例组件离容器下侧的距离。
# bottom 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比。
pos_bottom: Union[str, Numeric, None] = None,

# 图例列表的布局朝向。可选：'horizontal', 'vertical'
orient: Optional[str] = None,

# 图例标记和文本的对齐。默认自动（auto）
# 根据组件的位置和 orient 决定
# 当组件的 left 值为 'right' 以及纵向布局（orient 为 'vertical'）的时候为右对齐，即为 'right'。
# 可选参数：`auto`, `left`, `right`
align: Optional[str] = None,
```

```
    # 图例内边距，单位px，默认各方向内边距为5
    padding: int = 5,

    # 图例每项之间的间隔。横向布局时为水平间隔，纵向布局时为纵向间隔。
    # 默认间隔为 10
    item_gap: int = 10,

    # 图例标记的图形宽度。默认宽度为 25
    item_width: int = 25,

    # 图例标记的图形高度。默认高度为 14
    item_height: int = 14,

    # 图例关闭时的颜色。默认是 #ccc
    inactive_color: Optional[str] = None,

    # 图例组件字体样式，参考 `series_options.TextStyleOpts`
    textstyle_opts: Union[TextStyleOpts, dict, None] = None,

    # 图例项的 icon。
    # ECharts 提供的标记类型包括 'circle', 'rect', 'roundRect', 'triangle', 'diamond', 'pin', 'arrow', 'none'
    # 可以通过 'image://url' 设置为图片，其中 URL 为图片的链接，或者 dataURI。
    # 可以通过 'path://' 将图标设置为任意的矢量路径。
    legend_icon: Optional[str] = None,
)
```

VisualMapOpts：视觉映射配置项

class pyecharts.options.VisualMapOpts

```
class VisualMapOpts(
    # 是否显示视觉映射配置
    is_show: bool = True,

    # 映射过渡类型，可选，"color", "size"
    type_: str = "color",

    # 指定 visualMapPiecewise 组件的最小值。
    min_: Union[int, float] = 0,

    # 指定 visualMapPiecewise 组件的最大值。
    max_: Union[int, float] = 100,

    # 两端的文本，如['High', 'Low']。
    range_text: Union[list, tuple] = None,

    # visualMap 组件过渡颜色
    range_color: Union[Sequence[str]] = None,

    # visualMap 组件过渡 symbol 大小
    range_size: Union[Sequence[int]] = None,

    # visualMap 图元以及其附属物（如文字标签）的透明度。
    range_opacity: Optional[Numeric] = None,

    # 如何放置 visualMap 组件，水平（'horizontal'）或者竖直（'vertical'）。
```

```
orient: str = "vertical",

# visualMap 组件离容器左侧的距离。
# left 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比，
# 也可以是 'left', 'center', 'right'。
# 如果 left 的值为'left', 'center', 'right'，组件会根据相应的位置自动对齐。
pos_left: Optional[str] = None,

# visualMap 组件离容器右侧的距离。
# right 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比。
pos_right: Optional[str] = None,

# visualMap 组件离容器上侧的距离。
# top 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比，
# 也可以是 'top', 'middle', 'bottom'。
# 如果 top 的值为'top', 'middle', 'bottom'，组件会根据相应的位置自动对齐。
pos_top: Optional[str] = None,

# visualMap 组件离容器下侧的距离。
# bottom 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比。
pos_bottom: Optional[str] = None,

