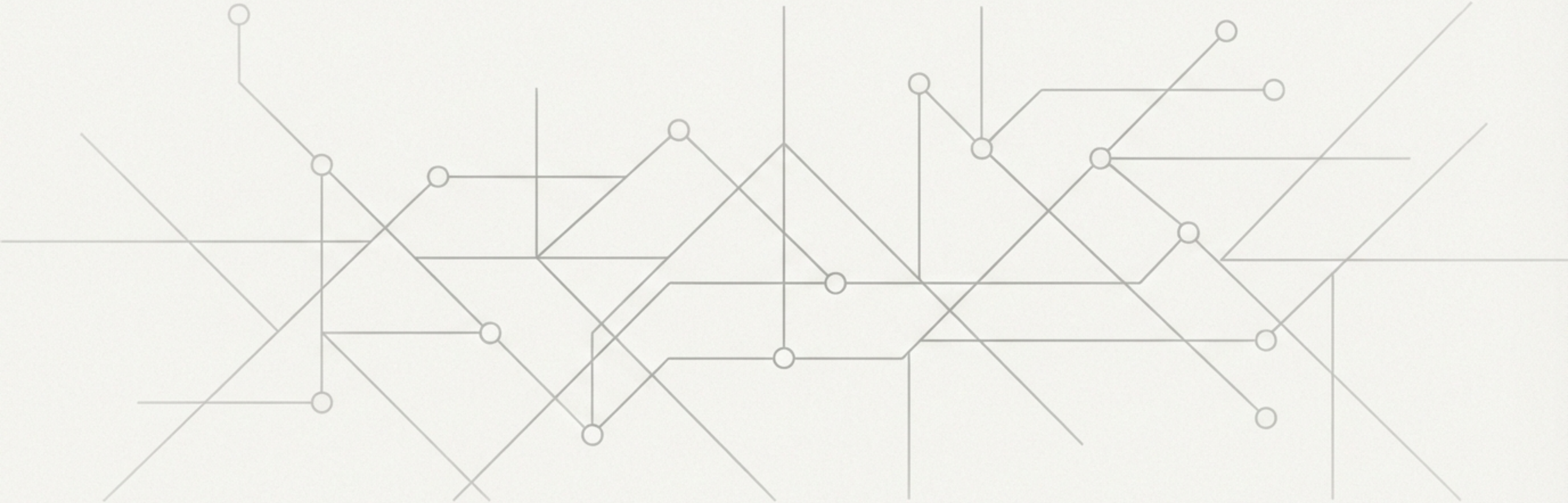
A New Architecture for HTML Manipulation

Introducing the Html_MGraph Multi-Graph Model



Internal Briefing | v1.2.0 | December 2025 | the-cyber-boardroom/MGraph-AI__Service__Html__Graph

