PyCon China 2024

For Good . For fun.

2024/11/23 中国 上海

PYCHINA
Python 中国社区

PyCon China 2024

>> 2024/11/23 上海

>> For good . For fun.

深入浅出 pyecharts

不深入一下了解，怎么了解 pyecharts 是啥呢?

孙海林 / Leo

- pyecharts 核心维护者（二代目）
- Python / Golang 开发者，Lua / CUDA 学习中
- 折腾了几年 K8S 和 NVIDIA GPU 基础设施
- 在一家全球化快递公司搞了几年 DL（AI） 落地
- 最近开始折腾 AIGC（有兴趣可以一起聊聊）

目录

- Why Echarts？ Why pyecharts?
- 深入 pyecharts
- 浅出 pyecharts
- pyecharts 的未来

Why Echarts?

Apache ECharts 是一个由百度开源的**高性能**、**跨平台**、**多图表**数据可视化工具库，凭借着良好的交互性，精巧的图表设计，得到了众多开发者的认可。

2018 年捐赠给 Apache 基金会，2021 年 1 月成为 Apache 顶级项目

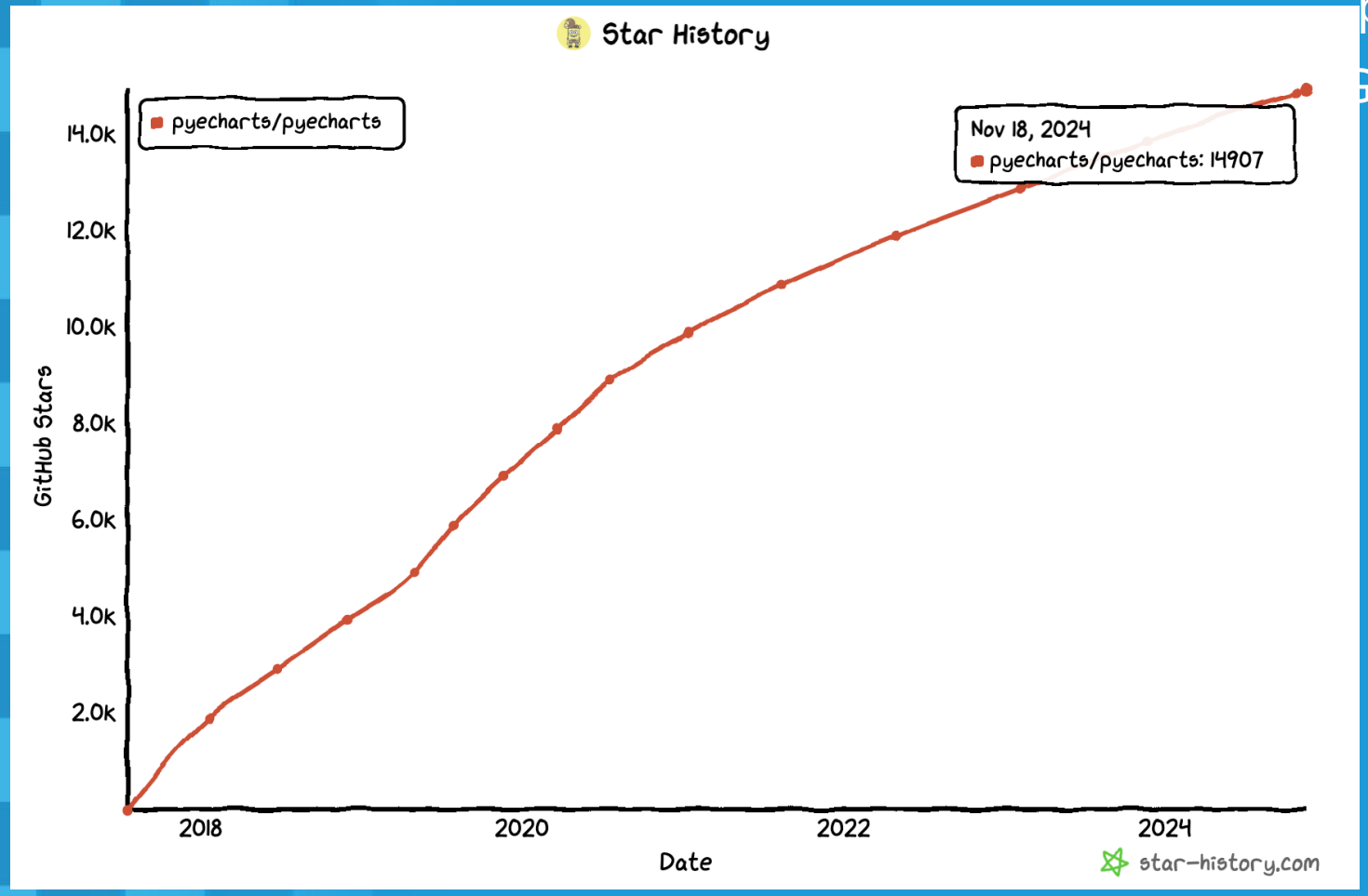
Why pyecharts?

