JAVASCRIPT

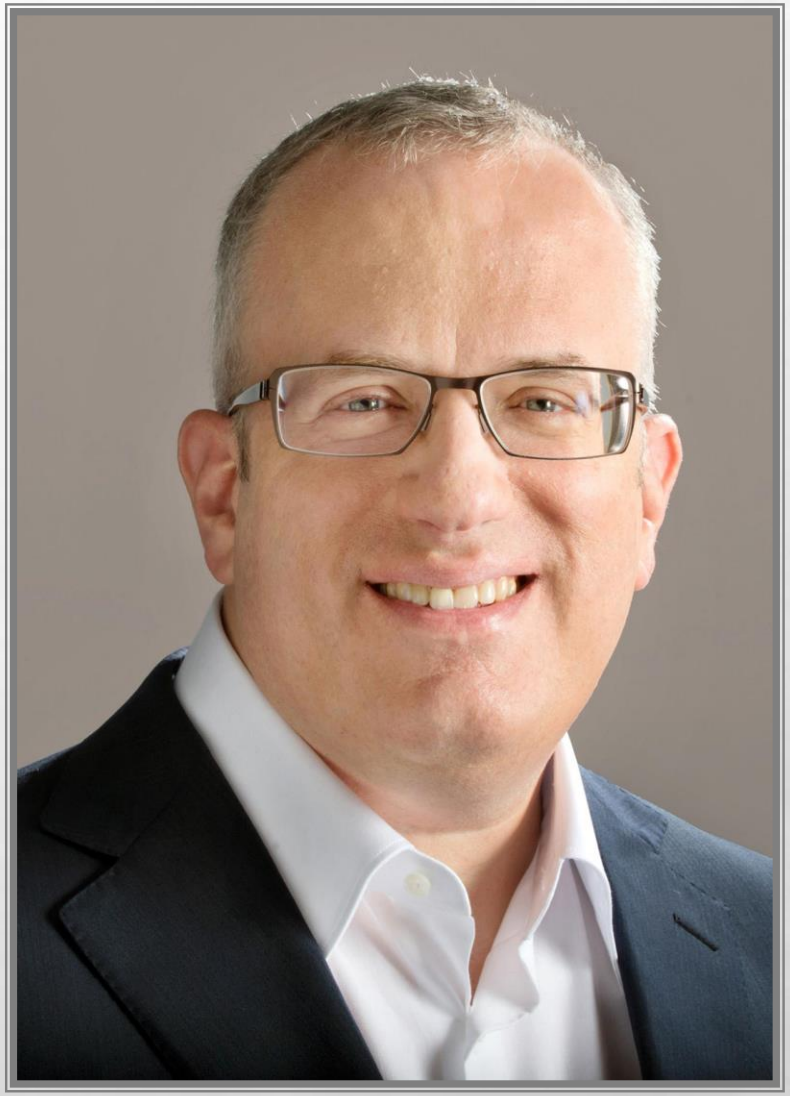
JAVASCRIPT - INITIATION

- LE JAVASCRIPT EST UN LANGAGE DE SCRIPT ORIENTÉ OBJET DYNAMIQUEMENT TYPÉ.
- LANGAGE DE SCRIPT : NÉCESSITANT UN INTERPRÉTEUR POUR ÊTRE LU. DANS LE CAS DU JAVASCRIPT L'INTERPRÉTEUR EST DANS LE NAVIGATEUR
- ORIENTÉ OBJET : LES ÉLÉMENTS DU LANGAGE SONT DES OBJETS. UNE CHAINE DE CARACTÈRE EST UN 'OBJET STRING'
- PAS OU FAIBLEMENT TYPÉ : LES VARIABLES CONTIENNENT UNIQUEMENT DES OBJETS, DE TYPE DIFFÉRENT MAIS TOUJOURS DES OBJETS. UNE VARIABLE CONTENANT UNE CHAINE DE CARACTÈRE '5' PEUT ÊTRE REDÉFINI POUR CONTENIR LE NOMBRE 5.
- LE JAVASCRIPT EST PRINCIPALEMENT UTILISÉ DU COTÉ CLIENT POUR DYNAMISER ET AJOUTER DES INTERACTIONS SUR DES PAGES HTML RENVOYÉ PAR LE SERVEUR MAIS PEUT AUSSI ÊTRE UTILISÉ COTÉ SERVEUR AVEC NODE JS.

JAVASCRIPT - HISTOIRE

EN 1995 BRENDAN EICH DE CHEZ NETSCAPE DÉVELOPPA LE LANGAGE INITIALEMENT APPELÉ LIVESCRIPT DESTINÉ À ÊTRE INSTALLÉ DIRECTEMENT DANS LES NAVIGATEURS NETSCAPE. LE LANGAGE SERA ENSUITE RENOMMÉ JAVASCRIPT EN HOMMAGE AU JAVA. MICROSOFT DÉVELOPPA L'ANNÉE QUI SUIT LE LANGAGE JSCRIPT POUR RESTER DANS LA COURSE DES NAVIGATEURS.

TOUJOURS AU COURS DE L'ANNÉE 1996 NETSCAPE ENVOYÈRENT UNE DEMANDE POUR STANDARDISER LE LANGAGE EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION (ECMA). DEPUIS LE LANGAGE A BEAUCOUP ÉVOLUER ET LA DERNIÈRE VERSION STABLE EST L'ECMASCRIPT 5 SORTIE EN 2009. L'ECMASCRIPT 6 EST SORTIE EN MAI 2015 ET LE 7 SORTIRA SOUS PEU CAR IL A ÉTÉ DÉVELOPPÉ EN PARALLÈLE DU 6. BIEN QUE L'ECMASCRIPT 6 SOIT DÉJÀ IMPLÉMENTER DANS CERTAINS NAVIGATEURS, IL NE MARCHERA PAS DANS LES ANCIENS NAVIGATEURS. SON UTILISATION RESTE DONC TRÈS LIMITÉE. DES SOLUTIONS EXISTENT POUR DÉVELOPPER AVEC L'ECMASCRIPT 6 QUI EST ENSUITE COMPILÉ EN ECMASCRIPT 5.



JAVASCRIPT - EN QUELQUES LIGNES

- JAVASCRIPT PERMET D'EFFECTUER DES TRAITEMENTS (DANS LE NAVIGATEUR) SANS INTERVENTION DU SERVEUR.
- LE CODE EST EMBARQUÉ DANS LA PAGE HTML.
- LANGAGE ORIENTÉ OBJET.
- LES NAVIGATEURS OFFRENT UNE COLLECTION D'OBJETS PRÉDÉFINIS EXPOSANT L'ENVIRONNEMENT D'EXÉCUTION DE LA PAGE COURANTE (LE DOCUMENT OBJECT MODEL).
- AUCUNE CONFIDENTIALITÉ AU NIVEAU DU CODE SOURCE.

JAVASCRIPT - EN QUELQUES LIGNES

- On dit que le HTML, le CSS et le JavaScript sont des standards du web car les principaux navigateurs web (Google Chrome, Safari, Firefox, etc.) savent tous « lire » (ou « comprendre » ou « interpréter ») ces langages et les interprètent généralement de la même façon ce qui signifie qu'un même code va généralement produire le même résultat dans chaque navigateur.

JAVASCRIPT - EN QUELQUES LIGNES

HTML

- DYNAMISATION DES PAGES
  - ✓ AFFICHAGE RÉACTIF AUX MOUVEMENTS DE SOURIS
  - ✓ MENUS DÉROULANTS
- PAGES PERSONNALISÉES PAR RAPPORT À L'ENVIRONNEMENT DU VISITEUR.
  - ✓ EN FONCTION DU NAVIGATEUR
  - ✓ EN FONCTION DES CONTRAINTES DE SON ÉCRAN

AJAX

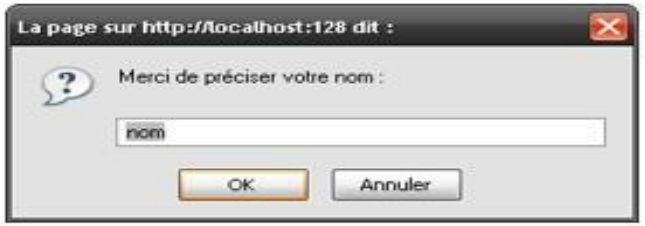
- POUR DES RECHARGEMENTS TRANSPARENTS ET UNE MISE À JOUR DE FRAGMENTS DE PAGE UNIQUEMENT.
- POUR LA VALIDATION DES FORMULAIRES SANS RECHARGEMENT DE LA PAGE.

