Développement Web:
Javascript & PHP

CHAPITRE 3

LES STRUCTURES DE CONTRÔLE

PLAN

- Les structures conditionnelles
- LES structures itératives

LES STRUCTURES CONDITIONNELLES

Structure conditionnelle simple (if…)

- Syntaxe:

```
if  (condition)
    {  Bloc d'instructions }
```

- Exemple:

```
if (x==2)
     document.write("X vaut 2");
```

Structure conditionnelle composée (if…else)

- Syntaxe:

```
if  (condition)
    {  Bloc d'instructions 1   }
else
    {  Bloc d'instructions 2   }
```

- Exemple:

```
if (x==2)
     document.write("X vaut 2");
else
     document.write("Autre valeur");
```

Structure conditionnelle à choix multiple (switch… case)

- Syntaxe:

```
switch (Variable)
{
case Valeur1 : Liste d'instructions 1;
                    break ;
case Valeur2 : Liste d'instructions 2 ;
                    break ;
case ValeurN : Liste d'instructions N ;
                    break ;
default : Liste d'instructions par défaut ;
           break ;
}
```

- Exemple:

```
let x=prompt("Entrez un chiffre entre 1 et 3") ;

switch (x) {
    case "1" : alert("La valeur 1") ; break ;
    case "2" : alert("La valeur 2") ; break ;
    case "3" : alert("La valeur 3") ; break ;
    default : alert("Vous n'avez pas entré une
valeur autorisée !") ;
}
```

Les structures itératives

Boucle for

- Syntaxe:

```
for (initialisation ; condition ; itération)
   { Bloc d'instructions     }
```

- Exemple:

```
for (let i = 0; i < 10; i++)
{
    document.write(i + " ");
}
```

Boucle while

- Syntaxe:

```
while (condition)
{
    Bloc d'instructions
}
```

- Exemple:

```
let compt=1;
while (compt<5) {
    document.write ("ligne : " + compt + "<BR>");
    compt++;
}
```

Boucle do … while

- Syntaxe:

```
do
{
    Bloc d'instructions
}
while(condition);
```

- Exemple:

```
let compt=1;
do{
    document.write ("ligne : " + compt +
"<BR>");
    compt++;
}while(compt<5)
```

Développement Web:
Javascript & PHP

CHAPITRE 4

TABLEAUX ET FONCTIONS

PLAN

- Les tableaux
- Les fonctions

Les Tableaux

Les tableaux

- Tableau: Ensemble ordonné de valeurs accessibles par un indice numérique
- En JavaScript
  - comme en C, C++, Java … cet indice démarre à 0
  - Les tableaux sont des objets:
    - **`Array`** est le type objet prédéfini (fonction constructeur) utilisé pour créer un tableau
    - **`length`** est un attribut qui donne la taille du tableau
  - Les éléments peuvent être de n'importe quel type (pas forcément le même type pour tous les éléments du tableau)

```
let tab = [ "un" , "deux", "trois" ];
// tab = new Array("un" , "deux", "trois");
let tab1 = [ 1, "deux", 3];
console.log(typeof tab);  //object
console.log(tab.length);  //3
```

Création d'un tableau

- En utilisant le constructeur **`Array`** :

```
let colors = new Array();
let colors2 = new Array(5);
let colors3 = new Array("red","blue","green");
```

- En utilisant les **`[…]`**
  - Les tableaux peuvent utiliser les propriétés et méthodes du constructeur Array() .

```
let colors4 = ["red","blue","green",];
console.log(colors4.length)
```

Accéder à une valeur dans un tableau

- Pour accéder à une valeur en particulier dans un tableau, il suffit de préciser le nom du tableau puis l'indice associé à la valeur à laquelle on souhaite accéder entre crochets **[]**.

```
let colors= new Array("red","blue","green");
colors[1]="yellow"
alert(colors[1]); // "yellow""
```

- **Parcourir toutes les valeurs d'un tableau**

```
let tab = [] ;
for ( let i = 0 ; i < 5 ; i++) {
    tab [i] = i;
    document.write (tab[i]+" ");
}
```

Parcours d'un tableau: boucle for … of

- **boucle sur les éléments :**

```
let tab = [ "un" , "deux" , "trois" ];
for (let nb of tab) {
    console.log(nb);
}
```

