

# Web components

## Integrantes:

- Francisco Olivares
- Gabriel Valenzuela

# Item de retail (Retail Item)

Así se ve utilizado dentro del `index.html`

```
<retail-item  
url="https://www.ing.uc.cl/"  
product_name='Televisor 44'"  
old_price="699000"  
new_price="499000"  
image_url="https://totalmarcas.com/2885-home_default/televisor-samsung-43-pulgadas-full-hd-flat-smart-tv-j5290a-series-5.jpg" />
```

## Cosas más relevantes

- Contrucción y manejo de estilo
- Lectura de atributos
- Manejo de eventos

# Web Component: Esquema general

```
export default class MyElement extends HTMLElement {
  constructor() {
    // Constructor padre
    super()
    // Referencia al shadow DOM
    this.root = this.attachShadow({ mode: 'open' })
    // Binding de funcion auxiliar
    this.myCustomFunc = this.myCustomFunc.bind(this)
    ...
  }
  // Se ejecuta después de ejecutar el constructor
  connectedCallback() {...}
  // Función que se hara cargo de ejecutar acciones cuando se clikee
  myCustomFunc() {...}
  // Getter y Setters para referirse a this.myAttribute_1
  get myAttribute_1() {
    return this.getAttribute('myAttribute_1')
  }
  set myAttribute_1(newValue) {
    return this.setAttribute('myAttribute_1', newValue)
  }
  // Función que se ejecuta cuando el elemento es removido del DOM
  disconnectedCallback() {...}
}
window.customElements.define('element-tag', MyElement)
```

# Retail Item: Contrucción y manejo de estilos

```
export default class RetailItem extends HTMLElement {  
  constructor() {  
    super()  
    this.root = this.attachShadow({ mode: 'open' })  
    this.handleClick = this.handleClick.bind(this)  
  }  
  ...  
}
```

Se inicializan los atributos en caso de que no se hayan ingresado al momento de instanciar el `retail-item`

```
connectedCallback() {
  let percentage = true
  // Se inicializan los parámetros
  if (!this.hasAttribute('url')) {
    this.setAttribute('url', 'www.google.com')
  }
  if (!this.hasAttribute('product_name')) {
    this.setAttribute('product_name', 'attr product_name is missing')
  }
  if (!this.hasAttribute('old_price')) {
    this.setAttribute('old_price', 'attr old_price is missing')
    percentage = false
  }
  if (!this.hasAttribute('new_price')) {
    this.setAttribute('new_price', 'attr new_price is missing')
    percentage = false
  }
  let sale_percentage = 0
  // Se puede calcular el porcentaje de descuento
  if (percentage) {
    sale_percentage = Math.round((this.old_price - this.new_price) / (this.old_price) * 100)
  }
  ...
}
```

Antes de continuar veamos el template. ¿Cómo se le da estilo al template, si es que este va ser cargado dentro del shadowDOM?

```
<template id="retail-item">
  <!-- ACA SE CARGAN LOS ESTILOS!!! -->
  <link rel="stylesheet" href="app.css">
  <div id="container" class="columns is-inline-flex is-multiline is-centered is-gapless">
    <div class="box" style="position: relative;">
      <div class="column is-12" style="padding-bottom: 0px;">
        <div style="display: flex; justify-content: center;">
          
        </div>
      </div>
      <div class="column is-12" style="padding-bottom: 0px;">
        <span id="product-name"> </span>
      </div>
      <div class="column is-12" style="font-size: 1.05rem; padding-bottom: 0px;">
        $ <span id="new-price"></span> (Oferta)
      </div>
      <div class="column is-12" style="padding-bottom: 0px;">
        $ <span id="old-price"></span>
      </div>
      <div class="sale-container">
        <span id="sale-percentage"> </span>
      </div>
    </div>
  </div>
</template>
```

## Continuando...

```
connectedCallback() {  
  ...  
  // Se obtiene el template y se rellena según los atributos  
  const { shadowRoot } = this  
  const template = document.getElementById('retail-item')  
  const node = document.importNode(template.content, true)  
  
  this.mainContainer = node.getElementById("container")  
  this.mainContainer.addEventListener('click', this.handleClick)  
  
  node.getElementById("product-name").innerHTML = this.product_name  
  node.getElementById("old-price").innerHTML = this.old_price  
  node.getElementById("new-price").innerHTML = this.new_price  
  if (percentage) {  
    node.getElementById("sale-percentage").innerHTML = String(sale_percentage) + "% DCTO"  
  } else {  
    node.getElementById("sale-percentage").innerHTML = "X% DCTO"  
  }  
  
  if (this.hasAttribute('image_url')) {  
    node.getElementById("product-image").src = this.image_url  
  }  
  
  shadowRoot.appendChild(node)  
}
```

# Lectura de atributos

```
get url() {  
    return this.getAttribute('url')  
}  
get product_name() {  
    return this.getAttribute('product_name')  
}  
get old_price() {  
    return this.getAttribute('old_price')  
}  
get new_price() {  
    return this.getAttribute('new_price')  
}  
get image_url() {  
    return this.getAttribute('image_url')  
}
```

/\* Después para acceder a alguno de los atributos se llama de la forma  
this.atributo ej: let url = this.url\*/



# Manejo de Eventos

```
constructor() {  
    /* más código arriba... */  
    this.handleClick = this.handleClick.bind(this)  
}  
  
connectedCallback(){  
    /* más código arriba... */  
    this.mainContainer = node.getElementById("container")  
    this.mainContainer.addEventListener('click', this.handleClick)  
    /* más código abajo... */  
}  
  
handleClick() {  
    window.open(this.url, '_blank')  
}
```

# Barra de rating (Stars Bar)

Uso en html

```
<stars-bar max_value="6" selected_value="2" />
```

## Cosas más relevantes

- Generación dinámica de estrellas
- Manejo del click

# Generación dinámica de estrellas

Se generaron funciones para crear los 2 tipos de estrella

```
getStar(number_of_star) {  
    let img = document.createElement('img')  
    img.src = "/static/star.png"  
    img.className = "star"  
  
    return img  
}  
getInactiveStar() {  
    return ``  
}
```

En la función `connectedCallback` según el valor de los atributos `max_value` y `selected_value` se añaden las estrellas

```
const { shadowRoot } = this
const template = document.getElementById('stars')
const node = document.importNode(template.content, true) // Se copia el contenido del template

// Se procede a rellenar las estrellas
const container = node.getElementById('container')

let i = 1

while (this.selected_value >= i) {
  container.appendChild(this.getStar())
  i += 1
}

while (this.max_value >= i) {
  container.innerHTML += this.getInactiveStar()
  i += 1
}

container.childNodes.forEach((x, index) => {
  x.addEventListener('click', this.handleClick(index + 1))
})

shadowRoot.appendChild(node)
```

# Manejo del evento click

Se generaron 2 funciones auxiliares

```
// Función que ve el estado de rating y en base a eso asigna las clases css
updateRating() {
  const { shadowRoot } = this
  let container = shadowRoot.getElementById('container')
  container.childNodes.forEach((x, index) => {
    if (index + 1 <= this.status['rating']) {
      x.className = "star"
    }
    else {
      x.className = "star is-inactive"
    }
  })
}

// Función que se ejecuta cuando clickean las estrellas
// x viene a ser el número de la estrella
handleClick(x) {
  return () => {
    this.status['rating'] = x
    this.updateRating()
  }
}
```