NotebookLM

The Challenge: Achieving a Perfect HTML Round-Trip

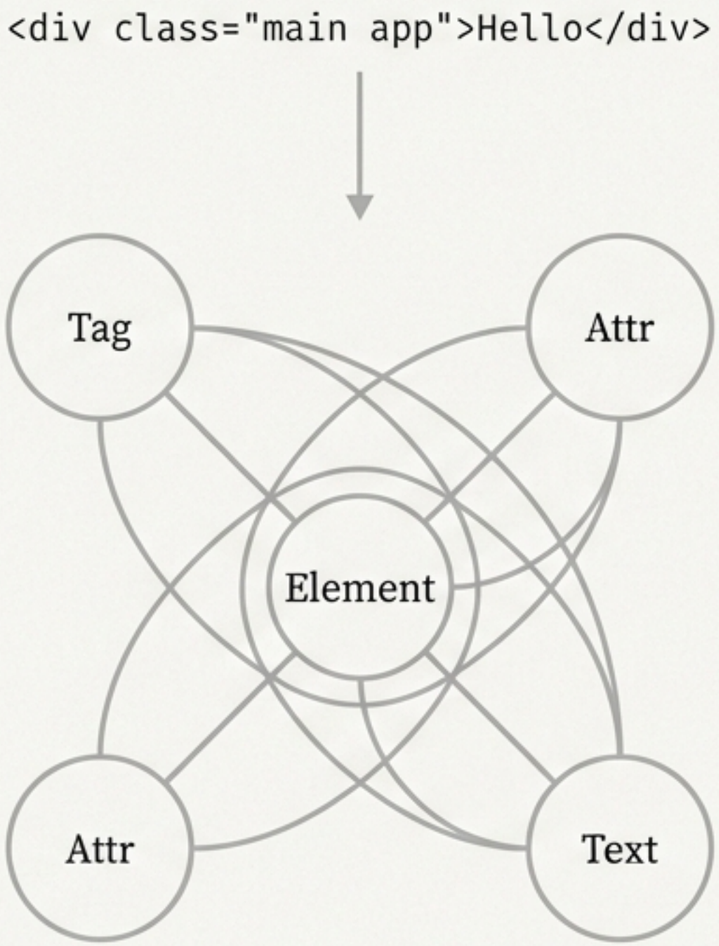
The Goal

1. **Visualization:** See HTML structure as a graph.
2. **Manipulation:** Transform the graph (e.g., collapse nodes, merge text).
3. **Round-trip:** Convert back to valid HTML after transformations.

The Current Reality (v1.x)

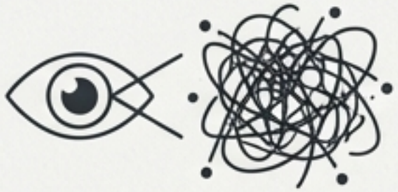
We use a single, monolithic graph to represent everything: elements, text, tags, attributes, scripts, and styles.

Single-Graph Representation



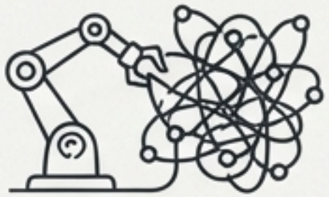
NotebookLM

The Limitations of a Monolithic Graph



1. Unreadable Visualisations

Even simple pages create a dense web of nodes and edges, making the graph impossible to interpret. The primary purpose of visualisation is defeated.

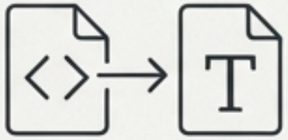


2. Difficult Manipulations

Transformations must carefully preserve tag and attribute edges, adding significant complexity to algorithms that should only care about structure and text.

3. <head> Section Noise

The `<head>` contains critical metadata for round-trips but clutters the graph with information irrelevant to content manipulation.



4. Code as Text

JavaScript and CSS content are treated as simple text nodes, cluttering content analysis and preventing future specialised parsing (e.g., creating an AST).

NotebookLM

The Path to a Solution: A Series of Experiments

To address these limitations, we conducted a series of targeted experiments. Each attempt to simplify the graph or improve transformations revealed a fundamental tension in the single-graph model. These experiments were not failures; they were essential steps in diagnosing the root cause.

NotebookLM

What We Learned: Three Experiments, One Conclusion

Experiment 1: Config Flags

Approach
Used schema flags to exclude tags and attributes from the graph.

Result
The graph was simpler, but it **broke the round-trip**. Information, once lost, could not be reconstructed.

Experiment 2: Transformation Pipeline

Approach
A 5-phase pipeline to modify the graph at different stages.

Result
Enabled some transformations, but they were destructive. The round-trip remained **broken** as essential data was destroyed during manipulation.

Experiment 3: Body-Only Transformation

Approach
Extracted and operated on the `<body> content exclusively.

Result
Achieved a clean graph for manipulation, but **lost the <head> entirely**. No round-trip was possible.

The Common Theme: Every attempt to simplify the graph for manipulation broke the fidelity required for reconstruction.

NotebookLM

The Key Insight: One Graph Cannot Serve Conflicting Masters

The fundamental issue is not our transformations; it's that a single graph is being forced to serve multiple, opposing purposes.

Manipulation & Visualisation Needs

- A clean, simple structure.
- Focus on elements and text only.
- Low noise, high clarity.

