Développement d'applications multiplateforme avec EnyoJS

EnyoJS est un cadre de développement qui permet de créer des applications pour n'importe quelle plateforme : ordinateur, tablette/smartphone Android, IOS, Window Phone, etc….

EnyoJS est basé sur le langage Javascript comme l'indique les deux lettres « JS » accolées au nom. Ainsi, pour pouvoir coder une application avec EnyoJS, vous apprendrez à utiliser les langages web (HTML, Javascript, CSS) que vous pourrez ré-utiliser dans n'importe quel contexte de développement web.

EnyoJS est Open Source et gratuit que ce soit pour créer des applications payantes ou gratuites.

Pour l'ISN, nous utiliserons la version 2.2 d'enyoJS.

Liens utiles :

http://enyojs.com

Le sampler enyoJS (voir rubriques Enyo Core, Layout, **Onyx UI**) : http://enyojs.com/sampler/latest/ (nous n'utiliserons pas Moonstone UI)

PREREQUIS :

Vous aurez besoin des logiciels suivants sur votre clé :

- Notepad++ pour éditer les fichiers
- Chrome portable
- Le « bootplate » enyoJS

BUT :

Créer une application/site web comportant plusieurs pages sur un sujet de votre choix. L'application devra disposer de texte mis en forme, d'images disponibles hors ligne, de boutons pour naviguer.

I. Installation de notre environnement de travail :

Ouvrir Chrome Portable et Notepad++. Dézipper le « bootplate » enyoJS. Vous obtenez l'arborescence suivante :

api assets enyo lib source tools debug.html deploy.md icon.png index.html package.js README.md	Pour lancer l'application, **déplacer debug.html dans le navigateur Chrome**. Un **fond gris indique que tout est ok**… mais l'application est vide pour le moment. Les seuls fichiers que nous utiliserons sont debug.html **et les fichiers contenus dans le répertoire /source**, le reste des autres fichiers permet le fonctionnement d'enyoJS. Nous allons maintenant regarder le contenu du répertoire /source. Vous trouverez un fichier nommé **App.js**, ouvrez le dans Notepad++. C'est ce fichier qui va contenir l'ensemble de la logique de notre première application codée en EnyoJS.

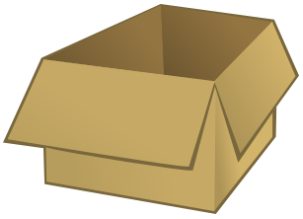
Structure du fichier :

- Zone d'installation de l'interface utilisateur : permet d'installer des éléments d'interface comme des boutons par exemple
- Zone des fonctions permet d'acceuillir les fonctions : ensemble de commandes pour réaliser des tâches, des calculs, et autres actions. Celle est composée de :
    - La fonction create : Cette fonction est appellée automatiquement au démarrage de l'applicatioh et permet d'effectuer des actions avant même que l'utilisateur n'interagisse avec l'application
    - D'autres fonctions dont je donne un exemple MaPremiereFonction qui n'est appellée par aucun élément d'interface pour le moment

```
enyo.kind({

    name: "App",
    fit: true,
    components:[
// Zone d'installation de l'interface utilisateur ---------

    ],
// Zone des fonctions -------------------------------------
create:function() {
    // La fonction create est appelée au démarrage de l'application. La
ligne ci dessous permet de faire passer les propriétés de l'application
    this.inherited(arguments);
    // placer ici les choses faire au démarrage de l'application

},
MaPremiereFonction:function() {
    // voici votre premiere fonction !

},
});
```

Voilà, nous n'aurons pas besoin de plus pour commencer à coder !

II. La structure de base d'enyoJS :



Pour les moins observateurs d'entre vous, voici une boite.

Vous pouvez admirer ci-dessus, une magnifique boite. Disons que cette boite est une boite pour ranger vos chaussures. Celle-ci sera remplie de vos chaussures de sport, de tongues pour la plage et de chaussons pour rester au coin du feu.