# 对于连续型数据，自动平均切分成几段。默认为5段。连续数据的范围需要 max 和 min 来指定
split_number: int = 5,

# 指定取哪个系列的数据，默认取所有系列。
series_index: Union[Numeric, Sequence, None] = None,

# 组件映射维度
```

```
dimension: Optional[Numeric] = None,

# 是否显示拖拽用的手柄（手柄能拖拽调整选中范围）。
is_calculable: bool = True,

# 是否为分段型
is_piecewise: bool = False,

# 是否反转 visualMap 组件
is_inverse: bool = False,

# 数据展示的小数精度。
# 连续型数据平均分段，精度根据数据自动适应。
# 连续型数据自定义分段或离散数据根据类别分段模式，精度默认为0（没有小数）。
precision: Optional[int] = None,

# 自定义的每一段的范围，以及每一段的文字，以及每一段的特别的样式。例如：
# pieces: [
#   {"min": 1500}, // 不指定 max，表示 max 为无限大（Infinity）。
#   {"min": 900, "max": 1500},
#   {"min": 310, "max": 1000},
#   {"min": 200, "max": 300},
#   {"min": 10, "max": 200, "label": '10 到 200（自定义label）'},
#   {"value": 123, "label": '123（自定义特殊颜色）', "color": 'grey'}, //表示 value 等于 123 的情况
#   {"max": 5}     // 不指定 min，表示 min 为无限大（-Infinity）。
# ]
pieces: Optional[Sequence] = None,

# 定义 在选中范围外 的视觉元素。（用户可以和 visualMap 组件交互，用鼠标或触摸选择范围）
```

```
    #  可选的视觉元素有：
    #  symbol: 图元的图形类别。
    #  symbolSize: 图元的大小。
    #  color: 图元的颜色。
    #  colorAlpha: 图元的颜色的透明度。
    #  opacity: 图元以及其附属物（如文字标签）的透明度。
    #  colorLightness: 颜色的明暗度，参见 HSL。
    #  colorSaturation: 颜色的饱和度，参见 HSL。
    #  colorHue: 颜色的色调，参见 HSL。
    out_of_range: Optional[Sequence] = None,

    # 图形的宽度，即长条的宽度。
    item_width: int = 0,

    # 图形的高度，即长条的高度。
    item_height: int = 0,

    # visualMap 组件的背景色。
    background_color: Optional[str] = None,

    # visualMap 组件的边框颜色。
    border_color: Optional[str] = None,

    # visualMap 边框线宽，单位px。
    border_width: int = 0,

    # 文字样式配置项，参考 `series_options.TextStyleOpts`
    textstyle_opts: Union[TextStyleOpts, dict, None] = None,
)
```

TooltipOpts：提示框配置项

class pyecharts.options.TooltipOpts

```
class TooltipOpts(
    # 是否显示提示框组件，包括提示框浮层和 axisPointer。
    is_show: bool = True,

    # 触发类型。可选：
    # 'item'：数据项图形触发，主要在散点图，饼图等无类目轴的图表中使用。
    # 'axis'：坐标轴触发，主要在柱状图，折线图等会使用类目轴的图表中使用。
    # 'none'：什么都不触发
    trigger: str = "item",

    # 提示框触发的条件，可选：
    # 'mousemove'：鼠标移动时触发。
    # 'click'：鼠标点击时触发。
    # 'mousemove|click'：同时鼠标移动和点击时触发。
    # 'none'：不在 'mousemove' 或 'click' 时触发，
    trigger_on: str = "mousemove|click",

    # 指示器类型。可选
    # 'line'：直线指示器
    # 'shadow'：阴影指示器
    # 'none'：无指示器
    # 'cross'：十字准星指示器。其实是种简写，表示启用两个正交的轴的 axisPointer。
```

```
axis_pointer_type: str = "line",

# 是否显示提示框浮层，默认显示。
# 只需 tooltip 触发事件或显示 axisPointer 而不需要显示内容时可配置该项为 false。
is_show_content: bool = True,

# 是否永远显示提示框内容，
# 默认情况下在移出可触发提示框区域后一定时间后隐藏，设置为 true 可以保证一直显示提示框内容。
is_always_show_content: bool = False,

# 浮层显示的延迟，单位为 ms，默认没有延迟，也不建议设置。
show_delay: Numeric = 0,

# 浮层隐藏的延迟，单位为 ms，在 alwaysShowContent 为 true 的时候无效。
hide_delay: Numeric = 100,

# 提示框浮层的位置，默认不设置时位置会跟随鼠标的位置。
# 1、通过数组配置：
# 绝对位置，相对于容器左侧 10px, 上侧 10 px ===> position: [10, 10]
# 相对位置，放置在容器正中间 ===> position: ['50%', '50%']
# 2、通过回调函数配置
# 3、固定参数配置：'inside', 'top', 'left', 'right', 'bottom'
position: Union[str, Sequence, JSFunc] = None,