Python 是一门富有表达力的语言，很适合用于数据处理。当数据分析遇上数据可视化时，pyecharts 诞生了。

Python ❤ ⑧ B@e^oqp : mvb`e^oqp

pyecharts 官网：**https://pyecharts.org**

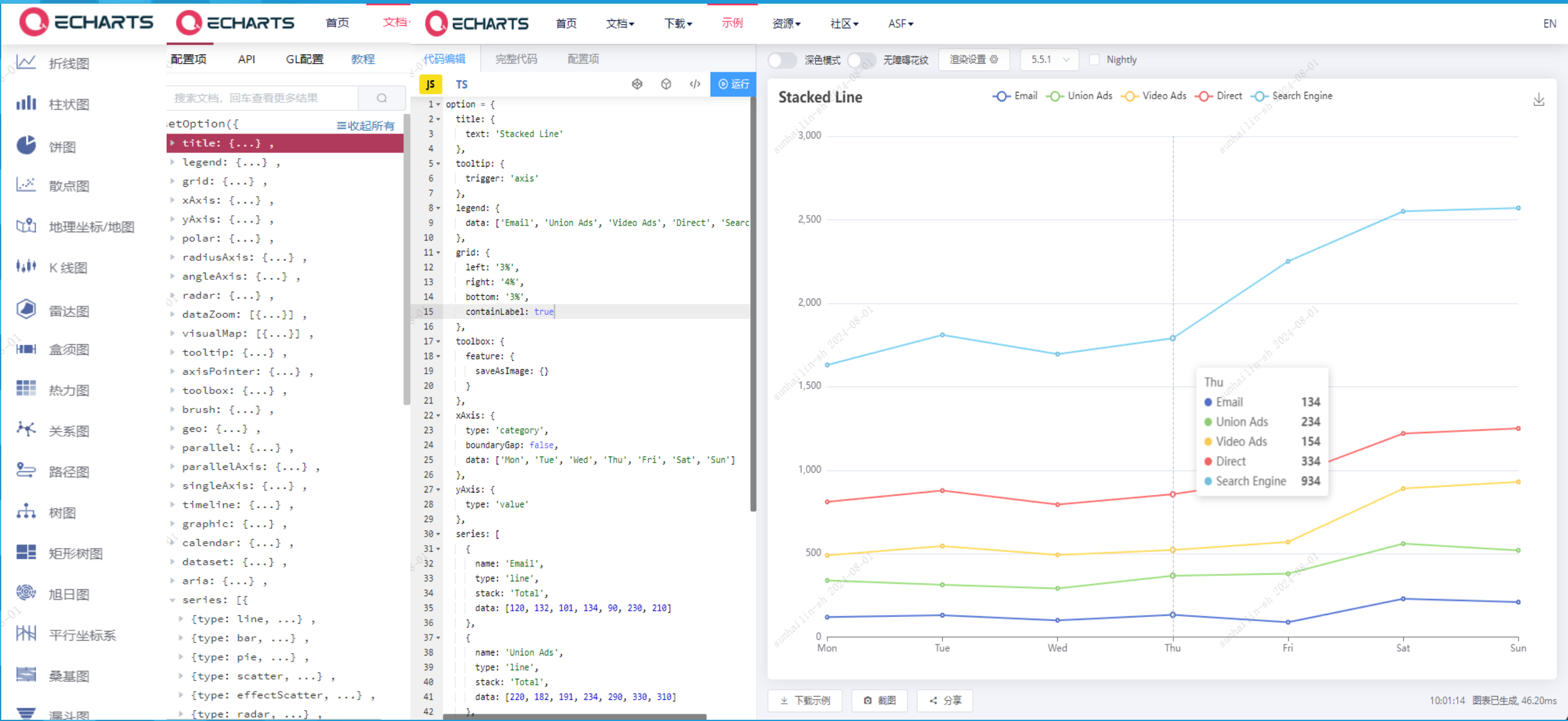
Gallery： **https://gallery.pyecharts.org**



深入 pyecharts

看看 Echarts 怎么做的?