LES FONCTIONS TYPE “BOITE DE DIALOGUE”

`alert(“salut”)` : affiche une popup JavaScript

`confirm(“continuer ?” )` : retourne true ou false

`prompt( )` : Retourne la saisie utilisateur



LES FONCTIONS TYPE “BOITE DE DIALOGUE”

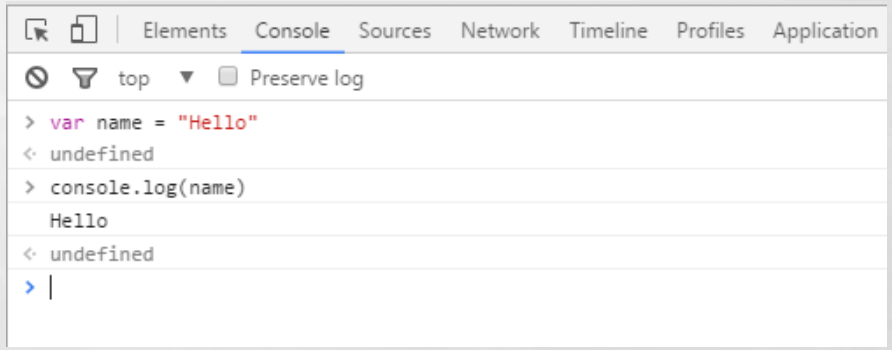
CONSOLE.LOG

CETTE FONCTION NOUS PERMET DE FAIRE DES TESTS, VOIR DE DEBUGGER NOS CODES DIRECTEMENT DANS LE NAVIGATEUR.

ESSAYONS UN SIMPLE CALCUL ARITHMÉTIQUE, ET OUI !

LES MATHS ÇA MARCHE AVEC JAVASCRIPT :

```
CONSOLE.LOG(5 * 2);
```

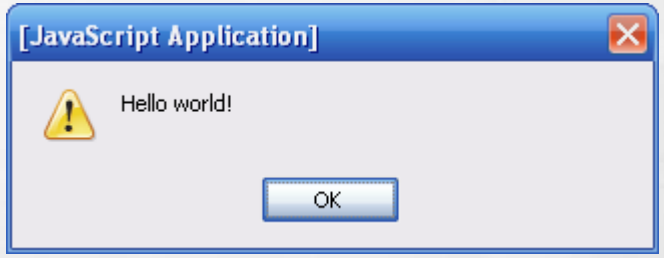


LES FONCTIONS TYPE “BOITE DE DIALOGUE”

LA FONCTION ALERT :

PERMET L'OUVERTURE D'UN POP-UP ET AFFICHE LE TEXTE ENTRE LES GUILLEMETS.

ALERT(“HELLO WORLD”);

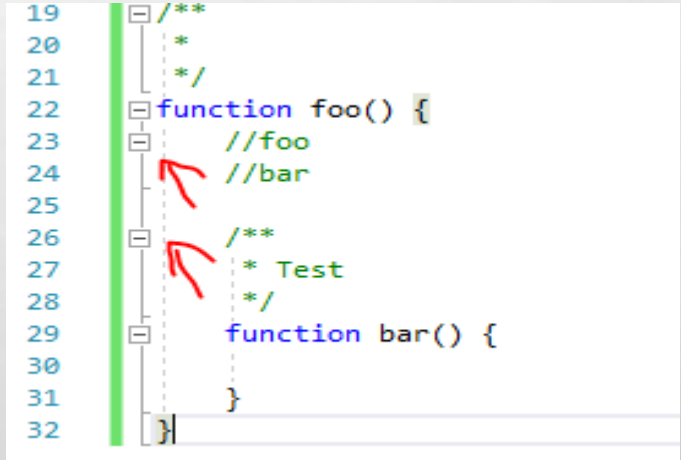


LES COMMENTAIRES :

UN COMMENTAIRE NE SE VOIT PAS DANS LE NAVIGATEUR C'EST UNE AIDE POUR LES DÉVELOPPEUR POUR SAVOIR CE QUE TU DOIS FAIRE.

// EXEMPLE DE COMMENTAIRE SUR UNE SEULE LIGNE

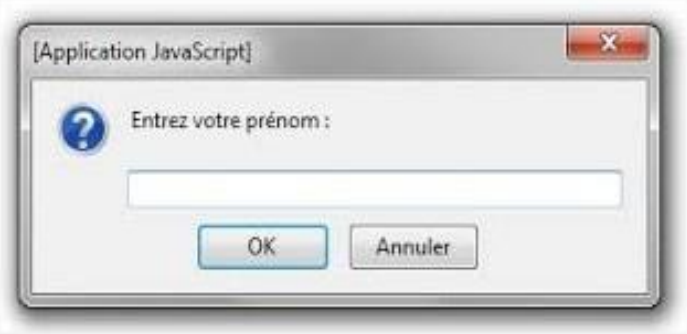
/* COMMENTAIRE SUR PLUSIEURS LIGNE */



LA FONCTION PROMPT()

Affiche une boîte de dialogue le paramètre message, une zone de saisie de texte avec le paramètre par défaut défaut et 2 boutons Valider et Annuler.

L'utilisateur peut saisir du texte. la méthode retourne le texte saisi si le visiteur clique sur Valider et retourne null si le visiteur clique sur Annuler ou appuye sur la touche Echap.



exemple :

```
var nombre = prompt("Entez un nombre supérieur à 18 :");
```

JAVASCRIPT – LES BASES

2.1 LES VARIABLES

UNE VARIABLE EST UN CONTENEUR QUI PEUT CONTENIR À PEU PRÈS TOUT. UN NOMBRE, UNE CHAINE DE CARACTÈRE, UN TABLEAU… ON SE RAPPELLERA QUE LES NOMBRE, CHAINE DE CARACTÈRES… SONT DES OBJETS.

IL EXISTE DEUX FAÇONS DE DÉCLARER UNE VARIABLE EN JS. VOUS POUVEZ DIRECTEMENT DONNER UN NOM À UNE VARIABLE ET UNE VALEUR OU PRÉCÉDER LE NOM DE LA VARIABLE PAR VAR COMME CECI :

```
MAVARIABLE = 5 ;

VAR MAVARIABLE = 5 ;
```

MÊME SI LES DEUX AFFECTENT UNE VALEUR DE 5 À LA VARIABLE MAVARIABLE, IL Y A UNE DIFFÉRENCE CONSIDÉRABLE ENTRE LES DEUX. EN PRÉCÉDANT LE NOM DE LA VARIABLE PAR LE MOT CLÉ VAR LA VARIABLE NE SERA DISPONIBLE QUE DANS UN ‘SCOPE’ DÉFINI. BIEN QUE LE LANGAGE LE PERMET, C’EST UNE TRÈS MAUVAIS PRATIQUE DE DÉCLARER UNE VARIABLE SANS LE PRÉCÉDER DE VAR. ON VA DONC TOUJOURS DÉCLARER NOS VARIABLES EN LES PRÉCÉDANT PAR VAR. ON Y REVIENDRA PLUS TARD POUR LE SCOPE MAIS POUR L’INSTANT RETENEZ JUSTE QU’IL Y A UNE DIFFÉRENCE ENTRE LES DEUX TYPES DE DÉCLARATION.

JAVASCRIPT – LES BASES

RÈGLES SUR LES VARIABLES (LE NOMMAGE DES VARIABLES)

Concernant le nom de nos variables, nous avons une grande liberté dans le nommage de celles-ci mais il y a quand même quelques règles à respecter :

Le nom d'une variable doit obligatoirement commencer par une lettre ou un underscore (_) et ne doit pas commencer par un chiffre ;
Le nom d'une variable ne doit contenir que des lettres, des chiffres et des underscores mais pas de caractères spéciaux ;
Le nom d'une variable ne doit pas contenir d'espace.

