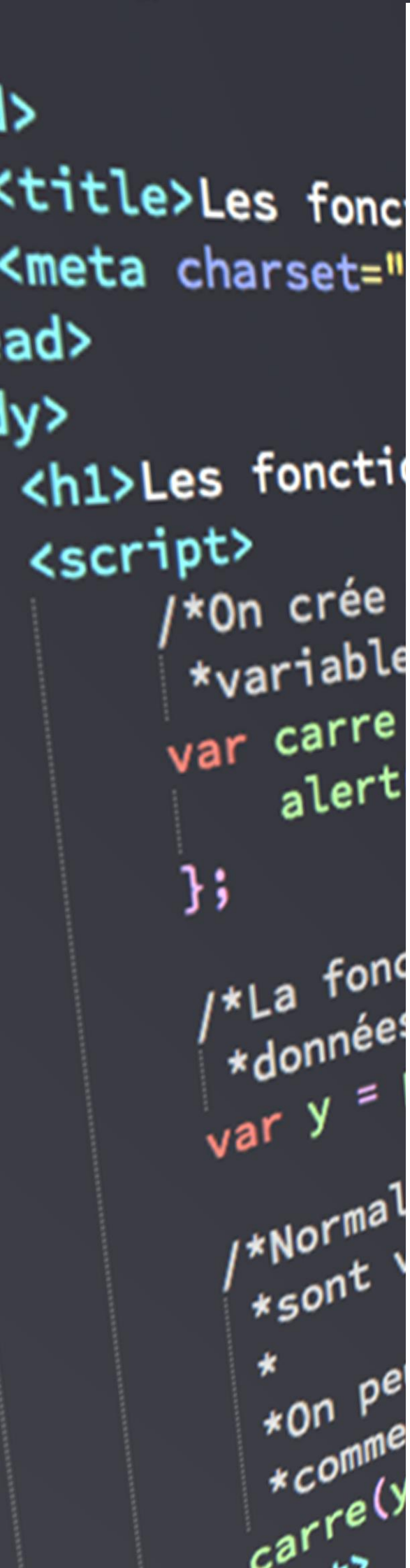
l>
<title>Les fonc
<meta charset="
ad>
ly>
<h1>Les fonctio
<script>
/*On crée
*variable
var carre
alert
};
/*La fonc
*donnée
var y =
/*Normal
*sont
*
*On pe
*comme
carre(y

JavaScript

Table des matières

Les Variables

Les propriétés

Une variable est une zone de stockage d'information. Elle permet de mémoriser des valeurs pour pouvoir les réutiliser dans le programme.
Une variable possède 3 propriétés principales :

- Son **NOM**, qui permet de l'identifier : Il peut contenir des lettres, majuscules ou minuscules, des chiffres (sauf en première position), le dollar **$** et le underscore _. Attention, la casse (majuscule ou minuscule) est importante ; la variable ***test*** et ***Test*** sont deux variables différentes.
- Sa **VALEUR**, qui est la donnée mémorisée dans la variable.
- Son **TYPE**, déterminant le rôle et les opérations applicables à cette variable

JavaScript n'impose pas de définir un type, il est déduit de la valeur enregistrer dans la variable. Le type peut changer au cours de l'exécution du programme.

Déclaration

Avant de pouvoir utiliser une variable, il faut la déclarer. En terme informatique, on réserve dans la mémoire de l'ordinateur, une zone qui sera attribuée à cette variable, il pourra ensuite lire ou écrire des informations, des données, dans cette zone en manipulant la variable.

Exemple de déclaration et d'affichage d'une variable.

```
var a;
console.log(a) ;
```

En JavaScript, la déclaration d'une variable se fait par le mot-clé *var* suivi du nom de la variable.

Avec les nouvelles versions de JavaScript (ECMAScript 2015), il est apparu deux mots-clefs pour déclarer des variables :

- *let* qui permet de définir une variable d'une portée restreinte
- *const* qui permet de définir une variable qui ne pourra être modifiée après sa première affectation

Affectation

Au cours de l'exécution du programme, la valeur d'une variable peut changer. Pour donner une valeur à la variable on utilise l'opérateur **=** appelé **opérateur d'affectation**.

```
a = 3.14 ;
```

La ligne précédente se lit « *a reçoit la valeur 3,14* ». (Ne pas confondre avec le « égal » mathématique.

On peut combiner déclaration et affectation d'une valeur en une seule ligne.

```
let a = 3.14;
```

Les types de variables

Les nombres

```
let a = 2;
let b = 3.1415;
let c = -63;
let d = 2/5;
```

Les chaines de caractères

```
let a = "Bonjour le monde";
let b = 'Hello World !';
```

Il n'y a pas de différence entre les simples et les doubles quotes sauf si notre chaine de caractère contient des ' ou "

```
let a = "Ceci n'est pas problématique";
let b = 'Ceci n\'est pas problématique';
```

Les booléens

Un booléen est soit true (vrai) soit false (faux), il ne peut pas avoir d'autres valeurs.

