
	<i>Contrôle domotique à distance</i>	<div data-bbox="1157 123 1197 179">T</div> 
SIN	Interface	

Objectifs de formation :

09 - Gérer la vie d'un système

Compétence attendue :

CO9.sin2. Installer, configurer et instrumenter un système réel Mettre en oeuvre la chaîne d'acquisition puis acquérir, traiter, transmettre et restituer l'information

Programme

3.1 Réalisation d'un prototype

Taxonomie : 3

Problématique de la séquence

On souhaite pouvoir contrôler à distance une application domotique.

Afin de réaliser le projet, on a retenu la solution suivante :

L'application domotique est gérée par un contrôleur programmable qui récupère l'information température issue de capteurs présents dans la maison et commande les convecteurs électriques.

Ce contrôleur programmable dispose d'une liaison Ethernet et met à disposition certaines valeurs -- en lecture et en écriture -- au travers du protocole Modbus/TCP.

Un serveur Web communique avec le contrôleur programmable et crée une IHM (Interface Homme Machine) grâce à une page Web.

Un PC ou un terminal mobile (tablette, smartphone...) doit pouvoir accéder à la page Web de n'importe quel endroit.

L'objectif de cette activité est de réaliser une interface dynamique afin d'améliorer l'ergonomie de m'application pour l'utilisateur.

I. Premiers programmes en Javascript

- On va intégrer une image SVG dans une page HTML. **Enregistrez** le code suivant dans un fichier que vous appellerez **document.html**., puis **visualisez** le résultat dans un navigateur Web.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<!DOCTYPE html>

<html>

<head>

</head>

<body>

<object type="image/svg+xml" height="80" width="300">

    <svg xmlns="http://www.w3.org/2000/svg" width="300" height="80">

        <rect id="rect1" x="60" y="10" width="60" height="60" fill="blue" />

        <rect id="rect2" x="160" y="10" width="60" height="60" fill="green" />

    </svg>

</object>

<p></p>

<input type="button" value= « Valider »/>

</body>

</html>
```

Dans ce TP, nous allons utiliser une bibliothèque Javascript appelée JQuery, utilisée par de nombreux sites comme Google, Amazon, etc....

- Placer la bibliothèque [Jquery](#) dans votre répertoire de travail.
- Pour intégrer cette bibliothèque à votre page HTML, il est nécessaire de rajouter la ligne suivante à l'intérieur des balises <head> :

```
<script src="jquery-2.0.3.js" type="text/javascript"></script>
```

Il est possible de récupérer la valeur d'un attribut d'un objet. Le code suivant permet de récupérer la couleur du rectangle.

```
$("#rect1").attr("fill")
```

- Modifier la fonction action() pour afficher la couleur du rectangle 2.
- Modifier le fichier pour que la couleur du rectangle s'affiche lorsque l'on passe la souris dessus. *Indication : On pourra créer une fonction changer_couleur() et passer le nom du rectangle en paramètre à la fonction.*

Il est également possible de modifier l'attribut d'un objet. Le code suivant permet de mettre en rose le rectangle.

```
$("#rect1").attr("fill", "pink")
```

- Modifier le fichier pour que le rectangle survolé par la souris devienne rouge.
- Modifier le fichier pour que le rectangle soit rouge lors du survol de la souris, puis reprenne sa couleur quand la souris sort. *Indication : On pourra créer une deuxième fonction retrouver_couleur() et passer le nom et la couleur du rectangle en paramètres à la fonction.*

II. Réalisation de l'interface

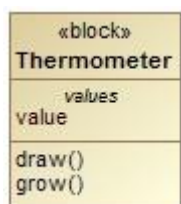
Dans cette partie on va utiliser la librairie [RGraph](#), à installer dans votre répertoire de travail.

Q1) En utilisant la bibliothèque RGraph, créer un thermomètre permettant de visualiser la valeur de la température.

Q2) Modifier votre page pour que la température de la simulation soit précisée au chargement de la page.

Q3) Créer un bouton "Rafraîchir".

Q4) Rédiger un script afin de rafraîchir la valeur de la température lors d'un clic sur le bouton. On utilisera pour cela les attributs et méthodes de la classe Thermometer :



Q5) En vous appuyant sur la fonction [setInterval](#), modifier le script pour que la température soit rafraîchie toutes les 2 minutes.

Q6) Créer un formulaire afin de pouvoir modifier la consigne :

Consigne :

Actualiser

Q7) Rédiger un script afin d'appeler le fichier PHP qui permet la modification de la consigne.

Q8) Modifier le script pour que la page affiche soit la confirmation de la mise-à-jour de la consigne, soit une notification d'erreur.

Restitutions :

Vous proposerez une synthèse sous forme numérique qui sera présenté au reste de la classe, selon la répartition suivante :

Groupe 1 : le paramétrage du thermomètre de la librairie RGraph

Groupe 2 : le principe de l'architecture AJAX

Groupe 3 : la mise à jour de la valeur de la température

Groupe 4 : la mise à jour de la consigne

Groupe 5 : le principe de la librairie JQuery

ANNEXE : la fonction setInterval

SetTimeout et setInterval: Les délais en JavaScript

On peut en JavaScript déclencher des actions après un intervalle de temps donné, ou de la répéter après un intervalle de temps.



Les méthodes *setTimeout* et *setInterval* sont des méthodes de l'objet Window. On peut donc écrire *setTimeout* ou *window.setTimeout*.

Ce sont des processus indépendants qui, quand ils sont lancés par une instruction, ne bloquent pas l'affichage du reste de la page ni les actions de l'utilisateur.

SetTimeout indique un délai avant exécution

Le méthode *setTimeout* définit une action à exécuter et un délai avant son exécution. Elle retourne un identifieur pour le processus.

```
var x = setTimeout("instructions", délai en millisecondes).
```

Exemple:

```
function mafonction() { ...}  
  
setTimeout(mafonction, 5000);
```

La fonction sera exécutée après 5 secondes (5 000 millisecondes).

Le délai commence à l'instant où *setTimeout* est exécutée par l'interpréteur JavaScript, donc à partir du chargement de la page si l'instruction est dans un script exécuté au chargement, où à partir d'une action de l'utilisateur si le script est déclenché par celui-ci.

setInterval déclenche une opération à intervalles réguliers

Similaire à *setTimeout*, elle déclenche répétitivement la même action à intervalles réguliers.

```
setInterval("instructions", délai)
```

Exemple:

```
setInterval("alert(('bip'))", 10000);
```

Affiche un message toutes les dix secondes.

L'action sera recommencée jusqu'à ce que l'on quitte la page ou que *clearInterval* soit exécutée.

Une fonction lambda peut être un argument

On peut définir une fonction dans les arguments de *setTimeout* ou *setInterval*.

Exemple:

```
var x = setTimeout(function() { alert("bip"); }, 2000);
```

Source : <https://www.xul.fr/>