Nom de variable correct	Nom de variable incorrect	Raison
Variable	Nom de Variable	comporte des espaces
Nom_De_Variable	123Nom_De_Variable	commence par un chiffre
nom_de_variable	toto@mailcity.com	caractère spécial @
nom_de_variable_123	Nom-de-variable	signe - interdit
_707	transient	nom réservé

LES PORTÉES DE VARIABLES

RÈGLES SUR LES VARIABLES (LES PORTÉES)

JAVASCRIPT A DEUX TYPES DE PORTÉES : GLOBALE ET LOCALE. SI VOUS DÉCLAREZ UNE VARIABLE EN DEHORS D'UNE DÉFINITION DE FONCTION, IL S'AGIT D'UNE VARIABLE GLOBALE ET SA VALEUR EST ACCESSIBLE ET MODIFIABLE DANS TOUT LE PROGRAMME. SI VOUS DÉCLAREZ UNE VARIABLE AU SEIN D'UNE DÉFINITION DE FONCTION, IL S'AGIT D'UNE VARIABLE LOCALE. ELLE EST CRÉÉE ET DÉTRUITE CHAQUE FOIS QUE LA FONCTION EST EXÉCUTÉE ; EN DEHORS DE CETTE FONCTION, AUCUN CODE NE PEUT Y ACCÉDER.

```
<!DOCTYPE html>
<html>
    <head>
        <title>La portée en JavaScript</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>La portée</h1>
        <script>
            var x = 4;

            function locale(){
                var x = 7;
                return x;
            }

            alert('x globale : ' + x +
                  '\nx locale : ' + locale());
        </script>
    </body>
</html>
```

JAVASCRIPT – LES BASES

- ## 2 .2 LES CONSTANTES
- **Une constante est similaire à une variable au sens où c'est également un conteneur pour une valeur. Cependant, à la différence des variables, on ne va pas pouvoir modifier la valeur d'une constante.**

EXEMPLE:

```
const number = 42;

console.log(number);
```

- **Déclarer une constante en JavaScript**
  - **Pour créer ou déclarer une constante en JavaScript, nous allons utiliser le mot clef const.**
  - **On va pouvoir déclarer une constante exactement de la même façon qu'une variable à la différence qu'on va utiliser const à la place de var**

JAVASCRIPT – LES BASES - OPERATEURS

LES OPÉRATEURS ARITHMÉTIQUES

Voici la liste des différents opérateurs de calcul disponible en JavaScript.
Essayons un simple calcul arithmétique, et oui les maths ça marche avec JavaScript :

+	opérateur d'addition	Ajoute deux valeurs	`console.log(5+5);`
-	opérateur de soustraction	Soustrait deux valeurs	`console.log(10-5);`
*	opérateur de multiplication	Multiplie deux valeurs	`console.log(5*25);`
/	opérateur de division	Divise deux valeurs	`console.log(25/5);`
%	opérateur modulo	Retourne le reste de la division	`console.log(%2);`
=	opérateur d'affectation	Affecte une valeur à une variable	`var test = 20;`

LES OPÉRATEURS DE COMPARAISON

ILS COMPARENT DES NOMBRES OU DES EXPRESSIONS QUI RETOURNENT *TRUE* (VRAI) SI LA COMPARAISON RÉUSSIT OU *FALSE (FAUX)* SI ELLE ÉCHOUE. QUAND LES OPÉRANDES SONT DE TYPES DIFFÉRENTS, ILS SONT CONVERTIS AVANT COMPARAISON.

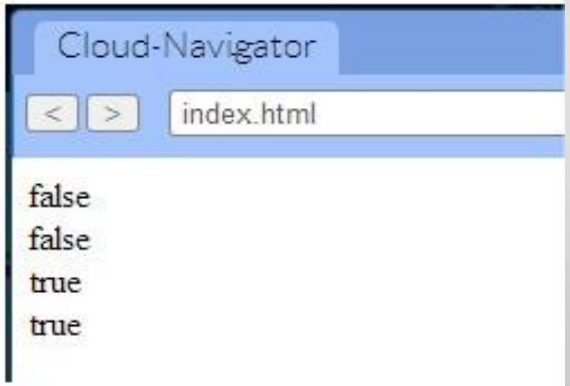
Les opérateurs:

==	égal
<	inférieur
>	supérieur
<=	inférieur ou égal
>=	supérieur ou égal
!=	différent

Exemple:

```
var chiffre1 = 4;
var chiffre2 = 10;
console.log(chiffre1 == chiffre2);
console.log(chiffre1 > chiffre2);
console.log(chiffre1 < chiffre2);
console.log(chiffre1 != chiffre2);
```

Résultat :



LES OPÉRATEURS LOGIQUES

Aussi appelés opérateurs booléens, ses opérateurs servent à vérifier deux ou plusieurs conditions.

Signe	Nom	Exemple	Signification
&&	et	(condition1) && (condition2)	condition1 et condition2
||	ou	(condition1) || (condition2)	condition1 ou condition2

Exemple :

```
if(a <= 18 && b == 20){
      //instruction
}
```

Exemple :

```
if(a == 18 || b == 20){
      //instruction
}
```

Exemple :

```
if((a == 18 && b == 20)|| (a ==
20 && bb == 18))
{
      //instruction
}
```

LA CONCATÉNATION

Le terme "concaténer" signifie joindre deux chaînes bout à bout pour n'en former qu'une seule.

En javascript, il suffit d'utiliser le caractère plus (+) ou d'utiliser la méthode **concat()**.
Dans les deux exemples ci-dessous, la variable chaîne 3 contient la chaîne "Bonjour tout le monde" :

```
var chaine1 = "Bonjour ";
var chaine2 = "tout le monde";
var chaine3 = chaine1+chaine2;
```

L'exemple ci-dessus est équivalent à l'exemple suivant :

```
var chaine1 = "Bonjour ";
var chaine2 = "tout le monde";
var chaine3 = chaine1.concat(chaine2);
```

JAVASCRIPT – LES BASES

2.3 LES CONDITIONS

QUAND VOUS DÉVELOPPEZ UNE APPLICATION VOUS AVEZ BESOINS PAR MOMENT DE SAVOIR SI UNE VARIABLE EXISTE OU SI ELLE CONTIENT QUELQUE CHOSE OU ENCORE SI ELLE EXISTE ET QU'ELLE CONTIENT BIEN UNE VALEUR…. EN FAIT VOUS ALLER CHERCHER À TESTER PLUSIEURS CAS POSSIBLE AVANT DE FAIRE UNE ACTION.

PAR EXEMPLE, SUR UN SITE DE VENTE DE EBOOKS EN LIGNE, VOUS NE LAISSEREZ L'UTILISATEUR TÉLÉCHARGER SON EBOOK QUE LORSQUE VOUS AVEZ VÉRIFIÉ SI VOUS AVEZ LES CORDONNÉES DE LA PERSONNE ET SI LE PAIEMENT EST BIEN VALIDÉ OU SI LE PRIX EST GRATUIT ET QUE VOUS N'AVEZ PAS BESOINS DES CORDONNÉES DE LA PERSONNE AVANT DE VALIDÉ VOTRE PROCESSUS.

C'EST LÀ QUE RENTRE EN JEUX LES CONDITIONS. ILS VONT JUSTEMENT NOUS PERMETTRE DE DIRE À NOTRE APPLICATION SI LE PRIX N'EST PAS GRATUIT IL FAUT QUE J'AI REÇU LE PAIEMENT AVANT DE LAISSER L'UTILISATEUR TÉLÉCHARGER SON LIVRE.

LES STRUCTURES CONDITIONNELLES - IF

LES STRUCTURES CONDITIONNELLES - IF & ELSE

L'instruction *if* dans sa forme basique ne permet de tester qu'une condition, or la plupart du temps on aimerait pouvoir choisir les instructions à exécuter **en cas de non réalisation de la condition**... L'expression *if... else* permet d'exécuter une autre série d'instruction en cas de non-réalisation de la condition.

```
var nombre = prompt("Entez un nombre inférieur à 18 :");
     if (nombre >= 18)
                {
                     console.log("Vous êtes majeur et vacciné !");
                }
     else
                {
                     console.log("Vous êtes mineur");
```

LES STRUCTURES CONDITIONNELLES - IF & ELSE IF & ELSE

Vous pouvez ajouter des conditions supplémentaires à l'aide de else if, voici la synthaxe :

```
if ( première condition ) {
}
else if ( seconde condition ) {
}
else{
}
```

Noté que "`else if`" est sur deux mots. Si vous utilisez celui-ci en un seul mot, vos conditions ne marchera pas.

La condition if (si) ;
La condition if… else (si… sinon) ;
La condition if… elseif… else (si… sinon si… sinon).

LES STRUCTURES CONDITIONNELLES - SWITCH CASE

L'instruction *switch* permet de faire plusieurs tests de valeurs sur le contenu d'une même variable. Ce branchement conditionnel simplifie beaucoup le test de plusieurs valeurs d'une variable, car cette opération aurait été compliquée (mais possible) avec des *if* imbriqués. Sa syntaxe est la suivante :

```
switch        (Variable)        {
      case                Valeur1:
      Liste       d'instructions;
      break;
      case                Valeur2:
      Liste       d'instructions;
      break;
      default:
      Liste       d'instructions;
      break; }
```

Les parenthèses qui suivent le mot clé *switch* indiquent une expression dont la valeur est testée successivement par chacun des *case*. Lorsque l'expression testée est égale à une des valeurs suivant un *case*, la liste d'instruction qui suit celui-ci est exécuté. Le mot clé *break* indique la sortie de la structure conditionnelle. Le mot clé *default* précède la liste d'instructions qui sera exécutée si l'expression n'est jamais égale à une des valeurs.

