CHEAT SHEET

WEB COMPONENTS

JS

Created by @Manz (https://twitter.com/Manz) https://lenguajejs.com/

T Template — Inert document fragment

HTML TAG

<template> inert tag for html content

```
<template shadowroot="open">
  #DOCUMENT-FRAGMENT
    <div>...</div>
</template>
```
declarative shadow dom

API TEMPLATE — HTMLTemplateElement

DF	**.content** ref to document fragment node
SR	**.shadowRoot** shadow dom root node

D DOM HTML Access — DOM manipulation

API HTML

s	**.innerHTML** get/replace HTML markup
s	**.outerHTML** idem (includes HTML tag)
s	**.textContent** get/replace text content

```
element.innerHTML = `<div>text</div>`;
```

CREATE HTML / ADD ELEMENT

e **document.createElement(tag)** create node

```
document.createElement("div");
```

o **.appendChild(child)** add inside element

```
element.appendChild(node);
```

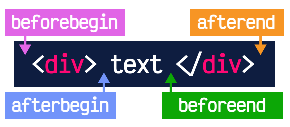
INSERT HTML / ELEMENT

▨ **.insertAdjacentHTML(pos, html)** add

```
el.insertAdjacentHTML("afterend",
  `<div><p>Content</p></div>`);
```

▨ **.insertAdjacentElement(pos, node)**

```
el.insertAdjacentElement("afterend", node);
```



html tag position to insert

F Find HTML Elements — DOM search

TRADITIONAL DOM SEARCH API

o	**document.getElementById(id)** find by #id
a	**.getElementsByName(name)** name attr
a	**.getElementsByClassName(class)** .class
a	**.getElementsByTagName(tag)** html tag

```
document.getElementById("name");
```

MODERN DOM SEARCH API

o	**.querySelector(selector)** return first elem
a	**.querySelectorAll(selector)** ret all elems
o	**.closest(selector)** return closest ancestor
b	**.matches(selector)** matches with elem?

```
document.querySelector(".menu > p");
```

HTML ATTRIBUTE API

b	**.hasAttributes()** element w/attributes?
s	**.getAttributeNames()** return attrs array
b	**.hasAttribute(name)** check attribute
s	**.getAttribute(name)** return value attr
▨	**.removeAttribute(name)** delete attribute
▨	**.setAttribute(name, value)** modify attr
b	**.toggleAttribute(name, force)** add/del

```
element.setAttribute("name", "value");
```

Custom Elements · Templates · Shadow DOM · Modules · CSS Scopes

C Custom Elements — HTML Components

HTML STRUCTURE

<prefix-component> custom html tag

```
<base-name attribute="value">
  <div>content</div>   SHADOW DOM
  <div>content</div>   LIGHT DOM
</base-name>
```

CLASS API — JS PROPERTIES & METHODS

o	**constructor()** initial method — super()
🟨	**public()** public properties/methods
🟨	**#private()** private properties/methods

```
class BaseName extends HTMLElement {
  ...
}
customElements.define("base-name",BaseName)
```

COMPONENT LIFECYCLE HOOKS — WHEN?

▨	**connectedCallback()** added to DOM
▨	**disconnectedCallback()** remove from DOM
▨	**adoptedCallback()** move to new doc.

```
class BaseName extends HTMLElement {
  constructor() {
    super();
    ...
  }
  connectedCallback() { ... }
  disconnectedCallback() { ... }
  adoptedCallback() { ... }
}
customElements.define("base-name", BaseName)
```
don't forget super() on constructor

ATTRIBUTE OBSERVATION

a	**get observedAttributes()** notify changes
▨	**attributeChangedCallback(attr, old, now)**

```
class BaseName extends HTMLElement {
  static get observedAttributes() {
    return ["name1", "name2"];
  }
  attributeChangedCallback(name, old, now){
    ...
  }
}
customElements.define("base-name", BaseName)
```
fire this callback when a observed attribute changes

CUSTOM ELEMENTS REGISTRY — GLOBAL REGISTRY

▨	**customElements.define(name, class)** reg
o	**customElements.get(name)** register elem
▨	**customElements.upgrade(node)** update el.
p	**customElements.whenDefined(name)** fire.