# 标签内容格式器，支持字符串模板和回调函数两种形式，字符串模板与回调函数返回的字符串均支持用 \n 换行。
# 字符串模板 模板变量有：
# {a}：系列名。
# {b}：数据名。
# {c}：数据值。
```

```
# {@xxx}：数据中名为 'xxx' 的维度的值，如 {@product} 表示名为 'product'` 的维度的值。
# {@[n]}：数据中维度 n 的值，如{@[3]}` 表示维度 3 的值，从 0 开始计数。
# 示例：formatter: '{b}: {@score}'
#
# 回调函数，回调函数格式：
# (params: Object|Array) => string
# 参数 params 是 formatter 需要的单个数据集。格式如下：
# {
#    componentType: 'series',
#    // 系列类型
#    seriesType: string,
#    // 系列在传入的 option.series 中的 index
#    seriesIndex: number,
#    // 系列名称
#    seriesName: string,
#    // 数据名，类目名
#    name: string,
#    // 数据在传入的 data 数组中的 index
#    dataIndex: number,
#    // 传入的原始数据项
#    data: Object,
#    // 传入的数据值
#    value: number|Array,
#    // 数据图形的颜色
#    color: string,
# }
formatter: Optional[str] = None,

# 提示框浮层的背景颜色。
```

```
    background_color: Optional[str] = None,

    # 提示框浮层的边框颜色。
    border_color: Optional[str] = None,

    # 提示框浮层的边框宽。
    border_width: Numeric = 0,

    # 文字样式配置项，参考 `series_options.TextStyleOpts`
    textstyle_opts: TextStyleOpts = TextStyleOpts(font_size=14),
)
```

AxisLineOpts: 坐标轴轴线配置项

> ***class pyecharts.option.AxisLineOpts***

```
class AxisLineOpts(
    # 是否显示坐标轴轴线。
    is_show: bool = True,

    # X 轴或者 Y 轴的轴线是否在另一个轴的 0 刻度上，只有在另一个轴为数值轴且包含 0 刻度时有效。
    is_on_zero: bool = True,

    # 当有双轴时，可以用这个属性手动指定，在哪个轴的 0 刻度上。
    on_zero_axis_index: int = 0,

    # 轴线两边的箭头。可以是字符串，表示两端使用同样的箭头；或者长度为 2 的字符串数组，分别表示两端的箭头。
```

```
    # 默认不显示箭头，即 'none'。
    # 两端都显示箭头可以设置为 'arrow'。
    # 只在末端显示箭头可以设置为 ['none', 'arrow']。
    symbol: Optional[str] = None,

    # 坐标轴线风格配置项，参考 `series_optionsLineStyleOpts`
    linestyle_opts: Union[LineStyleOpts, dict, None] = None,
)
```

AxisTickOpts: 坐标轴刻度配置项

class pyecharts.option.AxisTickOpts

```
class AxisTickOpts(
    # 是否显示坐标轴刻度。
    is_show: bool = True,

    # 类目轴中在 boundaryGap 为 true 的时候有效，可以保证刻度线和标签对齐。
    is_align_with_label: bool = False,

    # 坐标轴刻度是否朝内，默认朝外。
    is_inside: bool = False,

    # 坐标轴刻度的长度。
    length: Optional[Numeric] = None,

    # 坐标轴线风格配置项，参考 `series_optionsLineStyleOpts`
```

```
    linestyle_opts: Union[LineStyleOpts, dict, None] = None,
)
```

AxisPointerOpts: 坐标轴指示器配置项

class pyecharts.option.AxisPointerOpts

```
class AxisPointerOpts(
    # 默认显示坐标轴指示器
    is_show: bool = True,

    # 不同轴的 axisPointer 可以进行联动，在这里设置。联动表示轴能同步一起活动。
    # 轴依据他们的 axisPointer 当前对应的值来联动。
    # link 是一个数组，其中每一项表示一个 link group，一个 group 中的坐标轴互相联动。
    # 具体使用方式可以参见：https://www.echartsjs.com/option.html#axisPointer.link
    link: Sequence[dict] = None,

