WebForce3

Du 9 mai au 30 août 2018

# Informations diverses

## Formateurs

Formateur HTML, CSS/SASS, JavaScript, Angular/Ionic, Symfony :

Formateur : Mike

Tél : 06.68.00.51.26

Mail : [msylvestreets@yahoo.fr](mailto:msylvestreets@yahoo.fr)

## Communication

Il y a le site de WebForce3 <https://cours.wf3.fr/> mais Discord <https://discordapp.com/> est à privilégier.

# JavaScript

## Introduction

On place ce code entre les balises <script> et </script>. On pourrait placer du code en d’autres langages de programmation, mais il faudrait alors spécifier lequel. Il y a plusieurs endroits où l’on peut placer le bloc. Notamment dans le body, juste avant la fermeture de la balise.

<!DOCTYPE html>

<html lang="en">

<head>

(...)

</head>

<body>

<script>

alert("Hello world !");

</script>

</body>

</html>

Ce script affiche une boite de dialogue avec un « Hello World ! »

## Les variables

Il est nécessaire de déclarer les variables avant de les utiliser. Le mot-clef classique est var, plus récent, à préférer, let. Préférable parce qu’elle donne à la variable une portée plus restreinte.

JavaScript connait les types string, integer, float et boolean. Il y a aussi un type undefined. Le type par défaut est number (qui englobe integer et float).

Quelques utilisations :

<script>

alert("Hello world");

var a = "2" // string

var b = +a // avec le plus a devient un nombre.

var b = parseInt(a) // devient int

var b = parseFloat(a) // devient float

var b = number(a) // devient number (qui reste le type de base)

</script>

Le scope (portée de la variable)

considérer :

} else if (valeur.length > 4) {

let array = [];

for (let i = 0; i < valeur.length; i++) {

console.log(i);

}

}

...

<script>

// Différence entre Let et Var

function duifferenceVarLet(){

var variable1=0;

var variable1 = 1 ;

let variable2 = 0 ;

let variable2=1

}

</script>

Var est le moyen classique de déclarer une variable. Let est le moyen recent.

une variable déclarée avec var peut être redéclaré. Avec let non; une fois déclaré, redéclaré génère une erreur.

Une variable déclarée dans une fonction n'est visible que dans la fonction. Une variable déclarée dans une boucle for n'est pas visible en dehors si elle a été déclarée avec let, alors que si elle est avec var, elle restera visible en dehors.

for(...){

var variable1 ; // visible en dehors de for.

let variable2 ; // portée limitée au for.

}

Il est conseillé de n'utiliser que let, quelque soit la situation.

Les objets, par exemple les tableaux, ont des méthodes disponibles. Par exemple sort(), push(), pop() et bien d'autres.

## Les fonctions

### Les fonctions nommées

Une fonction est décomposée en trois parties :

* Le mot-clef "function" et son nom.
* Les parenthèses.
* Les accolades.

### Les fonctions anonymes

Les fonctions anonymes sont un bloc d’instructions.

function{

(instruction 1)

(instruction 2)

(...)

}

## Les conditionnels

### If.

L’opérateur d’égalité == compare deux valeurs. L’opérateur === compare les valeurs et le type.

"2" et 2 seront considérés comme égaux si on ne compare pas aussi le type avec ===.

### Switch.

## Opérateurs logiques

Pour combiner deux conditions, on utilisera volontiers l'opérateuir logique EZT ou l'opérateur logique OU (inclusif). Dans le premier cas, les deux conditions doivent être vraies pour que la combinaison soit vraie. Dans le cas du OU inclusif, il suffif que l'une des conditions au moins soit vraie, pour que la combinaison soit vraie.

C'est formalisé dans des tables de vérité :

ET FAUX VRAI

FAUX FAUX FAUX

VRAI FAUX VRAI

OU FAUX VRAI

FAUX FAUX VRAI

VRAI VRAI VRAI

ET s'écrit && et OU s'écrit ||.

## Les boucles

### For

### While

## Output

Pour savoir où en est le programme, on peut faire des sorties sur une boite de dialogue ou sur la console, dans l’inspecteur du navigateur.

