

CORRIGE - exercice 2 - HTML_CSS_TypeScript

Architecture

Se reporter au fichier « **Application HTML_CSS_TS préparation et exécution.pdf** » - **étape 1** pour les explications détaillées.

Sur votre « devhom », emplacement où se trouve vos applications web sur le serveur « devweb » de l'IUT de Metz, dans le dossier crée ou à créer IHM,

- a) créer le dossier TP2
- b) copier le contenu du dossier « squelette » (fichier compressé « squelette » à télécharger sur Arche), dans le dossier TP2
 - le fichier « tsconfig.json » et les dossiers « vue », « src/controleur » et « src/modele » apparaissent
 - le TP2 n'utilise pas de données extraites d'une base de données → supprimer le dossier « modele »

Travail à faire

Ecrire le fichier « **tp2.html** » dans le **dossier « vue »** et le fichier **tp2.css** associé dans le **sous-dossier « css »** du dossier « vue » à partir de la maquette ci-contre.

Maquette d'une interface web pour 'Mise en boîte de petits textes'. Elle comprend un titre, un champ de saisie avec le placeholder 'Taper votre texte puis Entrer', un bouton 'Supprimer', un bouton 'Vider', et une case à cocher 'liste triée'.

Conception HTML/CSS

Se reporter au fichier « **Application HTML_CSS_TS préparation et exécution.pdf** » - **étape 2**

Créer le fichier « vue/tp1.html »

- 1) Définir la structure globale du fichier « tp2.html » avec les parties
 - <head> avec le lien vers le fichier « tp2.css »
 - <body>
 - <script> avec l'appel du fichier « tp2.js »

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title> Mise en boîte de petits textes </title>
    <link type="text/css" rel="stylesheet" href="./css/tp2.css">
  </head>
  <body>
  </body>
  <script type="module" src="../controleur/tp2.js">
  </script>
</html>
```

CORRIGE - exercice 2 - HTML_CSS_TypeScript

2) Définir une zone identifiée « div_tp2 » dans <body>

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title> Mise en boîte de petits textes </title>
    <link type="text/css" rel="stylesheet" href= "./css/tp2.css">
  </head>
  <body>
    <div id='div_tp2'>
    </div>
  </body>
  <script type="module" src="../../controleur/tp2.js">
  </script>
</html>
```


3) Est-ce utile que la zone occupe entièrement l'espace écran ? **NON**

4) Définir le style associé à la zone ?

```
tp2.css
#div_tp2 {
  width      : 600px;
  height     : 400px;
  border     : 1px solid #808080;
  font-size  : 1.5rem;
}
```

CORRIGE - exercice 2 - HTML_CSS_TypeScript

- 5) Déterminer le nombre de zones horizontales y compris le titre et définir les zones sans définir leur contenu.



```
<div id='div_tp2'>
  <header>
  </header>
  <div>
    <label for="edt_saisie">Taper votre texte puis Entrer</label>
    <input type='text' id="edt_saisie"
      placeholder="entrer le texte ici" size='40'>
  </div>
  <div class="gestion_liste">
  </div>
  <div>
    <input id="chk_tri" name="chk_tri" type="checkbox" value="tri">
    <label for="chk_tri">liste triée</label>
  </div>
</div>
```

tp2.css

```
input {
  border      : 1px solid #5b5b5b;
  color       : #000000;
  font        : inherit;
  text-align  : left;
  margin      : 0.25rem;
}

.gestion_liste {
  display      : grid;
  grid-template-columns : 9fr 1fr;
  grid-gap     : 1rem;
}
```

CORRIGE - exercice 2 - HTML_CSS_TypeScript

- 6) Déterminer pour chaque zone,
- si un découpage en colonnes est nécessaire ?
 - définir le contenu et ajouter les styles nécessaires

6.1) *header* : **NON**

Mise en boîte de petits textes

```
<header>
Mise en boîte de petits textes
</header>
```

tp2.css

```
header {
  background-color : #808080;
  color            : #ffffff;
```

6.2) *gestion_liste* : **OUI (2 colonnes inégales)**



```
<div class='gestion_liste' >
  <div>
    <select id="select_texte" class="liste" size="10">
    </select>
  </div>
  <div>
    <input id="btn_supprimer" type="button" value="Supprimer">
    <input id="btn_vider" type="button" value="Vider">
  </div>
</div>
```