JAVASCRIPT – LES BASES

2.3.5 LES TERNAIRES

LES TERNAIRES SONT DES CONDITIONS TRÈS RAPIDE À ÉCRIRE MAIS QUI SONT DIFFICILE À LIRE POUR CEUX QUI NE SONT PAS HABITUER.

DE MANIÈRE GÉNÉRALE ON ÉVITERA DE LES UTILISER MÊME S'ILS SONT TRÈS PRATIQUES PARCE QUE VOTRE APPLICATION ÉVOLUERA ET SERA MAINTENU PAR D'AUTRES DÉVELOPPEURS QUI NE SONT PAS FORCÉMENT HABITUÉ À LEUR LECTURE.

```
VAR CONDITION = TRUE ;
VAR PROCESS = '';

PROCESS = CONDITION ? 'OK !' :  'FAILED !' ;

LA MÊME CHOSE ÉCRITE AVEC UN IF/ELSE:

IF ( CONDITION ) {
  PROCESS = 'OK!';
} ELSE {
  PROCESS = 'FAILED !';
}
```

LES CARACTÈRES D'ÉCHAPPEMENTS

N'oubliez pas Une chaîne de caractère est délimitée par des " ou des '.
Pour y insérer des " des ' ou des \ utiliser le caractère d'échappement :

caractère	codage	Exemple	La chaine résultante
'	\'	l'histoire	var a = 'l\'histoire';
"	\"	"bonjour" dit-il	var a = "\"bonjour\" dit-il";
\	\\		

Pour insérer des caractères spéciaux dans nos chaînes de caractères :

\b	retour en arrière	\r	retour chariot
\f	saut de page	\t	tabulation
\n	saut de ligne		

JAVASCRIPT – LES BASES

2.4 OU METTRE LE CODE JAVASCRIPT ?

LE JAVASCRIPT PEUT SE TROUVER DIRECTEMENT DANS VOTRE FICHIER HTML OU DANS UN FICHIER SÉPARÉ AVEC UNE EXTENSION .JS
POUR ÉCRIRE DU JAVASCRIPT DANS VOTRE FICHIER HTML VOUS DEVEZ LE PLACER DANS DES BALISES SCRIPT COMME CECI :
<SCRIPT></SCRIPT>

POUR ÉCRIRE DU JAVASCRIPT DANS UN FICHIER .JS POUR BIEN ARCHITECTURER VOTRE PROJET ET SÉPARÉ LE JAVASCRIPT DE VOTRE HTML VOUS DEVEZ RENSEIGNER DANS DES BALISES SCRIPT VIDE LE CHEMIN DEPUIS VOTRE FICHIER HTML À VOTRE FICHIER .JS AVEC L'ATTRIBUT SRC COMME CECI :

// EN SUPPOSANT QUE MON FICHIER JAVASCRIPT SE TROUVE DANS UN DOSSIER JS À LA RACINE DE MON PROJET ET QUE MON FICHIER HTML SE TROUVE AUSSI À LA RACINE DE MON PROJET

<SCRIPT SRC="JS/MONFICHIER.JS"></SCRIPT>

LES BOUCLES – WHILE (TANT QUE)

LES BOUCLES NOUS PERMETTENT DE RÉPÉTER UN CODE TANT QU'UNE CONDITION EST VRAIE OU QU'ON DISE À LA BOUCLE DE S'ARRÊTER. LA BOUCLE ARRÊTERA SON EXÉCUTION AUSSITÔT QUE LA CONDITION N'EST PLUS RESPECTÉE.

EXPLICATION

DANS L'EXEMPLE, LA VARIABLE INDEX EST DÉCLARÉ AVEC UNE VALEUR DE 0. A CHAQUE PASSAGE DANS LA BOUCLE ELLE EST INCRÉMENTER ET AFFICHE UN ALERT. TANT QUE LA VALEUR DE INDEX EST INFÉRIEUR À 3 UN ALERT S'AFFICHERA À CHAQUE PASSAGE DANS LA BOUCLE. LA FONCTION ALERT SERA DONC EXÉCUTER 3 FOIS.

```
WHILE(VRAI) {
  ALERT('J\'AIME LE JAVASCRIPT') ;
  ALERT('MAIS J\'AIME PAS LE JAVA') ;
}
EXEMPLE :
VAR INDEX = 0;
WHILE(INDEX < 3) {
  INDEX++; // INCRÉMENTATION
  ALERT('J\'AIME LE JAVASCRIPT') ;
}
```

LES BOUCLES - FOR

SIMILAIRE À LA BOUCLE WHILE, TOUTE LA PARTIE ‘LOGIQUE’ DE LA BOUCLE SE FAIT À UN ENDROIT CE QUI REND LE CODE PLUS FACILE À LIRE.

```
FOR(VAR INDEX=0; INDEX <3; INDEX++) {
  ALERT('J\'AIME LE JAVASCRIPT');
}
```

LES BOUCLES - LE DO WHILE (FAIT TANT QUE)

LA DIFFÉRENCE ENTRE LE WHILE ET LE DO WHILE EST UN PEU SUBTILE. DANS LA BOUCLE WHILE SI VOUS DÉCLAREZ LA VARIABLE INDEX AVEC UNE VALEUR SUPÉRIEUR À 3 LA BOUCLE NE SERA JAMAIS EXÉCUTER ALORS QUE AVEC LA BOUCLE DO WHILE LA BOUCLE SERA EXÉCUTÉE AU MOINS UNE FOIS ÉTANT DONNÉ QUE LA CONDITION N'EST VÉRIFIÉE QU'À LA FIN DE LA BOUCLE.

```
VAR INDEX = 0;
DO {
  INDEX++; // INCRÉMENTATION
  ALERT('J\'AIME LE JAVASCRIPT') ;
}
```

JAVASCRIPT - LES TABLEAUX

Un tableau, en Javascript, est une variable pouvant contenir plusieurs données indépendantes, indexées par un numéro, appelé **indice**.
L'indice d'un tableau est ainsi l'élément permettant d'accéder aux données qui y sont stockées.

Indice	0	1	2	3
Donnée	donnée 1	donnée 2	donnée 3	donnée 4

Pour créer un tableau en JavaScript :

```
// Création d'un tableau
var fruits = ["Bananes", "Oranges", "Fraises", "Citrons"];
```

JAVASCRIPT - LES TABLEAUX

Afficher les valeurs d'un tableau en JavaScript est très simple :

```
// Le tableau
var fruits = ["Bananes", "Oranges", "Fraises", "Citrons"];

// Afficher un indice du tableau
console.log(fruits[0]);  // Affiche le 1er élément du tableau (Bananes)
console.log(fruits[1]);  // Affiche le 2ème élément du tableau (Oranges)
console.log(fruits[2]);  // Affiche le 3ème élément du tableau (Fraises)
console.log(fruits[3]);  // Affiche le 4ème élément du tableau (Citrons)
```

JAVASCRIPT - LES TABLEAUX - EXEMPLES

VAR PRENOMS = [MOUSSA', 'NOÉMIE', 'ROBERT', 'JEAN'];

IL EXISTE DES FONCTIONS NATIVES DU JAVASCRIPT QUI NOUS PERMETTENT DE PARCOURIR CES TABLEAUX POUR AJOUTER, RÉCUPÉRER, CHANGER OU ENLEVER DES VALEURS OU DES CLÉS ENTRE AUTRE. ON UTILISERA SOUVENT LES BOUCLES FOR OU WHILE POUR LES PARCOURIR.

PAR EXEMPLE, PRENONS LE TABLEAU PRENOMS. SI JE VEUX CHANGER LA VALEUR ROBERT, IL FAUT QUE JE PARCOURE LE TABLEAU ET QUAND JE RENCONTRE LA VALEUR ROBERT, JE REMPLACE CETTE VALEUR PAR AUTRE CHOSE COMME DUJARDIN.