<script>

// Boite de dialogue.

alert("Hello world");

// Sortie sur console

console.log("Hello World!");

console.warn("Alerte!");

console.error("Danger");

</script>

Il est important d’utiliser la console. Quand on ajoute une fonction ou un évènement, c’est important de tester pour voir si jusque là tout va comme prévu avant d’aller plus loin. Sans ça, le débuggage risque de devenir laborieux.

## Le Document Object Model (DOM)

Il s’agit de la description de la page web. La raison d’être première de JavaScript est de manipuler le DOM.

Pour atteindre un élément dans le DOM, on dispose des méthodes suivantes :

* document.getElementById()
* document.getElementsByClassName()
* document.getElementsByName()
* document.getElementsByTagName()

Bien noter que lorsqu'on récupère un élément par son id, un seul élément est renvoyé (d'où le singulier). Pour les autres, c'est un tableau d'éléments qui est renvoyé (d'où le pluriel).

On utilisera plus volontiers :

* document.querySelector()
* documentQuerySelectorAll()

Le premier sélectionne la première occurrence uniquement ; le second un tableau de tous les éléments correspondant à la recherche.

## Les formulaires

<form action="" method= "get">

<input type="text" name="texte" value="Texte par défaut">

</form>

L’attribut *name* est obligatoire ; c’est celui qui va être utilisé pour récupérer les données du contrôle.

## Formulaires et JavaScript

Très souvent, JavaScript va fondamentalement servir à tester les réponses sur les formulaires avant de les envoyer au serveur.

<body>

<form action="" method="get" onsubmit="envoiDuFormulaire();">

<input type="text" name="valeur" id="valeur">

<br>

<button type="submit">Envoyer</button>

</form>

<script>

function envoiDuFormulaire() {

event.preventDefault();

console.log(document.getElementById("valeur"));

console.log(document.getElementsByName("valeur"));

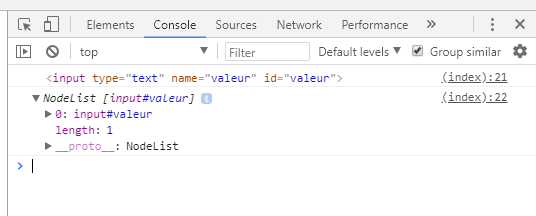
}

</script>

</body>

C’est pour les besoins de l’illustration. Il est déconseillé de mettre des évènements (ici onsubmit()) dans les balises. On préférera l’évènement onsubmit à onclick parce qu’il peut y avoir d’autres moyens pour soumettre le formulaire, par exemple à l’aide de la touche <Enter>.

On peut afficher les résultats sur la console :



La console permet aussi de voir toutes les propriétés et méthodes disponibles pour un objet:



A chaque fois que l'on travaille sur des formulaires ajouter l'instruction event.preventDefault pour annuler l'évènement par défaut. L'envoi du formulaire se fera manuellement.

Exercice 1. Vérifier que l’utilisateur entre bien « Mike » et afficher un message de bienvenue ou d’erreur selon le cas.

<script>

function exercice1() {

event.preventDefault();

let variable = document.getElementById("valeur");

if (variable.value == "Mike") {

console.log("Bienvenue");

} else {

console.log("Erreur de frappe");

variable.value = "";

}

}

</script>

Il convient de supprimer les espaces lorsqu’on récupère un input.

Exercice 2. La fonction doit accepter mike en majuscules ou minuscules, tout ou en partie, et des espaces avant et après.

<script>

function exercice2() {

event.preventDefault();

let valueInput = document.getElementById("valeur");

console.log(valueInput);

let valeur = valueInput.value.trim(); //

valeur = valeur.toUpperCase();

if (valeur == "MIKE") {

console.log("Bienvenue.");

} else {

console.error("Erreur de frappe.");

}

}

</script>

valueInput est la balise ; value un string.

## Opérations sur les chaînes de caractères

Pour récupérer un caractère à une position donnée: .charAt(n)

Par exemple, valeur.charAt(0) pour le premier caractère.

Ma méthode slice considère la chaine à partir de la position du caractère donné .slice(n)

Par exemple valeur.slice(1).

Pour concaténer, le signe plus : +.