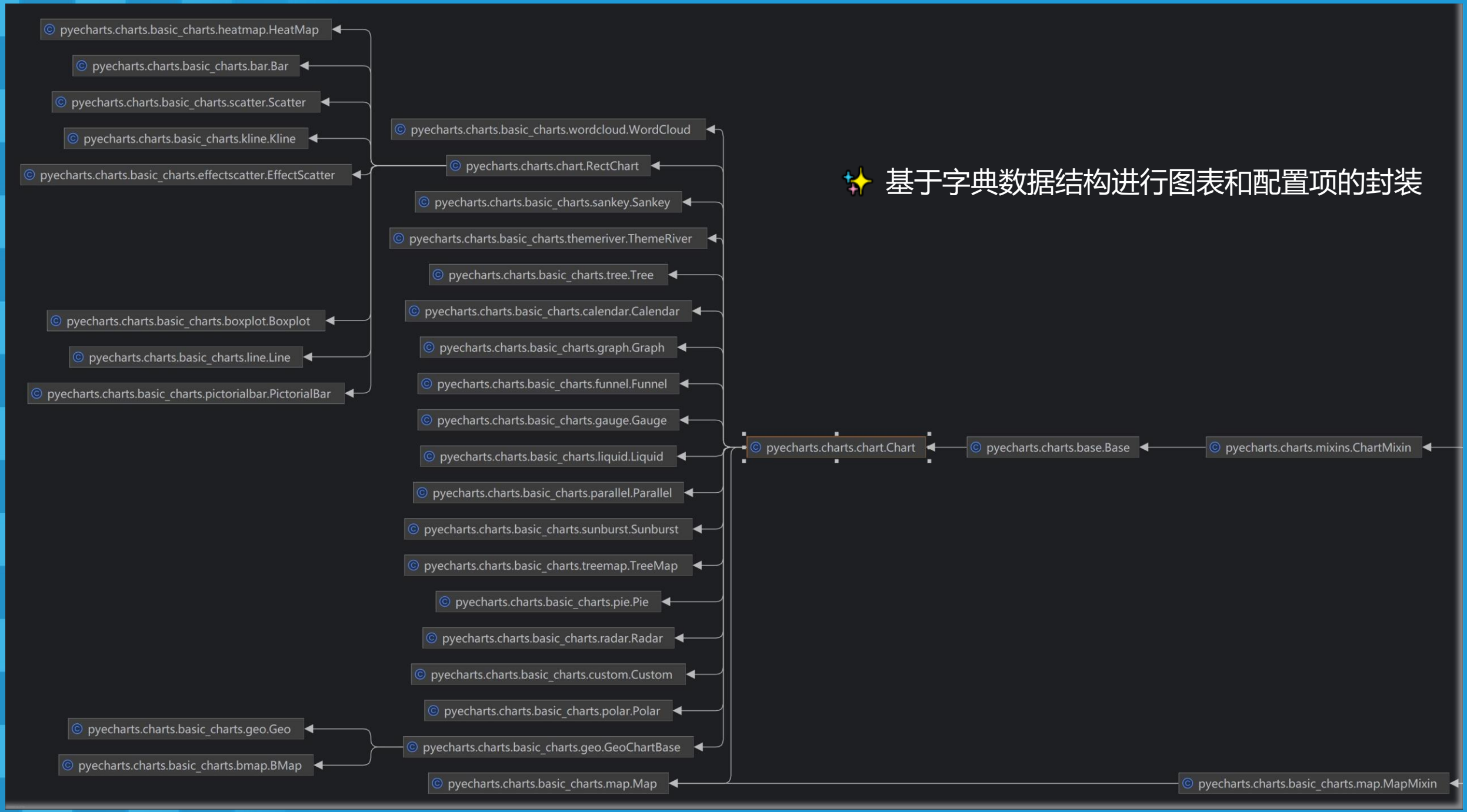
ECharts 核心通过 JSONObject 的方式对图表进行配置，并结合 ZRender 进行渲染