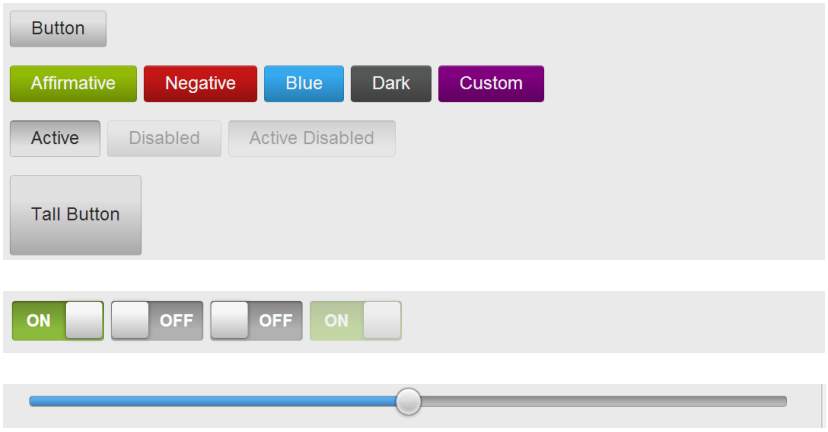
Dans le placard de rangement, on posera cette boite et on écrira dessus « Boite à chaussures » pour la retrouver rapidement parmi toutes les autres boites qui ont elles-mêmes leur propre fonction. A chaque fois qu'on ouvrira le placard, on sera dans l'attente qu'elle nous fournisse une jolie paire de chaussure pour l'activité du moment.

Avec EnyoJS on utilise aussi des boites, mais elle ressemble à cela ….

{ }

Pour les moins observateurs d'entre vous, ceci est toujours une boite.

Sauf qu'évidemment, ces boites pourront prendre les formes suivantes :



Interface utilisateur :

Les boutons, quelle que soit leur forme, permettent de créer une « interface utilisateur » pour établir une communication entre la machine et l'humain.

Pour créer une boite à chaussures, il nous faut du carton, pour créer une boite d'interface utilisateur, il nous faut une ligne de code que voici qui définit le type de boite, son contenu et ce qui va se passer si on l'ouvre.

```
{kind:"onyx.Button", content: "Appuyez ici !", ontap:"AppuiBouton"},
```

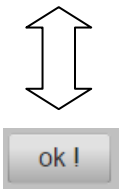
Ligne que nous allons insérer dans la zone d'installation :

```
enyo.kind({
    name: "App",
    fit: true,
    components:[
      {kind:"onyx.Button", content: "ok !", ontap:"MaPremiereFonction"},

    ],
});
```

Kind : Type de la boite : Boite à chaussures, boite à lettres, ….

On veut faire afficher un bouton : « onyx.Button »

Content : indique ce qui va être écrit dans le bouton



Evènement : Quand l'utilisateur appuie sur le bouton : vous appelez une fonction « `MaPremiereFonction` » qui va contenir le code à effectuer.

Si vous faites un tour dans le navigateur avec debug.html, appuyer sur F5 pour recharger, un bouton apparait. Il faut maintenant définir la fonction « `MaPremiereFonction` » qui est appelée lorsque le bouton est appuyé.

```
    MaPremiereFonction: function() {
                                    Zone dans laquelle il faut coder l'événement
                                    « l'utilisateur a appuyé sur le bouton »
    },
```

Cette fonction est placée dans la zone des ……. fonctions en toute logique !

```
enyo.kind({

    name: "App",
    fit: true,
    components:[
      {kind:"onyx.Button", content: "Appuyez ici !",
      ontap:"MaPremiereFonction"},
    ],
    create:function() {

    },
    MaPremiereFonction: function() {

    },
});
```

La console de Chrome : La console est très utilisée pour décrire pas à pas le déroulement logique d'un programme, elle permet de trouver les erreurs de programmations qui empêchent un algorithme de fonctionner correctement. **Accès : Clic droit dans l'application**

A chaque fois que l'utilisateur va appuyer sur le bouton, la fonction « `MaPremiereFonction`» va s'exécuter. Nous allons maintenant faire écrire dans la « console » de Chrome le mot « Hello World ».

```
enyo.kind({
    name: "App",
    fit: true,
    components:[
        {kind:"onyx.Button", content: "ok !", ontap:"MaPremiereFonction"},

    ],
    MaPremiereFonction: function() {
    console.log("Hello World");
    },
});
```

L'utilisateur appuie donc sur le bouton, action dont on peut voir les conséquences dans la console.

