Refactoring Attribute Storage: From Brittle to Robust

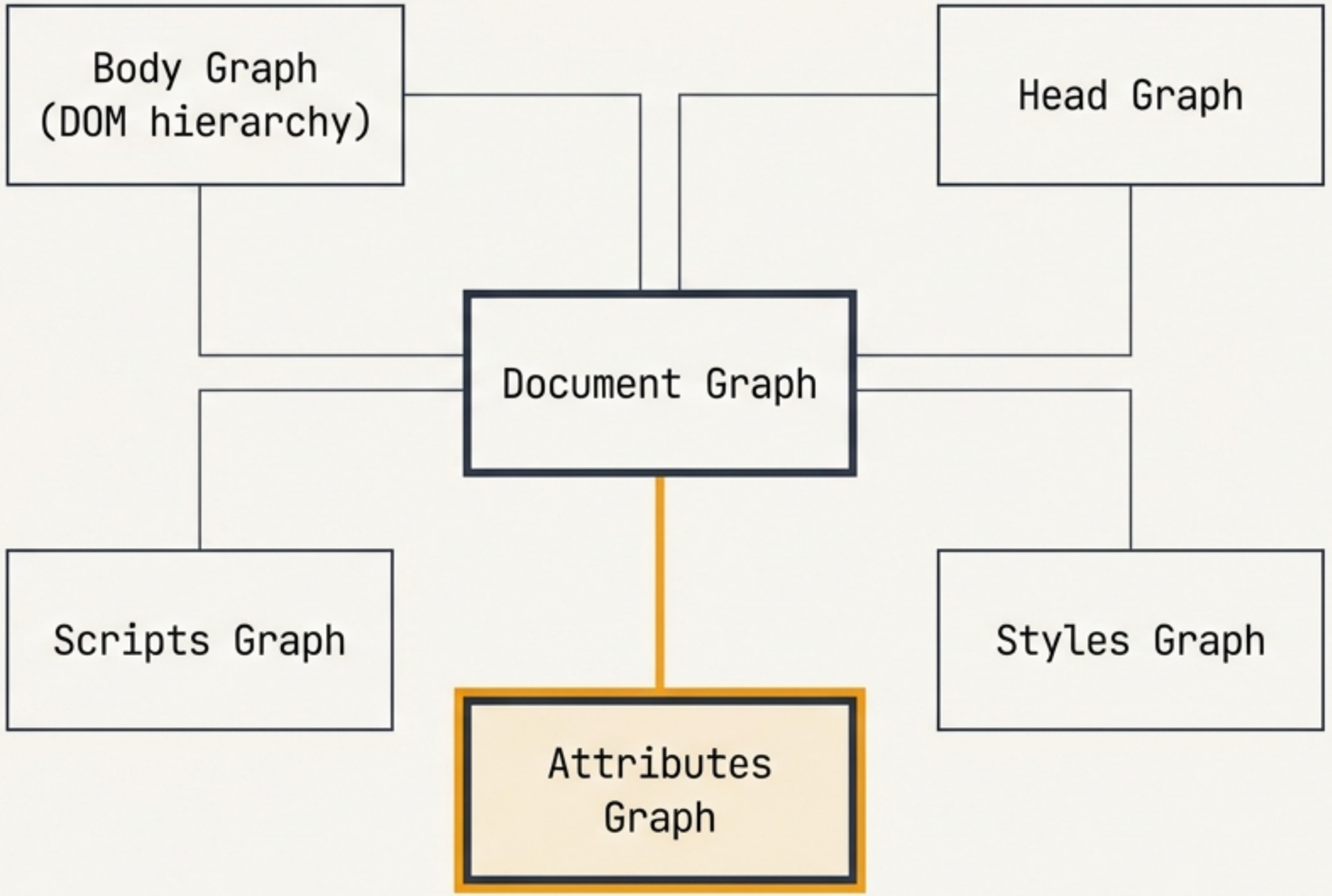
A deep-dive into the redesign of the `Html_MGraph__Attributes` model.

NotebookLM

The `Html_MGraph` Ecosystem

The `Html_MGraph` system represents a full HTML document as a collection of interconnected subgraphs.

Our focus today is on the **Attributes Graph**.

