**Lit Element Learning**

1. The three basic imports that we need

import { LitElement, html, css} from 'lit-element';

1. The render function should be pure function with out any side effects.
2. Define your component

customElements.define('basic-setup', BasicSetup);

1. LitElement will decode any html attributes set on a component as properties which can be used from javascript.
2. We set properites in a static getter above constructor. Any changes to properties defined in the static properties will trigger a re-render of the component.

// The supported types are String, Number, Boolean, Array Object.

  static get properties() {

    return {

      message: { type: String },

      count: { type: Number },

    };

  }

1. The constructor after the property setter

constructor() {

    super();

    // default values can be set from the constructor

    this.count = 0;

  }

1. Another setter example

  static get properties() {

    return {

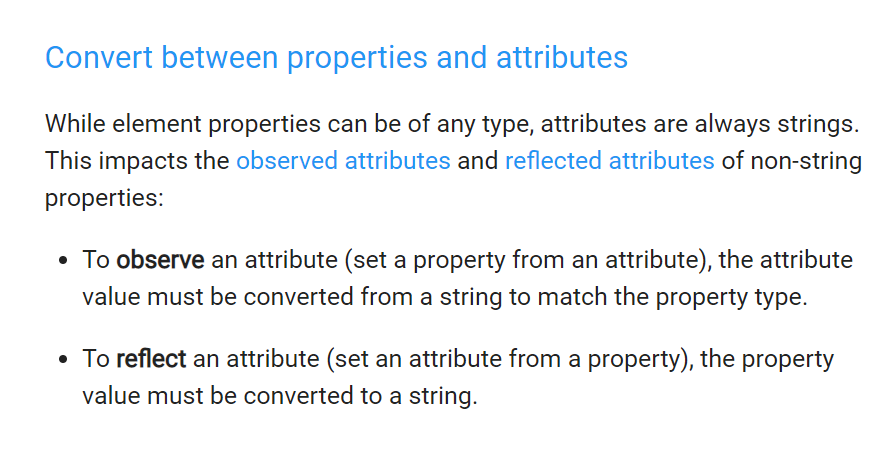
      href: { type: String },

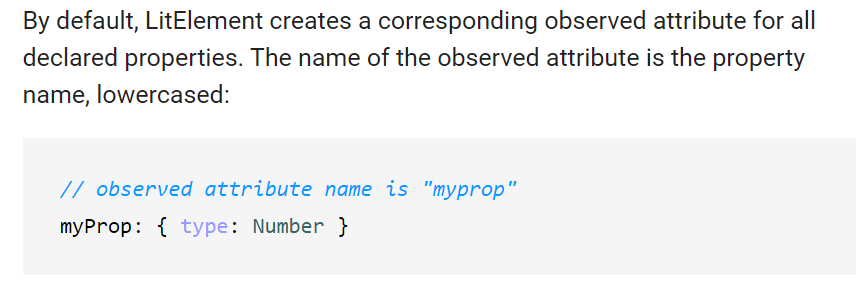
      checked: { type: Boolean }

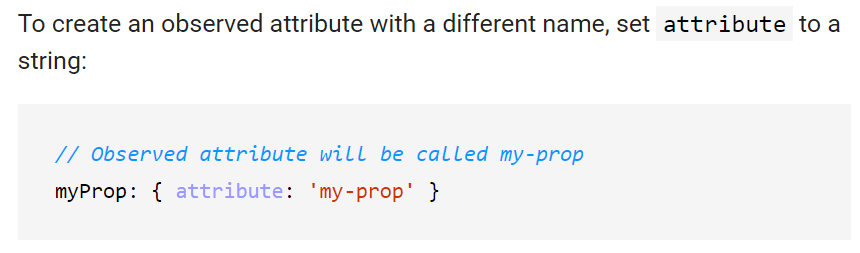
    }

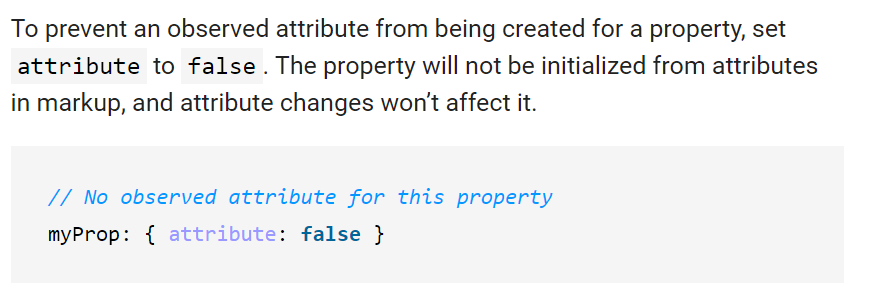
  }

1. Differences between attributes and properties:
   1. Attributes are defined in HTML, for example setting the id attribute: <div id="foo">
   2. Properties are defined in javascript, for example setting the id property:
      1. const div = document.createElement('div');
      2. div.id = 'foo';
2. If you'd like to reflect changes to the property as an attribute, flag the reflect boolean in the property descriptor.
3. While element properties can be of any type, attributes are always strings
4. Working Definition: attributes are in html and properties are in js. We can make attributes into properties and make them observalbe
5. An observed attribute fires the custom elements API callback attributeChangedCallback whenever it changes. By default, whenever an attribute fires this callback, LitElement sets the property value from the attribute using the property’s fromAttribute function.









An observed attribute can be used to provide an initial value for a property via markup.

Example

**class** MyElement **extends** LitElement {

**static** **get** properties() { **return** {

myProp: { attribute: **true** },

theProp: { attribute: **false** },

otherProp: { attribute: 'other-prop' },

};}

//…..other code

attributeChangedCallback(name, oldval, newval) {

console.log('attribute change: ', name, newval);

**super**.attributeChangedCallback(name, oldval, newval);

}

changeAttributes() {

**let** randomString = Math.floor(Math.random()\*100).toString();

**this**.setAttribute('myprop', 'myprop ' + randomString);

**this**.setAttribute('theprop', 'theprop ' + randomString);

**this**.setAttribute('other-prop', 'other-prop ' + randomString);

**this**.requestUpdate();

}

1. If we have our attribute reflect = true, it means that when ever we change this attribute from out side it will also change internally and the component will re-draw. This is in index.html

  <bilzaa-properties

  abc="This is an attribute from html"

  ></bilzaa-properties>

<button id="just">Click</button>

<script>

document.getElementById("just").addEventListener("click",function(){

  let elm = document.querySelector("bilzaa-properties");

  elm.abc= "the abc has been changed";

console.log('elm :', elm.abc);

})

</script>

The code inside the element is:

static get properties(){

    return {

      abc: {type: String,reflect: true},

      xyz: {type: Number},

      data: {type: Array},

      count:{type: Number}

    }

}

We can dispatch a custom event and send out any information that we want :

clickHandler() {

this.flag = (this.flag === true) ? false : true;

let event = new CustomEvent('bilzaa-toggle', {

    detail: {

      property: this.property,

      boxtype : this.boxtype,

      flag:  this.flag

    }

  });

this.dispatchEvent(event);

console.log('Flag :', event.detail.flag);

}