IMPORTANT : En cas de problème, la console vous indiquera les lignes comportant des problèmes de syntaxes. Dans l'écran ci-dessus, le numéro de la ligne appelant la console.

La console n'est pas visible de l'utilisateur et ne sert qu'au développeur pour s'assurer du fonctionnement correct de l'application, il faut maintenant afficher du texte à l'écran quand le bouton est appuyé. Pour cela nous allons créer une 2ème boite d'un autre type, une boite qui permet d'afficher du texte :

Pour afficher du texte à l'écran, il faut créer une boite vide en lui donnant un nom dans la zone d'installation

```
{name:"NomdelaBoite",
content:"maboite", style:""},
```

Puis remplir la boite ainsi dénommée, on utilise la ligne suivante dans la fonction appelée par le bouton.

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

```
enyo.kind({
    name: "App",
    fit: true,
    components:[
      {kind:"onyx.Button", content: "ok !", ontap:"MaPremiereFonction"},

      {name:"NomdelaBoite", content:"maboite", style:""},
    ],
    MaPremiereFonction: function() {
      console.log("Hello World");
      this.$.NomdelaBoite.setContent("Des lasagnes !!!!!");
    },
});
```

III. HTML :

Le langage HTML *(Hypertext Markup Langage)* est la base d'une page et est utilisé pour formater un texte brut *(gras, italique, centré, souligné, etc…).*

*Afin d'ajouter du texte formaté HTML dans enyoJS, il faut ajouter le paramètre allow**H**tml :true comme suit dans la zone d'installation*

```
{name:"NomdelaBoite", content:"maboite", style:"", allowHtml :true},
```

1. Formater un texte en HTML :

Nous allons dans ce premier chapitre aborder les bases de ce langage. Tout d'abord, « Markup langage » veut dire « langage à balise ».

Ci-dessous à gauche, voici un texte tel qu'il serait affiché dans une page d'un site web, et à droite, le code HTML permettant de l'afficher.

Principe de fonctionnement : Pour attribuer une modification de style à un mot, on l'entoure par 2 balises, l'une ouvrante, l'autre fermante.

<balise>contenu</balise>

Histoire C'est autour de **-100 avant JC**, qu'apparaissent les premiers moulins à eau, entre la *Grèce* et la *Turquie* avant de gagner **l'Empire Romain**. Sa principale utilisation était pour moudre le grain. Progressivement les usages s'étendent : on trouve quelques siècles plus tard des scies hydrauliques pour couper le marbre en Jordanie. Au XIX^{ème} siècle, on utilise ***l'énergie hydraulique*** notamment dans les mines, les forges et les filatures industrielles. Apparaissent ensuite les turbines modernes auxquelles on fixera des alternateurs vers 1880 : c'est la naissance de l'hydroélectricité.	`<h2>Histoire</h2>` `<p>C'est autour de <strong>-100 avant JC</strong>, qu'apparaissent les premiers moulins à eau, entre la <em>Grèce</em> et la <em>Turquie</em> avant de gagner <strong>l'Empire Romain</strong>.` `Sa principale utilisation était pour moudre le grain. </p>` ` Progressivement les usages s'étendent : on trouve quelques siècles plus tard des scies hydrauliques pour couper le marbre en Jordanie. Au XIX^{ème} siècle, on utilise l'énergie hydraulique notamment dans les mines, les forges et les filatures industrielles. Apparaissent ensuite les turbines modernes auxquelles on fixera des alternateurs vers 1880 : ` ` <center>c'est la naissance de l'hydroélectricité. </center> `

Il existe de nombreuses autres balises dont vous trouverez la liste sur
http://www.w3schools.com/html/

2. Comment créer une liste?

Il est souvent utile de lister des items et il existe une méthode pour mettre en valeur une liste.