L'expression suivante mettra le premier caractère en mayuscule.

console.log(valeur.charAt(0).toUpperCase() + valeur.slice(1) ;

## Exercices JavaScript sur Basic 88

On récupère le template basic 88 sur OS Templates (<https://www.os-templates.com/free-basic-html5-templates/basic-88>)

On note plusieurs fichiers index.html dans les différents dossiers. C’est l’ancienne façon d’empêcher le visiteur d’afficher la liste des fichiers et dossiers. A proscrire. Aujourd’hui on fait ça avec .htaccess .

Le charset est réglé sur la norme iso-8859-1 alors qu’on utilisera systématiquement UTF-8.

On créé un fichier scripts/webforce3.js

On peut activer JS à l’ancienne.

<!—index.html -->

<body onload="ready()">

(...)

<script src="scripts/webforce3.js"></script>

</body>

// scripts/webforce3.js

function ready() {

console.log("Le Saut de Sandra");

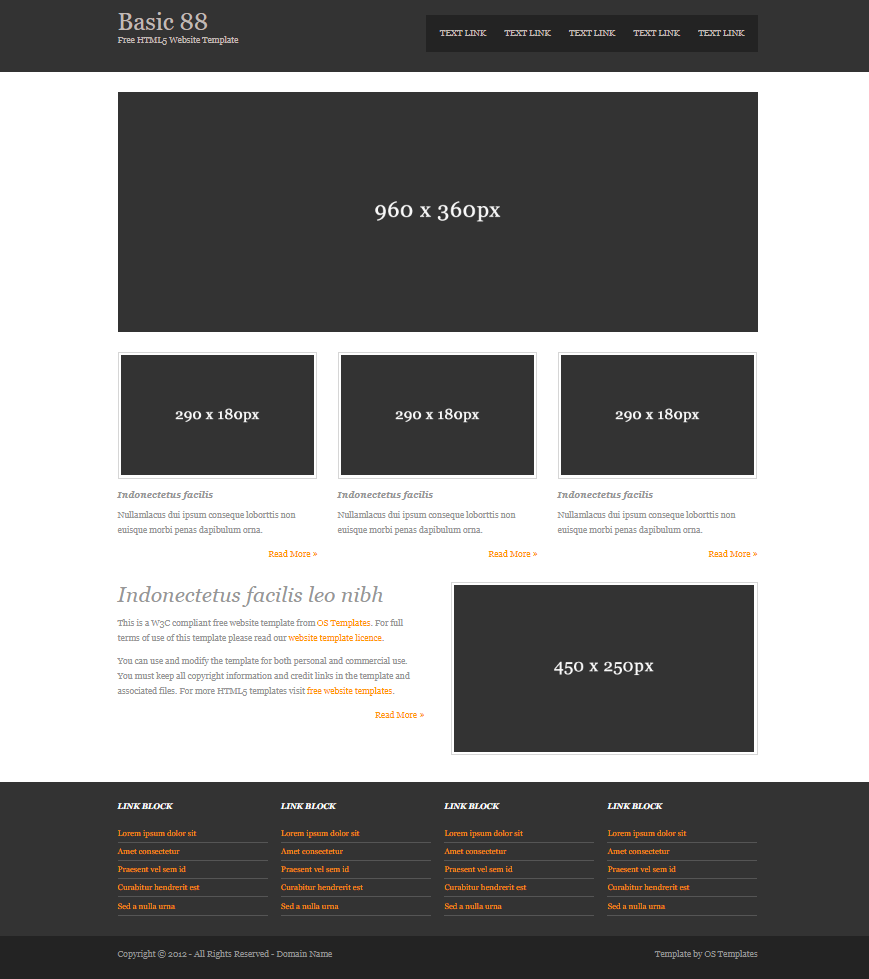
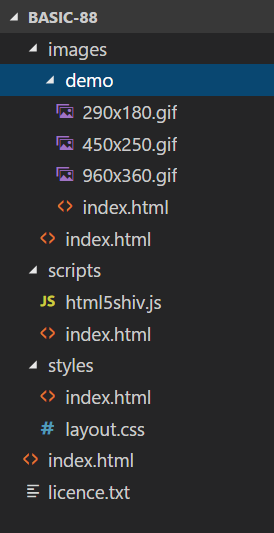
// Tout le code js vient ici dedans.