Les tableaux

Les tableaux permettent de stocker une liste d'information. Elle peut contenir n'importe quel autre type de variables (même un tableau).

```
let eleves = ['Jean', 'Marc', 'Marion'];
let demo = [true, 10, 'Marc'];
```

Les objets

Les objets permettent de stocker des informations plus complexes qu'une simple liste (tableau). Pour faire simple, on peut les imaginer comme une liste avec des index nommés.

```
let eleve = {
        clef : 'valeur',
        nom : 'Marc',
        age : 18,
        notes : [10, 4, 18]
}
```

De même que pour les listes, les objets peuvent contenir d'autres objets en valeur.

```
let eleve = {
        notes : {
                math : 18,
                francais : 14
        }
}
```

Incrémentation

Il est possible d'augmenter ou de diminuer la valeur d'un nombre avec les opérateurs ***+=*** ou ***++***.

Expression

Une **expression** est un morceau de code qui produit une valeur. On crée une expression en combinant des variables, des valeurs et des opérateurs. Toute expression produit une valeur et correspond à un certain type. Le calcul de la valeur d'une expression s'appelle **l'évaluation**. Lors de l'évaluation d'une expression, les variables sont remplacées par leur valeur. Une **expression** peut comporter des parenthèses, elles modifient la priorité des opérations lors de l'évaluation.

Opérateurs

JavaScript possède différents types d'opérateurs :

- Les opérateurs d'affectation
- Les opérateurs de comparaison
- Les opérateurs arithmétiques
- Les opérateurs logiques
- L'opérateur (ternaire) conditionnel

Opérateurs d'affectations

Nom	Opérateur composé	Signification
Affectation	x = y	x = y
Affectation après addition	x += y	x = x + y
Affectation après soustraction	x -= y	x = x - y
Affectation après multiplication	x *= y	x = x * y
Affectation après division	x /= y	x = x / y
Affectation du reste	x %= y	x = x % y
Affectation après décalage à gauche	x <<= y	x = x << y
Affectation après décalage à droite	x >>= y	x = x >> y
Affectation après décalage à droite non signé	x >>>= y	x = x >>> y
Affectation après ET binaire	x &= y	x = x & y
Affectation après OU exclusif binaire	x ^= y	x = x ^ y
Affectation après OU binaire	x |= y	x = x | y

Opérateurs de comparaison

```
var var1 = 3;
var var2 = 4;
```

Opérateur	Description	Exemples qui renvoient true
Égalité (==)	Renvoie true si les opérandes sont égaux après conversion en valeurs de mêmes types.	3 == var1 "3" == var1 3 == '3'
Inégalité (!=)	Renvoie true si les opérandes sont différents.	var1 != 4 var2 != "3"
Égalité stricte (===)	Renvoie true si les opérandes sont égaux et de même type. Voir Object.is()et égalité de type en JavaScript.	3 === var1
Inégalité stricte(!==)	Renvoie true si les opérandes ne sont pas égaux ou s'ils ne sont pas de même type.	var1 !== "3" 3 !== '3'
Supériorité stricte(>)	Renvoie true si l'opérande gauche est supérieur (strictement) à l'opérande droit.	var2 > var1 "12" > 2
Supériorité ou égalité (>=)	Renvoie true si l'opérande gauche est supérieur ou égal à l'opérande droit.	var2 >= var1 var1 >= 3
Infériorité stricte(<)	Renvoie true si l'opérande gauche est inférieur (strictement) à l'opérande droit.	var1 < var2 "2" < "12"
Infériorité ou égalité (<=)	Renvoie true si l'opérande gauche est inférieur ou égal à l'opérande droit.	var1 <= var2 var2 <= 5

Opérateurs arithmétiques

Opérateur	Description	Exemple
Reste (%)	Opérateur binaire. Renvoie le reste entier de la division entre les deux opérandes.	12 % 5 renvoie 2.
Incrément (++)	Opérateur unaire. Ajoute un à son opérande. S'il est utilisé en préfixe (++x), il renvoie la valeur de l'opérande après avoir ajouté un, s'il est utilisé comme opérateur de suffixe (x++), il renvoie la valeur de l'opérande avant d'ajouter un.	Si x vaut 3, ++x incrémente x à 4 et renvoie 4, x++ renvoie 3 et seulement ensuite ajoute un à x.
Décrément (--)	Opérateur unaire. Il soustrait un à son opérande. Il fonctionne de manière analogue à l'opérateur d'incrément.	Si x vaut 3, --x décrémente x à 2 puis renvoie2, x-- renvoie 3 puis décrémente la valeur de x.
Négation unaire (-)	Opérateur unaire. Renvoie la valeur opposée de l'opérande.	Si x vaut 3, alors -x renvoie -3.
Plus unaire (+)	Opérateur unaire. Si l'opérande n'est pas un nombre, il tente de le convertir en une valeur numérique.	+"3" renvoie 3. +true renvoie 1.

Opérateurs logiques

Opérateur	Usage	Description
ET logique (&&)	expr1 && expr2	Renvoie expr1 s'il peut être converti à false, sinon renvoie expr2. Dans le cas où on utilise des opérandes booléens, && renvoie true si les deux opérandes valent true, false sinon.
OU logique (||)	expr1 || expr2	Renvoie expr1 s'il peut être converti à true, sinon renvoie expr2. Dans le cas où on utilise des opérandes booléens, || renvoie true si l'un des opérandes vaut true, si les deux valent false, il renvoie false.
NON logique (!)	!expr	Renvoie false si son unique opérande peut être converti en true, sinon il renvoie true.

Opérateur conditionnel ternaire

condition ***?*** *val1* ***:*** *val2*

Si condition vaut *true*, l'opérateur vaudra *val1*, sinon il vaudra *val2*.