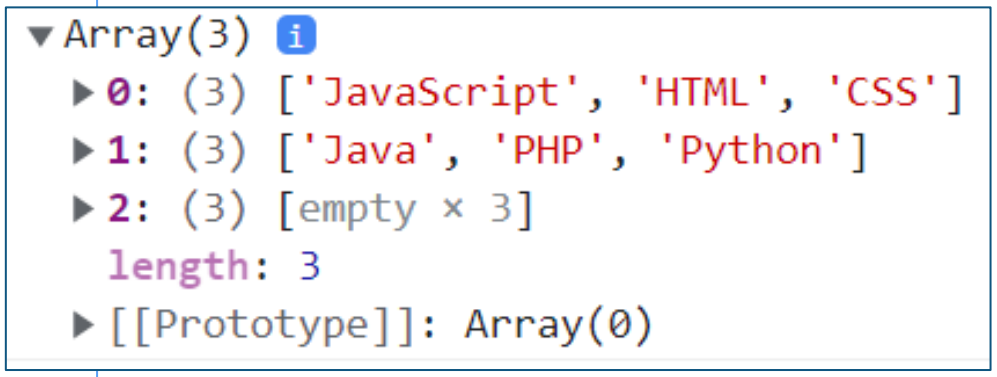
- C'est l'équivalent de la boucle for suivante:

```
for (let i = 0; i < tab.length; i++) {
    console.log(tab[i]);
}
```

Les tableaux multidimensionnels

```
let langages = new Array(3);
for (let i=0; i < 3; i++){
      langages[i] = new Array(3);
}
langages[0][0] = "JavaScript";
langages[0][1] = "HTML";
langages[0][2] = "CSS";

langages[1]=["Java", "PHP", "Python"];
console.log(langages);
```



Les Fonctions

Les fonctions

- On appelle fonction un sous-programme qui permet d'effectuer un ensemble d'instructions par simple appel de la fonction.
- En Javascript, il existe deux types de fonctions :
  - Les fonctions prédéfinies qui peuvent s'appliquer pour un objet spécifique, qu'on appelle **des méthodes**.
  - Les fonctions que vous définissez vous-même pour les besoins de votre script.
    - Une fonction doit être définie avant d'être utilisée

Déclaration et appel

- **Déclaration:** via le mot clé **`function`**

```
function nomDeLaFonction(liste des arguments){
    Liste des instructions;
    return valDeRetour; //si la fonction retourne une valeur
}
```

- **Appel:** via le nom de la fonction et (éventuellement) sa liste d'arguments

```
nomDeLaFonction(arg1, arg2, ...)
```

Déclaration et appel

- **Exemple:**

```
function positif(x) {
    if(x>=0)
        console.log("Nombre positif");
    else
        console.log("Nombre négatif")
}

function somme(a, b){
    return a+b;
}
positif(-6)
let s=somme(3,5);
```

La forme fléchée

- C'est une autre syntaxe de déclaration de fonctions qui est équivalente, côté fonctionnement, à celle proposée auparavant.

```
let nomDeLaFonction = (liste des arguments)=>
{    Liste des instructions; }
```

- Exemples:

```
//Déclaration
let myFunction = (a, b) => a * b;
//appel
myFunction(4, 5)
```

```
hello = () => {
    return "Hello World!";  }
```

- Voir: https://www.w3schools.com/js/js_arrow_function.asp

Portée des variables

- La « portée » d'une variable désigne l'espace du script dans laquelle elle va être accessible.
- En JavaScript, il n'existe que deux espaces de portée différents :
  - L'espace global: le script entier, à l'exception de l'intérieur des fonctions
  - L'espace local: l'espace dans une fonction.
- une variable définie dans l'espace global d'un script va être accessible à travers tout le script, même depuis une fonction.
- une variable définie dans une fonction n'est accessible que dans cette même fonction et ne peut pas être manipulée depuis l'espace global du script.

Portée des variables

```
//On déclare deux variables globales
let x = 5;
var y = 10;
//On définit une première fonction qui utilise les variables globales
function portee1(){
    document.write("Depuis portee1() : <br>x = " + x + "<br>y = " + y+ "<br>");
}
//On définit une deuxième fonction qui définit des variables locales
function portee2(){
    let a = 1;
    var b = 2;
    document.write("Depuis portee2() : <br> a = " + a + "<br>b = " + b+ "<br>");
}
//On définit une troisième fonction qui défnit également des variables locales
function portee3(){
    let x = 20;
    var y = 40;
    document.write("Depuis portee3() : <br>x = " + x + "<br>y = " + y+ "<br>");
}
```

Portée des variables

```
//Exécuter les fonctions
portee1();
portee2();
portee3();
//Afficher les variables globales puis locales depuis l'espace global
document.write("Depuis l'espace global : <br>x = " + x +" <br>y = " + y+ "<br>");
document.write("Depuis l'espace global : <br>a = " + a + "<br>b =" + b);
```

Depuis portee1() :
x = 5
y = 10
Depuis portee2() :
a = 1
b = 2
Depuis portee3() :
x = 20
y = 40

Depuis l'espace global :
x = 5
y = 10

Uncaught ReferenceError: a is not defined exp.html:35
at exp.html:35:58

Questions?