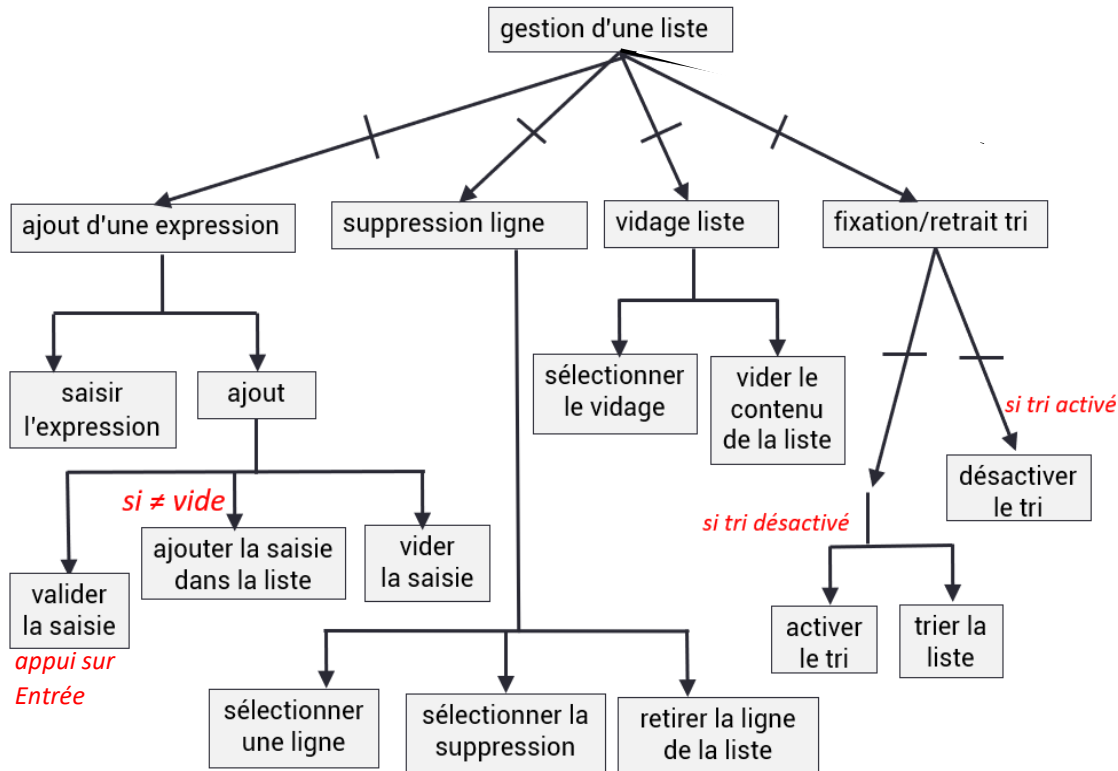
tp2.css

```
.liste {
  width      : 100%;
}
input[type=button] {
  cursor      : pointer;
  border      : none;
  background-color : #00b0ff;
  color       : #000000;
  height      : 2rem;
  vertical-align : middle;
  min-width   : 8rem;
  text-align   : center;
}
input[type=button]:hover {
  background-color : #0000ff;
  border          : 1px solid #ffffff;
  color           : #ffffff;
```

CORRIGE - exercice 2 - HTML_CSS_TypeScript

Travail à faire

Ecrire les fichiers « **class_tp2.ts** » et « **tp2.ts** » dans le dossier « **src/contrôleur** » à partir du modèle des tâches



Programmation TypeScript

Se reporter au fichier « **Application HTML_CSS_TS préparation et exécution.pdf** » - **étape 3**

Créer le fichier « src/contrôleur/class_tp2.ts »

1) Définir le type « TTp2Form »

```
type TTp2Form = {
    edtSaisie : HTMLInputElement
    , listeTexte : HTMLSelectElement
    , chkTri : HTMLInputElement
    , btnVider : HTMLInputElement
    , btnSupprimer: HTMLInputElement
}
```

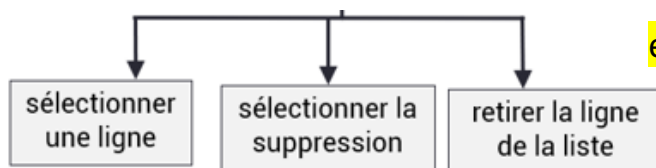
CORRIGE - exercice 2 - HTML_CSS_TypeScript

- 2) Créer la classe « VueTp2 » et définir l'attribut privé « _form », la fonction « init », le getter associé à l'attribut privé

```
class VueTp2 {  
    private _form : TTp2Form  
  
    init(form : TTp2Form) : void {  
        this._form = form  
    }  
  
    get form() : TTp2Form { return this._form }  
}
```

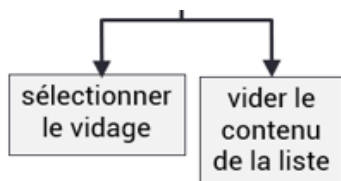
- 3) Quels sont les événements/actions déclenchés par l'utilisateur ?