C Custom Events — Send/Receive events

CUSTOM EVENTS API

b **.dispatchEvent(event)** send event

CUSTOM EVENT OBJECT

o	**.detail** data object with information
b	**.bubbles** bubbles up through the DOM
b	**.composed** send across shadow DOM

```
const event = new CustomEvent("message", {
  detail: { ... },
  bubbles: true,
  composed: true
});
```

S Slots — External slots

SIMPLE SLOT

<slot> external html to inside component

```
<slot></slot>
```

NAMED SLOT

<slot name="text"> multiple html to inside

```
<slot name="title"></slot>
```

CSS — HTMLTemplateElement

🟦 **::slotted(selector)** style to slotted tags

EVENTS — HTMLTemplateElement

🟨 **slotchange** detect slot-element changes

```
<base-name>
  <h2 slot="title">Default</h2>
  <p slot="description">Text</p>
</base-name>
```
MULTIPLE SLOT · SIMPLE SLOT

C CSS in WebComponents — Styles (CSS)

CSS WITHOUT SHADOW DOM — GLOBAL CSS

```
connectedCallback() {
  this.innerHTML = `
    <style>p { color: red; }</style>
  `;
}
```

CSS WITH SHADOW DOM — LOCAL CSS

```
connectedCallback() {
  this.shadowRoot.innerHTML = `
    <style>p { color: red; }</style>
  `;
}
```

CSS CONTAINER

🟦	**:host** style custom element (container)
🟦	**:host(selector)** idem, if match container
🟦	**:host-context(selector)** idem, ancestor

CSS PARTS

<span part="name"> define part

```
<span part="name"></span>
```

🟦 **::part(selector)** style surface parts

S Shadow DOM — .shadowRoot (DOM isolate)

WEBCOMPONENT WITHOUT SHADOW DOM

```
class BaseName extends HTMLElement {
  constructor() {
    super();
    this.innerHTML = `<div></div>`;
  }
}
```

WEBCOMPONENT WITH SHADOW DOM

s **.attachShadow(options)** add shadow dom

SHADOW DOM OPTIONS

s	**mode** encapsulation mode — open / closed
b	**delegatesFocus** shadow get focus — false

```
class BaseName extends HTMLElement {
  constructor() {
    super();
    this.attachShadow({ mode: "open" });
    this.shadowRoot.innerHTML = `<p></p>`;
  }
}
```

Emezeta.com

CHEAT SHEET

LIT-ELEMENT / LIT-HTML

lit-html

Created by @Manz (https://twitter.com/Manz) https://lenguajejs.com/

T Templates — Next-gen lit-html templates

SETUP & CONFIGURATION

```
<script type="module">
  import { html, render } from "...";
</script>
```

LIT-HTML SIMPLE TEMPLATES

- TR html`code` create HTML template
- ▨ render`template, element` update page

```
const tpl = html`<div>Hello</div>`;
render(tpl, document.body);
```

LIT-HTML DYNAMIC TEMPLATES

- TR html`code, ...values` template with data
- ▨ render`template(val), element` update

```
const p = (t) ⇒ html`<p>${t}</p>`;
render(tpl("Text"), document.body);
```

HELPERS (BIND)

<tag	value=${var}>	string attribute
<tag	?disabled=${var}>	boolean attribute
<tag	.value=${obj.value}>	bind object value
<tag	@event=${func}>	bind event to func.

CONDITIONALS (TERNARY) / NESTED TEMPLATES

```
html`${user.logged
  ? html`Welcome ${user.name}`
  : "User not logged in"                nothing
}`;
```

LOOPS

MAP LOOP

```
html`<ul>${
  arr.map(item ⇒ html`<li>${item}</li>`)
}</ul>`;
```

ARRAY LOOP

```
for (const item of items) {
  arr.push(html`<li>${item}</li>`);
}
html`<ul>${arr}</ul>`;
```

D Directives — lit-html/directives/name.js

DIRECTIVES

- TR asyncAppend(iterable) async add data
- TR asyncReplace(iterable) async change data
- TR cache(code) cache DOM for a bind/input
- o nothing render a empty text node
- o ifDefined(value) set value, else no-op
- o guard(deps, func) render func on change
- o live(value) update value (outside lit-html)
- TR repeat(items, keyfn, tpl) repeat template

REPEAT DIRECTIVE

```
import { repeat } from "...";
html`<ul>${repeat(items,
  (items) ⇒ items.id,
  (item) ⇒ html`<li>${item.name}</li>`)}
</ul>`;
```