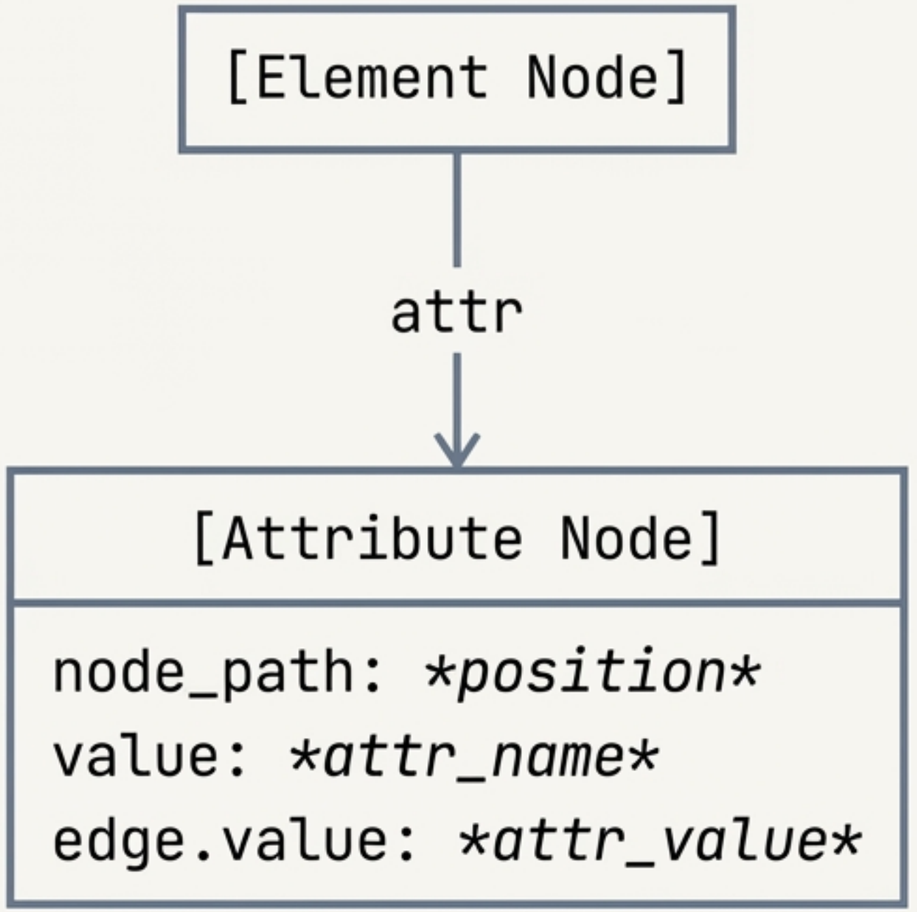


NotebookLM

The Original Model: Simple, but with a Hidden Flaw

Previously, each attribute was a single node connected to its parent element.

This design made a critical assumption: that an attribute attribute is an inseparable combination of a name, a value, and a position.