    # 指示器类型。
    # 可选参数如下，默认为 'line'
    # 'line' 直线指示器
    # 'shadow' 阴影指示器
    # 'none' 无指示器
    type_: str = "line",

    # 坐标轴指示器的文本标签，坐标轴标签配置项，参考 `series_options.LabelOpts`
    label: Union[LabelOpts, dict, None] = None,
```

```
    # 坐标轴线风格配置项，参考 `series_optionsLineStyleOpts`
    linestyle_opts: Union[LineStyleOpts, dict, None] = None,
)
```

AxisOpts：坐标轴配置项

class pyecharts.options.AxisOpts

```
class AxisOpts(
    # 坐标轴类型。可选：
    # 'value'：数值轴，适用于连续数据。
    # 'category'：类目轴，适用于离散的类目数据，为该类型时必须通过 data 设置类目数据。
    # 'time'：时间轴，适用于连续的时序数据，与数值轴相比时间轴带有时间的格式化，在刻度计算上也有所不同，
    # 例如会根据跨度的范围来决定使用月，星期，日还是小时范围的刻度。
    # 'log' 对数轴。适用于对数数据。
    type_: Optional[str] = None,

    # 坐标轴名称。
    name: Optional[str] = None,

    # 是否显示 x 轴。
    is_show: bool = True,

    # 只在数值轴中（type: 'value'）有效。
    # 是否是脱离 0 值比例。设置成 true 后坐标刻度不会强制包含零刻度。在双数值轴的散点图中比较有用。
    # 在设置 min 和 max 之后该配置项无效。
    is_scale: bool = False,
```

```
# 是否反向坐标轴。
is_inverse: bool = False,

# 坐标轴名称显示位置。可选：
# 'start', 'middle' 或者 'center','end'
name_location: str = "end",

# 坐标轴名称与轴线之间的距离。
name_gap: Numeric = 15,

# 坐标轴名字旋转，角度值。
name_rotate: Optional[Numeric] = None,

# 强制设置坐标轴分割间隔。
# 因为 splitNumber 是预估的值，实际根据策略计算出来的刻度可能无法达到想要的效果，
# 这时候可以使用 interval 配合 min、max 强制设定刻度划分，一般不建议使用。
# 无法在类目轴中使用。在时间轴（type: 'time'）中需要传时间戳，在对数轴（type: 'log'）中需要传指数值。
interval: Optional[Numeric] = None,

# x 轴所在的 grid 的索引，默认位于第一个 grid。
grid_index: Numeric = 0,

# x 轴的位置。可选：
# 'top', 'bottom'
# 默认 grid 中的第一个 x 轴在 grid 的下方（'bottom'），第二个 x 轴视第一个 x 轴的位置放在另一侧。
position: Optional[str] = None,

# Y 轴相对于默认位置的偏移，在相同的 position 上有多个 Y 轴的时候有用。
```

```
offset: Numeric = 0,

# 坐标轴的分割段数，需要注意的是这个分割段数只是个预估值，最后实际显示的段数会在这个基础上根据分割后坐标轴刻度显示的易读程度作调整。
# 默认值是 5
split_number: Numeric = 5,

# 坐标轴两边留白策略，类目轴和非类目轴的设置和表现不一样。
# 类目轴中 boundaryGap 可以配置为 true 和 false。默认为 true，这时候刻度只是作为分隔线，
# 标签和数据点都会在两个刻度之间的带(band)中间。
# 非类目轴，包括时间，数值，对数轴，boundaryGap是一个两个值的数组，分别表示数据最小值和最大值的延伸范围
# 可以直接设置数值或者相对的百分比，在设置 min 和 max 后无效。 示例：boundaryGap: ['20%', '20%']
boundary_gap: Union[str, bool, None] = None,