A partir du modèle des tâches



événement « clic » du bouton « **supprimer** »

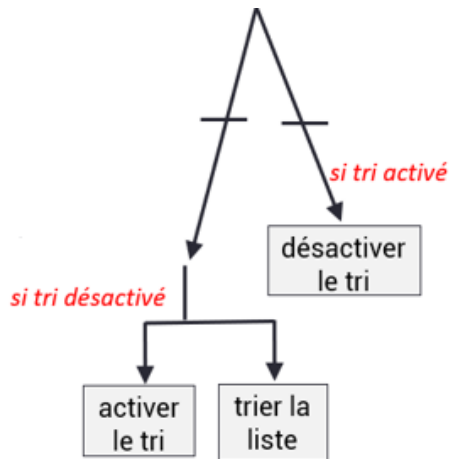
→ appel de la méthode **supprimerLigne()**



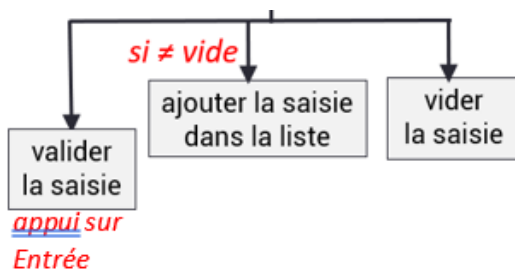
événement « clic » du bouton « **vider** »

→ appel de la méthode **viderListe()**

CORRIGE - exercice 2 - HTML_CSS_TypeScript



événement « **change** » de la case à cocher « **tri** »
 ➔ appel de la méthode **trierListe()**



- événement « **keypress** » de la zone de saisie
 - appel de la méthode **ajouterSaisie(ev:KeyboardEvent)**

CORRIGE - exercice 2 - HTML_CSS_TypeScript

- 4) Définir les méthodes de la classe `viderListe()`, `supprimerLigne()`, `trierListe()`, `ajouterSaisie()`

```
class VueTp2 {  
  
    • • •  
    viderListe() :void {  
        // à compléter  
    }  
    supprimerLigne() :void {  
        // à compléter  
    }  
  
    trierListe(liste :HTMLSelectElement):void {  
        const options : HTMLOptionsCollection = liste.options;  
        let optionsArray : HTMLOptionElement[] = [];  
        for (let i = 0; i < options.length; i++) {  
            optionsArray.push(options[i]);  
        }  
        optionsArray = optionsArray.sort(function  
            (a:HTMLOptionElement, b:HTMLOptionElement):number {  
                if (a.value > b.value) { return 1 } else { return -1 }  
            });  
        for (let i = 0; i <= options.length; i++) {  
            options[i] = optionsArray[i];  
        }  
    }  
  
    trierListe() :void {  
        // à compléter  
    }  
    ajouterSaisie(ev:KeyboardEvent) :void {  
        // à compléter  
    }  
}  
  
let vueTp2 = new VueTp2;  
  
export { vueTp2 }
```

5) Créer l'objet « vueTp2 », instanciation de la classe

6) Écrire la commande d'exportation de l'objet

CORRIGE - exercice 2 - HTML_CSS_TypeScript

7) Écrire le corps des méthodes de la classe

```
viderListe():void {
    this.form.listeTexte.length = 0;
}

supprimerLigne():void {
    const liste = this.form.listeTexte;
    const noLigne : number = liste.selectedIndex;
    if (noLigne > -1) {
        liste.remove(noLigne);
    }
}

trierListe() : void {
    if (this.form.chkTri.checked) {
        this.triListe(this.form.listeTexte)
    }
}

ajouterSaisie(ev:KeyboardEvent):void {
    if (ev.key === 'Enter') {
        const elt = this.form.edtSaisie;
        const liste = this.form.listeTexte;
        const chaine : string = elt.value.trim();
        if (chaine !== "") {
            const opt = new Option(chaine, chaine);
            liste.options.add(opt);
            this.trierListe();
        }
        elt.value = "";
        elt.focus();
    }
}
```

CORRIGE - exercice 2 - HTML_CSS_TypeScript

8) Ajouter la définition des événements dans la fonction « init »

```
init(form : TTp2Form) : void {  
    ...  
  
    // définition des événements  
    this.form.edtSaisie.onkeydown  
        = function (event):void { vueTp2.ajouterSaisie(event); }  
    this.form.chkTri.onChange  
        = function ():void      { vueTp2.trierListe(); }  
    this.form.btnVider.onclick  
        = function ():void      { vueTp2.viderListe(); }  
    this.form.btnSupprimer.onclick  
        = function ():void      { vueTp2.supprimerLigne(); }  
}
```

Créer le fichier « src/controleur/tp2.ts »

1) Écrire la commande d'importation de l'objet créé précédemment

```
import {vueTp2} from "../controleur/class_tp2"
```

2) Appeler la méthode « init » avec les paramètres

```
vueTp2.init (  
{  
    edtSaisie : document.querySelector('[id=edt_saisie]')  
    , listeTexte : document.querySelector('[id=select_texte]')  
    , chkTri : document.querySelector('[id=chk_tri]')  
    , btnVider : document.querySelector('[id=btn_vider]')  
    , btnSupprimer: document.querySelector('[id=btn_supprimer]')  
}) );
```

Transpiler vos fichiers « ts » et tester l'application

Se reporter au fichier « Application HTML_CSS_TS préparation et exécution.pdf » - **étape 4**