

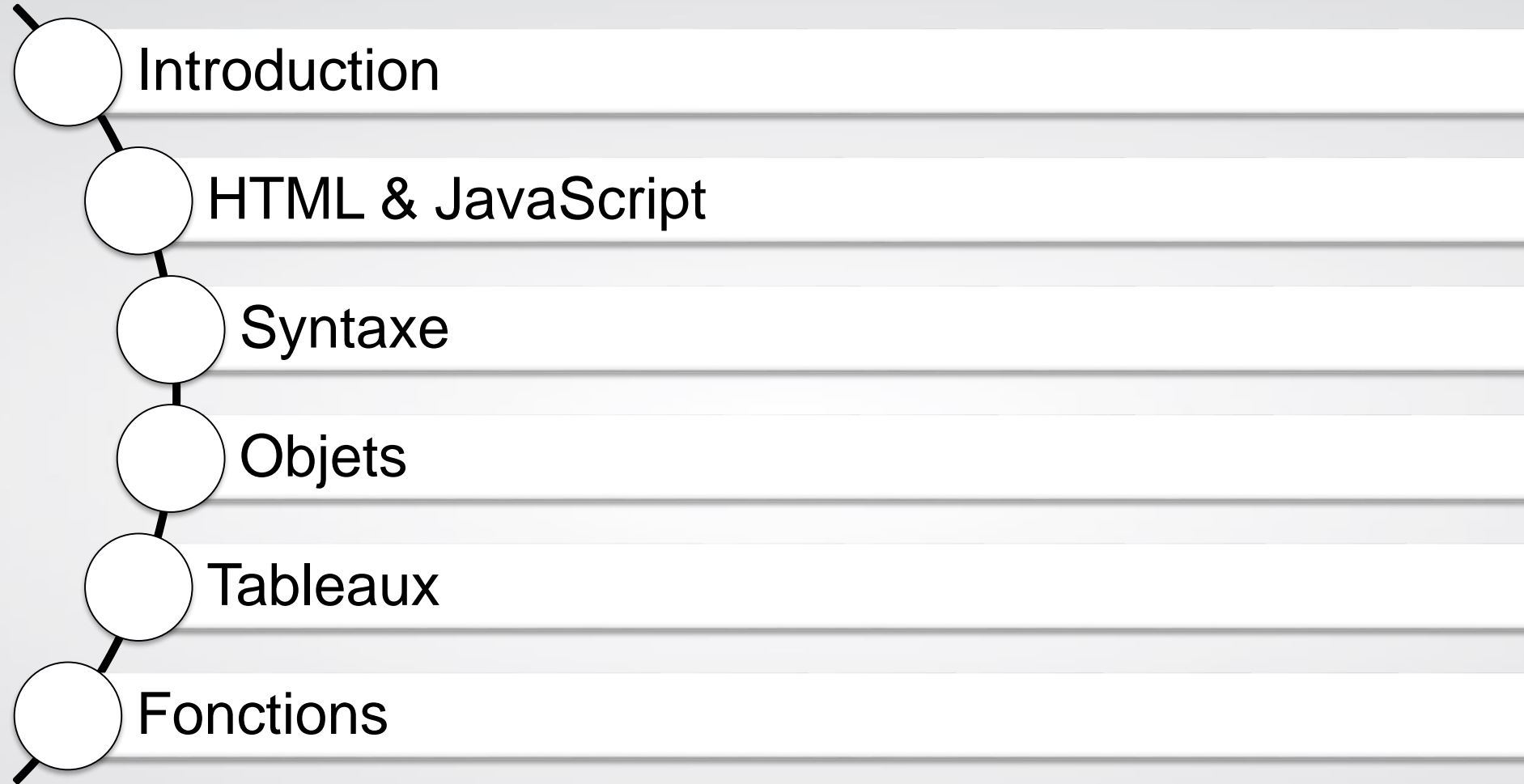


Chapitre 2: JavaScript

UP Web

AU: 2021/2022

Plan





Objectifs

- Manipuler le DOM.
- Différencier entre les événements.
- Ecrire un script en utilisant les fonctions prédéfinis, événement...

Prérequis

- HTML



Introduction

- **Javascript** permet de rendre **interactif** un site internet développé en HTML.
- **Javascript** est standardisé par un comité spécialisé, l'ECMA (European Computer Manufactures Association).
- **JavaScript** est un langage de programmation:
 - **scripté** (interprété) - pas de compilateur à proprement parler.
 - **côté client** - s'exécute dans un navigateur en général (il existe des environnements côté serveur : NodeJS).
 - **asynchrone** - plusieurs « morceaux » peuvent s'exécuter en parallèle.