# 坐标轴刻度最小值。
# 可以设置成特殊值 'dataMin'，此时取数据在该轴上的最小值作为最小刻度。
# 不设置时会自动计算最小值保证坐标轴刻度的均匀分布。
# 在类目轴中，也可以设置为类目的序数（如类目轴 data: ['类A', '类B', '类C'] 中，序数 2 表示 '类C'。
# 也可以设置为负数，如 -3）。
min_: Union[Numeric, str, None] = None,

# 坐标轴刻度最大值。
# 可以设置成特殊值 'dataMax'，此时取数据在该轴上的最大值作为最大刻度。
# 不设置时会自动计算最大值保证坐标轴刻度的均匀分布。
# 在类目轴中，也可以设置为类目的序数（如类目轴 data: ['类A', '类B', '类C'] 中，序数 2 表示 '类C'。
# 也可以设置为负数，如 -3）。
max_: Union[Numeric, str, None] = None,

# 自动计算的坐标轴最小间隔大小。
# 例如可以设置成1保证坐标轴分割刻度显示成整数。
```

```
    # 默认值是 0
    min_interval: Numeric = 0,

    # 自动计算的坐标轴最大间隔大小。
    # 例如，在时间轴（（type: 'time'））可以设置成 3600 * 24 * 1000 保证坐标轴分割刻度最大为一天。
    max_interval: Optional[Numeric] = None,

    # 坐标轴刻度线配置项，参考 `global_options.AxisLineOpts`
    axisline_opts: Union[AxisLineOpts, dict, None] = None,

    # 坐标轴刻度配置项，参考 `global_options.AxisTickOpts`
    axistick_opts: Union[AxisTickOpts, dict, None] = None,

    # 坐标轴标签配置项，参考 `series_options.LabelOpts`
    axislabel_opts: Union[LabelOpts, dict, None] = None,

    # 坐标轴指示器配置项，参考 `global_options.AxisPointerOpts`
    axispointer_opts: Union[AxisPointerOpts, dict, None] = None,

    # 坐标轴名称的文字样式，参考 `series_options.TextStyleOpts`
    name_textstyle_opts: Union[TextStyleOpts, dict, None] = None,

    # 分割区域配置项，参考 `series_options.SplitAreaOpts`
    splitarea_opts: Union[SplitAreaOpts, dict, None] = None,

    # 分割线配置项，参考 `series_options.SplitLineOpts`
    splitline_opts: Union[SplitLineOpts, dict] = SplitLineOpts(),

    # 坐标轴次刻度线相关设置，参考 `series_options.MinorTickOpts`
```

```
    minor_tick_opts: Union[MinorTickOpts, dict, None] = None,

    # 坐标轴在 grid 区域中的次分隔线。次分割线会对齐次刻度线 minorTick，参考 `series_options.MinorSplitLineOpts`
    minor_split_line_opts: Union[MinorSplitLineOpts, dict, None] = None,
)
```

SingleAxisOpts：单轴配置项

class pyecharts.SingleAxisOpts

```
class SingleAxisOpts(
    # 坐标轴名称。
    name: Optional[str] = None,

    # 坐标轴刻度最大值。
    # 可以设置成特殊值 'dataMax'，此时取数据在该轴上的最大值作为最大刻度。
    # 不设置时会自动计算最大值保证坐标轴刻度的均匀分布。
    # 在类目轴中，也可以设置为类目的序数（如类目轴 data: ['类A', '类B', '类C'] 中，序数 2 表示 '类C'。
    # 也可以设置为负数，如 -3）。
    max_: Union[str, Numeric, None] = None,

    # 坐标轴刻度最小值。
    # 可以设置成特殊值 'dataMin'，此时取数据在该轴上的最小值作为最小刻度。
    # 不设置时会自动计算最小值保证坐标轴刻度的均匀分布。
    # 在类目轴中，也可以设置为类目的序数（如类目轴 data: ['类A', '类B', '类C'] 中，序数 2 表示 '类C'。
    # 也可以设置为负数，如 -3）。
    min_: Union[str, Numeric, None] = None,
```