}

Sortie sur console :

>Le Saut de Sandra

Cependant, plutôt que de capturer les évènements dans les balises avec des onquelquechose(), on va créer des eventlisteners.

## Base 88 Le carrousel

Exercice : faire un carrousel sur l’image 960x360.

Pour régler le temps qu’une image reste affichée, il y a setIntervall() et setTimeOut().

Exemples :

setIntervall(function(){

console.log("Le Saut de Sandra") ;

}, 3000) ;

Tous les 3.000 millisecondes, la console affiche le message.

setTimeOut(function(){

console.log("Le Saut de Sandra") ;

}, 3000) ;

Au bout de 3.000 millisecondes, le message est affiché.

Nous voulons que tous les x secondes, l’image change.

Nous commençons par séparer l’espace du script en deux, en haut les déclarations de variables, plus bas les eventlisteners et en bas les fonctions.

Nous déclarons un tableau avec les url vers les images du carrousel, nous récupérons la balise image du carrousel et définissons un indice à zéro. Lorsque cet indice arrivera au nombre maximum d’images disponibles, il se remettra à zéro.

Ensuite, avec setInterval() on modifie l’image affichée tous les x secondes.

// scripts/webforce3.js

function ready() {

/\*\*\*\*\*\*\*\*\* Déclarations de variables \*\*\*\*\*\*\*\*\*/

let imagesHeader = [

"\*.jpg",

"\*.jpg",

"\*.jpg",

"\*.jpg",

"\*.jpg"

];

let imageHeader = document.querySelector("#slider img");

let indice = 0;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\* eventlisteners \*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* fonctions \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Carrousel

setInterval(function() {

if (indice == imagesHeader.length)

indice = 0;

imageHeader.src = imagesHeader[indice];

indice++;

}, 2500);

}

## Base 88 Images aléatoires

On veut que les images 290x180 s’affichent de façon aléatoire.

On commence par créer un tableau d’url d’images imagesArticle. Ensuite on crée une fonction qui récupère un tableau des trois images et les remplace par des images choisies de façon aléatoire. Ceci au bout d’un delais géré par setTimeOut().

let imagesArticle = [

"\*.jpg",

"\*.jpg",

"\*.jpg",

"\*.jpg",

"\*.jpg",

]

(...)

setTimeout(randomPicture, 2500); // S'execute apres 2,5s.

function randomPicture() {

let balisesImg = document.querySelectorAll("#services img");

balisesImg[0].src=imagesArticle[Math.round(Math.random()\*((imagesArticle.length - 1)))];

balisesImg[1].src=imagesArticle[Math.round(Math.random()\*((imagesArticle.length - 1))];

balisesImg[2].src=imagesArticle[Math.round(Math.random()\*((imagesArticle.length - 1)))];

}

On peut aussi préférer setInterval() pour ne pas que ça reste bloqué après un seul changement.

# jQuery

Table des matières

[Informations diverses 1](#_Toc518335684)

[Formateurs 1](#_Toc518335685)

[Communication 1](#_Toc518335686)

[JavaScript 1](#_Toc518335687)

[Introduction 1](#_Toc518335688)

[Les variables 1](#_Toc518335689)

[Les fonctions 3](#_Toc518335690)

[Les fonctions nommées 3](#_Toc518335691)

[Les fonctions anonymes 3](#_Toc518335692)

[Les conditionnels 3](#_Toc518335693)

[If. 3](#_Toc518335694)

[Switch. 3](#_Toc518335695)

[Opérateurs logiques 3](#_Toc518335696)

[Les boucles 4](#_Toc518335697)

[For 4](#_Toc518335698)

[While 4](#_Toc518335699)

[Output 4](#_Toc518335700)

[Le Document Object Model (DOM) 5](#_Toc518335701)

[Les formulaires 5](#_Toc518335702)

[Formulaires et JavaScript 5](#_Toc518335703)

[Opérations sur les chaînes de caractères 7](#_Toc518335704)

[Exercices JavaScript sur Basic 88 7](#_Toc518335705)

[Base 88 Le carrousel 8](#_Toc518335706)

[Base 88 Images aléatoires 9](#_Toc518335707)

[jQuery 10](#_Toc518335708)