

# 1. WebComponent

## 1.1. Notion de WebComponent

Un "WebComponent" est un **composant web** dans un **format se voulant normalisé/standard (W3C)** que l'on peut *utiliser en pur "html/javascript"* sans se soucier de la technologie qui a servi à le développer.

Autrement dit un composant web codé avec angular peut par la suite être utilisé dans une application web qui n'est pas basée sur angular ou vice versa.

## 1.2. WebComponent élémentaire en pur javascript

**hello-world-component.js**

```
class HelloWorldComponent extends HTMLElement {
  constructor() {
    super();
  }
  connectedCallback() {
    this.innerHTML = '<div style="color:red;font-size:24pt">Hello world!</div>'
  }
}

customElements.define("hello-world-component", HelloWorldComponent)
```

**with-web-component.html**

```
<html>
<head>
  <title>with-web-component</title>
  <script src="./hello-world-component.js" ></script>
</head>

<body>
  <h1>with-web-component</h1>

  <!-- NB:cette balise personnalisée correspond au webcomponent
  élémentaire codé dans le fichier hello-world-component.js -->
  <hello-world-component></hello-world-component>
</body>
</html>
```

Affichage résultant de ce code dans tous les navigateurs récents (chrome, firefox, edge) :

**with-web-component**

**Hello world!**

### 1.3. Code source des exemples

Référentiel <https://github.com/didier-mycontrib/angular12plus>  
et partie `vanilla-js/basic-web-component`

### 1.4. Web Component avec attributs

*with-attr-component.js*

```
class WithAttrComponent extends HTMLElement {
  constructor() {
    super();
    this.name="?";//default value
    this.color="red";//default value
  }

  //static get observedAttributes() to define list of observedAttributes
  static get observedAttributes() {
    return ['name', 'color'];
  }

  //attributeChangedCallback(property, oldValue, newValue) callback
  // to update new changed attribute/property value
  attributeChangedCallback(property, oldValue, newValue) {
    if (oldValue === newValue) return;
    this[ property ] = newValue;
  }

  connectedCallback() {
    this.innerHTML =
      `<div style="color:${this.color};font-size:24pt">Hello ${this.name}</div>`
  }
}

customElements.define("with-attr-component", WithAttrComponent)
```

Utilisation :

```
<html><head> ...
  <script src="/with-attr-component.js" ></script>
</head>
<body>
  <with-attr-component name="toto" color="blue"></with-attr-component>
  <with-attr-component name="titi" color="green"></with-attr-component>
...

```

Hello toto!  
Hello titi!

## 1.5. Web Component avec "shadow DOM" et "slot"

### with-shadow-component.js

```
class WithShadowComponent extends HTMLElement {
  constructor() {
    super();
  }

  connectedCallback() {
    const shadow = this.attachShadow({ mode: 'closed' }); //only accessible by web component
    //const shadow = this.attachShadow({ mode: 'open' }); //accessible outside with Element.shadowRoot
    shadow.innerHTML = `
      <style>
      div {
        text-align: center;
        font-weight: normal;
        padding: 1em;
        background-color: #eee;
        border: 1px solid blue;
      }

      .c1 { color: green; }

      :host {
        display: block;
        background-color: lightgreen;
        padding : 1em;
      }
      </style>
      <div class="c1 c2">With-shadow-DOM <slot name="comment"></slot></div>
    `;
  }
}

customElements.define("with-shadow-component", WithShadowComponent)

//NB: le selecteur :host permet de cibler la racine du webComponent
//ici la balise <with-shadow-component></...> contenant elle meme la <div>...</>
```

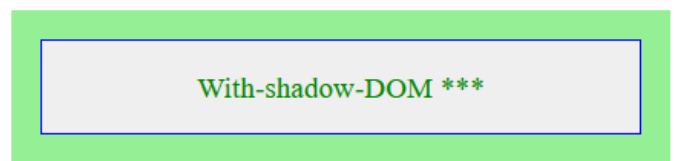
### Utilisation :

```
...
<p class="c1 c2">before</p>
  <with-shadow-component>
    <span slot="comment">***</span>
  </with-shadow-component>
<p class="c1 c2">after</p>

et

.c1 { color:blue;}
```