POUR NOTRE BOUCLE FOR OU WHILE, IL NOUS FAUDRA UNE CONDITION POUR SORTIR DE LA BOUCLE. CETTE CONDITION EST LA TAILLE DU TABLEAU, CÀD LE NOMBRE D'ÉLÉMENTS QU'IL CONTIENT. DANS NOTRE CAS ON VOIT BIEN QUE NOTRE TABLEAU CONTIENT 4 ÉLÉMENTS. MAIS CE TABLEAU POURRAIT CHANGER AU COURS DE SA VIE ET SI VOUS SORTEZ DE VOTRE BOUCLE AU BOUT DE 4 ÉLÉMENTS, VOTRE CODE NE SERA PLUS BON. DONC POUR RÉCUPÉRER LA TAILLE DU TABLEAU NOUS UTILISERONS CECI : PRENOMS.LENGTH

DEUXIÈME PROBLÈME, COMMENT RÉCUPÉRER UNE VALEUR PRÉCISE D'UN TABLEAU ET PLUS ENCORE, COMMENT LA CHANGER !!! ESSAYEZ-LE !

JAVASCRIPT – LES OBJETS

UN OBJET POSSÈDE UNE STRUCTURE SPÉCIFIQUE SÉPARÉ EN 3 PARTIES. LE CONSTRUCTEUR, LES PROPRIÉTÉS ET LES MÉTHODES. LE CONSTRUCTEUR EST UN BOUT DE CODE QUI EST EXÉCUTÉ À CHAQUE FOIS QU'ON UTILISE UN NOUVEL OBJET. LES PROPRIÉTÉS SONT LES VARIABLES AU SEIN D'UN OBJET ET LES MÉTHODES SONT LES FONCTIONS CONTENUES DANS L'OBJET QU'ON UTILISERA POUR MODIFIER L'OBJET.

LE LANGAGE DISPOSE D'OBJETS NATIFS QUI ONT DONC DÉJÀ LEURS PROPRIÉTÉS ET MÉTHODES. CES OBJETS NOUS LES AVONS DÉJÀ UTILISÉS.

EXEMPLE :

```
VAR CHAINE = 'CECI EST UN OBJET STRING';
ALERT(CHAINE.LENGTH);
ALERT(CHAINE.TOUPPERCASE());
```

COMME VOUS LE VOYEZ J'UTILISE ICI LA PROPRIÉTÉ LENGTH DE L'OBJET STRING ET SA MÉTHODE TOUPPERCASE() QUI TRANSFORMERA VOTRE CHAINE DE CARACTÈRE EN MAJUSCULE.

POUR CRÉER UN OBJET ON FERA :

```
VAR PERSONNES = {
  NOM: MOUSSA',
  MOYENNE: '15',
  ADMIS: 'OUI',
};
```

LA SYNTAXE RESSEMBLE UN PEU À LA SYNTAXE DE TABLEAU MAIS IL Y A QUAND MÊME QUELQUES DIFFÉRENCES.

POUR RÉCUPÉRER LA VALEUR DU NOM ON FERA AINSI :

```
VAR PERSONNES = {
  NOM: MOUSSA',
  MOYENNE: ‘15',
  ADMIS: 'OUI',
};

ALERT(PERSONNES.NOM);
```

ET POUR AFFECTER UNE VALEUR ON FERA :

```
PERSONNES.NOM = 'ALFRED';
```

NOUS AVONS VU DANS CE CHAPITRE LES BASES DU JAVASCRIPT, COMMENT DÉCLARER LES VARIABLES, FAIRE DES BOUCLES SUR DES TABLEAUX OU ENCORE FAIRE DES FONCTIONS ET LES UTILISER. MAIS LE JAVASCRIPT NOUS PERMETS DE FAIRE BIEN PLUS QUE DE LA PROGRAMMATION ALGORITHMIQUE, ELLE NOUS PERMET ENTRE AUTRE DE MODIFIER LE DOM, ET C'EST CE QUE NOUS ALLONS VOIR DANS LE CHAPITRE SUIVANT.

JAVASCRIPT – LES FONCTIONS

ON APPELLE FONCTION UN SOUS-PROGRAMME QUI PERMET D'EFFECTUER UN ENSEMBLE D'INSTRUCTIONS PAR SIMPLE APPEL DE LA FONCTION DANS LE CORPS DU PROGRAMME PRINCIPAL. CETTE NOTION DE SOUS-PROGRAMME EST GÉNÉRALEMENT APPELÉE FONCTION DANS LES LANGAGES AUTRES QUE LE JAVASCRIPT (TOUTEFOIS LEUR SYNTAXE EST GÉNÉRALEMENT DIFFÉRENTE...). LES FONCTIONS PERMETTENT D'EXÉCUTER DANS PLUSIEURS PARTIES DU PROGRAMME UNE SÉRIE D'INSTRUCTIONS, CELA PERMET UNE SIMPLICITÉ DU CODE ET DONC UNE TAILLE DE PROGRAMME MINIMALE.

- UNE FONCTION C'EST UN BLOC DE CODE QU'ON POURRA EXÉCUTER À LA DEMANDE. CE SONT UN PEU COMME DES VARIABLES. CE SONT DES CONTENEURS.

CONTRAIREMENT AUX VARIABLES ILS POURRONT CONTENIR PLUSIEURS CHOSES ET ON POURRA MÊME LUI DIRE D'EXÉCUTER UN CODE DIFFÉREMMENT AVEC UN OU PLUSIEURS PARAMÈTRES.

Passons directement à un exemple.

```
function afficheHelloWorld()
{
   var hello = 'hello world';
   alert(hello);
}
```

Détaillons un peu ce bout de code. Tout d'abord pour créer une fonction il faut le mot clé function, le nom de la fonction, des parenthèses et des accolades.

On verra par la suite que le JavaScript est plus flexible que ça mais pour le moment on s'en tiendra à cette syntaxe.

A l'exécution de ce code dans la console de votre navigateur, vous voyez… rien !

Vous vous dites surement que cette fonction exécutera toujours la même chose à chaque fois et qu'il n'y a pas moyen de changer cette fonction et vous avez raison. Mais vous pouvez aussi lui passer des paramètres…. Modifions notre fonction pour illustrer ce fameux paramètre :

```
function afficheHelloWorld(texte)
{
   alert(texte);
}
afficheHelloWorld('Hello world');
afficheHelloWorld('Hello Moussa');
```

Désormais à chaque appel à la fonction on lui donne le texte à afficher.

JAVASCRIPT – LES FONCTIONS AVEC PARAMETRES

Considérons qu'une fonction est comme une machine à laver. Celle-ci est composé de trois niveau :

- Le premier niveau est une entrée,insertion de vêtements sales.
- Le deuxième niveau est la fonction en elle même de la machine, laver les vêtements.
- Le troisième niveau est la sortie des vêtements propres.

L'équivalent des entrées dans une fonction s'appelle "les paramètres" d'une fonction.

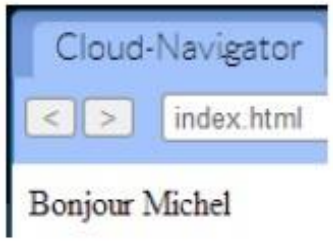
Exemple :

```
var direBonjour = function(nom) {
    console.log("Bonjour " + nom);
};
```

Appel de la fonction :

```
direBonjour
("Michel");
```

Résultat :



JAVASCRIPT – LES FONCTIONS AVEC PARAMETRES

Sans fonctions !

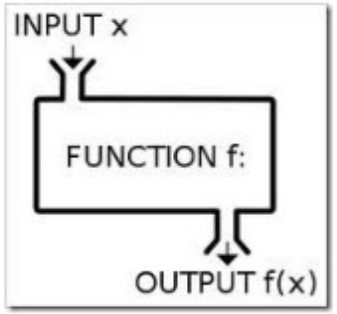
```
var nom;
var age;

// Présentation de Michel
nom = "Michel";
age = 24;
console.log(nom + " a " + age + "
ans");

// Présentation de Nicolas

nom = "Nicolas";
age = 18;
console.log(nom + " a " + age + "
ans");
```

```
// Présentation de Jennifer
nom = "Jennifer";
age = 32;
console.log(nom + " a " + age + "
ans");
```



JAVASCRIPT – LES FONCTIONS AVEC PARAMETRES

Avec fonction !

```
// Création de la fonction presenterUnePersonne ; Cette fonction attend 2
paramètres : le nom, et l'âge de la personne

var presenterUnePersonne = function(nom, age) {
    console.log(nom + " a " + age + " ans");
};

// Il nous reste plus qu'à appeler la fonction pour chacune des personnes :

presenterUnePersonne("Michel", 24);
presenterUnePersonne("Nicolas", 18);
presenterUnePersonne("Jennifer", 32);
```

JAVASCRIPT – LES FONCTIONS AVEC PARAMETRES

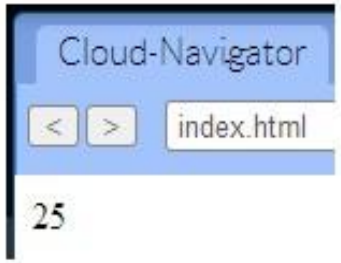
Nous avons vu l'entrée d'une fonction (paramètre), le traitement, passons à la sortie.
Le **return** permet de terminer la fonction et ainsi retourner une valeur que vous avez traité dans votre fonction.

