Igor's Node.bind Notes

*Authors: igor@angularjs.org*

This is a collection of notes and questions gathered while I researched and reviewed [NodeBind](https://github.com/Polymer/NodeBind/) project.

## Documentation

- [source is the documentation](https://github.com/Polymer/NodeBind/)

- no documentation in the code

- no documentation on the polymer site

- [outdated doc](https://github.com/Polymer/docs/blob/f316b977903c019921df58b782244595058604f1/platform/node_bind.md) in the polymer repo

## API Signature

The signature is: node.bind(propertyName, valueOrBinding, oneTime)

- propertyName - the name of the Node property to bind to

- some property names are [special-cased](https://github.com/Polymer/docs/blob/f316b977903c019921df58b782244595058604f1/platform/node_bind.md#binding-types) (examples: TextArea element has 'value' binding) and have custom implementation

- valueOrBinding - actual value to update the Node with (only for oneTime bindings) or a binding object that implements an interface compatible with PathObserver from observe-js

- oneTime - optional boolean value that tells the node that the binding that it should update the Node the the value provided to node.bind

The return value of calling node.bind is the binding passed in.

## node.bindings\_ property

If a Platform.enableBindingsReflection is enabled, each node that has been bound at least once has a property bindings\_. This property is a map of bound property names to Binding objects passed into node.bind.

Currently the primary use of this api is debugging and tooling.

## The Binding interface

The Binding interface represents an adapter between the templating layer and DOM.

It allows the templating layer to be agnostic of nuances of updating various DOM properties (e.g. checked property of input element should be set only with boolean value, value property of text input element should not be set to null or undefined because they'll be stringified by DOM to "null" or "undefined")

Additionally, it enables the DOM to signal to interested party that certain properties have changed. In DOM today, this is currently done via a myriad of events without any consistent mapping between the property name and event fired.

The interface has the following methods:

- open - called synchronously with domPropertySetter function when the binding is registered.

- close - called when another binding is registered with the same element, when a binding is unbound.

- setValue - called when the property value was changed (by the component, user interaction or binding).

- discardChanges - called after setValue call for an unclear reason (like something to do with preventing O.o() to notify us of this change?)

## Tests

- mocha syntax

- almost no unit tests, but instead integration tests that use observe-js

- majority of tests are async (mainly because of observe-js?)

- the karma setup is odd (optimized for CI environment and not for development)

- for some reason (bug in karma or in karma config) the exceptions often didn't include stack trace

- the test coverage should be improved

## Implementation Dependencies

The code depends on observe-js because of frequent Platform.performMicrotaskCheckpoint(); calls. This means that any app using node.bind prollyfill must also include observe-js.

## Misc Implementation notes

The code surprisingly contains unnecessary [polyfill for Document#contains](https://github.com/Polymer/NodeBind/blob/eb5ee7941f712cbee755da24ab7553e2d90fb99d/src/NodeBind.js#L47-L54) for IE and [also some unused helper method](https://github.com/Polymer/NodeBind/blob/eb5ee7941f712cbee755da24ab7553e2d90fb99d/src/NodeBind.js#L28-L45).

## Questions / Concerns

### Attributes vs Properties

The current implementation works with both Node [properties](https://github.com/Polymer/NodeBind/blob/eb5ee7941f712cbee755da24ab7553e2d90fb99d/src/NodeBind.js#L196-L198) and [attributes](https://github.com/Polymer/NodeBind/blob/eb5ee7941f712cbee755da24ab7553e2d90fb99d/src/NodeBind.js#L118-L128). It's not clear if this is just an inconsistency or conscious design decision.

It would be better to always work only with properties. Some properties are automatically synchronized by attributes so just by updating properties we get the attributes updated as well (which can be useful for constructing css selectors that match on attribute presence).

### One time bindings

It's unclear why node.bind needs to know about one-time vs non-one time binding. Because it does, and implements semantics for one-time binding any system that uses this feature will be locked to these semantics. (e.g. lazy-evaluated one-time binding is impossible because the default implementation uses eager evaluation)

### "Boolean bindings"

"boolean bindings" (`hidden?="{{ foo.bar }}"`) are handled by node.bind which seems good, but why does node.bind need to [inspect the attribute name](https://github.com/Polymer/NodeBind/blob/eb5ee7941f712cbee755da24ab7553e2d90fb99d/src/NodeBind.js#L137-L141) to determine if this is a "boolean binding" or just a regular binding? I think this is a responsibility of a higher layer and that layer should tell node.bind that this binding is a boolean biddings (just as oneTime bindings are currently signaled).

Additionally node.bind [removing the "template" attribute](https://github.com/Polymer/NodeBind/blob/eb5ee7941f712cbee755da24ab7553e2d90fb99d/src/NodeBind.js#L139) to clean up the DOM is a signal that if this feature is needed, it should be part of template instantiation and not node.bind

### Biding#discardChanges

it's unclear what's the purpose for discardChanges api on the Binding interface.

### Tight coupling with observe-js and PathObserver api

The level of coupling is surprising and results in a lot of code smell. Some examples:

* Numerous calls to [Platform.performMicrotaskCheckpoint()](https://github.com/Polymer/NodeBind/blob/eb5ee7941f712cbee755da24ab7553e2d90fb99d/src/NodeBind.js#L215)
* [Monkey-patching observer's close method](https://github.com/Polymer/NodeBind/blob/eb5ee7941f712cbee755da24ab7553e2d90fb99d/src/NodeBind.js#L219-L228).

It seems unnecessary to complicate the node.bind with any of the observe-js stuff. A good sign of a solid api is that it is useful on its own.