再看看 pyecharts

底层基于 jinja 渲染引擎和预设的 HTML 模板进行图表渲染。支持 jupyter 和 Web 框架的集成

- 基本图表（日历图，饼图，词云图等）
- 直角坐标系图表（柱状图，折线图，散点图等）
- 树型图表（树图、矩形树图）
- 地理图表（地理坐标系，地图等）
- 3D 图表（3D 柱状图，3D 散点图等）
- 组合图表（并行多图，时间轴轮播等）
- HTML 组件（表格，图片）



✨ 基于字典数据结构进行图表和配置项的封装

pyecharts 渲染流程

```python
class Base(ChartMixin):  20 usages
    def render(
        self,
        path: str = "render.html",
        template_name: str = "simple_chart.html",
        env: Optional[Environment] = None,
        **kwargs,
    ) -> str:
        self._prepare_render()
        return engine.render(self, path, template_name, env, **kwargs)
```

渲染前处理

```python
def _prepare_render(self):  4 usages
    self.json_contents = self.dump_options()
    self._use_theme()

    self._render_cache.clear()
    if self.render_options.get("embed_js"):
        self._render_cache[
            "javascript"
        ] = self.load_javascript().load_javascript_contents()
```

添加需要用到的 JS 和 CSS 依赖

```python
def get_options(self) -> dict:  2 usages
    return utils.remove_key_with_none_value(self.options)

def dump_options(self) -> str:  7 usages (5 dynamic)
    return utils.replace_placeholder(
        json.dumps(self.get_options(), indent=4, default=defa
    )

def dump_options_with_quotes(self) -> str:  1 usage
    return utils.replace_placeholder_with_quotes(
        json.dumps(self.get_options(), indent=4, default=defau
    )
```

深度遍历配置项，将 None 值配置进行过滤处理

```python
def _expand(dict_generator):  3 usages
    return dict(list(dict_generator))


def _clean_dict(mydict):...


def _clean_array(myarray):...


def remove_key_with_none_value(incoming_di
    if isinstance(incoming_dict, dict):
        return _expand(_clean_dict(incoming_dict))
    elif incoming_dict:
        return incoming_dict
    else:
        return None
```

pyecharts 渲染流程（二）

```python
class Base(ChartMixin):  20 usages
    def render(
        self,
        path: str = "render.html",
        template_name: str = "simple_chart.html",
        env: Optional[Environment] = None,
        **kwargs,
    ) -> str:
        self._prepare_render()
        return engine.render(self, path, template_name, env, **kwargs)
```