Exemple :

```
var carre = function(chiffre) {
    return chiffre * chiffre;
};

console.log(carre(5));
```

Résultat :



JAVASCRIPT - LES OBJETS PRÉDÉFINIS

JAVASCRIPT POSSÈDE NATIVEMENT DES DONNÉES (PROPRIÉTÉS) ET FONCTIONS (MÉTHODES) UTILISABLES DIRECTEMENT PAR LES PROGRAMMEURS. CEUX-CI SONT DIVISÉ EN 14 CATÉGORIES :

`window` - gestion de la fenêtre du navigateur

`document` - permet de travailler dans la page

`navigator` - méthodes lié au navigateur

`Array` (tableaux) - manipulation des tableaux

`Boolean` - gérer les booléans

`Date` - permet de manipuler les dates

`Error` - Gestion des erreurs

`Function` - gestion des fonctions

`Image` - travailler avec les images

`Math` - méthodes mathématiques

`Number` - méthodes lié au nombre

`Object` - création d'objet (POO)

`RegExp` - expression réguliaire

`String` - gestion des chaînes de caractères

JAVASCRIPT - LES OBJETS PRÉDÉFINIS

`uneChaine.toLowerCase()` : passe toute la chaîne en minuscule.

`uneChaine.toUpperCase()` : passe toute la chaîne en majuscule.

`uneChaine.big()` : formate la chaîne avec la balise <big>.

`uneChaine.small()` : formate la chaîne avec la balise <small>.

`uneChaine.link(url)` : formate la chaîne pour en faire un lien.

`uneChaine.anchor(name)` : formate la chaîne pour en faire une ancre.

LES OBJETS PRÉDÉFINIS TYPE STRING

Exemple :

```
var chaine = "en minuscule";
console.log(chaine.toUpperCase());
```

LES OBJETS PRÉDÉFINIS TYPE ARRAY (TABLEAU)

`Tableau.pop()` : enlève le dernier élément du tableau.

`Tableau.shift()` : enlève le premier élément du tableau.

`Tableau.join(séparateur)` : assemble un tableau pour en faire une chaîne.

`Tableau.reverse()` : inverse l'ordre des éléments.

`Tableau.sort([fonction])` : trie les éléments du tableau.

`Tableau.concat(tableau1, ...)` : concatène des tableaux.

LES OBJETS PRÉDÉFINIS TYPE ARRAY (TABLEAU)

Exemple :

```
var tableau1 = ["salut","tu"];
var tableau2 = ["vas","bien"];
var tableau3 = tableau1.concat(tableau2);
console.log(tableau3);
```

LES OBJETS PRÉDÉFINIS TYPE DATE

```
var uneDate = new Date();
```

Les méthodes :

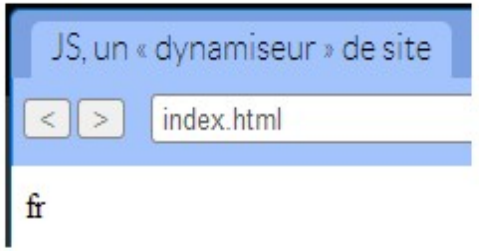
```
uneDate.getDate() : retourne le jour du mois (1 à 31).
uneDate.getMonth() : retourne le mois (0 à 11).
uneDate.getFullYear() : retourne l'année depuis 1900
uneDate.getHours() : retourne l'heure (0 à 23).
uneDate.getMinutes() : retourne les minutes (0 à 59)
uneDate.getSeconds() : retourne les secondes (0 à 59)
```

LES OBJETS PRÉDÉFINIS TYPE NAVIGATOR

Exemple :

Il est possible de trouver la langue par défaut du navigateur avec la propriété “language”.

```
console.log(window.navigator.language);
```



LA MANIPULATION DU DOM

NOUS POUVONS AUSSI ACCÉDER ET MODIFIER LE DOM (DOCUMENT OBJECT MODEL). POUR CE FAIRE PLUSIEURS MÉTHODES EXISTENT AU SEIN MÊME DU LANGAGE. TOUTEFOIS À FAIRE ATTENTION, CERTAINES MÉTHODES NE MARCHERONT PAS SUR TOUS LES NAVIGATEURS, NOTAMMENT LES VIEUX NAVIGATEURS, TRÈS PARTICULIÈREMENT INTERNET EXPLORER.

BIEN QUE TOUS LES NAVIGATEURS N'INTERPRÈTENT PAS LE JAVASCRIPT (COMME LE JAVASCRIPT EST INTERPRÉTER PAR LE NAVIGATEUR) DE LA MÊME MANIÈRE, IL Y A ASSEZ PEU DE DIFFÉRENCES ENTRE LES NAVIGATEURS. VOUS DEVEZ DONC VÉRIFIER SI LES MÉTHODES QUE VOUS UTILISÉ SONT COMPATIBLE AVEC VOTRE SCOPE NAVIGATEUR.

ACCÉDER AUX ÉLÉMENTS DU DOM

LE GETELEMENTBYID()

POUR ACCÉDER AUX ÉLÉMENTS DU DOM IL EXISTE ENTRE AUTRE LA MÉTHODE GETELEMENTBYID('ID') QUI PREND EN PARAMÈTRE L'ID QUE VOUS RECHERCHEZ. COMME SON NOM L'INDIQUE, IL PERMET DE RÉCUPÉRER UN 'ID'.

CETTE MÉTHODE VOUS RETOURNE COMME VOUS POUVEZ LE VOIR DANS VOTRE CONSOLE, LE DIV CONTENANT L'ID SPÉCIFIÉ. VOUS REMARQUEREZ QUE J'AI PRÉFIXÉ CETTE MÉTHODE DU MOT CLÉ 'DOCUMENT'. CECI EST PARCE QUE DOCUMENT REPRÉSENTE ICI LE NŒUD DE PREMIER NIVEAU DE VOTRE DOM. C'EST LA BALISE HTML, VOTRE PAGE WEB.

IL EXISTE UN AUTRE MOT CLÉ TRÈS IMPORTANT, C'EST LE MOT CLÉ 'WINDOW' QUI LUI REPRÉSENTE LA FENÊTRE DU NAVIGATEUR.

EXEMPLE :

```
<!DOCTYPE HTML>
<HTML LANG="EN">
 <HEAD>
  <META CHARSET="UTF-8">
  <TITLE>JAVASCRIPT COURSE - IKNSA</TITLE>
 </HEAD>
 <DIV ID="TEST"></DIV>
 <BODY>
  <!-- PAGE CONTENT -->
  <SCRIPT>
    CONSOLE.LOG(DOCUMENT.GETELEMENTBYID('TEST'));
  </SCRIPT>
 </BODY>
</HTML>
```

ACCÉDER AUX ÉLÉMENTS DU DOM

GETELEMENTSBYTAGNAME ET GETELEMENTSBYCLASSNAME

CES DEUX MÉTHODES VOUS PERMETTENT DE RÉCUPÉRER LES ÉLÉMENTS AVEC LES MÊME BALISES ET CLASSE RESPECTIVEMENT. VOUS NOTEREZ QUE ICI IL Y A UN ‘S’ À ‘ELEMENT’. C’EST PARCE QUE CONTRAIREMENT À LA MÉTHODE PRÉCÉDENTE QUI VOUS RETOURNE UN OBJET, CES DEUX MÉTHODE VOUS RETOURNE UN TABLEAU DES OBJETS RECHERCHÉ.

DEPUIS LA CONSOLE DE VOTRE NAVIGATEUR VOUS POUVEZ AUSSI VOIR LES NOMBREUSES MÉTHODES ET PROPRIÉTÉ AUXQUELLES VOUS POUVEZ Y ACCÉDER. NOUS ALLONS EN VOIR QUELQUES-UNS POUR VOIR COMMENT ILS FONCTIONNENT.

