Dièse Dev <1> - Introduction à JavaScript

Introduction

Cette première formation est dédié à l'apprentissage du language JavaScript. JavaScript est au coeur du web, normalement utilisé côté client, c'est un langage de programmation de scripts principalement employé dans les pages web interactives. Avec l'arrivée de NodeJs, il est maintenant utilisé également côté serveur ce qui fait de lui un language complet et essentiel pour le web.

React Native est une librairie JavaScript. C'est donc en JavaScript que nous développerons nos applications mobiles.

> C'est un langage orienté objet.
>
> -- *fr.wikipedia.org/wiki/JavaScript*

Pour lancer un script JavaScript, on utilisera un navigateur classique

Afficher dans la console

```
console.log('👋 Hello World!')
```

Affiche la chaine de caractère "Hello World!" dans la console.

Déclarer une variable ou une constante

> Le mot clé **const** permet de déclarer une constante accessible uniquement en lecture.

On ne peux pas modifier une constante 😅 .

```
const a = 10
a = 12 //Erreur
const b = a + 2
//b = 12
```

> Pour déclarer une variable, deux mots clés existent : **var** et **let**

Ces deux mots clés ont la même utilité mais il existe quelques subtilités.

```
let a = 5
a = 10
console.log(a)
>> 10

var b = 10
console.log(b)
```

```
>> 10

b = 42
console.log(b)
>> 42
```

Les subtilités entre **var** et **let** :

```
/* ✅ Fonctionne ✅ */
//On peut utiliser une variable avant de la déclarer mais il faut au moins
la déclarer après l'avoir utiliser.
prenom = "Jack"
var prenom

var age = 16
var age = 21 // On peut redéclarer une variable avec var

/* ❌ Ne fonctionne pas ❌ */
//Il faut declarer la variable avant de l'utiliser
prenom = "Jack"
let prenom

let age = 16
let age = 21 //On ne redeclare pas les variables avec let
```

Les conditions

> JavaScript possède des opérateurs de comparaisons stricts et des opérateurs de comparaisons qui effectuent des conversions. Une comparaison strict (ex. : ===) ne sera vraie que si les deux opérandes sont du même type. La comparaison d'égalité faible (==) convertira les deux opérandes en un même type avant d'effectuer la comparaison. Pour les comparaisons relationnelles (ex. : <=), les opérandes sont tout d'abord converties en valeurs, puis en valeurs du même type, enfin la comparaison est effectuée.
> *https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Op%C3%A9rateurs/Op%C3%A9rateurs_de_comparaison*

Égalité simple (==)

```
1 == 1     ✅
1 == '1'   ✅
0 == false ✅
1 == 2     ❌
```

Inégalité simple (!=)

```
1 != 2;      ✅
1 != "1";    ❌
1 != '1';    ❌
1 != true;   ❌
```

Égalité stricte (===)

```
1 === 1   ✅
1 === '1' ❌
```

Inégalité stricte (!==)

```
3 !== '3' ✅
4 !== 3   ✅
```

If..else

> L'instruction if exécute une instruction si une condition donnée est vraie ou équivalente à vrai. Si la condition n'est pas vérifiée, il est possible d'utiliser une autre instruction.
>
> *https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Instructions/if...else*

```
if(condition){
    ...
}else{
    ...
}
```

Attention à ne pas confondre les valeurs booléennes « primitives » true et false avec les valeurs true et false d'un objet Boolean. Toute valeur qui n'est pas false, undefined, null, 0, -0, NaN ou la chaîne vide (""), et tout objet, y compris un objet Boolean dont la valeur est false, seront évalués à true lors d'une instruction conditionnelle if. Ainsi :

```
var b = new Boolean(false);
if (b) // la condition sera évaluée à true car b est définie mais il est
différent de false

if(a) //la condition sera évaluée à false car a est undefined
```

Exemple :

```
var nombreSoda = 5
if(nombreSoda){
    console.log("la variable nombreSoda est définie")
}

if(nombreSoda > 0){
    console.log('Il reste du 🥤 ')
}else{
    console.log("Il n'y a plus de 🥤 ")
}

>>"la variable nombreSoda est définie"
>>"Il reste du 🥤 "
```

> Pour concaténer des chaines de caractères entre elles ou avec des variables, on utilise l'opérateur **+**

```
const prixSoda = 0.99

console.log('🥤  : $'+prixSoda)
>>'🥤  : $0.99'
```

Les fonctions

On peut déclarer des fonctions de différentes manières en JavaScript.

```
function carré(a){
    return a * a
}

const carré = function (a){
    return a * a
};

const carré = (a) => {
    return a * a
}
```

> À noter : on ne peut pas accéder à des variables définies dans une fonction en dehors de la fonction.

```
// Les variables suivantes sont globales
var num1 = 20,
    num2 = 3,
    nom = "Licorne";

// Cette fonction est définie dans la portée globale
function multiplier() {
```

```
  return num1 * num2;
}

multiplier(); // Renvoie 60

// Un exemple de fonction imbriquée
function getScore () {
    var a = 12
    var num1 = 2,
    num2 = 3;

    function ajoute() {
        return nom + " a marqué " + (num1 + num2);
    }

    return ajoute();
}

getScore(); // Renvoie "Licorne a marqué 5"

console.log(a) // ❌ Erreur : on ne pas avoir accès à une variable en
dehors de la fonction où elle est déclaré.
```

Les objets

On peut créer un objet simplement de la manière suivante :

```
var menu = {
    burger:'Bacon CheeseBurger',
    accompagnement: 'Frites',
    boisson:'Coca Vanille Cerise'
}

console.log('Le client a commandé un '+menu.burger+' accompagné de
'+menu.accompagnement+' et il sirotera un '+ menu.boisson)

>> "Le client a commandé un Bacon CheeseBurger accompagné de Frites et il
sirotera un Coca Vanille Cerise"
```

On peut également modifier les objets :

```
menu.boisson = "Sprite"
console.log(menu)
>>{
    burger:'Bacon CheeseBurger',
    accompagnement: 'Frites',
    boisson:'Sprite'
}
```