Round-Trip Reconstruction Needs

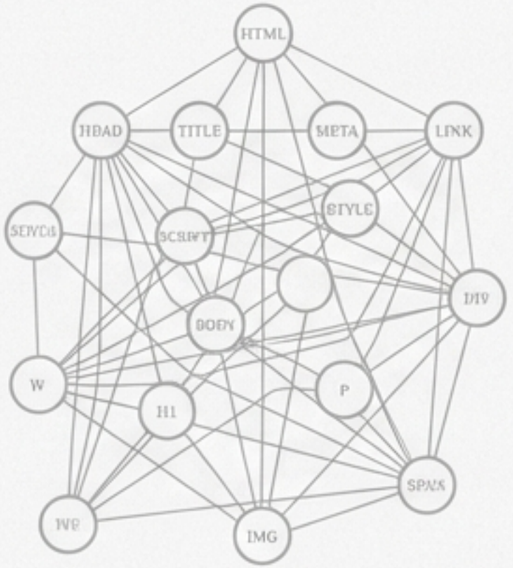
- 100% data fidelity.
- Requires tags, attributes, <head>, scripts, and styles.
- Completeness over simplicity.

The Inevitable Solution: We must separate these concerns into dedicated graphs that can be recombined.

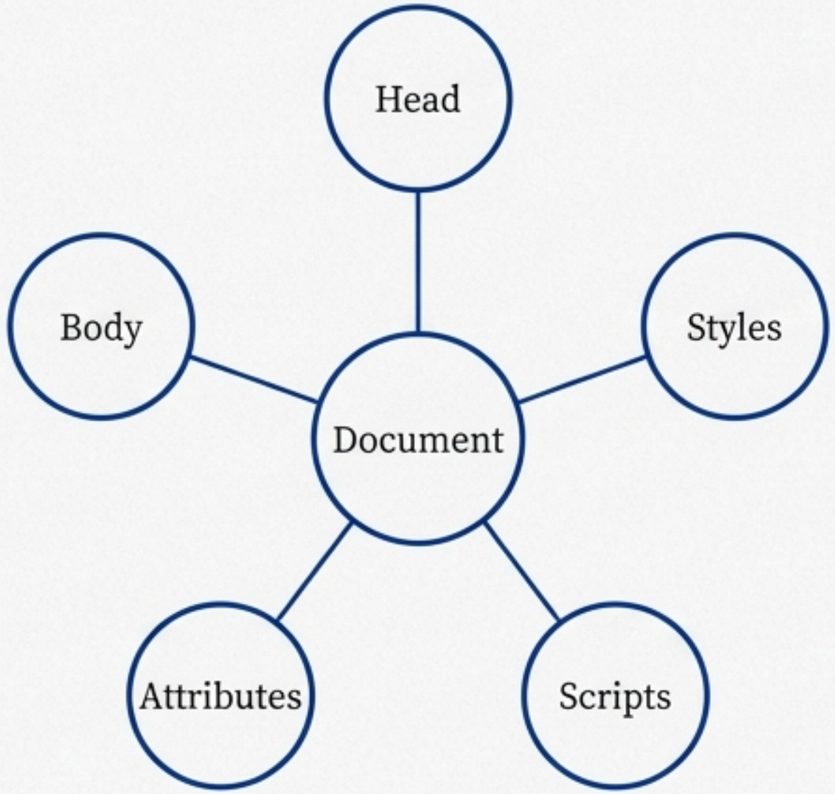
NotebookLM

The Solution: A Multi-Graph Architecture

Instead of one monolithic graph, we separate the HTML into **5 specialised graphs + 1 orchestrator.** Each graph has a single, focused responsibility.