EXEMPLE :

```
<!DOCTYPE HTML>
<HTML LANG="EN">
 <HEAD>
  <META CHARSET="UTF-8">
  <TITLE>JAVASCRIPT COURSE - IKNSA</TITLE>
 </HEAD>
 <DIV CLASS="TEST"></DIV>
 <DIV CLASS="TEST"></DIV>
 <DIV CLASS="TEST"></DIV>
 <BODY>
  <!-- PAGE CONTENT -->
  <SCRIPT>
    CONSOLE.LOG(DOCUMENT.GETELEMENTSBYTAGNAME('DIV'));
    CONSOLE.LOG(DOCUMENT.GETELEMENTSBYCLASSNAME('TEST'));
  </SCRIPT>
 </BODY>
</HTML>
```

ACCÉDER AUX ÉLÉMENTS DU DOM

CES DEUX MÉTHODES PRENNENT TOUS LE DEUX EN PARAMÈTRE UNE CHAINE DE CARACTÈRE SEMBLABLE À VOTRE SÉLECTEUR CSS.

HTTPS://GITHUB.COM/IKNSA-FORMATION/JAVASCRIPT/TREE/MODIFICATION_DOM

EXEMPLE :

```
<!DOCTYPE HTML>
<HTML LANG="EN">
 <HEAD>
  <META CHARSET="UTF-8">
  <TITLE>JAVASCRIPT COURSE - IKNSA</TITLE>
 </HEAD>
 <BODY>
  <DIV ID="TEST">
    <P CLASS="PARAGRAPH">
      <A HREF="#" CLASS="LIEN">CECI EST UN LIEN</A>
    </P>
  </DIV>
  <script>
    console.log(document.querySelector('#TEST .PARAGRAPH .LIEN'));
  </script>
 </BODY>
</HTML>
```

ACCÉDER AUX ÉLÉMENTS DU DOM

NOUS SAVONS MAINTENANT RÉCUPÉRER UN OBJET DU DOM MAIS IL SERAIT ENCORE MIEUX DE POUVOIR LE MODIFIER. C'EST CE QUE NOUS POUVONS FAIRE AVEC INNERHTML.

ANALYSONS L'EXEMPLE PRÉCÉDENT. NOUS AVONS UNE DIV AVEC L'ID TEST, UNE BALISE P ET DU TEXTE DEDANS. NOTRE PREMIÈRE LIGNE DE CODE JAVASCRIPT AFFICHE LE CONTENU HTML DE NOTRE DIV DANS LA CONSOLE ALORS QUE LA DEUXIÈME LIGNE AFFECTE UNE NOUVELLE VALEUR HTML À NOTRE DIV TEST.

```
<HTML LANG="EN">
 <HEAD>
  <META CHARSET="UTF-8">
  <TITLE>JAVASCRIPT COURSE - IKNSA</TITLE>
 </HEAD>
 <DIV ID="TEST">
   <P>
     LOREM IPSUM DOLOR SIT AMET.
   </P>
 </DIV>
 <BODY>
  <SCRIPT>
    CONSOLE.LOG(DOCUMENT.GETELEMENTBYID('TEST').INNERHTML);
    DOCUMENT.GETELEMENTBYID('TEST').INNERHTML = '<A HREF="#"><STRONG>MODIFICATION AVEC LE
JAVASCRIPT</STRONG></A>';
    CONSOLE.LOG(DOCUMENT.GETELEMENTBYID('TEST').INNERHTML);
  </SCRIPT>
 </BODY>
</HTML>
```

ACCÉDER AUX ÉLÉMENTS DU DOM

SACHEZ QUE NOUS POUVONS AUSSI ACCÉDER AUX ATTRIBUTS DES ÉLÉMENTS DU DOM AVEC GETATTRIBUTES OU RAJOUTER/ENLEVER DES CLASSES SUR UN ÉLÉMENT DU DOM.

COMME NOUS POUVONS LE VOIR ICI, AU DÉBUT NOUS AVONS BIEN UNE CLASS ‘ALFRED’ DANS NOTRE DIV. NOUS RAJOUTONS ENSUITE UNE CLASSE ‘NOUVELLE’ POUR FINALEMENT ENLEVER LA CLASSE ‘ALFRED’. COMME TOUT EST LOGGÉ EN CONSOLE NOUS POUVONS FACILEMENT SUIVRE LE PROCESSUS.

```
<!DOCTYPE HTML>
<HTML LANG="EN">
 <HEAD>
  <META CHARSET="UTF-8">
  <TITLE>JAVASCRIPT COURSE - IKNSA</TITLE>
 </HEAD>
 <BODY>
 <DIV ID="TEST" CLASS="ALFRED"></DIV>
  <SCRIPT>
     CONSOLE.LOG(DOCUMENT.GETELEMENTBYID('TEST').CLASSLIST);
     DOCUMENT.GETELEMENTBYID('TEST').CLASSLIST.ADD('NOUVELLE');
     CONSOLE.LOG(DOCUMENT.GETELEMENTBYID('TEST').CLASSLIST);
     DOCUMENT.GETELEMENTBYID('TEST').CLASSLIST.REMOVE('ALFRED');
     CONSOLE.LOG(DOCUMENT.GETELEMENTBYID('TEST').CLASSLIST);
  </SCRIPT>
 </BODY>
</HTML>
```

LA MANIPULATION DU DOM

- **GETELEMENTBYID()** : RECHERCHE UN ÉLÉMENT D'APRÈS SON IDENTIFIANT (ATTRIBUT ID),
- **GETELEMENTSBYTAGNAME()** : RECHERCHE DES ÉLÉMENTS D'APRÈS LEUR TYPE (BALISE),
- **GETELEMENTSBYCLASSNAME()** : RECHERCHE DES ÉLÉMENTS D'APRÈS LEUR CLASSE (ATTRIBUT CLASS)
- **QUERYSELECTOR()** : RETOURNE LE PREMIER ÉLÉMENT TROUVÉ SATISFAISANT AU SÉLECTEUR (TYPE DE RETOUR : ELEMENT), OU NULL SI AUCUN OBJET CORRESPONDANT N'EST TROUVÉ.
- **QUERYSELECTORALL()** : RETOURNE TOUS LES ÉLÉMENTS SATISFAISANT AU SÉLECTEUR, DANS L'ORDRE DANS LEQUEL ILS APPARAISSENT DANS L'ARBRE DU DOCUMENT (TYPE DE RETOUR : NODELIST), OU UN TABLEAU NODELIST VIDE SI RIEN N'EST TROUVÉ.

DOM:getElementById & innerHTML

Pour ce qui concerne les “id”, il faut utiliser la méthode document.getElementById() qui permet d’allez chercher un l’ID que vous désirez dans le DOM.
Pour ce faire nous allons ajouter une fonction bien pratique qui permet d’ajouter, supprimer du HTML dans notre DOM.

```
<div id="ma_div">
Bonjour le monde !
</div>
```

```
document.getElementById(“ma_div”).innerHTML(“mon contenu”);
```

mon contenu

DOM: getElementsByClass

Pour ce qui concerne les “`class`”, il existe depuis peu la méthode `GetElementsByClass`, celle-ci permet de sélectionner une classe afin de faire un traitement sur plusieurs valeurs à l’aide d’un tableau. Nous allons utiliser une méthode très pratique appelé “`style`” permettant d’intégrer du style sur notre page.

```html
<div class="ma_div">
Bonjour le monde 1
</div>
<div class="ma_div">
bonjour le monde 2
</div>
```

```javascript
var ma_div = document.getElementsByClassName("ma_div");

ma_div[0].style.backgroundColor = "yellow";
ma_div[1].style.fontWeight = "bold";
```

Bonjour le monde !

bonjour le monde !

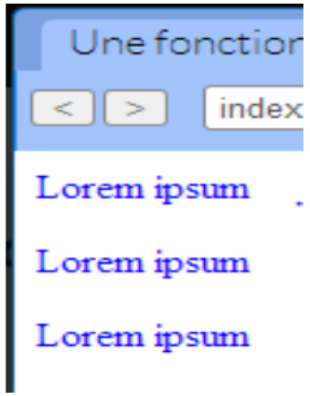
DOM: getElementByTagName

`getElementsByTagName` retourne un tableau, il faut donc utiliser une boucle afin de traiter l'ensemble.

```
<p>Lorem ipsum</p>
<p>Lorem ipsum</p>
<p>Lorem ipsum</p>
```

```
var paragraphes = document.getElementsByTagName("p");

for (var compteur = 0; compteur < paragraphes.length;
compteur++) {
     paragraphes[compteur].style.color = "blue";
}
```



DOM: getElementsByName

Retourne un tableau (Array) d'objets HTML ayant nom défini dans la propriété `name` de la balise de l'objet.
Ne pas confondre la propriété id et la propriété `name`. Pour trouver un objet à partir de son id, utilisez `getElementById()`.

```
<form>

<input type="text" name="login"
value="mon login"/>

</form>
```

```
var a =document.getElementsByName("login")[0];
alert(a.value);
```

Les évenements

- Les évènements sont des actions qui se produisent et auxquelles on va pouvoir répondre en exécutant un code.
  - Par exemple, on va pouvoir afficher ou cacher du texte suite à un clic d'un utilisateur sur un élément, on change la taille d'un texte lors du passage de la souris d'un utilisateur sur un élément.