```
# single 组件离容器左侧的距离。
# left 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比，
# 也可以是 'left', 'center', 'right'。
# 如果 left 的值为'left', 'center', 'right'，组件会根据相应的位置自动对齐。
pos_left: Optional[str] = None,

# single组件离容器右侧的距离。
# right 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比。
pos_right: Optional[str] = None,

# single组件离容器上侧的距离。
# top 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比，
# 也可以是 'top', 'middle', 'bottom'。
# 如果 top 的值为'top', 'middle', 'bottom'，组件会根据相应的位置自动对齐。
pos_top: Optional[str] = None,

# single组件离容器下侧的距离。
# bottom 的值可以是像 20 这样的具体像素值，可以是像 '20%' 这样相对于容器高宽的百分比。
pos_bottom: Optional[str] = None,

# single 组件的宽度。默认自适应。
width: Optional[str] = None,

# single 组件的高度。默认自适应。
height: Optional[str] = None,

# 轴的朝向，默认水平朝向，可以设置成 'vertical' 垂直朝向。
orient: Optional[str] = None,
```

```
    # 坐标轴类型。可选：
    # 'value'：数值轴，适用于连续数据。
    # 'category'：类目轴，适用于离散的类目数据，为该类型时必须通过 data 设置类目数据。
    # 'time'：时间轴，适用于连续的时序数据，与数值轴相比时间轴带有时间的格式化，在刻度计算上也有所不同，
    # 例如会根据跨度的范围来决定使用月，星期，日还是小时范围的刻度。
    # 'log' 对数轴。适用于对数数据。
    type_: Optional[str] = None,
)
```

GraphicGroup：原生图形元素组件

class pyecharts.GraphicGroup

```
class GraphicGroup(
    # 图形的配置项
    graphic_item: Union[GraphicItem, dict, None] = None,

    # 根据其 children 中每个图形元素的 name 属性进行重绘
    is_diff_children_by_name: bool = False,

    # 子节点列表，其中项都是一个图形元素定义。
    # 目前可以选择 GraphicText，GraphicImage，GraphicRect
    children: Optional[Sequence[BaseGraphic]] = None,
)
```

GraphicItem：原生图形配置项

class pyecharts.GraphicItem

```
class GraphicItem(
    # id 用于在更新或删除图形元素时指定更新哪个图形元素，如果不需要用可以忽略。
    id_: Optional[str] = None,

    # setOption 时指定本次对该图形元素的操作行为。可选：
    #  'merge'：如果已有元素，则新的配置项和已有的设定进行 merge。如果没有则新建。
    #  'replace'：如果已有元素，删除之，新建元素替代之。
    #  'remove'：删除元素。
    action: str = "merge",

    # 平移（position）：默认值是 [0, 0]。表示 [横向平移的距离, 纵向平移的距离]。右和下为正值。
    position: [Sequence, Numeric, None] = None,

    # 旋转（rotation）：默认值是 0。表示旋转的弧度值。正值表示逆时针旋转。
    rotation: Union[Numeric, JSFunc, None] = 0,

    # 缩放（scale）：默认值是 [1, 1]。表示 [横向缩放的倍数, 纵向缩放的倍数]。
    scale: Union[Sequence, Numeric, None] = None,

    # origin 指定了旋转和缩放的中心点，默认值是 [0, 0]。
    origin: Union[Numeric, Sequence, None] = None,

    # 描述怎么根据父元素进行定位。
    # 父元素是指：如果是顶层元素，父元素是 echarts 图表容器。如果是 group 的子元素，父元素就是 group 元素。
```

```
# 值的类型可以是：
# 数值：表示像素值。
# 百分比值：如 '33%'，用父元素的高和此百分比计算出最终值。
# 位置：如 'center'：表示自动居中。
# 注：left 和 right 只有一个可以生效。如果指定 left 或 right，则 shape 里的 x、cx 等定位属性不再生效。
left: Union[Numeric, str, None] = None,

