Echarts (Pie)

ECharts Pie Chart 구현 가이드

📌 시작하기 전에

- 자바스크립트를 몰라도 괜찮습니다
- 아래 코드를 **전체 복사** 후 필요한 부분만 수정하세요
  - (가장 하단의 `clearInterval(this.myInterval)`은 destroy 탭에)
- `this.`는 각 컴포넌트를 구분하는 표시이므로 그대로 두세요

🔧 전체 코드 구조 (REST API)

```
// 1. 기본 함수 정의
this.createChart = (element) => echarts.init(element);

this.settingChart = fx.curry((chart, option) => chart.setOption(option));

this.setPieOption = fx.curry((option, response) => {
  option.title = {
      text: 'Referer of a Website',
      subtext: 'Fake Data',
      left: 'center'
  },
  option.tooltip = {
      trigger: 'item',
  };
  option.legend = {
      orient: 'vertical',
      left: 'left',
      top: 'center'
  }
  return response;
});
```

```
this.setPieSeries = fx.curry((option, response) => {
    option.series = [{
        type: 'pie',
        radius: '50%',
        data: response.data,
        emphasis: {
            itemStyle: {
                shadowBlur: 10,
                shadowOffsetX: 0,
                shadowColor: 'rgba(0, 0, 0, 0.5)'
            }
        }
    }];
    return response;
});

this.getChartData = async apiArguments => {
    const { response } = await WKit.fetchData(...Object.values(apiArgument
s)).catch(console.error);
    return response;
};

// 2. 차트 생성 및 설정
const chartElement = this.element.querySelector('#echarts');
this.myPieChart = this.createChart(chartElement);

const chartOption = {};

// 🔴 API 연결 정보 (수정 필요)
const apiArguments = {
    page: this.page,
    datasetName: 'api_pie_chart',    // ← 여러분의 API 이름으로 변경
    params: { port: 3000, categories: 4, percent: true } // ← 필요한 파라미터로
변경
};

// 3. 차트 로드 함수
const loadChart = () => fx.go(
```

```
    this.getChartData(apiArguments),
    this.setPieOption(chartOption),
    this.setPieSeries(chartOption),
    (_) => this.settingChart(this.myPieChart, chartOption)
);

loadChart();

// 🟡 자동 새로고침 설정 (필요시 수정)
this.REFRESH_INTERVAL = 5000;  // 5초마다 새로고침
this.myInterval = setInterval(loadChart, this.REFRESH_INTERVAL);

// 4. 정리 (destroy 탭에 위치)
clearInterval(this.myInterval);
```

🔧 전체 코드 구조 (DB Query)

```
this.createChart = (element) => echarts.init(element);

this.settingChart = fx.curry((chart, option) => chart.setOption(option));

this.setPieOption = fx.curry((option, response) => {
    option.title = {
        text: 'Referer of a Website',
        subtext: 'Fake Data',
        left: 'center'
    },
    option.tooltip = {
        trigger: 'item',
    };
    option.legend = {
        orient: 'vertical',
        left: 'left',
        top: 'center'
    }
    return response;
    return response;
```

```
});

this.setPieSeries = fx.curry((option, response) => {
    option.series = [{
        type: 'pie',
        radius: '50%',
        data: response.data,
        emphasis: {
            itemStyle: {
                shadowBlur: 10,
                shadowOffsetX: 0,
                shadowColor: 'rgba(0, 0, 0, 0.5)'
            }
        }
    }];
    return response;
});

this.getChartData = async apiArguments => {
    const { response } = await WKit.fetchData(...Object.values(apiArgument
s)).catch(console.error);
    return formatChartResponse(response);
};

function formatChartResponse(response) {
    const data = response.rows || response;
    const total = data.reduce((sum, item) => sum + item.value, 0);

    return {
        data: data.map(item => ({
            name: item.name,
            value: item.value,
            percentage: ((item.value / total) * 100).toFixed(1)
        })),
        total: total,
        metadata: {
            generated: new Date().toISOString(),
            categories: data.length,
```

```
            includeValue: true,
            type: 'pie-chart'
        }
    };
}


// 2. 차트 생성 및 설정
const chartElement = this.element.querySelector('#echarts');
this.myPieChart = this.createChart(chartElement);

const chartOption = {};

// 🔴 API 연결 정보 (수정 필요)
const apiArguments = {
    page: this.page,
    datasetName: 'api_pie_chart_query',    // ← 여러분의 API 이름으로 변경
    params: {  categories: "4", percent: "true" } // ← 필요한 파라미터로 변경, ← 필요한 파라미터로 변경, query는 parameter를 ""로 변경
};

// 3. 차트 로드 함수
const loadChart = () => fx.go(
    this.getChartData(apiArguments),
    this.setPieOption(chartOption),
    this.setPieSeries(chartOption),
    (_) => this.settingChart(this.myPieChart, chartOption)
);

loadChart();

// 🟡 자동 새로고침 설정 (필요시 수정)
this.REFRESH_INTERVAL = 5000;  // 5초마다 새로고침
this.myInterval = setInterval(loadChart, this.REFRESH_INTERVAL);

// 4. 정리 (destroy 탭에 위치)
clearInterval(this.myInterval);
```