Introduction



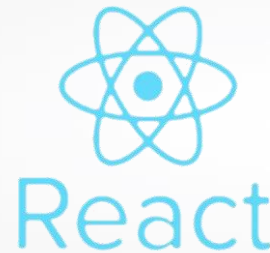
- JavaScript, permet :
 - de spécifier des changements sur le document :
 - sur le contenu: la structure, le style...
 - en interceptant des évènements: souris, clavier, ...
 - échanger avec un serveur (AJAX)
 - dessiner (canvas - bitmap - ou svg - vectoriel)
 - se géolocaliser
 - enregistrer localement du contenu
 - jouer des fichiers audio ou vidéo

► Introduction

- Utilitaires JavaScript



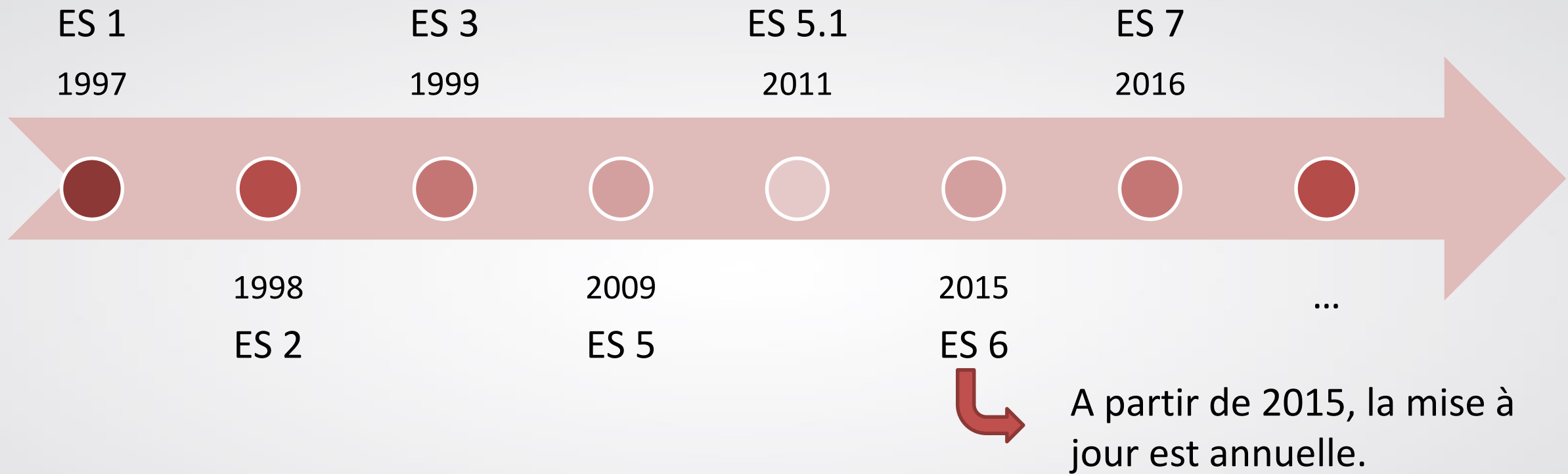
ANGULARJS



- On peut créer une base logicielle entièrement codée en JavaScript qui peut tourner sur **MEAN** → **M**ongoDB, **E**xpress.js, **A**ngular.js, et **N**ode.js au lieu de **LAMP** → **L**inux, **A**pache, **M**ySQL, **P**HP.



Introduction



- ✓ ES6 est le standard supporté par la plupart des nouvelles versions de navigateurs.
<https://kangax.github.io/compat-table/es6/>



HTML & JavaScript

Console
développeur

- Firefox: Ctrl+Shift+K
- Chrome / Edge: Ctrl+Shift+I

Balise HTML

- `Cliquez-moi !`

Code HTML
(interne)

- `<script> ... </script>`

Fichier séparé
(externe)

- `<script src="script.js"></script>`

Exemple

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7    </head>
8    <body>
9      <p id="item">Mon premier paragraphe</p>
10     <p>Mon deuxième paragraphe</p>
11
12     <script>
13       // Instructions
14     </script>
15  </body></html>
```

- Tester le code suivant:

