Afin d'améliorer l'ergonomie utilisateur et de prendre en compte les différents types d'interfaces (tactile, souris, clavier), il est important d'utiliser tous les moyens possibles, notamment, les nombreux éléments d'interface offerts par enyoJS. Tous les éléments présentés ci-après sont modifiables avec du CSS, il suffit de rajouter le paramètre `style:"color:red;etc….",` dans la zone d'installation à la suite des paramètres existants.

Consulter le EnyoJS Sampler pour obtenir une version plus détaillée de ce document à l'adresse suivante : http://enyojs.com/sampler/2.2.0/

(attention, nous utilisons la version 2.2)

Content : permet d'afficher du texte à l'écran

Zone d'installation :

```
{name:"NomdelaBoite", content:"affichage de l'écran",style:"color:red;"},
```

Zone des fonctions :

```
UneFontion: function(inSender, inEvent) {

this.$.NomdelaBoite.setContent("Des lasagnes !!!!!");

this.$.NomdelaBoite.setStyle("color: blue;");

},
```

Buttons : cf activité introductive

Inputs : les champs de texte

Zone d'installation :

```
{kind: "onyx.InputDecorator", components: [

{name:"NOMDUCHAMP", kind: "onyx.Input", placeholder: "Enter text here",
onchange:"inputChanged"},

]},
```

Zone des fonctions :

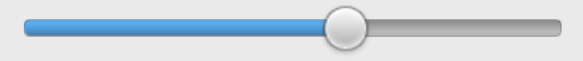
```
inputChanged: function(inSender, inEvent) {

      this.$.result.setContent("Input: " + inSender.getValue());

},
```

Ou alors, on peut accéder au contenu du champ de texte via une fonction directement :

```
NIMPORTEQUELLEFONCTION: function(inSender, inEvent) {

Texte = this.$.NOMDUCHAMP.getValue();

this.$.NOMDUCHAMP.setValue("hello");

},
```

Les sliders

Les sliders : permettent une saisie rapide d'une donnée chiffrée par l'utilisateur et s'avère très pratique sur un écran tactile.



Propriétés :

- lockbar : permet d'afficher une barre bleue en arrière
- value : de 0 à 100, indique la position initiale
- ShowStripes : affiche des barres alternées de couleur

Evénements :

- onChanging : renvoit dans une fonction nommée « sliderChanging », tant que l'utilisateur a le doigt sur le slider la valeur de « value » de 0 à 100
- OnChange : renvoit dans une fonction nommée « sliderChanging », quand l'utilisateur relâche le doigt du slider la valeur de « value » de 0 à 100

Zone d'installation :

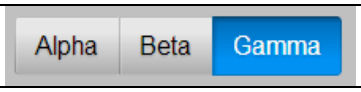
```
{kind: "onyx.Slider", lockBar: false, value: 50, showStripes:true,
onChanging:"sliderChanging", onChange:"sliderChanged"},
```

Zone des fonctions : *récupérer la valeur du slider*

```
sliderChanging: function(inSender, inEvent) {
Celcius = inSender.getValue();
console.log(Celcius);
},
```

groupedButtons

Les *groupedButtons* permettent à l'utilisateur de choisir un élément parmi plusieurs.



Zone d'installation :

```
{kind: "onyx.RadioGroup", onActivate:"radioActivated", components: [
      {content: "Alpha", active: true},
      {content: "Beta"},
      {content: "Gamma"}
]},
```

Zone des fonctions :

```
radioActivated: function(inSender, inEvent) {
if (inEvent.originator.getActive()) {
console.log(inEvent.originator.getContent() + " selected.");
}
},
```

IconButtons

Les *IconButtons* permettent d'utiliser une icône pour faire un bouton. Pour simuler l'appui du bouton, l'icône utilisée dispose d'une version appuyée et d'une version relâchée du bouton. Par convention, les icônes sont stockées dans le répertoire assets


Icone utilisée

Zone d'installation : *(évidemment, il faut mettre l'icône dans le répertoire indiqué, sinon cela marche moins bien…)*

```
{kind: "onyx.IconButton", src: "assets/menu-icon-bookmark.png",
ontap:"iconTapped"},
```

Zone des fonctions :

```
iconTapped: function(inSender, inEvent) {
      console.log("icon tapped");
},
```

toggleButtons

Les *toggleButtons* sont très similaire aux *groupedButtons* mais permettent uniquement de faire des choix binaires.

ou

Zone d'installation: Notez que la paramètre `value:true` permet de fixer l'état du bouton au départ.

```
{kind:"onyx.ToggleButton", onChange:"toggleChanged", value: true,
onContent:"ON", offContent:"OFF"},
```

Zone des fonctions:

```
toggleChanged: function(inSender, inEvent) {

     console.log(inSender.getValue());

},
```

checkBoxes

Les *checkBoxes* sont très similaires aux *toggleButtons* et permettent de faire des choix binaires mais avec une interface graphique différente.