- Les évènements et leur prise en charge sont l'un des mécanismes principaux du JavaScript qui vont nous permettre d'ajouter un vrai dynamisme à nos pages Web.

Les évenements

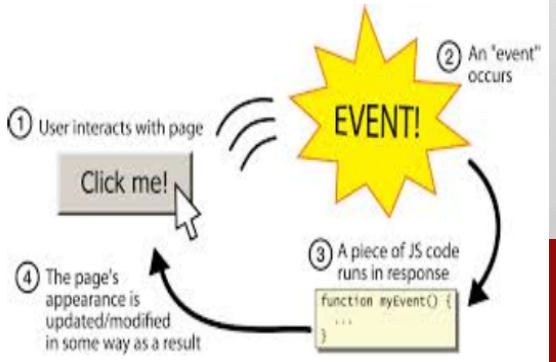
- Il existe de nombreux évènements répertoriés en JavaScript (plus d'une centaine). Ces évènements peuvent être très différents les uns des autres :
- Le chargement du document est un évènement :
  - Un clic sur un bouton effectué par un utilisateur est un évènement ;
  - Le survol d'un élément par la souris d'un utilisateur est un évènement ;
  - Etc.

Les évenements

- Ces attributs HTML de « type évènement » possèdent souvent le nom de l'évènement qu'ils doivent écouter et gérer précédé par « on » comme par exemple :
  - L'attribut onclick pour l'évènement « clic sur un élément » ;
  - L'attribut onmouseover pour l'évènement « passage de la souris sur un élément » ;
  - L'attribut onmouseout pour l'évènement « sortie de la souris d'élément » ;
  - Etc.

Evénements applicables à tous les éléments HTML :

`onclick`: sur un clic.
`ondblclick` : sur un double clic .
`onmousedown` : sur une pression d'un bouton de la souris
`onmouseup` : au relâchement d'un bouton de la souris .
`onmouseover`: sur le survol d'un élément par le pointeur de la souris.
`onmouseout` : sur le départ du pointeur de la zone de l'élément.



La fonction addEventListener

- La méthode addEventListener() d'EventTarget installe une fonction à appeler chaque fois que l'événement spécifié est envoyé à la cible.
- Elle permet d'ajouter un évenement vu précedennement c'est-à-dire un évenement au click, clavier,lié au document

```
addEventListerer("évènement",la fonction à déclencher);
```

Un exemple : je veux au click sur un bouton changer la couleur du fond de ma page

```
<div id="cible">
clicker ici
</div>
```

```
function changeBG(){
     document.body.style.background = "blue";
}
document.getElementById("cible").addEventListener("click",changeBG);
```

La fonction addEventListener

Étudions la fonction addEventListener() de plus près.

```
document.getElementById("cible") .addEventListerer ("mouseover",changeBG);
```

Sur l'ID "cible" ajouter un écouteur qui "lors du passage de la souris" lance la fonction changeBG

```
<span class="menu"
>menu1</span>

<span
class="menu">menu2
</span>

<div id="contenu">
</div>
```

```
function change(){
    document.getElementById("contenu").innerHTML = "Bienvenue sur le menu
1";
}

document.getElementsByClassName("menu")[0].addEventListener("mouseover",
change);
```

this

- Le mot clé this est trés important en JS, en effet celui-ci permet de trouver un chemin d'un élément précis du DOM sans même le specifier.
- la variable this a comme valeur l'objet qui est en train d'être construit.

```html
<span class="menu"
>menu1</span>

<span
class="menu">menu2
</span>
```

```javascript
function change(){
this.style.backgroundColor = "blue";
}

var tableau = document.getElementsByClassName("menu");
for(i=0;i < tableau.length; i++){
     tableau[i].addEventListener("mouseover",change);
}
```

L'objet: event

L'objet `event` est la petite cerise sur gâteau ! En effet celui-ci est crée lors de l'utiliation d'un évèvement (click, keydown, etc)...
Il permet de fournir une multitudes d'informations sur l'évènement enclenché tels que les coordonnées du curseur, la touche du clavier actuellement enfoncé, l'évènement utilisé, etc.
Il existe quelques différences sur IE mais nous rentrerons pas dans le détails ici.

```
<div id="contenu">
</div>
```

```
function showme(event){
var coordonneeX = event.clientX;
var coordonneeY = event.clientY;
document.getElementById("contenu").innerHTML = "Coordonnée X " +
coordonneeX + "Coordonnée Y " +coordonneeY;
}

document.body.addEventListener("mouseover",showme);
```

L'objet: event

Voici quelques propriétés de l'objet `event` .

DOM	Description
altKey	L'état de la touche ALT
stopPropagation()	**Empêche l'événement de se propager.**
clientX	Coordonnées X dans la fenêtre.
clientY	Coordonnées Y dans la fenêtre.
ctrlKey	L'état de la touche CTRL
relatedTarget	L'élément depuis lequel le pointeur a bougé.
keyCode	Contient le code de la touche pressée ('keypress').
preventDefault()	**Empêche l'action par défaut de se déclencher.**
screenX	Coordonnées X relative à l'écran.
screenY	Coordonnées Y relative à l'écran.
shiftKey	L'état de la touche SHIFT
target	**L'élément auquel l'événement a été envoyé.**
currentTarget	L'élément vers lequel la souris s'est déplacée.
type	Renvoie le nom de l'événement.

Interface node: propriétés et méthodes

Propriétés principales :

`childNodes` : liste des nœuds enfant
`parentNode` : le nœud parent
`nextSibling` : le prochain nœud
`previousSibling` : le nœud précédent
`firstChild` : premier nœud enfant
`lastChild` : dernier nœud enfant
`nodeName` : nom du nœud
`nodeValue` : valeur du nœud

Méthodes principales :

`appendChild(node)` : ajoute un nœud enfant en dernière position
`removeChild(node)` : retire un nœud enfant
`insertBefore(new, ref)` : ajoute un nœud juste avant le nœud de référence
`replaceChild(new, old)` : remplace l'ancien nœud par le nouveau
`cloneNode(bool)` : retourne un clone du nœud
`hasAttributes()` : indique si le nœud a des attributs
`hasChildNodes()` : indique si le nœud a des fils

Exemple

Sauvegarde un tableau de tous les nœuds enfant disponibles d'un nœud.


```html
<ul id="test">
     <li>node1</li>
     <li>node2</li>
     <li>node3</li>
</ul>
```


```javascript
var noeud=document.body.childNodes.length;
alert(noeud);
```

Interface node: nextSibling, previousSibling

Sauvegarde l'objet du premier nœud enfant d'un nœud.

```html
<ul id="test">
     <li>node1</li>
     <li>node2</li>
     <li>node3</li>
</ul>
```

```javascript
test = document.getElementsByTagName("ul")[0].firstChild;
alert(test.nextSibling.nodeName);
```

```html
<ul id="test">
     <li>node1</li>
     <li>node2</li>
     <li>node3</li>
</ul>
```

```javascript
test = document.getElementsByTagName("ul")[0].firstChild;
alert(test.PreviousSibling.nodeName);
```

Interface node: appendChild

`document.createElement` permet de créer un objet node de type element
`document.createTextNode` permet de créer objet de type node text
`appendChild` permet d'imbriquer les nodes et de l'insérer dans le DOM en position de dernier


```
<ul id="test">
     <li>node1</li>
     <li>node2</li>
     <li>node3</li>
</ul>
```


```
var nouvelleLI = document.createElement("li");
var LItexte = document.createTextNode("node4");
nouvelleLI.appendChild(LItexte);
document.getElementsByTagName("li")[0]. appendChild
(nouvelleLI);
```

Interface node: insertBefore

`document.createElement` permet de créer un objet node de type element
`document.createTextNode` permet de créer objet de type node text
`insertBefore` permet d'imbriquer les nodes et de l'insérer dans le DOM en position de dernier


```
<ul id="test">
     <li>node1</li>
     <li>node2</li>
     <li>node3</li>
</ul>
```


```
var a = document.createElement("p");
var b = document.createTextNode("yes !");
a.appendChild(b);

document.getElementById("ma_div"). insertBefore(a)
```

EXERCICES + TPS

- Un peu d’xercices et tps sont les bienvenus