使用内置模版进行渲染

simple_chart.html

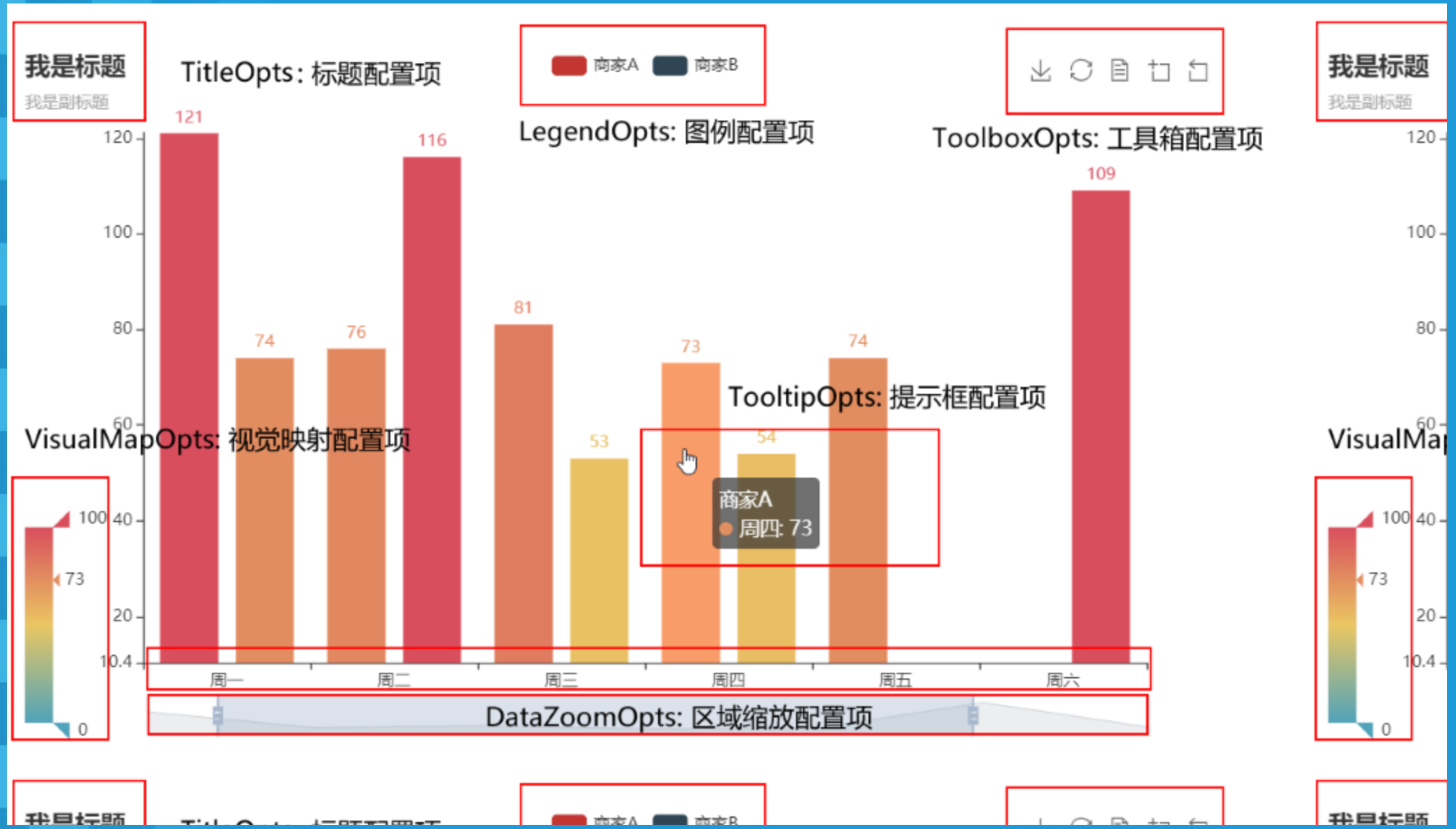
```jinja
{% import 'macro' as macro %}
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>{{ chart.page_title }}</title>
    {{ macro.render_chart_dependencies(chart) }}
</head>
<body {% if chart.fill_bg %}style="..."{% endif %}>
    {{ macro.render_chart_content(chart) }}
</body>
</html>
```

使用 jinja 语法渲染 HTML

macro

```jinja
{%- macro render_chart_content(c) -%}
    <div id="{{ c.chart_id }}" class="chart-container" style="width:{{ c.width }}; height:{{ c.height }}; {{ c.horiz
    {% if c._geo_json_name and c._geo_json %}
    <script>
        (function (root, factory) {
            if (typeof define === 'function' && define.amd) {
                // AMD. Register as an anonymous module.
                define(['exports', 'echarts'], factory);
            } else if (typeof exports === 'object' && typeof exports.nodeName !== 'string') {
                // CommonJS
```