Un atome est composé :

- de protons
- de neutrons
- d'électrons

Voici ces particules classées par ordre de masse croissante :

1. neutrons
2. protons
3. électrons

```
<h2>Un atome est composé :</h2>

<ul>

<li>de protons</li>

<li>de neutrons</li>

<li>d'électrons</li>

</ul>

<h2>Voici ces particules classées par
ordre de masse décroissante : </h2>

<ol>

<li>neutrons</li>

<li>protons</li>

<li>électrons</li>

</ol>
```

3. Comment créer un tableau ?

Un tableau est souvent utile pour organiser des données de manière visuelle. Voici un exemple qui montre comment créer un tableau.

Colonne 1 / ligne 1	Colonne 2 / ligne 1	Colonne 3 / ligne 1
Colonne 1 / ligne 2	Colonne 2 / ligne 2	Colonne 3 / ligne 2

```
<table border="1" cellpadding="10">

<tr>

<td>Colonne 1 / ligne 1</td>

<td>Colonne 2 / ligne 1</td>

<td>Colonne 3 / ligne 1</td>

</tr>

<tr>

<td>Colonne 1 / ligne 2</td>

<td>Colonne 1 / ligne 2</td>

<td>Colonne 1 / ligne 2</td>

</tr>

</table>
```

4. Ajouter des images et des liens hypertexte :

Il est possible d'ajouter des images et des liens hypertext pour donner de l'interactivité à une page. C'est la base de la navigation sur n'importe quel site web.

Consulter l'article sur Wikipédia sur les dauphins

```
<img
src="http://upload.wikimedia.org/wikipedi
a/commons/thumb/9/90/Dolphins_in_sea.jpg/
800px-Dolphins_in_sea.jpg" width=320
height=240 alt="des dauphins">

<br/>

Consulter l'article sur Wikipédia <a
href=http://fr.wikipedia.org/wiki/Dauphin
target="_blank">sur les dauphins</a>
```

IV. CSS :

Le langage CSS vient s'articuler avec le langage HTML pour permettre une plus grande maitrise de la mise en page. Comment ajouter du CSS dans enyoJS ?

Si nous reprenons toujours la ligne suivante de la zone d'installation

```
{name:"NomdelaBoite", content:"maboite", style:"", allowHtml :true},
```

Vous remarquez la présence d'un paramètre `style:""`, c'est ici que nous allons insérer les paramètres CSS.

1. Les couleurs :

ce texte est orange

ce texte est rouge

ce texte est bleu
ce texte est vert

```
{name:" NomdelaBoite", content:"<h4>Ce texte est orange</h4>",
style:"color:orange;", allowHtml :true},
{name:" NomdelaBoite", content:"Ce texte est rouge",
style:"color:red;", allowHtml :true},
{name:" NomdelaBoite", content:"Ce texte est bleu",
style:"color:rgb(0,0,255) ;", allowHtml :true},
{name:" NomdelaBoite", content:"Ce texte est vert",
style:"color:#00ff00;", allowHtml :true},
```

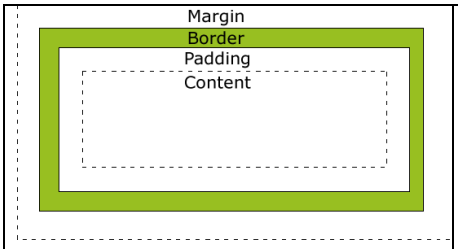
Attention, si vous attribuez plusieurs propriétés CSS à un élément, chacune des propriétés CSS devront être séparées par des points virgules.