```
alert("Bonjour tout le monde!!!");
```

```
console.log("Texte à afficher");
```

```
document.getElementById('item').innerHTML = "<p> Nouveau paragraphe</p>";
```



Exemple

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7    </head>
8    <body>
9      <p id="item">Mon premier paragraphe</p>
10     <p>Mon deuxième paragraphe</p>
11
12     <script>
13       // Instructions
14     </script>
15  </body></html>
```

- Résultat:

Nouveau paragraphe

Mon deuxième paragraphe

Éléments Console Sources Réseau >>

▶ ⛔ sommet ▼ 🔍 Filtrer Niveau

Texte à afficher

>

Syntaxe



Les commentaires

- Par ligne:

```
// un commentaire sur une ligne
```

- Par Bloc:

```
/* un commentaire plus  
long sur plusieurs lignes  
*/
```

- Remarque:

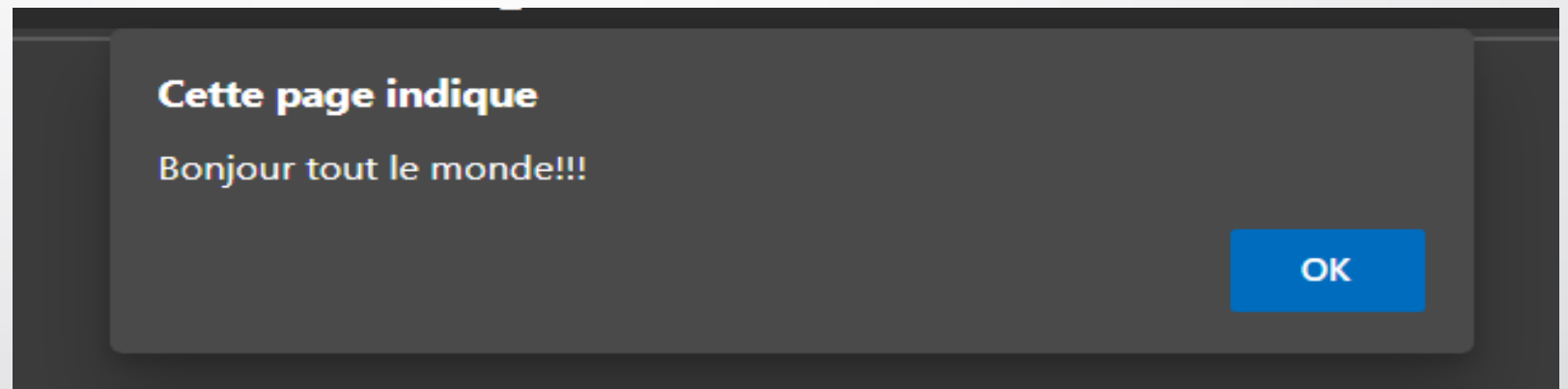
```
/* Par contre on ne peut pas /* imbriquer des commentaires */ SyntaxError */
```

Syntaxe



Les boîtes de dialogue

```
<script>  
    alert("Bonjour tout le monde!!!");  
</script>
```



