

# JAVASCRIPT 2/3

**Formateur: NSEKE Charles**

# PLAN

- I. Les objets
- II. Les opérateurs
- III. Les structures conditionnelles
- IV. Les fonctions
- V. Le DOM
- VI. Les évènements

LES OBJETS

# LES OBJETS

Il s'agit d'un type de variable qui peut contenir des propriétés et des fonctions.

```
var monObjet = {};
```

```
var monObjet2 = {propriete1: 1, propriete2: "une propriété"};
```

```
var monObjet3 = new Object();
```

Pour accéder à la valeur d'une propriété d'un objet:

```
monObjet2.propriete2;
```

# LES OBJETS

On peut ajouter des propriétés à un objet de deux façons:

```
var monObjet = {};
```

```
monObjet.maPropriete = 1222;
```

```
monObjet["maPropriete2"] = "une autre propriété";
```

# LES OPÉRATEURS

# OPÉRATEURS D'AFFECTATION

## Opérateurs d'affectation

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

# OPÉRATEURS DE COMPARAISON

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



# EXERCICE

que vaut :

`1 == 1`

`1 === 1`

`1 == '1'`

`1 === '1'`

`[] == ''`

`[] === ''`

que vaut :


`1 >= '1'`

`1 >= '2'`

`1 >= 'a'`

`'a' >= 'a'`

# 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 renvoie 2, 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.
Opérateur d'exponentiation (**) (puissance) 	Calcule un nombre (base) élevé à une puissance donnée (soit <code>base<sup>puissance</sup></code> )	2 ** 3 renvoie 8  10 ** -1 renvoie 0.1

# EXERCICE

que vaut :

`"1" + 2`

`1 + 1`

`"1" + "1"`

`[] + +[] + 1` /// rajouter

`+[] + +[] + 1`

# OPÉRATEURS LOGIQUES

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

# QUELQUES BONUS

Opérateur delete: qui permet de supprimer la valeur d'une variable ou d'une propriété d'un objet.

```
valeur1 = 12;
```

```
delete valeur1;//
```

```
var voiture = {marque:"Volkswagen", puissanceDynamique: "150  
CV"}
```

```
delete voiture.marque;
```

# QUELQUES BONUS

Opérateur `typeof`: retourne le type de la variable qu'on met à sa droite.

```
var nombre = 12;
```

```
console.log(typeof nombre);
```

# QUELQUES BONUS

Opérateur condition ternaire:

**condition ? valeur1 : valeur2**

*valeur2 si la condition est false*

*valeur1 si la condition est true*

# LES STRUCTURES CONDITIONNELLES



# IF/ELSE

```
if(condition) {  
  
} else if(condition2) {  
  
} else {  
  
}///  
  
if(condition) {  
  
}  
  
if(condition2) {  
  
} else {  
  
}
```

# LES BOUCLES

for, while, do... while

pour sortir d'une boucle on peut utiliser break;

# SWITCH CASE

```
switch(valeur) {  
  
  case 'valeur possible':  
  
    // traitements  
  
    break;  
  
  case 'valeur possible 2':  
  
    // traitements  
  
    break;  
  
  default:  
  
    // traitements  
  
}
```

# EXERCICE

```
pour a = 3, 2, 12;
```

```
switch(a) {
```

```
case:2
```

```
case:3
```

```
alert(a);
```

```
break;
```

```
default:
```

```
alert("on ne sait pas gérer ça");
```

```
}
```

# TP

1- créer trois objets ayant les propriétés : prenom, dateDeNaissance, moyenne.

(Karine, 16/04/1991, 12),

(Célia, 05/05/1995, 14),

(Christophe, 09/09/1997, 15);

2- mettez les objets créés dans un tableau

3- affichez l'âge des personnes : prénom + âge(à l'aide d'une boucle)

# TP

- 4- affichez le prénom des personnes qui ont plus de 25ans (à l'aide d'une boucle)
- 5- affichez les personnes (toutes les propriétés) dont le prénom commence par la lettre 'c'
- 6- affichez la moyenne des moyennes des personnes
- 7- affichez les personnes qui ont une moyenne inférieure à la moyenne des trois moyennes
- 8- supprimer la propriété moyenne des trois objets

# LES FONCTIONS

# LES FONCTIONS

En JavaScript une fonction est un objet. Une fonction est un bout de code qui peut être appelé dans un autre bout de code.

On manipule généralement des fonctions anonymes (sans nom) et des fonctions nommées.

```
function () {} // fonction anonyme
```

```
function maFonction() {} // fonction nommée
```



# LES FONCTIONS

On peut déclarer une fonction dans un fonction

```
function perimetreRectangle(longueur, largeur) {  
  function somme(val1, val2) {  
    return val1 + val2;  
  }  
  return somme(longueur, largeur) * 2  
}
```

# LES FONCTIONS DEPUIS ES2015

Les fonctions se définissent comme ceci:

```
() => {}; // anonyme
```

```
const maFonction = () => {}; // fonction nommée
```

```
const perimetreRectangle = (longueur, largeur) => {
```

```
    const somme = (a, b) => a + b;
```

```
    return somme(longueur, largeur) * 2;
```

```
}
```

LE DOM

# DEFINITION DU DOM

Le DOM (Document Object Model) est une API qui représente et interagit avec tout document HTML ou XML.

Le DOM est l'une des API les plus utilisées sur le Web car il permet au code exécuté dans un navigateur d'accéder à chaque nœud du document et d'interagir avec lui. Les nœuds peuvent être créés, déplacés et modifiés. Des écouteurs d'événements peuvent être ajoutés aux nœuds et déclenchés lors de la survenue d'un événement donné.

# MANIPULATION DU DOM

pour manipuler le dom de la page html dans laquelle code javascript s'exécute, il faut utiliser l'objet document.

Une liste de méthode à utiliser pour manipuler et effectuer des traitements sur le DOM est disponible ici:

[https://developer.mozilla.org/en-US/docs/Web/API/Document\\_Object\\_Model/Introduction#DOM\\_interfaces](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction#DOM_interfaces)

LES ÉVÈNEMENTS

# LES ÉVÈNEMENTS

Un évènement est une action qui est déclenchée par l'utilisateur sur une page web.

Exemple d'évènements: click, survol d'une zone par la souris, tape une touche...

# LES ÉVÈNEMENTS

Pour attacher un évènement à un élément du DOM on peut:

```
document.elementSelectionne.addEventListener('click', function() {alert("il a cliqué");});
```

```
document.elementSelectionne.onclick = function() {alert("il a cliqué");};
```





TP

récupérer le code html de cette page :

<https://github.com/nsekecharles/formation-js/blob/master/exercice-evenements.html> suivez les instructions