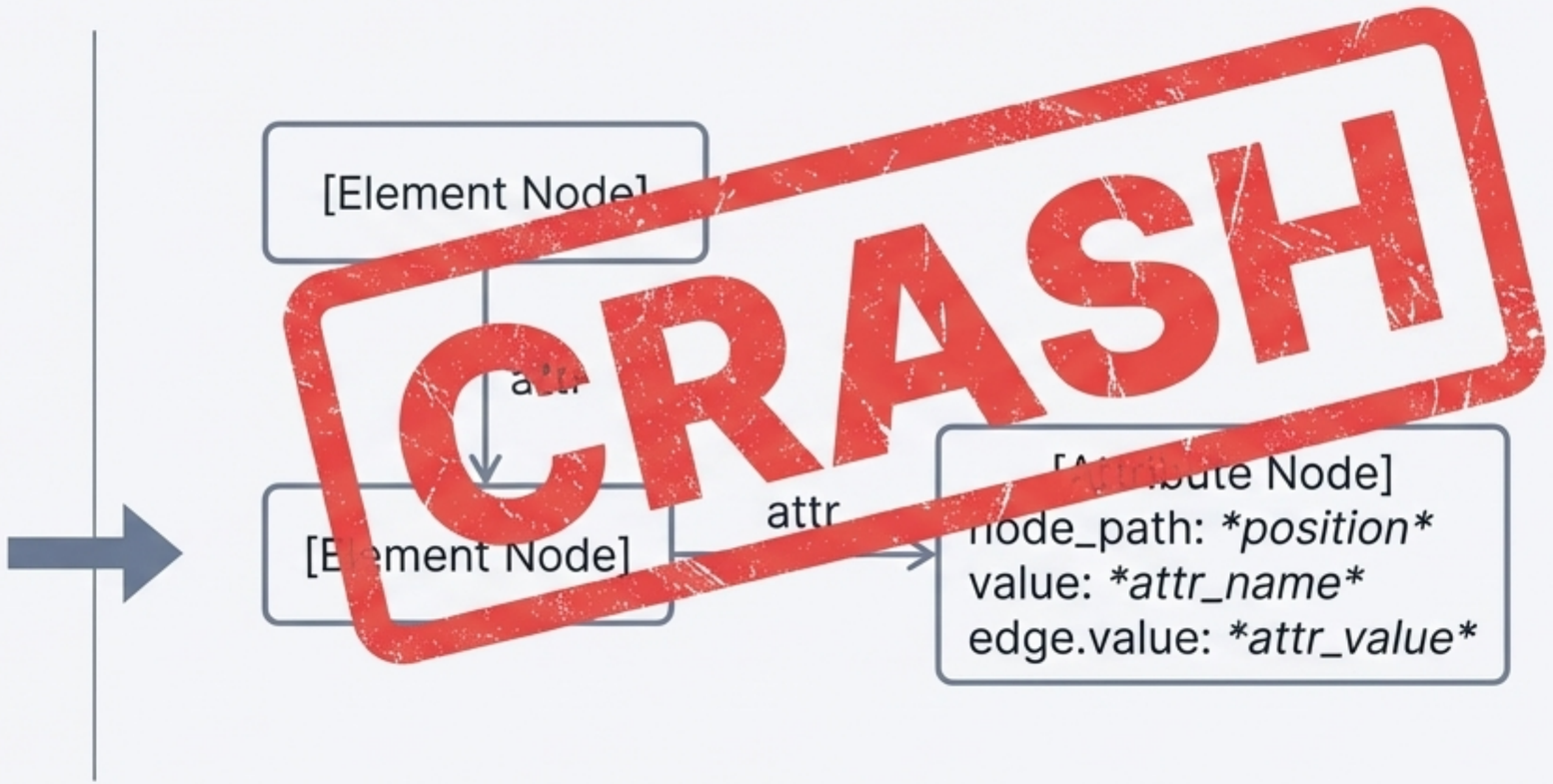


NotebookLM

The Breaking Point: A Single Boolean Attribute

The model's core assumption failed when processing standard HTML with boolean attributes (attributes without a value). The conversion from `Html-Dict` to `Html_MGraph` crashed the system.

```
<input required>
```