📊 데이터 구조 이해하기

API 응답 구조:


```
{
  "data": [
    { "name": "Sales", "value": 397, "percentage": "15.9" },
    { "name": "Marketing", "value": 328, "percentage": "13.1" },
    { "name": "Engineering", "value": 114, "percentage": "4.6" },
    { "name": "Support", "value": 422, "percentage": "16.9" },
    { "name": "Finance", "value": 928, "percentage": "37.1" }
  ],
  "total": 2189,
  "metadata": {
    "generated": "2025-08-08T05:10:26.935Z",
    "categories": 5,
    "includeValue": true,
    "type": "pie-chart"
  }
}
```


DB Query 응답 구조


```
[
    {
        "value": 424,
        "name": "Sales",
        "percentage": "11.8"
    },
    {
        "value": 954,
        "name": "Marketing",
        "percentage": "26.6"
    },
    {
        "value": 368,
        "name": "Engineering",
        "percentage": "10.3"
```

```
    },
    {
      "value": 611,
      "name": "Support",
      "percentage": "17.0"
    },
    {
      "value": 385,
      "name": "Finance",
      "percentage": "10.7"
    }
  ],
```

중요 포인트:

- `data`: 파이 차트의 각 조각 데이터
- 각 항목은 `name`(라벨), `value`(값), `percentage`(선택)를 가집니다
- `total`: 전체 합계 (참고용)
- ECharts가 자동으로 백분율을 계산하므로 `percentage`는 선택사항입니다

🎯 데이터 흐름 설명

`loadChart` 함수의 동작:

```
const loadChart = () => fx.go(
  this.getChartData(apiArguments),    // 1단계: API 호출
  this.setPieOption(chartOption),     // 2단계: 툴팁 설정
  this.setPieSeries(chartOption),     // 3단계: 파이 데이터 설정
  (_) => this.settingChart(this.myPieChart, chartOption)  // 4단계: 차트 그리기
)
```

단계별 설명:

1. **API 데이터 가져오기**: `{ data: [...] }` 형태의 response
2. **기본 옵션 설정**: 툴팁 등 기본 설정
3. **Pie Series 설정**: 데이터를 파이 차트 형태로 변환

4. **차트 렌더링**: 완성된 옵션으로 차트 그리기

데이터 전달 과정:

API 호출 → response 객체 → response.data → 파이 차트 표시

✏️ 수정 가능한 부분

1. API 정보 변경 (🔴 필수)

```
const apiArguments = {
    page: this.page,
    datasetName: 'department_stats',     // API 이름 변경
    params: {
        port: 3000,
        categories: 5,                   // 5개 카테고리
        percent: true,                   // 백분율 포함
        year: 2025                       // 추가 파라미터
    }
}
```

2. 파이 차트 스타일 변경 (🟡 선택)

```
// 도넛 차트로 변경
this.setPieSeries = fx.curry((option, response) => {
    option.series = [{
        type: 'pie',
        radius: ['40%', '70%'],  // 내부 40%, 외부 70% (도넛 모양)
        data: response.data,
        // 라벨 위치 조정
        label: {
            show: true,
            position: 'outside',
            formatter: '{b}: {d}%'  // 이름: 퍼센트%
        },
        emphasis: {
            itemStyle: {
```

```
            shadowBlur: 10,
            shadowOffsetX: 0,
            shadowColor: 'rgba(0, 0, 0, 0.5)'
        }
    }
  }];
  return response;
});
```

