I/O Summary of <>

General Overview

```
// attached files need to be in same dir !
results = FileAttachment("results.json").json({ typed: true })

ticks_per_second = results.TimerResolution

// Function to format axis labels
function formatBytesAxis(bytes) {
  if (bytes >= 1e15) return (bytes / 1e15).toFixed(2) + " PB";
  if (bytes >= 1e12) return (bytes / 1e12).toFixed(2) + " TB";
  if (bytes >= 1e9)  return (bytes / 1e9).toFixed(2) + " GB";
  if (bytes >= 1e6)  return (bytes / 1e6).toFixed(2) + " MB";
  if (bytes >= 1e3)  return (bytes / 1e3).toFixed(2) + " KB";
  return bytes + " B";
}

function formatTimeAxis(seconds) {
  if (seconds >= 3600) return (seconds / 3600).toFixed(2) + " h";
  if (seconds >= 60)   return (seconds / 60).toFixed(2) + " min";
  if (seconds >= 1)    return seconds.toFixed(2) + " s";
  if (seconds >= 1e-3) return (seconds * 1e3).toFixed(2) + " ms";
  if (seconds >= 1e-6) return (seconds * 1e6).toFixed(2) + " µs";
  return (seconds * 1e9).toFixed(2) + " ns";
}
```

By paradigm

```
// Extract IOOperations
ioData = Object.entries(results.IOOperations).map(([paradigm, ops]) => ({
    paradigm,
    bytes: ops.Bytes,
    meta: ops.MetaOperationTime / ticks_per_second,
    transfer: ops.TransferOperationTime / ticks_per_second,
    count: ops.Count
}));

// Create charts
chartBytes = Plot.plot({
    marks: [Plot.barY(ioData, {x: "paradigm", y: "bytes", fill: "paradigm"})],
    y: {label: "Bytes", tickFormat: formatBytesAxis},
    width: 400,
    height: 300
});

chartTransfer = Plot.plot({
    marginLeft: 70,
    marks: [Plot.barY(ioData, {x: "paradigm", y: "transfer", fill: "paradigm"})],
    y: {label: "IOOperationTime", tickFormat: formatTimeAxis},
    width: 400,
    height: 300
});

chartMeta = Plot.plot({
    marginLeft: 70,
    marks: [Plot.barY(ioData, {x: "paradigm", y: "meta", fill: "paradigm"})],
    y: {label: "MetaOperationTime", tickFormat: formatTimeAxis},
    width: 400,
    height: 300
});

// Return horizontal container
html`<div style="display:flex; gap:20px;">${chartBytes}${chartTransfer}${chartMeta}</div>`
```

```
data = FileAttachment("./palmer-penguins.csv").csv({ typed: true })
viewof bill_length_min = Inputs.range(
  [32, 50],
  {value: 35, step: 1, label: "Bill length (min):"}
```

```
)
viewof islands = Inputs.checkbox(
  ["Torgersen", "Biscoe", "Dream"],
  { value: ["Torgersen", "Biscoe"],
    label: "Islands:"
  }
)

filtered = data.filter(function(penguin) {
  return bill_length_min < penguin.bill_length_mm &&
         islands.includes(penguin.island);
})

Plot.rectY(filtered,
  Plot.binX(
    {y: "count"},
    {x: "body_mass_g", fill: "species", thresholds: 20}
  ))
  .plot({
    facet: {
      data: filtered,
      x: "sex",
      y: "species",
      marginRight: 80
    },
    marks: [
      Plot.frame(),
    ]
  }
)
```