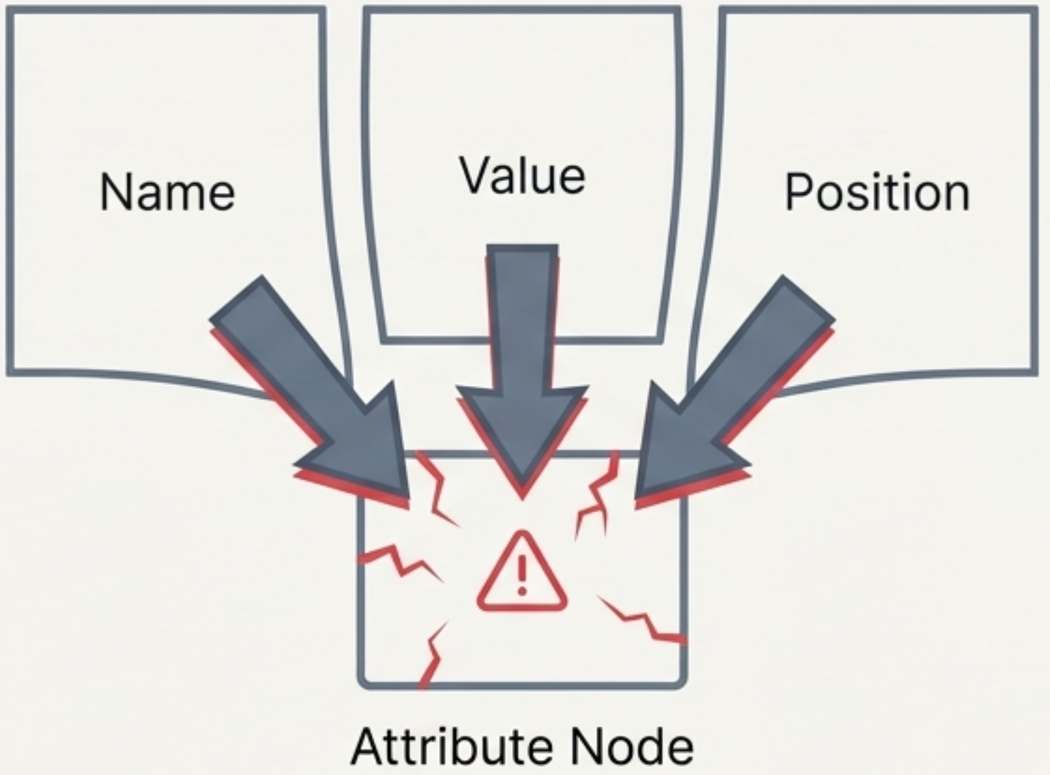


NotebookLM

Root Cause: A Flawed Premise

The single-node model fundamentally conflated three distinct concepts into one entity:

1. **Attribute Name** (e.g., `` `class` ``, `` `required` ``) - Should always exist.
2. **Attribute Value** (e.g., `` "container"` ``) - Is optional; absent for booleans.
3. **Attribute Position** - The order of the attribute on the element.



Key Insight: The model had no way to represent a name *without* a value.

NotebookLM

The Struggle: Why Simple Hacks Failed

Three attempts to patch the model failed because they violated the system's integrity, proving the flaw was structural.

Attempt 1: Use an empty string for the value.

Result: Incorrect graph data. The attribute name was lost.

Attempt 2: Store the name in `node_path`.

Result: Broke downstream consumers like `get_attributes()` which expect `node_path` to be be an integer position.

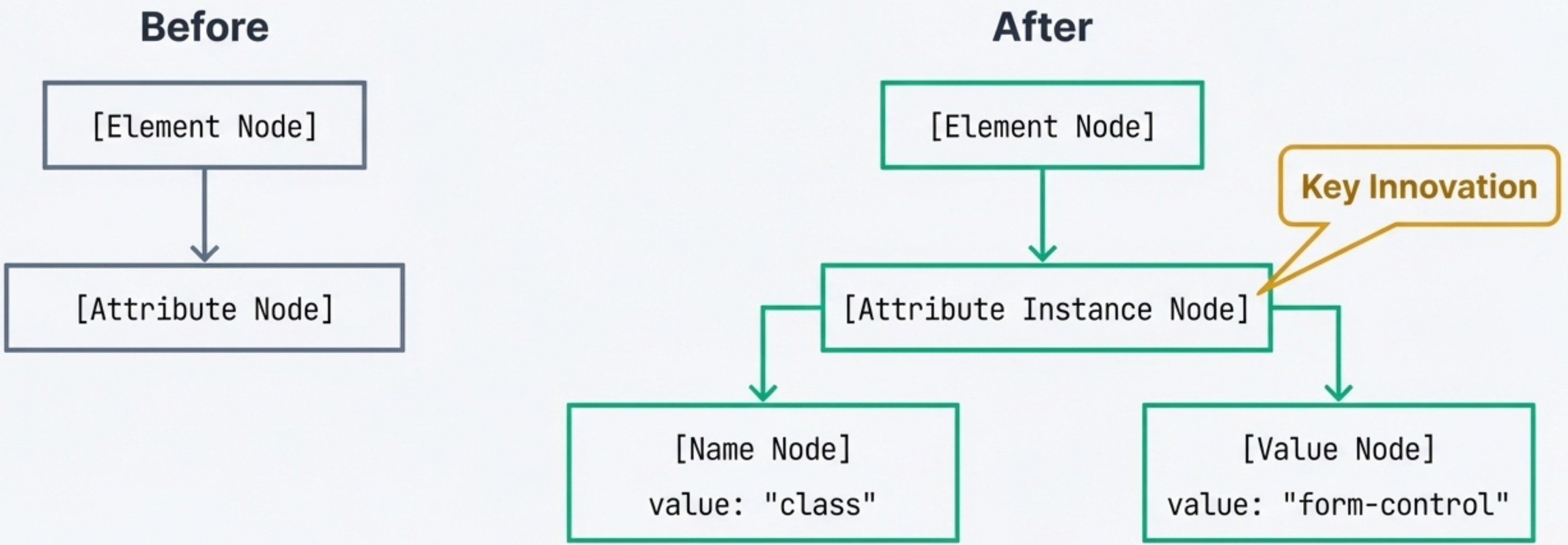
Attempt 3: Accept `""` but render it.

