Can `morphdom` be used with any virtual DOM implementation?

No, `morphdom` cannot be used with any virtual DOM implementation. To keep code size small and fast, morphdom requires that virtual DOM nodes implement the minimal set of methods and properties based described above. One of the goals of `morphdom` is to stay as close as possible to the real DOM while allowing for a more optimized virtual DOM when it makes sense.

Using `morphdom` with a virtual DOM

`morphdom` has always supported diffing real DOM nodes with real DOM nodes and this will continue to be supported. Support for diffing real DOM with a *virtual* DOM was introduced in `v2.1.0`. Virtual DOM nodes are expected to implement a minimal DOM API that consists of the following methods and properties:

- `node.firstChild`
- `node.nextSibling`
- `node.nodeType`
- `node.nodeName`
- `node.namespaceURI`
- `node.nodeValue`
- `node.attributes`
- `node.value`
- `node.selected`
- `node.disabled`
- `node.hasAttributeNS(namespaceURI, name)`
- `node.actualize(document)` [1]
- `node.isSameNode(anotherNode)` [2]

NOTES:

1. In addition to the standard DOM node methods and properties, a virtual DOM node must also provide a

`node.actualize(document)` method. The `node.actualize(document)` will be called when the virtual DOM node needs to be upgraded to a real DOM node so that it can be moved into the real DOM.

2. A virtual DOM node may choose to implement `isSameNode(anotherNode)` to short-circuit diffing/patching a particular DOM subtree by treating two nodes as the "same"
3. A virtual DOM node may choose to implement the non-standard `assignAttributes(targetNode)` to optimize copying the attributes from the virtual DOM node to the target DOM node. If virtual DOM node implements `assignAttributes(targetNode)` then it is not necessary to implement `node.attributes`.

marko-vdom is the first virtual DOM implementation that is compatible with `morphdom` and it can be used as a reference implementation.

FAQ

Why support a virtual DOM?

Working with real DOM nodes is fast, but real DOM nodes do tend to have more overhead (the amount of overhead associated with real DOM nodes will vary drastically by browser). In order to be 100% compliant with the DOM specification, real DOM nodes require a lot of internal "bookkeeping" and validation checks that slow certain operations down. Virtual DOM nodes have the advantage that they can be optimized to use less memory and enable better performance since they are not required to be compatible the entire DOM specification.

When using `morphdom` to update the view, performance will be largely dictated by how much time it takes to render the view to a virtual DOM/real DOM and how long it takes to walk the tree (including iterating over attributes). We are seeing signficant performance improvements when utilizing a virtual DOM. Please see the marko-vdom benchmarks to better understand the performance characteristics of the virtual DOM and the real DOM.