Before

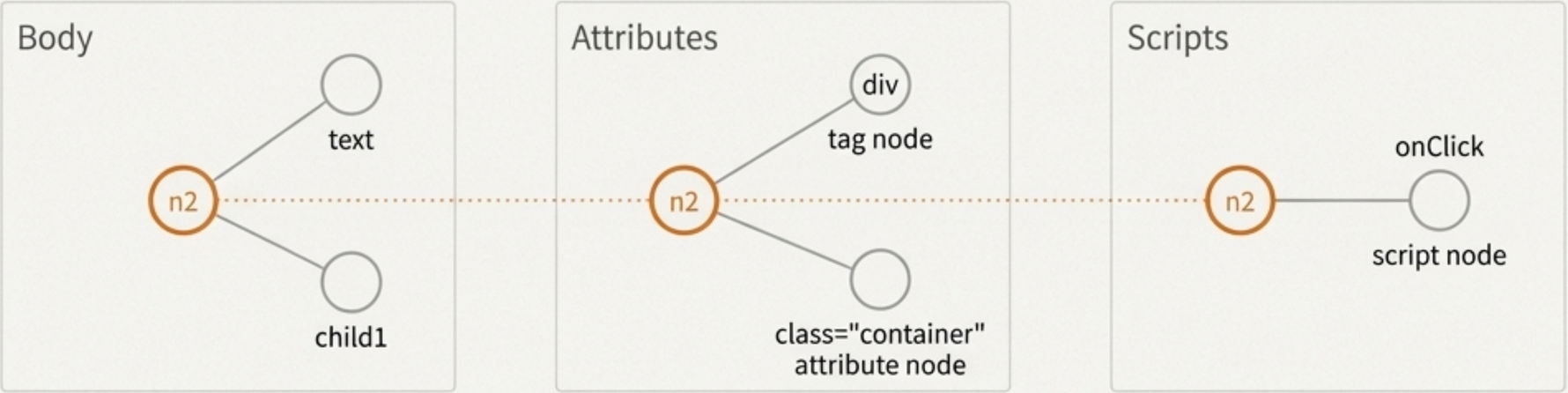


After: The Multi-Graph Model

NotebookLM

The Linchpin: How It All Connects with Shared Node_Ids

Element nodes use the **exact same Node_ Id** across all graphs in which they appear. This shared ID is the key to recombination.



Consider a `<div>` element with ID `n2`:

- In `Html_MGraph__Body`, `n2` is an element node connected to children and text.
- In `Html_MGraph__Attributes`, `n2` appears under the `div` tag node, connected to its attributes.

Benefit: This allows us to query across graphs. For example: "Find element `n2` in the `Body` graph, then find its attributes by looking up `n2` in the `Attributes` graph."

NotebookLM

The 5+1 Graphs: A Purpose-Built System

Graph	Purpose	Visualisation Complexity
`Html_MGraph__Document`	**Orchestrator** holding references to the other five graphs.	Minimal (6 nodes)
`Html_MGraph__Head`	<head> element structure and simple text content (<title>)	Low
`Html_MGraph__Body`	<body> element structure and <body> and text content. **The main working graph.**	Medium
`Html_MGraph__Attributes`	Tags and attributes for ALL elements across the document.	Medium
`Html_MGraph__Scripts`	JavaScript content from `<script>` tags.	Low
`Html_MGraph__Styles`	CSS content from `<style>` tags.	Low