2. Apparence générale :

1. style="**text-decoration:overline;**" 2. style="**text-decoration:line-through;**" 3. style="**text-decoration:underline;**" 4. style="**letter-spacing:2px;**" 5. style="**letter-spacing:-2px;**" 6. style="**line-height:350%;**" 7. style="**word-spacing:30px;**" 8. style="**font:20px arial;**"	1. Voici un nouveau chapitre 2. ~~Voici un nouveau chapitre~~ 3. Voici un nouveau chapitre 4. Voici un nouveau chapitre 5. Voici un nouveau chapitre 6. Voici un nouveau chapitre 7. Voici un nouveau chapitre 8. Voici un nouveau chapitre

D'autres paramètres CSS : Les bordures : http://www.w3schools.com/css/css_border.asp

3. Placement et marge :

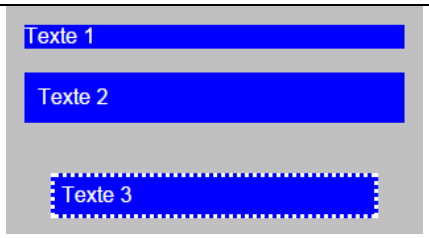


La propriété de feuille de style margin css peut prendre les valeurs de (dans l'ordre énoncé):
margin-top valeur de la marge haute,
margin-right, valeur de la marge droite,
margin-bottom, valeur de la marge basse,
margin-left, valeur de la marge gauche.
auto, valeur seule, marge calculée automatiquement.

Les valeurs des marges sont des valeurs numériques suivies de px ou % ou pt ou em ou simplement auto (mal interprété sur les anciens navigateurs).
Il est possible d'omettre des valeurs.
Si la feuille de style margin css est suivie **d'une valeur** alors cette dernière sera appliquée à toutes les marges.
Si la feuille de style css margin est suivie **deux valeurs** alors la première sera appliquée aux marges verticales et la seconde aux marges horizontales.

Exemples

```
{name:" NomdelaBoite", content:"Texte 1",
style:"padding:0px;background-
color:blue;color:white", allowHtml :true},
       {content:"<br/>", allowHtml :true},
{name:" NomdelaBoite", content:"Texte 2",
style:"padding:10px;background-
color:blue;color:white", allowHtml :true},
       {content:"<br/>", allowHtml :true},
{name:" NomdelaBoite", content:"Texte 3",
style:"margin:20px;padding:5px;border-
style:dotted;background-color:blue;color:white",
allowHtml :true},
```



A voir le CSS3, dernière évolution du CSS : http://www.w3schools.com/css/css3_intro.asp

V. Javascript : coder la logique d'une application

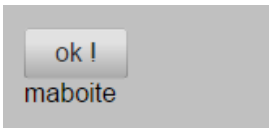
Le langage Javascript est extrêmement documenté sur Internet, une recherche pourra souvent vous faire découvrir de nouvelles fonctionnalités. Inutile par contre de chercher avec le mot clé EnyoJS trop réducteur.

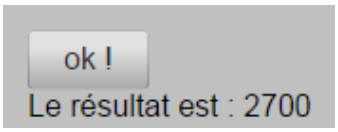
Dans enyoJS, Javascript est le langage utilisé dans les fonctions… dans la zone des fonctions.

1. Déclarer des variables et calcul simple :

La déclaration des variables est très simple, quel que soit leur type :

```
enyo.kind({
    name: "App",
    fit: true,
    components:[
        {kind:"onyx.Button", content: "ok !", ontap:"MaPremiereFonction"},
        {name:"NomdelaBoite", content:"maboite", style:"",allowHtml:true},
    ],
    MaPremiereFonction: function() {
        a = 50; // un nombre entier.
        b = 54,5; // un nombre décimal dit flottant.
        c = true; // un booléen qui peut prendre 2 états : true, false.

        // Faire un calcul
        c = a*b;

        this.$.NomdelaBoite.setContent("Le résultat est : " + c);
    },
});
```





2. Le conditions :

Une condition permet de définir deux cas

- Un cas rempli par la condition
- Les autres cas

Exemple :

Si j'ai 14 au contrôle de maths, alors mes parents m'offrent 20€, si j'ai moins, je n'ai rien.

```
if (ControleMaths>=14 ) {
argent = 20;
} else {
argent = 0;
}
```

Voici la liste des opérateurs de condition qui existent dont celui que nous venons d'utiliser.