Result: Failed roundtrip fidelity. Produced `<input required="">` instead of the correct `<input required>`.

NotebookLM

The Solution: Separating Concerns with an 'Instance' Node

The new model introduces an intermediary **Attribute Instance** node. This decouples the attribute's existence and position from its name and value, resolving the core design flaw.

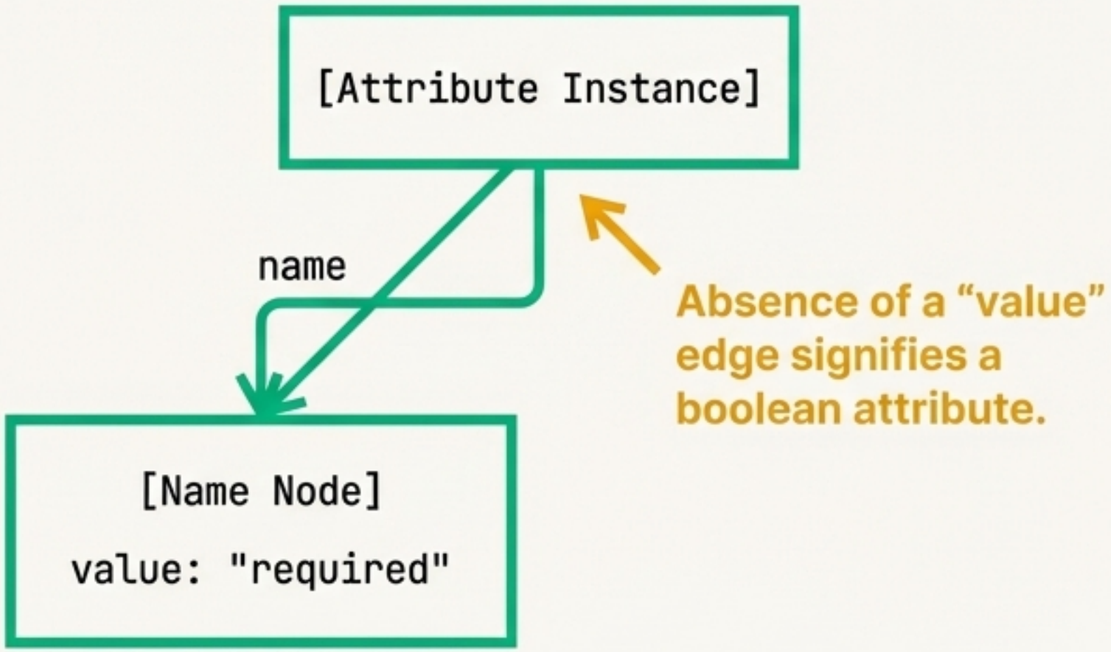


NotebookLM

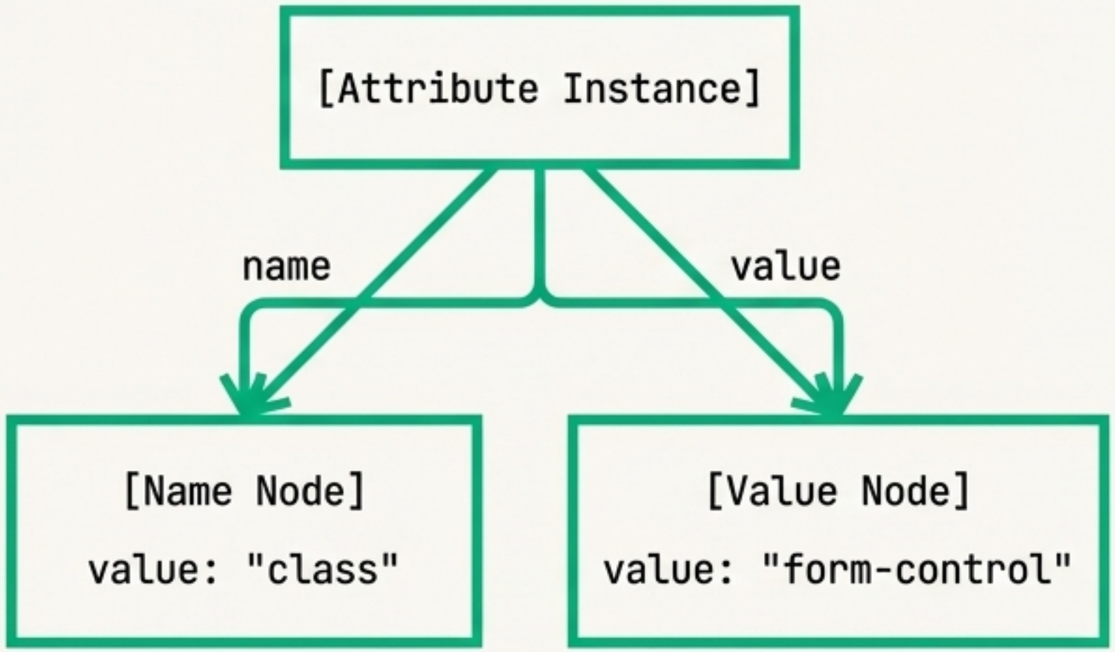
Anatomy of the New Model