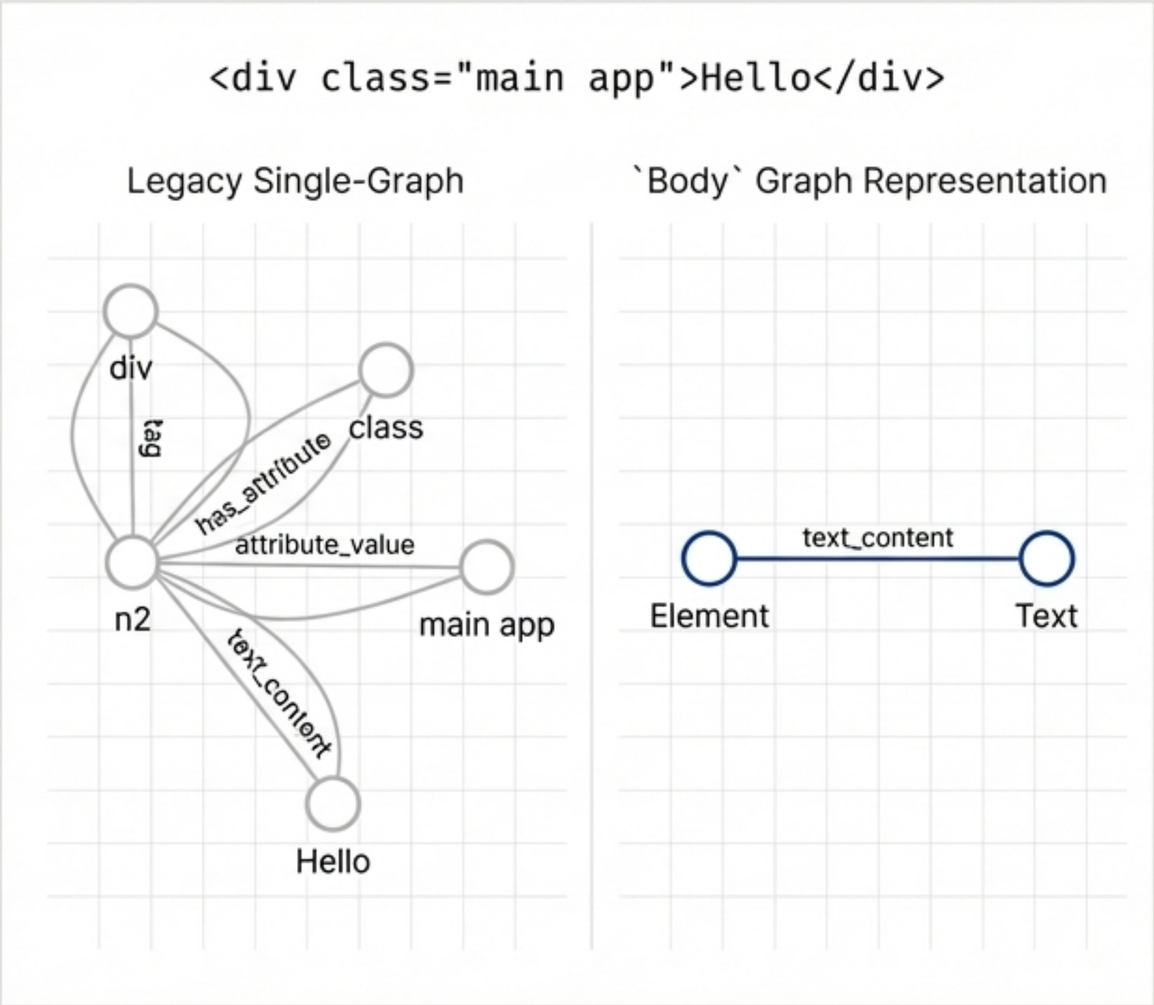
NotebookLM

The Workspace: Clean Manipulation in the `Body` Graph

`Html_MGraph__Body` is the primary working graph for most transformations and visualisations. By offloading other concerns, it becomes a pure representation of the document's content structure.

Key Characteristics:

- Contains only element structure and text content from the `<body>`.
- **No clutter:** Free from tag nodes, attribute nodes, scripts, or styles.
- **Ideal for algorithms:** Perfect for use cases like Html_Use_Case__2` (collapsing nodes, merging text).
- **Perfect for visualisation:** Renders clean, readable graphs in tools like D3.js or Graphviz."



