ReflectToAttribute: This property is used to reflect properties as attributes of the element (in the tag in rendered element). There are two ways to reflect a property to attribute:

* One, when you set a property when calling the element like <paper-card elevation=2></paper-card>. Now elevation will appear in attributes property of the element (this.attributes).
* Second, you can set reflectToAttribute property true for that property. Eg
  + In case of elevation there is some default value but it still appears in attributes list of paper-card
  + Read only properties can be reflected using this property

**Note**: Reflect to attribute has performance impact.

Returning function as value of a property

generateRowCss:{

type:Object,

value:function(){

return function(item){

console.log('app.generateRowCss');

console.log(item);

var levels = ['#FFFFFF', '#FFEBEE', '#FFCDD2', '#EF9A9A'];

var min = 150;

var max = 450;

var level = Math.floor((item.calories - min)/(max-min)\*levels.length);

console.log(level);

console.log('background:'+levels[level]+';');

return 'background:'+levels[level]+';';

}

}

}

<http://plnkr.co/edit/35FF8rdnQICaLYSgaAyY?p=preview>

Element can make a call to itself

To call Polymer Function from dom-bind use



Instantiating from js does not create a shadow-root directly

Ones mentioned in properties will not be accessible before ready(ie in created)

getComputedStyleValue to get value of custom vars.

Polymer.dom (this) & Polymer.dom(this.root) are different former returns content children and lateral local dom’s and are different from this which will return rendered one

Shady VS Shadow dom

Doc structure is different as a result traversing is different from non-polymer js atleast

Shady-dom drawback: It’s leaking details, confusing CSS, and all the rest because of classic Jquery structure it renders.

HTMLImports.whenReady(cb) is from webcomponentsjs polyfill.

Use single binding for iron-selector selected attribute for routing as otherwise will it create issue whileback button is pressed as both data-path and href gets stored in location history

**Rules on creating custom elements**

1. The name of a custom element **must contain a dash (-)**. So <x-tags>, <my-element>, and <my-awesome-app>are all valid names, while <tabs> and <foo\_bar> are not. This requirement is so the HTML parser can distinguish custom elements from regular elements. It also ensures forward compatibility when new tags are added to HTML.
2. You can't register the same tag more than once. Attempting to do so will throw a DOMException. Once you've told the browser about a new tag, that's it. No take backs.
3. Custom elements cannot be self-closing because HTML only allows [a few elements](https://html.spec.whatwg.org/multipage/syntax.html#void-elements) to be self-closing. Always write a closing tag (<app-drawer></app-drawer>).