The presence or absence of a `value` edge from the instance node elegantly defines the attribute type.

Boolean Attribute (`required`)



Valued Attribute (`class="form-control""`)

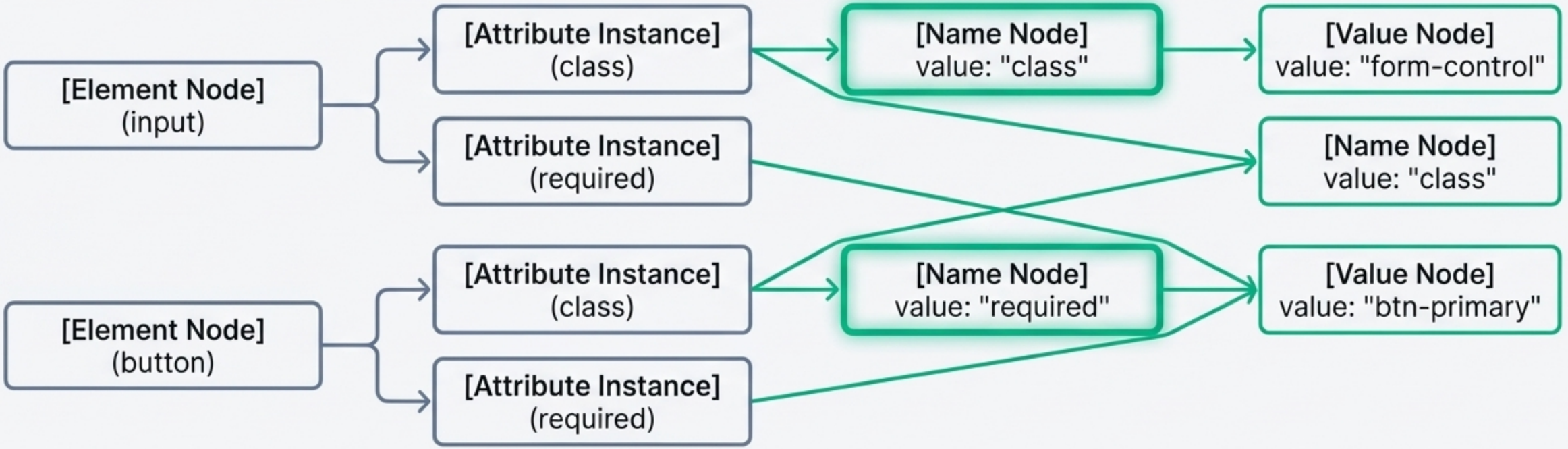


NotebookLM

The Power of Reuse: Name and Value Deduplication

A key advantage of the new structure is that name and value nodes are canonical and reused across the entire document graph. An attribute name like `class` or `required` now exists as only one node, no matter how many times it is used.