NotebookLM

The Reference Library: The `Attributes` Graph

Purpose

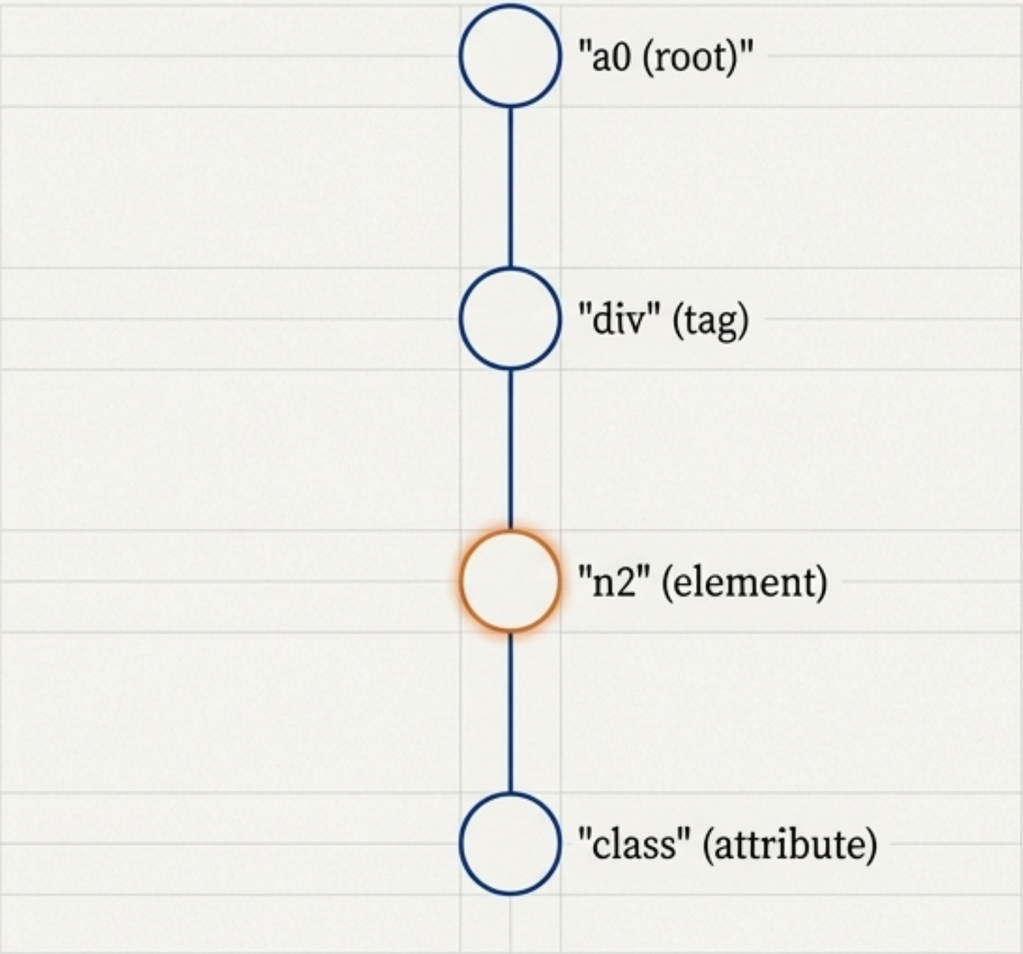
To store all tags and attributes for every element in the document (<html>`, <head>`, and `<body>`) in a structured, queryable way.

Key Feature: A Smart Hierarchy

The graph is structured logically:
`Root -> Tag -> Element -> Attribute`

Benefits of this Structure

- **Find an element's tag:** Traverse one edge UP from the element's node (e.g., from `n2` up to the `div` tag node).
- **Find all elements of a type:** Get all children of a tag node (e.g., all children of the `div` node).
- **Preserves order:** Attribute order is preserved via the `node_path` property on value nodes, ensuring perfect round-trip reconstruction.



NotebookLM

Future-Proofing: Dedicated Graphs for Scripts and Styles

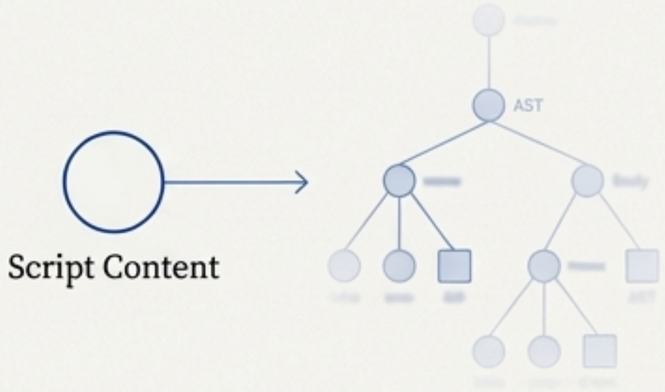
Purpose: To isolate code content from document content, preparing it for more advanced processing.

Html_MGraph__Scripts

- Holds JavaScript content from inline `<script>` tags.
- Element nodes share `Node_Id`s with the `Head`/`Body` graphs.
- External scripts (`src=...`) are represented by a node, but the `src` attribute itself lives in the `Attributes` graph.

Html_MGraph__Styles

- Holds CSS content from `<style>` tags.
- Element nodes share `Node_Id`s with the `Head` graph.



The Strategic Advantage

This separation creates a clean foundation for future work, such as feeding the content directly into an AST parser to create structured representations of JavaScript or CSS.