# 同上
right: Union[Numeric, str, None] = None,

# 配置和 left 及 right 相同， 注：top 和 bottom 只有一个可以生效。
top: Union[Numeric, str, None] = None,

# 同上
bottom: Union[Numeric, str, None] = None,

# 决定此图形元素在定位时，对自身的包围盒计算方式。可选：
# 'all'：（默认） 表示用自身以及子节点整体的经过 transform 后的包围盒进行定位。这种方式易于使整体都限制在父元素范围中。
# 'raw'：表示仅仅用自身（不包括子节点）的没经过 tranform 的包围盒进行定位。这种方式易于内容超出父元素范围的定位方式。
bounding: str = "all",

# z 方向的高度，决定层叠关系。
z: Numeric = 0,

# 决定此元素绘制在哪个 canvas 层中。注意，越多 canvas 层会占用越多资源。
z_level: Numeric = 0,

# 是否不响应鼠标以及触摸事件。
is_silent: bool = False,
```

```
    # 节点是否可见。
    is_invisible: bool = False,

    # 节点是否完全被忽略（既不渲染，也不响应事件）。
    is_ignore: bool = False,

    # 鼠标悬浮时在图形元素上时鼠标的样式是什么。同 CSS 的 cursor。
    cursor: str = "pointer",

    # 图形元素是否可以被拖拽。
    is_draggable: bool = False,

    # 是否渐进式渲染。当图形元素过多时才使用。
    is_progressive: bool = False,

    # 用于描述此 group 的宽。这个宽只用于给子节点定位。
    # 即便当宽度为零的时候，子节点也可以使用 left: 'center' 相对于父节点水平居中。
    width: Numeric = 0,

    # 用于描述此 group 的高。这个高只用于给子节点定位。
    # 即便当高度为零的时候，子节点也可以使用 top: 'middle' 相对于父节点垂直居中。
    height: Numeric = 0,
)
```

GraphicBasicStyleOpts：原生图形基础配置项

class pyecharts.GraphicBasicStyleOpts

```
class GraphicBasicStyleOpts(
    # 填充色。
    fill: str = "#000",

    # 笔画颜色。
    stroke: Optional[str] = None,

    # 笔画宽度。
    line_width: Numeric = 0,

    # 阴影宽度。
    shadow_blur: Optional[Numeric] = None,

    # 阴影 x 方向偏移。
    shadow_offset_x: Optional[Numeric] = None,

    # 阴影 y 方向偏移。
    shadow_offset_y: Optional[Numeric] = None,

    # 阴影颜色。
    shadow_color: Optional[str] = None,
)
```

GraphicShapeOpts：原生图形形状配置项

class pyecharts.GraphicShapeOpts

```
class GraphicShapeOpts(
    # 图形元素的左上角在父节点坐标系（以父节点左上角为原点）中的横坐标值。
    pos_x: Numeric = 0,

    # 图形元素的左上角在父节点坐标系（以父节点左上角为原点）中的横坐标值。
    pos_y: Numeric = 0,

    # 图形元素的宽度。
    width: Numeric = 0,

    # 图形元素的高度。
    height: Numeric = 0,

    # 可以用于设置圆角矩形。r: [r1, r2, r3, r4]，  左上、右上、右下、左下角的半径依次为r1、r2、r3、r4。
    #  可以缩写，例如：
    #  r 缩写为 1 相当于 [1, 1, 1, 1]
    #  r 缩写为 [1] 相当于 [1, 1, 1, 1]
    #  r 缩写为 [1, 2] 相当于 [1, 2, 1, 2]
    #  r 缩写为 [1, 2, 3]1 相当于[1, 2, 3, 2]`
    r: Union[Sequence, Numeric, None] = None,
)
```

GraphicImage：原生图形图片配置项

class pyecharts.GraphicImage

```
class GraphicImage(
    # 图形的配置项，参考 GraphicItem
    graphic_item: Union[GraphicItem, dict, None] = None,

    # 图形图片样式的配置项
    graphic_imagestyle_opts: Union[GraphicImageStyleOpts, dict, None] = None,
)
```

GraphicImageStyleOpts：原生图形图片样式配置项

class pyecharts.GraphicImageStyleOpts