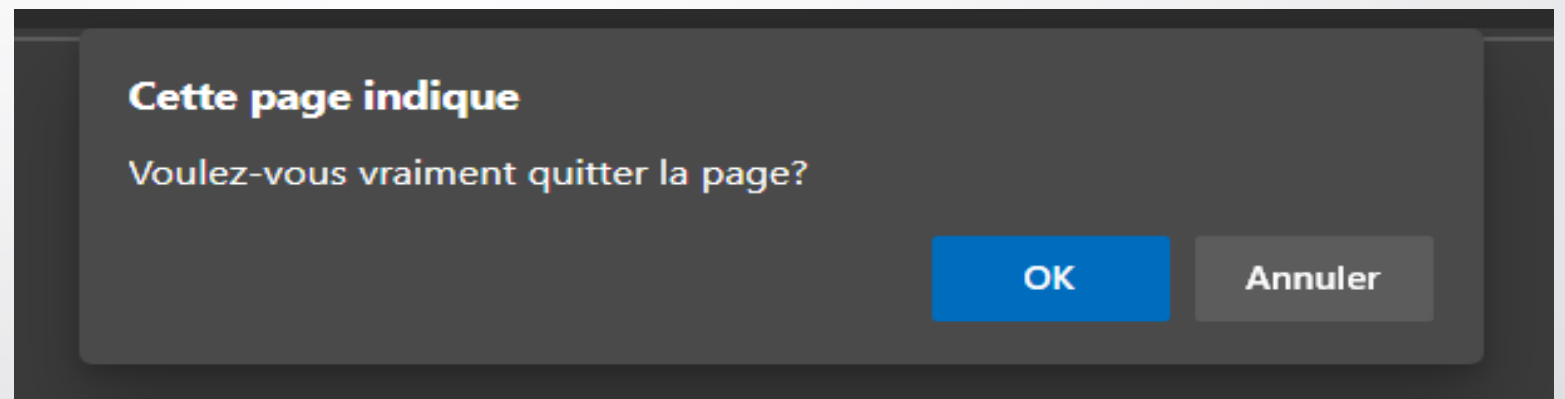
► Syntaxe



Les boîtes de dialogue

```
<script>  
    confirm("Voulez-vous vraiment quitter la page?");  
</script>
```

confirm: renvoie
true ou false



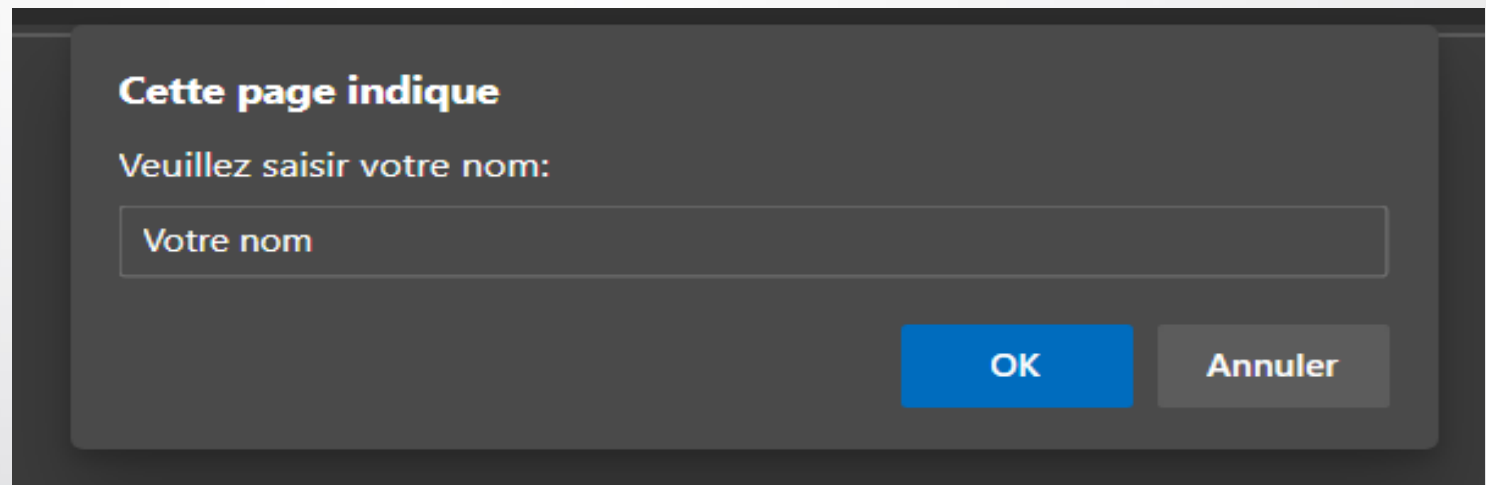
► Syntaxe



Les boîtes de dialogue

```
<script>  
    prompt("Veuillez saisir votre nom:", "Votre nom");  
</script>
```

prompt: renvoie la
valeur saisie ou Null



Syntaxe



Variable

- **JavaScript** est un langage pauvrement typé, il n'est pas indispensable de déclarer préalablement le type de variable.
- Il existe trois types de déclarations de variable en JavaScript.
 - **var**: déclare une variable, en initialisant sa valeur éventuellement.
 - **let**: déclare une variable dont la portée se limite au bloc courant.
 - **const**: déclare une constante, dont la portée se limite au bloc courant et accessible en lecture seule.

Syntaxe



Variable

- Le nom d'une variable doit commencer par:
 - Lettre
 - Tiret bas (_)
 - Symbole dollar (\$)
- Les caractères qui suivent peuvent inclure les lettres minuscules et/ou majuscules et les chiffres.
- **Par convention:**
 - Noms de variables et fonctions écrits en CamelCase
 - Noms de constantes écrits en majuscule
- **Remarque:**
 - JavaScript est sensible à la casse: maVariable est différente de MaVariable.