pyecharts 组件示意图



浅出 pyecharts

pyecharts 基础操作

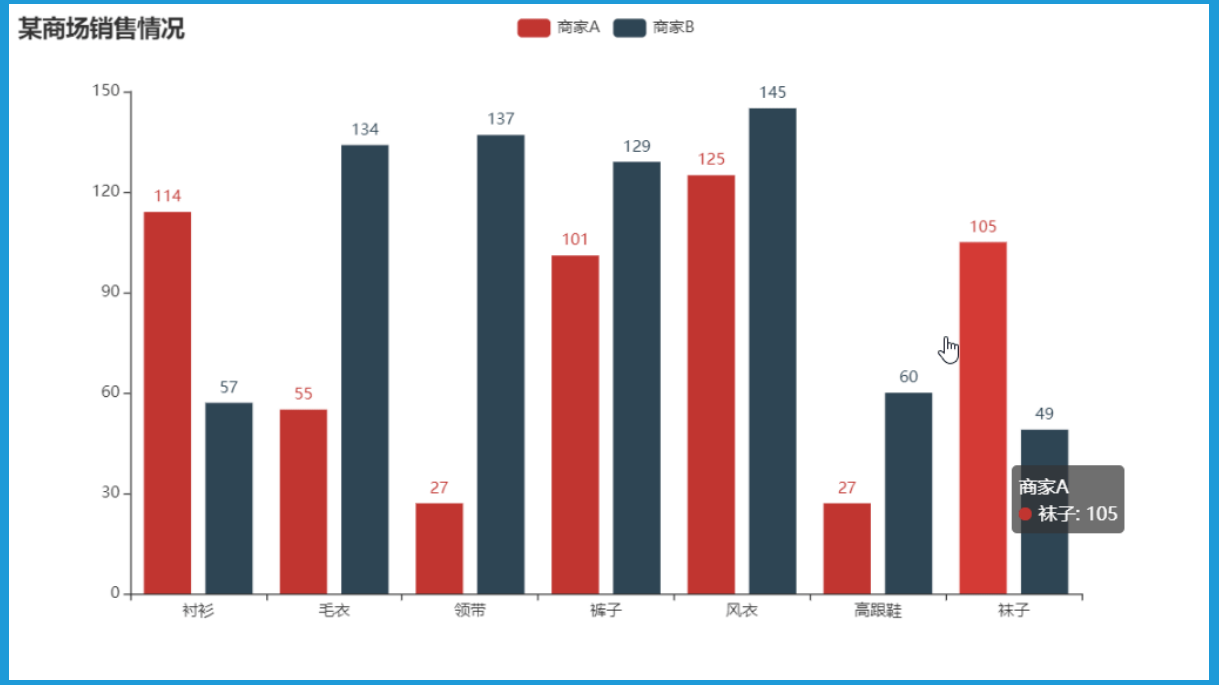
通过简单的几行代码即可实现一个可视化的 demo

生成 HTML

```python
from pyecharts.charts import Bar
from pyecharts import options as opts

# V1 版本开始支持链式调用
bar = (
    Bar()
    .add_xaxis(["衬衫", "毛衣", "领带", "裤子", "风衣", "高跟鞋", "袜子"])
    .add_yaxis("商家A", [114, 55, 27, 101, 125, 27, 105])
    .add_yaxis("商家B", [57, 134, 137, 129, 145, 60, 49])
    .set_global_opts(title_opts=opts.TitleOpts(title="某商场销售情况"))
)
bar.render()

# 不习惯链式调用的开发者依旧可以单独调用方法
bar = Bar()
bar.add_xaxis(["衬衫", "毛衣", "领带", "裤子", "风衣", "高跟鞋", "袜子"])
bar.add_yaxis("商家A", [114, 55, 27, 101, 125, 27, 105])
bar.add_yaxis("商家B", [57, 134, 137, 129, 145, 60, 49])
bar.set_global_opts(title_opts=opts.TitleOpts(title="某商场销售情况"))
bar.render()
```



pyecharts 高阶操作

```python
from pyecharts import options as opts
from pyecharts.charts import Bar
from pyecharts.commons.utils import JsCode
from pyecharts.globals import ThemeType

list2 = [
    {"value": 12, "percent": 12 / (12 + 3)},
    {"value": 23, "percent": 23 / (23 + 21)},
    {"value": 33, "percent": 33 / (33 + 5)},
    {"value": 3, "percent": 3 / (3 + 52)},
    {"value": 33, "percent": 33 / (33 + 43)},
]

list3 = [
    {"value": 3, "percent": 3 / (12 + 3)},
    {"value": 21, "percent": 21 / (23 + 21)},
    {"value": 5, "percent": 5 / (33 + 5)},
    {"value": 52, "percent": 52 / (3 + 52)},
    {"value": 43, "percent": 43 / (33 + 43)},
]

c = (
    Bar(init_opts=opts.InitOpts(theme=ThemeType.LIGHT))
    .add_xaxis([1, 2, 3, 4, 5])
    .add_yaxis("product1", list2, stack="stack1", category_gap="50%")
    .add_yaxis("product2", list3, stack="stack1", category_gap="50%")
    .set_series_opts(
        label_opts=opts.LabelOpts(
            position="right",
            formatter=JsCode(
                "function(x){return Number(x.data.percent * 100).toFixed() + '%';}"
            ),
        )
    )
    .render("stack_bar_percent.html")
)
```