[before](#)



[after](#)

## 1.6. toggle-panel webComponent (slot , shadow , attribute, ...)

### *toggle-panel-component.js*

```
class TogglePanelComponent extends HTMLElement {
  constructor() {
    super()
    this.label="TogglePanelComponent"; //default value
    this.toggleP = false;
  }

  initComponentInnerHtml(){
    this.componentCssTemplateString = `
    .my-card { margin-top: 0.1em; margin-bottom: 0.1em; }
    .my-card-header { border-top-left-radius: 0.3em; border-top-right-radius: 0.3em;
padding: 0.1em; margin-bottom: 0px;}
    a { text-decoration: none;}
    .my-card-body {border: 0.1em solid blue; border-bottom-left-radius: 0.3em;
border-bottom-right-radius: 0.3em; padding: 0.2em;}
    .my-bg-primary { background-color: blue;}
    .my-text-light { color : white;}
    .my-icon { color : blue; background-color: white; margin: 0.2em;
padding-left: 0.2em; padding-right : 0.2em;
min-width: 1em; font-weight: bold;}
    .my-collapse { display : none;}
    .my-show { display : block ;}
    `

    this.componentHtmlTemplateString = `
    <div class="my-card">
      <h4 class="my-card-header my-bg-primary" id="myCardHeader">
        <a class="my-text-light" >
          <span class="my-icon" id="myIconShow">+</span>
          <span class="my-icon" id="myIconCollapse" style="display:none">-</span>${this.label}
        </a>
      </h4>
      <div id="myCardBody" class="my-card-body my-collapse">
        <slot></slot>
      </div>
    </div>
    `

    this.componentInnerHtml = `
    <style>
    ${this.componentCssTemplateString}
    </style>

    ${this.componentHtmlTemplateString}
    `
  }

  // component attributes
  static get observedAttributes() {
    return ['label'];
  }
}
```

```

}

// attribute change
attributeChangedCallback(property, oldValue, newValue) {
  if (oldValue === newValue) return;
  this[ property ] = newValue;
}

connectedCallback() {
  const shadow = this.attachShadow({ mode: 'closed' });//only accessible by web component
  //const shadow = this.attachShadow({ mode: 'open' });//accessible outside with Element.shadowRoot
  this.initComponentInnerHtml();
  shadow.innerHTML = this.componentInnerHtml;
  shadow.getElementById("myCardHeader").addEventListener("click", function(evt){
    this.toggleP = ! this.toggleP;
    //console.log("toggleP="+this.toggleP);
    const myCardBody = shadow.getElementById("myCardBody");
    myCardBody.setAttribute("class", "my-card-body my-collapse"
      + (this.toggleP?"my-show":""))

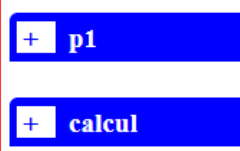
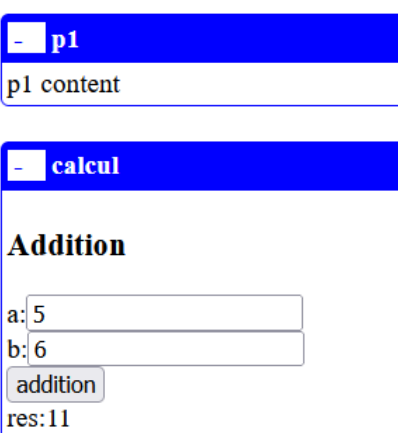
    const myIconShow = shadow.getElementById("myIconShow");
    const myIconCollapse = shadow.getElementById("myIconCollapse");
    myIconShow.style.display=this.toggleP?'none':'inline-block';
    myIconCollapse.style.display=this.toggleP?'inline-block':'none';
  });
}

}

customElements.define("toggle-panel-component", TogglePanelComponent)

```

Utilisation :

<pre> &lt;toggle-panel-component label="p1"&gt;   &lt;div&gt;p1 content&lt;/div&gt; &lt;/toggle-panel-component&gt;  &lt;toggle-panel-component label="calcul"&gt;   ... &lt;/toggle-panel-component&gt; </pre>		
---	--	---