Zone d'installation: Notez que la paramètre `value:true` permet de fixer l'état au départ

```
{kind:"onyx.Checkbox", onchange:"checkboxChanged", checked: true},
```

Zone des fonctions:

```
checkboxChanged: function(inSender, inEvent) {

     console.log(inSender.getValue());

},
```

checkBoxes : highlander, il ne peut y en avoir qu'un !

Les *CheckBoxes* peuvent être utilisée avec le paramètre « highlander » et permet de gérer automatiquement des ensembles de *checkboxes* afin qu'une seule ne puisse être sélectionnée.

 ou  ou 

Zone d'installation: Notez que la paramètre `value:true` permet de fixer l'état au départ

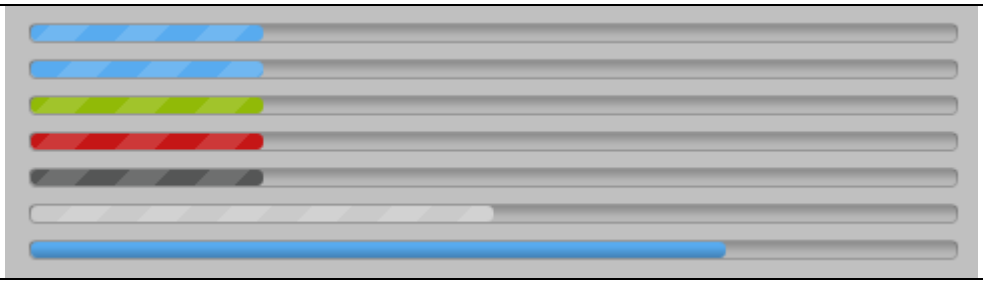
```
{kind: "Group", classes: "onyx-sample-tools group",
onActivate:"groupActivated", highlander: true, components: [

     {kind:"onyx.Checkbox", checked: true,
onchange:"checkboxChanged1",},

     {kind:"onyx.Checkbox", onchange:"checkboxChanged2"},

     {kind:"onyx.Checkbox", onchange:"checkboxChanged3"}

]},
```

Zone des fonctions:

```
checkboxChanged1: function(inSender, inEvent) {

     console.log("checkBox 1 : " + inSender.getValue());

},

checkboxChanged2: function(inSender, inEvent) {

     console.log("checkBox 2 : " + inSender.getValue());

},

checkboxChanged3: function(inSender, inEvent) {

     console.log("checkBox 3 : " + inSender.getValue());

},
```

progressBar

Les *progressBar* permettent d'indiquer visuellement une valeur, ou l'avancement d'une tâche.



Zone d'installation: Voici différents exemples

```
{kind: "onyx.ProgressBar", name:"progression", progress: 25},

{kind: "onyx.ProgressBar", animateStripes: false, progress: 25},

{kind: "onyx.ProgressBar", progress: 25, barClasses: "onyx-green"},

{kind: "onyx.ProgressBar", progress: 25, barClasses: "onyx-red"},

{kind: "onyx.ProgressBar", progress: 25, barClasses: "onyx-dark"},

{kind: "onyx.ProgressBar", animateStripes: false, barClasses: "onyx-
light", progress: 50},

{kind: "onyx.ProgressBar", showStripes: false, progress: 75},
```

Zone des fonctions: Pour affecter une valeur (comprise entre 0 et 100) à une *ProgressBar*, il suffit d'utiliser un bouton qui, dans sa fonction utilise l'appel ci-dessous. Vous remarquerez qu'il faut nommer avec le paramètre `name` la *progressBar* pour modifier sa valeur.

```
ButtonPressed: function(inSender, inEvent) {

      this.$.progression.animateProgressTo(74);

},
```

Drawer

Les *Drawers* permettent de cacher ou faire apparaitre une partie de l'interface utilisateur afin d'afficher le strict nécessaire pour l'utilisateur.