Syntaxe



Variable

- Typage dynamique

```
var maVariable = 2020; // maVariable est un nombre
maVariable = "hello"; // maVariable est une chaîne de caractères
maVariable = true; // maVariable est un booléen
```

- Types de données

- Primitifs:

- Booléen
 - Null
 - Undefined
 - Nombre
 - String

- Objet

Syntaxe



Variable

- Evaluation

```
var a;  
console.log("La valeur de a est " + a); // La valeur de a est undefined  
  
console.log("La valeur de b est " + b); // La valeur de b est undefined  
var b; // La déclaration de la variable est "remontée" (hoisting)  
  
console.log("La valeur de x est " + x); // signale une exception ReferenceError  
let x, y;  
console.log("La valeur de y est " + y); // La valeur de y est undefined
```

Syntaxe



Variable

- Portée

```
if (true) {  
  var z = 2020;  
}  
console.log(z); // z vaut 2020  
  
if (true) {  
  let y = 'Hello!!';  
}  
console.log(y); // Uncaught ReferenceError: y is not defined
```

Syntaxe



Variable

- L'opérateur **typeof** renvoie le type d'une variable.

```
var X, nom = prompt("Veuillez saisir votre nom:", "Votre nom");

console.log(typeof X);
// expected output: "undefined"

X = 2020;
console.log(typeof X);
// expected output: "number"

console.log(typeof nom);
// expected output: "string"

console.log(typeof true);
// expected output: "boolean"

console.log(typeof Symbol('test'));
// expected output: "symbol"

console.log(typeof null);
// expected output: "object"
```



Syntaxe

Opérateurs

Opérateur	Explication	Symbole
Addition	<ul style="list-style-type: none">Additionner des nombres (1+5;)Concaténer des chaînes ("Hello " + "World! ";;)	+
Arithmétique	<ul style="list-style-type: none">Les opérateurs mathématiques de base: soustraction, division et multiplicationOpérateur de puissance (**)Reste de la division (%)	- , / , * ** %
Assignation	<ul style="list-style-type: none">Affecte une valeur à une variableAffectation après addition, soustraction, division, multiplicationAffectation du reste (x %= y ➔ x = x % y)	= +=, -=, /=, *= % =
Négation	<ul style="list-style-type: none">Non Logique: Renvoie la valeur opposé (false devient true)Non Unaire: Renvoie l'opposé de l'opérandeNon binaire: Inverse les bits de l'opérande (~1 ➔ 0)	! - ~



Syntaxe

Opérateurs

Opérateur	Explication	Symbole
Incrémentation Décrémentation	Ajoute / soustrait une unité à son opérande <ul style="list-style-type: none">• Suffixe: renvoie la valeur avant l'incrément / décrémentation• Préfixe: renvoie la valeur après l'incrément / décrémentation	X++, X-- ++X, --X
Relationnel	<ul style="list-style-type: none">• Permet de comparer deux opérandes et de renvoyer une valeur booléenne	<, >, <=, >=
Binaire	<ul style="list-style-type: none">• ET binaire (AND)• OU binaire (OR)	&
Logique	<ul style="list-style-type: none">• ET logique (AND)• OU logique (OR)	&&
Egalité	<ul style="list-style-type: none">• (in)égalité faible• (in)égalité stricte	!=, == !=, ==

Syntaxe



Structure Conditionnelle

- `if (condition) {
 // instructions
}
else {
 // instructions
}`

```
let x = 'WEB'  
  
if (x === 'web') {  
    alert('Web Development');  
}  
else {  
    alert('Others');  
}
```



Syntaxe

Structure Itérative

- for ([exp. Initiale]; [Condition]; [incrément]) {
 // instructions
}
- do {
 // instructions
} while (condition);
- while (condition) {
 // instructions
}

```
for (var compteur = 0; compteur < 5; compteur++){  
    console.log("Compteur = " + compteur);  
}
```

Compteur = 0

Compteur = 1

Compteur = 2

Compteur = 3

Compteur = 4

>

► Objets



Définition

- Un objet est une entité à part entière qui possède des propriétés.
- Une propriété est une association entre un nom (clé) et une valeur.
- On accède à une propriété en utilisant les notations suivantes:
 nom_Objet.nom_Propriété
 nom_Objet[nom_Propriété]

► Objets



Définition

- La création d'un objet peut se faire :

```
var nom_Objet = new Object();  
var nom_Objet = { prop_1: valeur_1,  
                  'prop_2': valeur_2,  
                  prop_3: function() { ...},  
                  ...,  
                  10: valeur_10  
};
```

```
Object.create(nom_Objet);
```

