View encapsulation

Folosit si util pentru a nu supra-inscrie style-urile intre componente.

```
@Component({
  selector: 'list-view',
  template: `
    <h1>List view</h1>
    <tile *ngFor="let person of persons" [person]="person"></tile>
  `,
  styles: [`
    h1 { color: white }   // this style will be applied
    tile {
      h2 { color: white } // this styles won't be applied
    }
  `]
})
export class ListViewComponent {
}
```

Aici, de exemplu, nu ar trebui sa ai voie sa accesezi style-urile pentru componenta copil, care este ‘tile’ – nu putem controla ce sa facem cu h2 din interiorul lui tile, asta o vom face din interiorul componentei tile.

Poti specifica mai multe tipuri de encapsulari care sa se aplice pentru o componenta.

```
@Component({
  selector: 'app-no-encapsulation',
  template: `
    <h2>None</h2>
    <div class="none-message">No encapsulation</div>
  `,
  styles: ['h2, .none-message { color: red; }'],
  encapsulation: ViewEncapsulation.None,
})
export class NoEncapsulationComponent { }
```

- Shadow (browser built in api, nu toate broawserele supporta)
- Emulated (default)
- None (haos)

Angular aplica attribute generate pentru a identifica unic un element din DOM.S

```
<app-root _nghost-c0="">

    <h2 _ngcontent-c0="">Component Style Isolation example</h2>

    <button _ngcontent-c0="" class="red-button">Button</button>

</app-root>
```

Apoi toate style-urile scrise de tine vor fi inserate in styles.css (global), unde style-urile se vor aplica individual in functie de atributele tag-urilor.

```
h2[_ngcontent-c0]{
  color: white;
}
```

:host

O componenta este considerata un host/gazda al tuturor tag-urilor/componentelor care se afla in interiorul componentei host.

In exemplul de mai jos :

App-root este hostul/gazda lui h2/button/bluebutton , asta inseamna ca prin css poti modifica style-urile acestor tag-uri, din cauza view-encapsulation nu vei putea modifica h2 al lui blue-button

Blue-button este o componenta, este content pentru parintele app-root, iar acesta la randul lui reprezinta un host pentu h2 si button.

```
<app-root _nghost-c0="">

    <h2 _ngcontent-c0="">Component Style Isolation example</h2>

    <button _ngcontent-c0="" class="red-button">Button</button>

    <blue-button _nghost-c1="" _ngcontent-c0="">

        <h2 _ngcontent-c1="">Blue button component</h2>

        <button _ngcontent-c1="" class="blue-button">Button</button>

  </blue-button>

</app-root>
```

Bad practice nefolosind selectorul :host

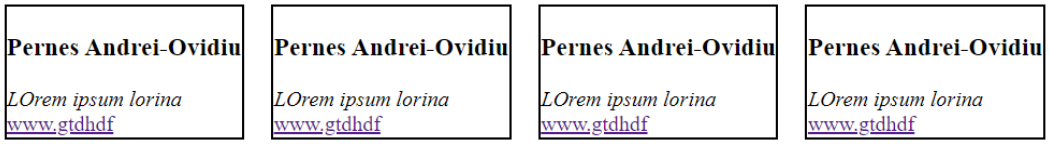
Avem o componenta denumita 'list-item' care are urmatorul cod html :

```
<div style="border: 2px solid black;display: inline-block;" >
  <h3>Pernes Andrei-Ovidiu</h3>
  <em>LOrem ipsum lorina</e
  <a href="#" class="website">www.gtdhdf</a>
</div>
```

, iar parintele o foloseste asftel :

```
<app-list-item style="margin:10px"></app-list-item>
<app-list-item style="margin:10px"></app-list-item>
<app-list-item style="margin:10px"></app-list-item>
<app-list-item style="margin:10px"></app-list-item>
```

Rezultatul este:



Nu ne place deoarece trebuie sa folosim un div in plus in interiorul componentei degeaba, si apoi marginile sa le aplicam folosind css-ul din parinte.