Zone d'installation:

```
{name: "voiciunspoiler", kind: "onyx.Drawer", animated: true,
open:false, components: [

      {content: "Voici le fin du film : Les deux personnages, ..."}

]},
```

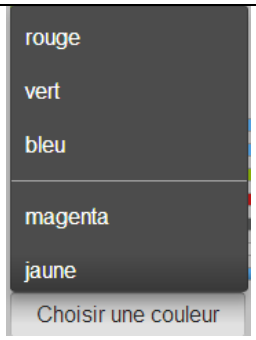
Zone des fonctions: Pour ouvrir ou fermer un *Drawer*, il suffit d'utiliser un bouton qui, dans sa

fonction utilise l'appel ci-dessous. Vous remarquerez qu'il faut nommer avec le paramètre `name` au *Drawer* pour modifier son statut.

```
// notez que ButtonPressed est une fonction appelée par un bouton,
il faut donc mettre en place un bouton (ou autre) avant de
commencer.

ButtonPressed: function(inSender, inEvent) {

      this.$.voiciunspoiler.setOpen(!this.$.voiciunspoiler.open);

      // ou alors -------

      this.$.voiciunspoiler.setOpen(true);

      // ou alors -------

      this.$.voiciunspoiler.setOpen(false);

},
```

Les menus

Les *Menus* permettent à l'utilisateur de choisir parmi une liste de chaine de caractère



Zone d'installation: Deux imbrications ici, similaire aux *Inputs*. Une fonction `itemSelected` est appelée à chaque fois qu'un élément est sélectionné.

```
{kind: "onyx.MenuDecorator", onSelect: "itemSelected", components: [

      {content: "Choisir une couleur"},

      {kind: "onyx.Menu", floating: true, components: [

            {content: "rouge"},

            {content: "vert"},

            {content: "bleu"},

            {classes: "onyx-menu-divider"},
```

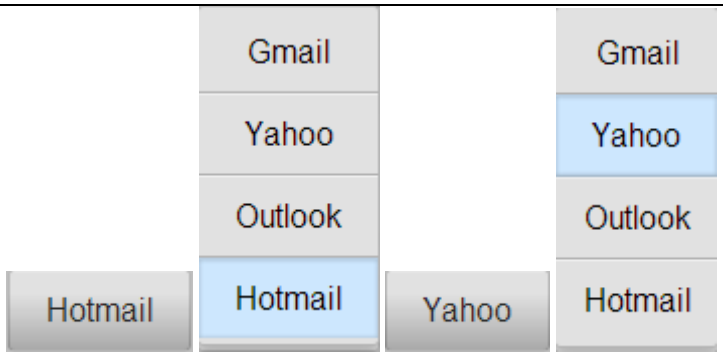
```
            {content: "magenta"},
            {content: "jaune"},
            {content: "cyan"},
      ]}
]},
```

Zone des fonctions: Voici la fonction permettant de savoir quel item l'utilisateur a choisi.

```
itemSelected: function(inSender, inEvent) {
            console.log(inEvent.content);
},
```

Les Pickers

Les *Pickers* permettent à l'utilisateur de choisir rapidement sur une interface tactile un nombre, ou une chaine de caractère. Légèrement diffèrent des menus, car le choix sélectionné est visible sur le bouton.



Zone d'installation: Le paramètre `active` permet de déterminer l'élément sélectionné par défaut sur l'interface.

```
{kind: "onyx.PickerDecorator", onSelect: "PickerSelected",
components: [
      {},
      {kind: "onyx.Picker", components: [
            {content: "Gmail"},
            {content: "Yahoo"},
            {content: "Outlook"},
```

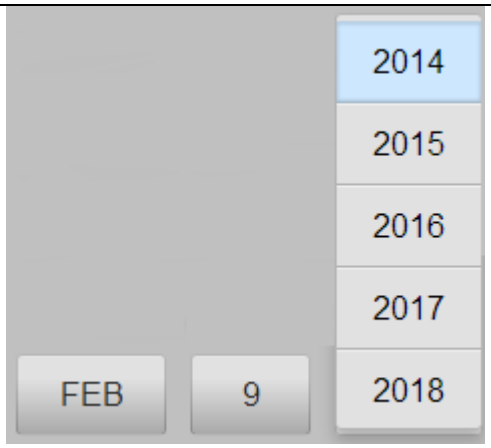
```
            {content: "Hotmail", active: true}

      ]}

]},
```

Zone des fonctions: Voici la fonction permettant de savoir quel item l'utilisateur a choisi.

```
PickerSelected: function(inSender, inEvent) {

            console.log(inEvent.selected.content);

},
```

Les DatePickers

Les *Pickers* permettent à l'utilisateur de choisir rapidement une date



Zone d'installation: Différents paramètres permettent d'afficher ou de cacher à l'utilisateur le choix du jour, du mois ou de l'année.

```
{kind:"onyx.DatePicker", minYear:2010, maxYear:2020, dayHidden:
false, monthHidden: false, yearHidden: false, onSelect:"getDates"},
```