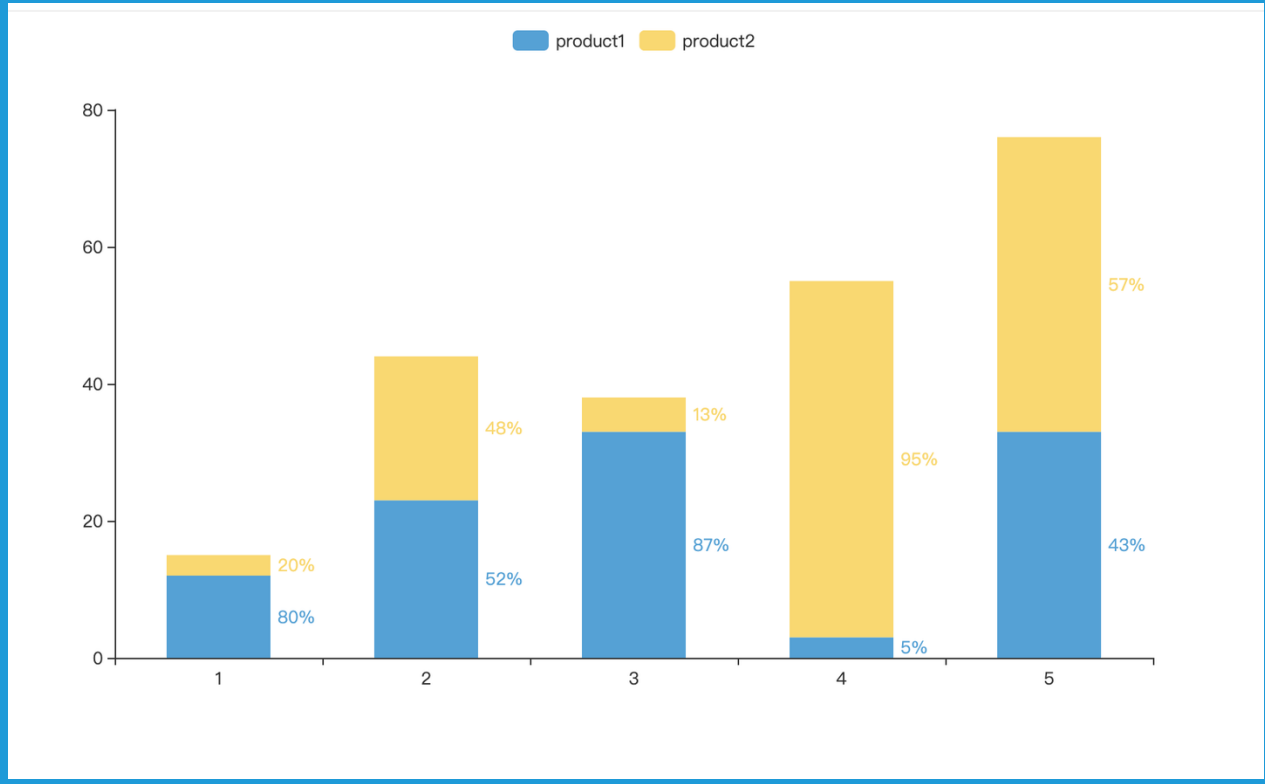
使用 JsCode 实现 JS 回调函数高阶操作

1、自定义计算结果

2、插入 JS 代码实现部分前端交互

3、动态更新操作

4、等等...