► Objets



Exemple 1

```
var car = new Object();  
car.manufacturer = "General Motors";  
car.model = "Corvette";  
car.year = 1953;  
  
console.log(car);  
console.log('manufacturer: ' + car['manufacturer']);  
console.log(car[2]);
```

```
▼ {manufacturer: "General Motors", model: "Corvette", year: 1953} ⓘ  
  manufacturer: "General Motors"  
  model: "Corvette"  
  year: 1953  
  ► __proto__: Object  
manufacturer: General Motors  
undefined
```

► Objets



Exemple 2

```
var transportation = {  
  type: "car",  
  afficherType : function() {  
    console.log('transportation type: ' + this.type);  
  }  
}  
  
var t1 = Object.create(transportation);  
t1.afficherType();
```

```
transportation type: car
```



Tableau

- JavaScript ne possède pas de type particulier pour représenter un tableau de données.
- Utiliser l'objet natif **Array** ainsi que ses méthodes pour manipuler des tableaux.
- Pour créer un tableau:
 - `var arr = new Array(élément0, élément1, ..., élémentN);`
 - `var arr = Array(élément0, élément1, ..., élémentN);`
 - `var arr = [élément0, élément1, ..., élémentN];`



Tableau

- Pour créer un tableau sans aucun élément initial et de longueur non nulle (l):
 - `var arr = new Array (l);`
 - `var arr = Array (l);`
 - `var arr = [];`
`arr.length = l;`
- Rq: l doit être un nombre

► Tableau



Exercice 1

- Tester les instructions suivantes:

```
var arr1 = [5];  
var arr2 = Array (5);  
var arr3 = Array (5.2);  
var arr4 = Array.of(5);  
  
console.log(arr1);  
console.log(arr2);  
console.log(arr3);  
console.log(arr4);
```

► Tableau



Exercice 2

- Tester les instructions suivantes:

```
var arr = [];  
arr[0] = "Un";  
arr[1] = 2;  
arr[2] = 3.14;  
  
console.log(arr);  
console.log(arr[0]);  
console.log(arr["length"]);  
arr.length = 0;  
console.log(arr);  
arr.length = 3;  
console.log(arr);
```


► Tableau

Exercice 3

- Tester les instructions suivantes:

```
var arr = [];  
arr[0] = "Un";  
arr[1] = 2;  
arr[2] = 3.14;  
  
for (var i = 0; i < arr.length; i++) {  
    console.log(arr[i]);  
}  
  
arr.forEach(function(nb) {  
    console.log('nb: ' + nb);  
});  
  
arr.forEach(nb => console.log('nb: ' + nb));
```



Fonctions



Syntaxe

```
function nom_fonction () {  
    // instructions  
}
```

```
function greetings()  
{  
    console.log("Welcome!");  
}  
  
greetings();
```



Fonctions



Syntaxe

```
function nom_fonction (arg1, arg2, ...) {  
    // instructions  
}
```

```
function greetings(name)  
{  
    return "Welcome " + name + "!";  
}  
  
console.log(greetings("Jane"));
```



Fonctions



Syntaxe

```
function nom_fonction (arg1 = value, arg2, ...) {  
    // instructions  
}
```

```
function greetings(name="John")  
{  
    return ("Welcome " + name + "!");  
}  
  
console.log(greetings()); // OUTPUT: Welcome John!
```



Fonctions



Syntaxe: Fonction anonyme

```
var x = function (arg1, arg2, ...) {  
    // instructions  
}
```

```
var greet = function (name="John")  
{  
    return ("Welcome " + name + "!");  
}  
  
console.log(greet()); // OUTPUT: Welcome John!
```



Fonctions

Syntaxe: Fonction anonyme

```
var x = function (arg1, arg2, ...) {  
    // instructions  
}
```

➔ Une fonction anonyme peut être passée en paramètre.

```
setTimeout(function ()  
{  
    console.log ("Welcome!");  
}, 2000); // OUTPUT: Welcome!
```



Fonctions

Syntaxe: Arrow function

Plusieurs paramètres

```
(arg1, arg2, ...) => {  
  // instructions  
}
```

Un seul paramètre

```
(arg1) => { // instructions }  
arg1 => { // instructions }
```

Sans paramètres

```
() => { // instructions }
```

```
var factorielle = function fac(n) { return n < 2 ? 1 : n * fac(n - 1) };  
console.log(factorielle(3)); // expected output: 6
```

```
var fact = n => n < 2 ? 1 : n * fact (n-1);  
console.log(fact(3));
```



► **Merci de votre attention**