==	Egal à
!=	Différent de
>=	Supérieur ou égal à
>	Supérieur strictement
<=	Inférieur ou égal à
<	Inférieur strictement

Combinaisons de plusieurs conditions

Si la note de maths est supérieure ou égale à 15 **ET** si la note d'anglais est supérieure ou égale à 13 alors c'est bien

```
if ((NoteMaths>=15)&&(NoteAnglais>=13)) {
this.$.Note.setContent("Bien !");
}
```

Si la note de maths est supérieure ou égale à 15 **OU** si la note d'anglais est supérieure ou égale à 13 alors c'est bien

```
if ((NoteMaths>=15)||(NoteAnglais>=13)) {
this.$.Note.setContent("Bien !");
}
```

 Attention à l'utilisation des parenthèses !

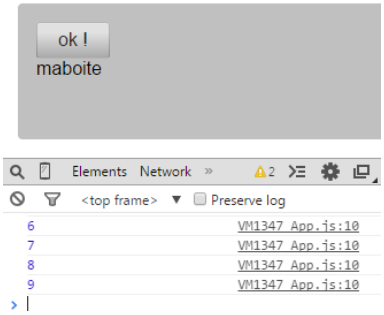
Exemple :

```
enyo.kind({
    name: "App",
    fit: true,
    components:[
        {kind:"onyx.Button", content: "ok !", ontap:"MaPremiereFonction"},
        {name:"NomdelaBoite", content:"maboite", style:"",allowHtml:true},
    ],
    MaPremiereFonction: function() {
        a = 50; // un nombre entier.
        b = 44,5; // un nombre décimal dit flottant.
        c = true; // un booléen qui peut prendre deux états : true ou
false.
        if (a>b ) {
        this.$.NomdelaBoite.setContent("a est plus grand que b");
        } else {
        this.$.NomdelaBoite.setContent("b est plus grand que a");
        }
    },
});
```

3. Les boucles for :

Pour répéter un grand nombre de fois une opération, on utilise des boucles. Une boucle exécute plusieurs la même instruction jusqu'à ce qu'une condition vienne interrompre la boucle.

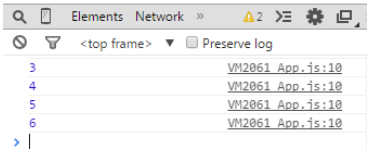
```
enyo.kind({
    name: "App",
    fit: true,
    components:[
        {kind:"onyx.Button", content: "ok !", ontap:"MaPremiereFonction"},
        {name:"NomdelaBoite", content:"maboite", style:"",allowHtml:true},
    ],
    MaPremiereFonction: function() {
        for (i=0;i<10;i++) {
        console.log(i);
        }
    },
});
```



Casser une boucle

Afin de stopper une boucle avant la fin prévue par le 2ème paramètre de la boucle for, on peut arrêter le déroulement de celle-ci à l'aide d'une condition :

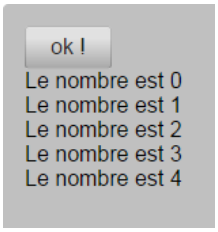
```
enyo.kind({
    name: "App",
    fit: true,
    components:[
      {kind:"onyx.Button", content: "ok !", ontap:"MaPremiereFonction"},
      {name:"NomdelaBoite", content:"maboite", style:"" , allowHtml:true},

    ],
    MaPremiereFonction: function() {
        for (i=0;i<10;i++) {
        console.log(i);
            if (i==6) {
            break;
            }
        }
    },
});
```



Les boucles while

Les boucles while comprennent directement une référence à une condition pour arrêter la boucle.

```
enyo.kind({
    name: "App",
    fit: true,
    components:[
        {kind:"onyx.Button", content: "ok !", ontap:"MaPremiereFonction"},
        {name:"NomdelaBoite", content:"maboite", style:"", allowHtml:true},
    ],
    MaPremiereFonction: function() {
        i=0; x="";
        while (i<5) {
        x = x + "Le nombre est " + i + "<br>";
        i++;
        this.$.NomdelaBoite.setContent(x);
        }
    },
});
```



4. Les tableaux (Array): Stocker et manipuler un grand nombre de variables

Les tableaux permettent de stocker en mémoire un grand nombre de données afin de les manipuler facilement.