```
class GraphicImageStyleOpts(
    # 图片的内容，可以是图片的 URL。
    image: Optional[str] = None,

    # 图形元素的左上角在父节点坐标系（以父节点左上角为原点）中的横坐标值。
    pos_x: Numeric = 0,

    # 图形元素的左上角在父节点坐标系（以父节点左上角为原点）中的纵坐标值。
    pos_y: Numeric = 0,

    # 图形元素的宽度。
```

```
    width: Numeric = 0,

    # 图形元素的高度。
    height: Numeric = 0,

    # 透明度 0 到 1。1 即完整显示
    opacity: Numeric = 1,

    # 图形基本配置项，参考 GraphicBasicStyleOpts
    graphic_basicstyle_opts: Union[GraphicBasicStyleOpts, dict, None] = None,
)
```

GraphicText：原生图形文本配置项

class pyecharts.GraphicText

```
class GraphText(
    # 图形的配置项，参考 GraphicItem
    graphic_item: Union[GraphicItem, dict, None] = None,

    # 图形文本样式的配置项
    graphic_textstyle_opts: Union[GraphicTextStyleOpts, dict, None] = None,
)
```

GraphicTextStyleOpts：原生图形文本样式配置项

class pyecharts.GraphicTextStyleOpts

```
class GraphicTextStyleOpts(
    # 文本块文字。可以使用 \n 来换行。
    text: Optional[JSFunc] = None,

    # 图形元素的左上角在父节点坐标系（以父节点左上角为原点）中的横坐标值。
    pos_x: Numeric = 0,

    # 图形元素的左上角在父节点坐标系（以父节点左上角为原点）中的纵坐标值。
    pos_y: Numeric = 0,

    # 字体大小、字体类型、粗细、字体样式。
    # 例如：
    # // size | family
    # font: '2em "STHeiti", sans-serif'
    # // style | weight | size | family
    # font: 'italic bolder 16px cursive'
    # // weight | size | family
    # font: 'bolder 2em "Microsoft YaHei", sans-serif'
    font: Optional[str] = None,

    # 水平对齐方式，取值：'left', 'center', 'right'。默认值为：'left'
    # 如果为 'left'，表示文本最左端在 x 值上。如果为 'right'，表示文本最右端在 x 值上。
    text_align: str = "left",
```

```
    # 垂直对齐方式，取值：'top', 'middle', 'bottom'。默认值为：'None'
    text_vertical_align: Optional[str] = None,

    # 图形基本配置项，参考 GraphicBasicStyleOpts
    graphic_basicstyle_opts: Union[GraphicBasicStyleOpts, dict, None] = None,
)
```

GraphicRect：原生图形矩形配置项

class pyecharts.GraphicRect

```
class GraphicRect(
    # 图形的配置项，参考 GraphicItem
    graphic_item: Union[GraphicItem, dict, None] = None,

    # 图形的形状配置项，参考 GraphicShapeOpts
    graphic_shape_opts: Union[GraphicShapeOpts, dict, None] = None,

    # 图形基本配置项，参考 GraphicBasicStyleOpts
    graphic_basicstyle_opts: Union[GraphicBasicStyleOpts, dict, None] = None,
)
```

PolarOpts：极坐标系配置

class pyecharts.PolarOpts

```
class PolarOpts(
    # 极坐标系的中心（圆心）坐标，数组的第一项是横坐标，第二项是纵坐标。
    # 支持设置成百分比，设置成百分比时第一项是相对于容器宽度，第二项是相对于容器高度。
    center: Optional[Sequence] = None,

    # 极坐标系的半径。可以为如下类型：
    # number：直接指定外半径值。
    # string：例如，'20%'，表示外半径为可视区尺寸（容器高宽中较小一项）的 20% 长度。
    # Array.<number|string>：数组的第一项是内半径，第二项是外半径。每一项遵从上述 number string 的描述。
    radius: Optional[Union[Sequence, str]] = None,

    # 本坐标系特定的 tooltip 设定。参考 `global_options.TooltipOpts`
    tooltip_opts: TooltipOpts = None,
)
```