3. 범례(Legend) 추가 (🟡 선택)

```
this.setPieOption = fx.curry((option, response) => {
  option.tooltip = { trigger: 'item' };
  option.legend = {
    orient: 'vertical',
    left: 'left',
    data: response.data.map(item => item.name)
  };
  return response;
});
```

4. 새로고침 주기 변경 (🟡 선택)

```
this.REFRESH_INTERVAL = 10000;  // 10초로 변경
// 또는 자동 새로고침을 원하지 않으면 setInterval 라인 삭제
```

🔍 Preview 섹션 이해

Preview는 개발 중 미리보기용입니다:

```
const targetElement = this.element.querySelector('#echarts')
this.previewChart = echarts.init(targetElement);

// API와 동일한 구조의 샘플 데이터
const response = {
  "data": [
    { "name": "Sales", "value": 397, "percentage": "15.9" },
```

```
    { "name": "Marketing", "value": 328, "percentage": "13.1" },
    { "name": "Engineering", "value": 114, "percentage": "4.6" },
    { "name": "Support", "value": 422, "percentage": "16.9" },
    { "name": "Finance", "value": 928, "percentage": "37.1" }
  ],
  "total": 2189
};

// 실제 코드와 동일한 방식으로 처리
let option = {};
option.tooltip = { trigger: 'item' };
option.series = [{
  type: 'pie',
  radius: '50%',
  data: response.data,  // response.data를 그대로 사용
  emphasis: {
    itemStyle: {
      shadowBlur: 10,
      shadowOffsetX: 0,
      shadowColor: 'rgba(0, 0, 0, 0.5)'
    }
  }
}];

this.previewChart.setOption(option);
```

Preview의 역할:

- API 연결 전에 차트가 잘 작동하는지 확인
- 실제 API 응답과 동일한 구조로 테스트
- 메인 코드의 처리 방식을 그대로 재현

⚠️ 자주 발생하는 문제

차트가 안 보일 때

- HTML에 `<div id="echarts"></div>` 확인
- div에 높이 지정: `<div id="echarts" style="height: 400px;"></div>`

- 브라우저 콘솔(F12)에서 에러 메시지 확인

데이터가 안 나올 때

- `datasetName`이 정확한지 확인
- **데이터 구조 확인:**
  - `setPieSeries`에서 사용하는 데이터(예: `response.data`)가 배열인지
  - 파이 차트에 필요한 형태인지 확인
  - 브라우저 콘솔에서 `console.log(response)`로 구조 파악
- **다른 구조의 API라면 함수 수정:**

```
// 예: API가 { items: [...] } 구조라면
this.setPieSeries = fx.curry((option, response) => {
    option.series = [{
        type: 'pie',
        radius: '50%',
        data: response.items,  // data → items로 변경
        // ...
    }];
    return response;
});
```

파이 조각이 너무 작을 때

```
// 작은 값들을 '기타'로 합치기
const threshold = 5; // 5% 미만
const others = response.data.filter(item => item.value < threshold);
const main = response.data.filter(item => item.value >= threshold);
if (others.length > 0) {
    main.push({
        name: '기타',
        value: others.reduce((sum, item) => sum + item.value, 0)
    });
}
```

✅ 체크리스트

- [ ] HTML에 `<div id="echarts" style="height: 400px;"></div>` 존재
- [ ] `datasetName`을 내 API 이름으로 변경
- [ ] `params`를 필요한 값으로 수정
- [ ] **`setPieSeries` 함수가 올바른 데이터를 받고 있는지 확인**
  - 파이 차트 데이터가 배열 형태로 전달되는지
  - ECharts가 요구하는 `[{name: "...", value: ...}, ...]` 형태인지
  - 필요시 `setPieSeries` 함수 내부의 데이터 경로 수정
- [ ] destroy 탭에 `clearInterval(this.myInterval)` 위치 확인