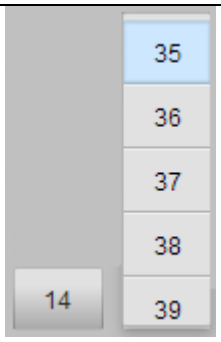
Zone des fonctions: Voici la fonction qui permet d'afficher dans la console la date sélectionnée

```
getDates: function(inSender, inEvent){

      console.log(inEvent.value);

},
```

Astuce : pour sélectionner l'information qui nous intéresse (heure, minute, …), on pourra utiliser la méthode suivante :

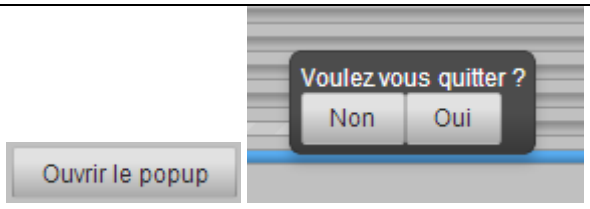
```
a = inEvent.value // on récupère la valeur de l'événement

b = a.substring(a,b) // ce qui permet de tronquer une chaine de
```

`caractère de la position a à la position b.`

Les TimePickers
Les Time*Pickers* permettent à l'utilisateur de choisir rapidement une heure.

Zone d'installation: `{name:"lheure", kind:"onyx.TimePicker", is24HrMode: true, onSelect:"getTime"},`

Zone des fonctions: Voici la fonction qui permet d'afficher dans la console la date sélectionnée
`getTime: function(inSender, inEvent){`
`console.log(inEvent.value);`
`},`

Astuce : pour sélectionné l'information qui nous intéresse (heure, minute, …), on pourra utiliser la fonction `.substring(a,b)` qui permet de tronquer une chaine de caractère de la position a à la position b

Les Popups
Les *Popups* permettent de mettre en avant un choix crucial (sauvegarde de donnée, quitter une application, etc…)

Zone d'installation: `{kind: "onyx.Button", content: "Ouvrir le popup", ontap:`

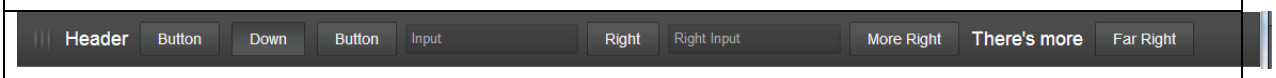
```
"showPopup"},

{name: "questionUtilisateur", classes: "onyx-sample-popup", kind:
"onyx.Popup", centered: true, modal: true, floating: true,
components: [

      {content:"Voulez vous quitter ?"},

      {kind: "onyx.Button", content: "Non", ontap: "nonPopup"},

      {kind: "onyx.Button", content: "Oui", ontap: "ouiPopup"}

]},
```

Zone des fonctions: Trois fonctions à ajouter. La fonction `showPopup` est une fonction appelée par un bouton qui permet l'affichage du popup. Les fonctions `ouiPopup` et `nonPopup` exécutent les actions des deux boutons. La construction d'éléments graphique dans les popup se fait de la même manière que dans la fenêtre prcinipale.

```
nonPopup: function() {

      console.log("non, je ne reste !");

},

ouiPopup: function(inSender) {

       console.log("Au revoir !");

      this.$.questionUtilisateur.hide();

},

showPopup: function(inSender) {

      this.$.questionUtilisateur.show();

},
```

Les Toolbars

Les *Toolbars* permettent de regrouper les fonctionnalités principales d'une application au même endroit pour augmenter la qualité de l'ergonomie



Zone d'installation:

```
{kind: "onyx.Toolbar", components: [
      {kind: "onyx.Grabber"},
      {content: "Header"},
      {kind: "onyx.Button", content: "Button"},
      {kind: "onyx.Button", content: "Down", classes: "active"},
```

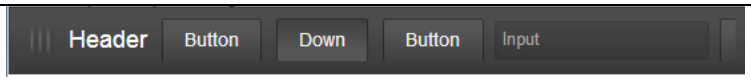
```
        {kind: "onyx.Button", content: "Button"},
        {kind: "onyx.InputDecorator", components: [
        {kind: "onyx.Input", placeholder: "Input"}
        ]},
        {kind: "onyx.Button", content: "Right"},
        {kind: "onyx.InputDecorator", components: [
        {kind: "onyx.Input", placeholder: "Right Input"}
        ]},
        {kind: "onyx.Button", content: "More Right"},
        {content: "There's more"},
        {kind: "onyx.Button", content: "Far Right"}
]},
```

Zone des fonctions: Vous pouvez ajouter des fonctions aux boutons pour gérer les événements utilisateurs.