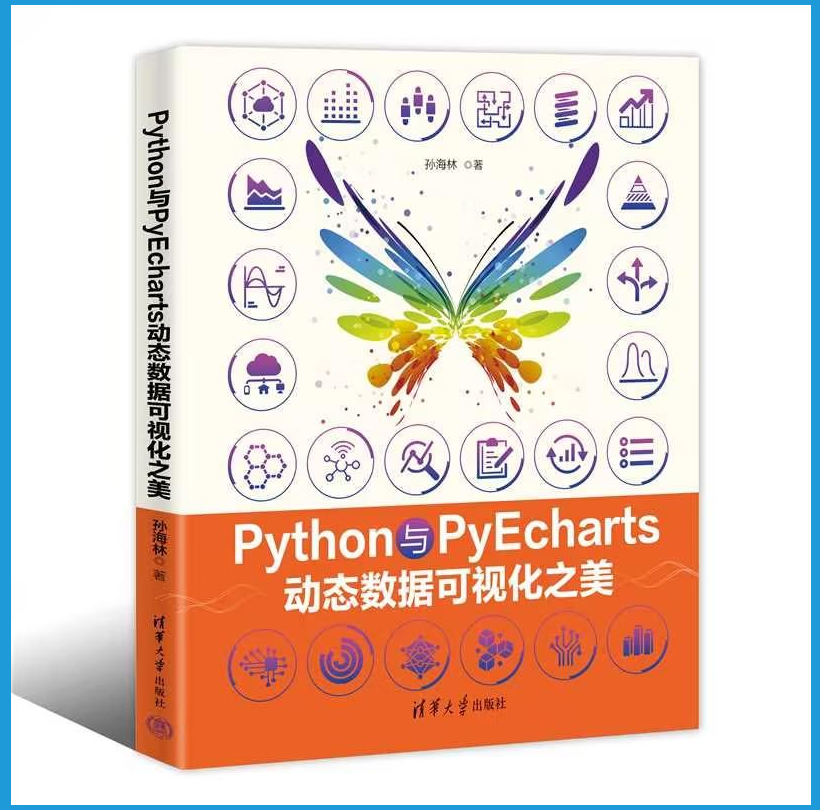
完整说明参见 pyecharts 知乎专栏：https://zhuanlan.zhihu.com/p/133533187

pyecharts 的未来

- 过去：
- V0.5.X 版本：初代 pyecharts，API 混乱，图表实现不统一
- V1.X 版本：重构 API 和图表实现方式，对接 Echarts 4.X
- V2.X 版本：改造部分 API 实现及图表参数，对接 Echarts 5.X

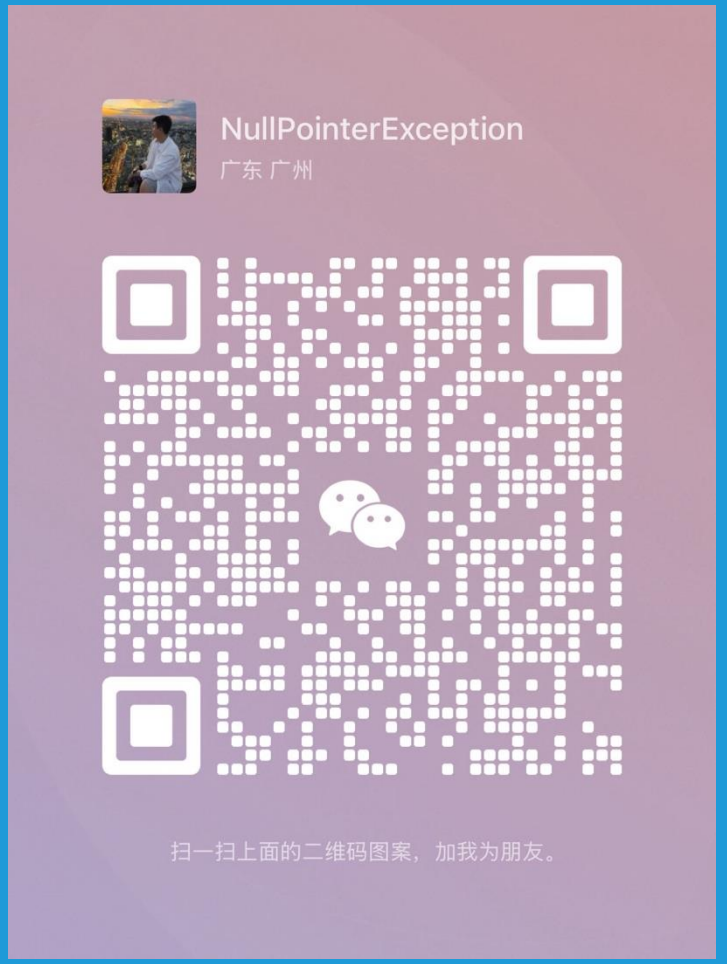
- 未来：
- Echarts 6.X?
- 再一次重构？V3.X?

One more thing



预计 2025 年 3 月出版

Thank You ！



Ps：对 AIGC 项目感兴趣的小伙伴可以多沟通，有好 idea ~