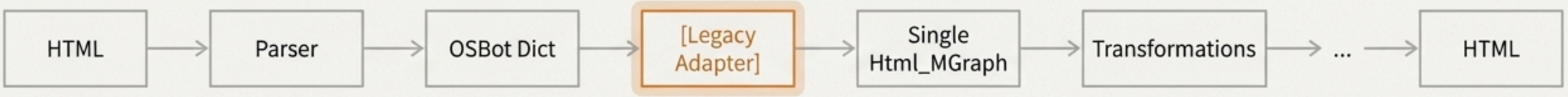
NotebookLM

The New Round-Trip Pipeline: From HTML to HTML

A clear before-and-after of the process flow, demonstrating the efficiency of the new Multi-Graph approach.

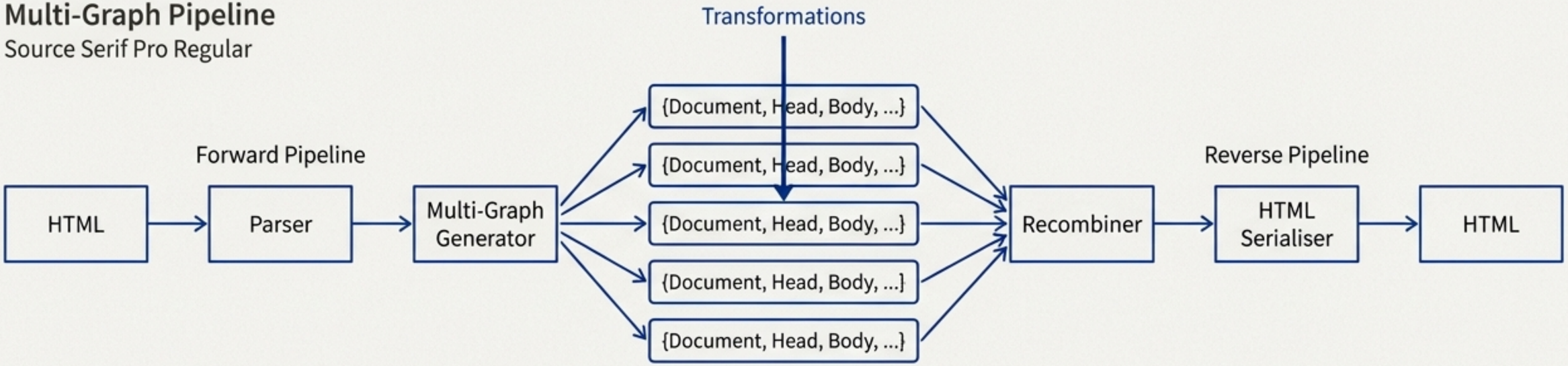
Legacy Pipeline

Source Serif Pro Regular



Multi-Graph Pipeline

Source Serif Pro Regular



NotebookLM

A Practical Migration Path

Ensuring Backward Compatibility

- The existing `Html_MGraph` class will remain for simple use cases.
- The new multi-graph architecture is an opt-in feature, allowing for gradual adoption.

Phased Implementation Plan

1. Refactor `Html_Dict__To__Html_MGraph` to remove legacy adapter **(DONE)**
2. Implement `Html_MGraph__Body` and `Html_MGraph__Head`
3. Implement `Html_MGraph__Attributes`
4. Implement `Html_MGraph__Scripts` and `Html_MGraph__Styles`
5. Implement the `Html_MGraph__Document` orchestrator
6. Implement the reverse pipeline for round-trip (`Html_MGraph__Document__To__Html`)
7. Migrate existing transformations to the new structure.

NotebookLM

A Summary of Our Design Principles

Choice Made	Rationale	Decision
5 + 1 orchestrator	Provides a clear and robust separation of concerns.	Number of graphs
Shared Node_Ids	The essential mechanism enabling cross-graph lookups and recombination.	Node Identity
As an intermediate layer in the Attributes graph	Eliminates a separate Tags graph and provides natural element grouping.	Tag Storage
node_path on value nodes	Preserves original attribute order for 100% round-trip fidelity.	Attribute Ordering
Separate, dedicated graphs	Declutters the **Body** graph and prepares for future AST parsing.	Script/Style Content
Every graph has a single root node	Guarantees a known starting point for any traversal algorithm.	Graph Traversal

NotebookLM