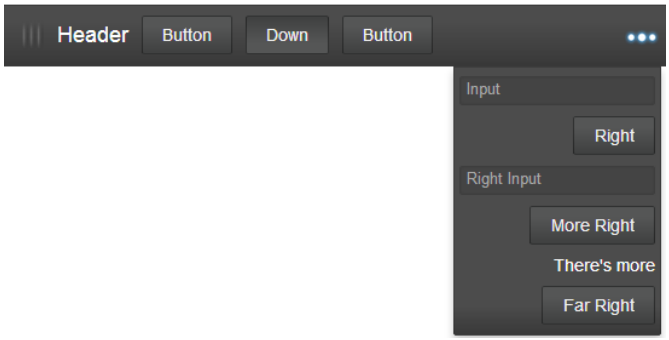
Commentaires : `onyx.Grabber` permet d'afficher une poignée sur la barre d'outils afin de donner la possibilité à l'utilisateur de la déplacer
Les boutons peuvent être affectés d'une classe `classes: "active"` afin d'être inutilisable.

Les Toolbars auto adaptables

Dans le cas de largeur d'écran faible (smartphone), il est possible que la barre d'outils soit trop longue, il existe deux solutions.



Ou alors



Zone d'installation:

```
{kind: "Scroller", classes:"onyx-toolbar", touchOverscroll:false,
touch:true, vertical:"hidden", style:"margin:0px;", thumb:false,
components: [
{classes: "onyx-toolbar-inline", style: "white-space: nowrap;",
components: [
      {kind: "onyx.Grabber"},
      {content: "Header"},
      {kind: "onyx.Button", content: "Button"},
      {kind: "onyx.Button", content: "Down", classes: "active"},
      {kind: "onyx.Button", content: "Button"},
```

```
      {kind: "onyx.InputDecorator", components: [
      {kind: "onyx.Input", placeholder: "Input"}
      ]},
      {kind: "onyx.Button", content: "Right"},
      {kind: "onyx.InputDecorator", components: [
      {kind: "onyx.Input", placeholder: "Right Input"}
      ]},
      {kind: "onyx.Button", content: "More Right"},
      {content: "There's more"},
      {kind: "onyx.Button", content: "Far Right"}
]}
]},
```

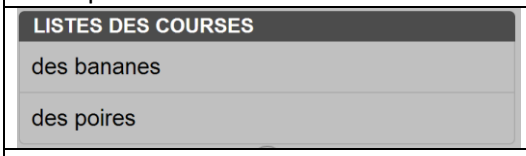
Ou alors

```
{kind: "onyx.MoreToolbar", components: [
      {kind: "onyx.Grabber"},
      {content: "Header"},
      {kind: "onyx.Button", content: "Button"},
      {kind: "onyx.Button", content: "Down", classes: "active"},
      {kind: "onyx.Button", content: "Button"},
      {kind: "onyx.InputDecorator", components: [
      {kind: "onyx.Input", placeholder: "Input"}
      ]},
      {kind: "onyx.Button", content: "Right"},
      {kind: "onyx.InputDecorator", components: [
      {kind: "onyx.Input", placeholder: "Right Input"}
      ]},
      {kind: "onyx.Button", content: "More Right"},
      {content: "There's more"},
      {kind: "onyx.Button", content: "Far Right"}
]},
```

Zone des fonctions: Vous pouvez ajouter des fonctions aux boutons pour gérer les événements utilisateurs.

Groupbox

Les groupbox permettent de mettre en place rapidement une interface verticale, idéale pour un smartphone.



Zone d'installation:

```
{kind: "onyx.Groupbox", components: [
    {kind: "onyx.GroupboxHeader", content: "Listes des courses"},
    {content: "des bananes", style: "padding: 8px;"},
    {content: "des poires", style: "padding: 18px;"}
]},
```

Les Tooltips

Les *Tooltips* permettent de donner des informations au survol d'un bouton

Zone d'installation:

```
{kind: "onyx.TooltipDecorator", components: [

     {kind: "onyx.Button", content: "Bouton"},

     {kind: "onyx.Tooltip", content: "Pour info, il faut appuyer
sur le bouton..."}

]},
```

Les scrollers

Les **scrollers** permettent de pouvoir naviguer verticalement ou horizontalement dans un contenu trop grand pour s'afficher sur l'écran.

Zone d'installation:

```
{kind: "Scroller", classes: "enyo-fit", touch: true, components: [

// ici les éléments d'interface à insérer

 ]},
```