- TR templateContent(tag) render <template>
- TR unsafeHTML(html) render unsafe code
- TR unsafeSVG(svg) render unsafe svg code
- TR until(...values) render w/priority (1=more)

UNTIL DIRECTIVE

```
import { until } from "...";
html`${until(fetchPromise,
             html`<p>Loading...</p>` )}`;
```

C Component — Lit-Element web component

SETUP & CONFIGURATION

```
<script type="module">
  import { LitElement, html } from "..."
</script>
```

DEFINE COMPONENT — LIT-ELEMENT

```
class BaseName extends LitElement {
  ...
}
customElements.define("base-name", BaseName)
```

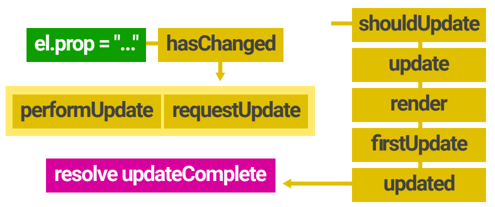
BASIC LIFECYCLE HOOKS — WHEN?

- ▨ update(props) reflect prop to attr & render
  if override, super.update(props) or no render
- TR render() use lit-html to render template

```
class BaseName extends LitElement {
  constructor() {
    super();
    ...
  }
  connectedCallback() {                 don't forget
    super.connectedCallback();          call super on
  }                                     parent hooks
  update() { ... }
  render() {
    html`<div>Component</div>`;
  }
}
customElements.define("base-name", BaseName)
```

ADVANCED LIFECYCLE HOOKS — WHEN?

- p requestUpdate() manually start an update
- p requestUpdate(propName, old) prop setter
- ▨ performUpdate() microtask after ev.loop
- b shouldUpdate(props) update proceed
- ▨ firstUpdate(props) called on first update
- ▨ updated(props) DOM updated & rendered
- p .updateComplete true=no pending updates



S Shadow DOM — By default, uses ShadowDOM

LITELEMENT WITHOUT SHADOW DOM

```
class BaseName extends LitElement {
  createRenderRoot() {
    return this;
  }
}
```

D Decorators — Typescript or Babel needed

DECORATORS

```
import { customElement, property } ...

@customElement("base-name")
class BaseName extends LitElement {
  @property()
  prop1 = "value";
  prop2 = "value";
  ...
  render() {
    html`<div>Component</div>`;
  }
}
```

P Properties — Properties != Attributes

PROPERTIES DECLARATION SYNTAX

```
static get properties() {
  return {
    prop1: { type: String, ... },
    prop2: { type: Boolean, ... }
  }
}
```

PROPERTIES OPTIONS

- o type hint for convert between props/attr
- o converter custom func or object props/attr
  - f fromAttribute(value, type) convert to prop
  - f toAttribute(value, type) convert to attr
- f hasChanged(now, old) true=requestUpdate
- b attribute associate prop with a attribute
- b noAccessor avoid generate def. accesor
- b reflect autoset prop value to attribute

S Styling — CSS in Components

SETUP & CONFIGURATION

```
<script type="module">
  import { ..., html, css } from "..."
</script>
```

STYLES IN COMPONENTS — ALL INSTANCES

- var(--name) set css variable from in/out
- ${var} set javascript variable
- CR unsafeCSS(css) set unsafe css code

```
static get styles() {
  return css`                          --theme: ...
    :host { color: var(--theme) }
    div { color: red }                 css`...`
    button { color: ${bgColor} }
  `;
}   return [super.styles, css`...`, css`...`]
```

STYLES IN COMPONENTS — SPECIFIC INSTANCES

```
render() {
  return html`
    <style>
      div { color: red }
    </style>
    <div>Hello!</div>
  `;
}
```

D CSS Directives — lit-html/directives/name.js

CLASSMAP

```
import { classMap } from "...";
const classes = { selected : true };
const result = html`
  <div class=${classMap(classes)}>
    Content
  </div>
`;
```

STYLEMAP

```
import { styleMap } from "...";
const styles = { color : "red" };
const result = html`
  <div style=${styleMap(styles)}>
    Content
  </div>
`;
```

U https://unpkg.com/lit-element?module
https://unpkg.com/lit-html/directives/class-map.js?module

https://cdn.skypack.dev/lit-element
https://cdn.skypack.dev/lit-html/directives/class-map.js

Emezeta.com