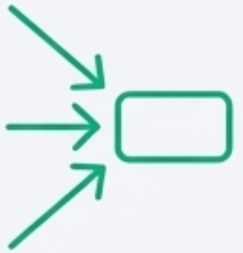
```
<input class="form-control" required>
<button class="btn-primary" required>
```



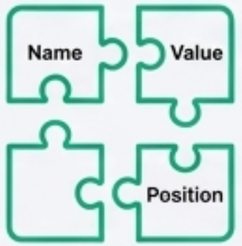
NotebookLM

Key Design Wins

✅ **Native Boolean Support:** The model now correctly represents attributes with or without values. The presence/absence of a `value` edge defines the type.



✅ **Full Deduplication:** Name and value nodes are created once and reused, reducing graph size and memory footprint for documents with common attributes.



✅ **Semantic Clarity:** Clean and explicit separation of an attribute's name, its value, and its positional instance on an element.

✅ **High-Fidelity Roundtrip:** HTML parsed into the graph and regenerated now maintains perfect semantic integrity (e.g., `` `required` `` stays `` `required` ``).

NotebookLM

Implementation: API Impact for Consumers

The primary change for developers using the `Html_MGraph` is the return type of the `get_attributes()` method. It now correctly reflects that an attribute's value can be absent.

Old Signature

```
def get_attributes(element_id: str)
    -> Dict[str, str]:
```

New Signature

```
def get_attributes(element_id: str)
    -> Dict[str, Optional[str]]:
```

NotebookLM

⚠ Managing the Transition: Breaking Changes

This refactor introduces breaking changes. A “clean break” strategy is acceptable as the graph is ephemeral and all consumers can be updated simultaneously.

Graph Incompatibility

Graph Incompatibility

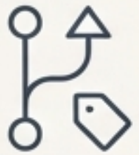
Existing serialised graphs are incompatible with the new structure.

API Contract Change

API Contract Change

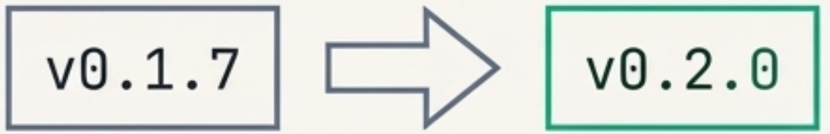
The `get_attributes()` return type has changed (as detailed previously).

Responsible Versioning



Responsible Versioning

To signal this breaking change, the package version will be bumped from `v0.1.7` to `v0.2.0`, following semantic versioning.



NotebookLM

Confidence Through Coverage: A Multi-Layered Testing Strategy

The new implementation is supported by a thorough testing strategy to ensure correctness, fidelity, and prevention of regressions.

Unit Tests

- Focus on `add_attribute()` and `get_attributes()` methods.
- Cover boolean, valued, and empty-valued attribute cases.

Roundtrip Fidelity Tests

- Ensure that HTML -> Graph -> HTML conversion is lossless.
- Verify that `<input required>`does not become `<input required="">`.

Graph Visualization Tests

- Use visual inspection of generated graphs to confirm the new structure is correctly formed for complex documents.

NotebookLM

The Evolution of a Model: Before & After

Aspect	Before	After
Boolean Attr Support	✗ Crashes	☑ Native
Name Deduplication	✗ None	☑ Full
Value Deduplication	✗ None	☑ Full
Roundtrip Fidelity	✗ `required=""`	☑ `required`
Graph Semantics	Conflated	☑ Clean separation
Nodes per Attribute	1	2-3 (with reuse)

NotebookLM

Appendix: Code & Resources

For those wishing to examine the implementation, the core changes are located in the following files:

1. `Html_MGraph__Attributes.py` (Core logic)
2. `Html_MGraph__Document.py` (Wrapper method updates)
3. `Html_MGraph__Document__To__Html_Dict.py` (Consumer updates)
4. `Html_MGraph__Data__Extractor.py` (Visualization data)
5. `Html_MGraph__Render__Labels.py` (Visualization labels)
6. All related test files.

Questions?

NotebookLM