Folosind selectorul :host

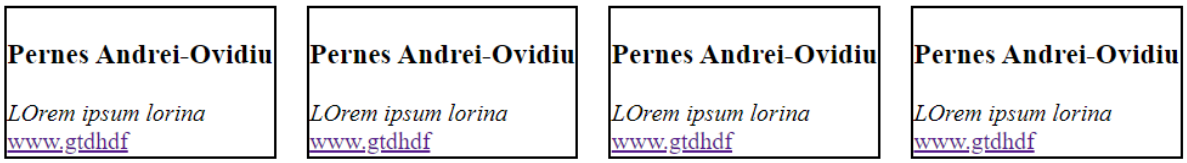
Style-ul pentru margine,display si border im vom pune in list-item.css folosind :host si vom sterge div-ul din interiorul template-ului.la fel si marginile applicate din parinte

```
:host{
  margin-left: 20px;
  display: inline-block;
  border: 2px solid black;
}
```

```
<h3>Pernes Andrei-Ovidiu</h3>
<em>LOrem ipsum lorina</em>
<a href="#" class="website">www.gtdhdf</a>
```

```
<app-list-item ></app-list-item>
<app-list-item ></app-list-item>
<app-list-item ></app-list-item>
<app-list-item ></app-list-item>
```

Rezultatul fiind acelasi :



Selectorul host poate fi folosit si impreuna cu alt selector :

```
:host(.active)
 {
 font-weight: bold;
 background-color      : red;
 }
```

Textul din interiorul gazdei va fi bold si background rosu doar daca host-ul are clasa .active

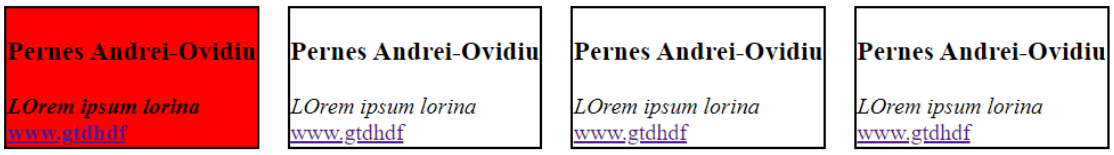
Cod in interioul list-item.css

```
  :host{
    margin-left: 20px;
    display: inline-block;
    border: 2px solid black;
  }
  :host(.active){
    font-weight: bolder;
    background-color: red;
  }
```

Parinte :

```
<app-list-item class="active"></app-list-item>
<app-list-item ></app-list-item>
<app-list-item ></app-list-item>
<app-list-item ></app-list-item>
```

Rezultat:



:host-context

Acest selector il folosim atunci cand vrem sa aplicam anumit style host-ului in functie daca parintele(elementul care continue componenta pe care dorim noi sa l aplicam styles) are o anumita clasa.

Este folosit doar impreuna cu alte selectoare

Exemplu :

Avem componenta 'themeable-button'

```
@Component({
  selector: 'themeable-button',
  template: `
        <button class="btn btn-theme">Themeable Button</button>
  `,
  styles: [`
      :host-context(.red-theme) .btn-theme {
        background: red;
      }
      :host-context(.blue-theme) .btn-theme {
          background: blue;
      }
  `]
})
export class ThemeableButtonComponent {

}
```

Acest component este reutilizabil si ar trebui sa arate altfel, in functie de locul unde este folosit.

```
<div class="blue-theme">
    <themeable-button></themeable-button>
</div>
```

Daca dorim ca butonul sa fie albastru, vom aplica clasa blue-theme parintelui

Daca dorim sa adaugam o noua culoare va trebui sa cream style-ul necesar in interiorul componentei themable-button

Documentatie :

Angular :host, :host-context, ::ng-deep - The Complete Guide (angular-university.io)

Techniques to style component host element in Angular - Angular inDepth

Angular - Component styles