Parser Abstraction Notes

As a part of the Change Detection implementation currently being ported from Angular.dart (<https://github.com/caitp/watchtower.js>), one of the things which is present in the Dart code, which is somewhat problematic, is the tight binding betweem the core parser and the dirty checking AST.  
  
While it may be possible for me to come up with something usable by the WatchGroup implementations independently from the parser’s implementation, this is potentially not very helpful as it may require a conversion from the parser’s parse tree into something the WatchGroup implementation can understand. Since Angular 2.0 is aiming for a more modular approach, it would be ideal if the change detection code is not tightly bound to another module in the library. However, it must still be able to interoperate with parsed Angular expressions.  
  
So because of this, I would be interested in figuring what what sort of abstraction is really going to work.

My goals here, as stated, include:

1. Remove tight binding between Parser and Change Detection
2. Change Detection’s interaction with Parser should happen through a very simple abstraction, which is easy for other applications to make use of.

The change detection essentially needs to know, I believe, 3 things:

1. If a token is a function call (and therefore, which arguments are to be passed to it)
2. If a token is a field name
3. If a token is a constant value

Unfortunately, this isn’t necessarily a simple amount of information to transmit, and it may actually be necessary to send an entire parse-tree along with it.  
  
One thing that could be possible is to make sure the parse-tree/AST is in a format compatible with Esprima (one of the more commonly used ASTs), but this feels like a difficult undertaking, and hard to maintain. On the other hand, this would help de-couple the implementations a fair bit.