1. Déclarer un tableau :

Les tableaux se déclarent en début de code afin qu'ils soient accessibles par toutes les fonctions (vous pouvez cependant déclarer un tableau dans une fonction, mais ce dernier ne sera pas accessible aux autres fonctions)

```
myArray = ['Simon', 'Marc', 'Maurine', 'Cécile', 'Valentin'];
myArray2 = [42, 12, 6, 3];
myArray3 = [42, 'Simon', 12, 'Cécile'];

enyo.kind({
    name: "App",
    fit: true,
    components:[
        {kind:"onyx.Button", content: "ok !", ontap:"MaPremiereFonction"},
        {name:"NomdelaBoite", content:"maboite", style:"", allowHtml:true},
    ],
    MaPremiereFonction: function() {

    },
});
```

Les différentes variables contenues dans un tableau sont numérotées à partir de 0.

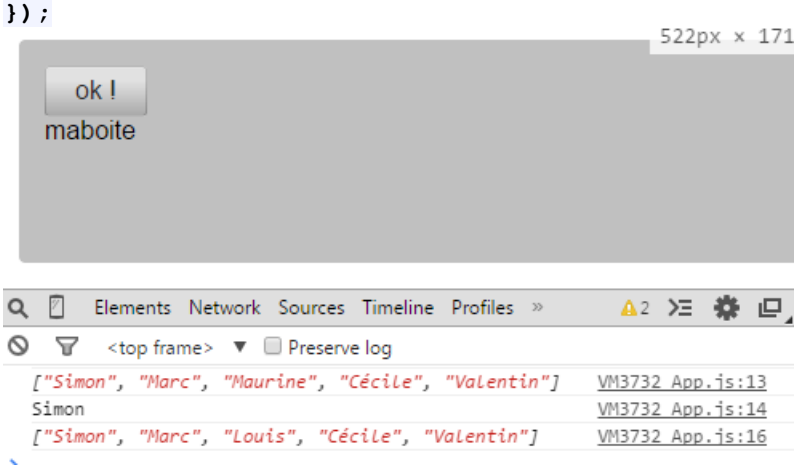
	Numéro de la case
1ère case	0
2ème case	1
3ème case	2
4ème case	3

Les chiffres sont stockés directement dans le tableau tandis que les chaines de caractères sont indiquées entre guillemets simples ou doubles.

2. Lire ou modifier les variables d'un tableau :

Voici un exemple pour récupérer la valeur d'une case ou pour en changer.

```
myArray = ['Simon', 'Marc', 'Maurine', 'Cécile', 'Valentin'];
myArray2 = [42, 12, 6, 3];
myArray3 = [42, 'Simon', 12, 'Cécile'];

enyo.kind({
    name: "App",
    fit: true,
    components:[
        {kind:"onyx.Button", content: "ok !", ontap:"MaPremiereFonction"},
        {name:"NomdelaBoite", content:"maboite", style:"", allowHtml:true},
    ],
    MaPremiereFonction: function() {
     console.log(myArray) // envoyer dans la console tout le tableau.
     console.log(myArray[0]) // envoyer dans la console la 1ere case
     myArray[2] = 'Louis'; // modifie la variable placée sur la 3eme case.
     console.log(myArray) // envoyer dans la console tout le tableau.
    },
});
```



Voici un exemple supplémentaire pour ajouter ou supprimer des valeurs d'un tableau

```
myArray = ['Simon', 'Marc', 'Maurine', 'Cécile', 'Valentin'];
myArray2 = [42, 12, 6, 3];
myArray3 = [42, 'Simon', 12, 'Cécile'];

enyo.kind({
    name: "App",
    fit: true,
    components:[
        {kind:"onyx.Button", content: "ok !", ontap:"MaPremiereFonction"},
        {name:"NomdelaBoite", content:"maboite", style:"", allowHtml:true},
    ],
    MaPremiereFonction: function() {
     console.log(myArray) // envoyer dans la console tout le tableau.
     myArray.push('Ludovic'); // ajoute en dernière position une variable
     console.log(myArray) // envoyer dans la console tout le tableau.
     myArray.pop(); // supprimer la dernière valeur
     console.log(myArray) // envoyer dans la console tout le tableau.
      },
});
```