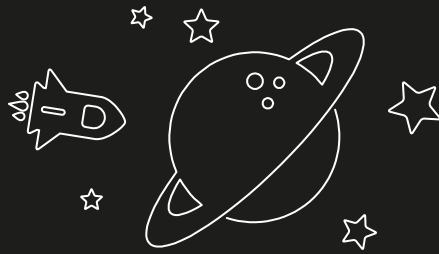


Les fondements

JS



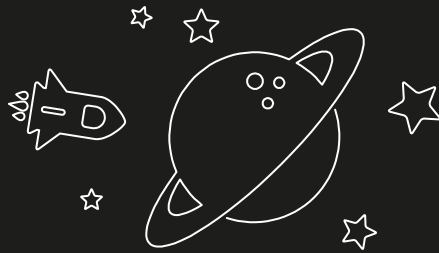


# Structure

2

```
/*  
 * @param input  
 * @return boolean  
 */  
function echo(input) {  
  // Display message into the console  
  console.log("message");  
  return true;  
}
```

3



# Types de données

4



5

Typage faible

6 types différents

2 catégories

4 type de déclarations  
possibles



# 6

## Les types de données

### Types primitifs

String => "message"

Number => 2 2.3

Boolean => true false

Undefined => undefined

### Types complexes

Object

object => { prop1: true, prop2: "yes" }

null => null

array => [ 1, 2, 3 ]

Function

“

**MENSONGES !!!!**

7

A close-up of Pinocchio's face from the Disney movie. He has a surprised expression with wide eyes and an open mouth. A white square box is drawn over his face, containing the number 8.

# 8

## JAVASCRIPT EST UN LANGAGE ORIENTE OBJET A PROTOTYPE

- Chaque type est “wrappé” par un objet
- Panel de fonctions intégrées et customisables

```
// Number
1.390.toFixed(3)
// String
“message”.split()
// Boolean
true.toString()
// Array
[13, 9, 0].length
// Function
function split(arg1){}
split.arguments
```



9

## Les opérateurs

	+	-	*	/	.
Number	Add	Sub	Mult	Div	
String	Concat				
Object					Access
Boolean	Add	Sub	Mult	Div	

## Résultats inattendus

NaN

Infinity

# 10

## Déclaration de variables

**foo = 3**

Déclaration globale

**var foo = 3**

Déclaration locale

**let foo = 3**

Déclaration au block

**const foo = 3**

Let + constante de référence

## Attention !!! Phénomène de Hoisting

Javascript remonte automatiquement les déclarations de variables sans les initialiser en haut du scope courant pour le keyword **var**







## Les closures

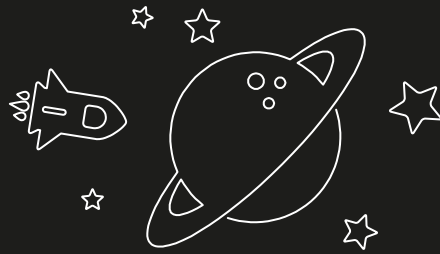
Fonction dans une fonction

Javascript se souvient de l'environnement de chaque fonction.

Une variable qui ne devrait plus exister peut donc continuer à être appelée.

```
function creerFonction() {  
  var nom = "Mozilla";  
  function afficheNom() {  
    console.log(nom);  
  }  
  return afficheNom;  
}
```

```
var maFonction = creerFonction();  
maFonction();
```



# **BLOC DE STRUCTURE**

13



# 14

## Les boucles

### Boucle for

```
for(init;cond;acc) {  
    //Something  
}
```

### Boucle for ... in

```
for(x in object) {  
    //Something  
}
```

### Boucle do ... while

```
do {  
    //Something  
} while (cond)
```

### Boucle while

```
while (cond) {  
    //Something  
}
```

### Break X

Stop les itérations de  
X boucles

### Continue

Passe à l'itération  
suivante



# 15

## Les conditions

if

```
if (cond) {  
    //Something  
} else if (cond2) {  
    //Something  
} else {  
    //Something  
}
```

Opérateurs

>, >=

<, <=

==

===

switch

```
switch(var) {  
    case v1:  
        //Something  
        break;  
    default:  
        //Something  
        break;  
}
```



# **Exercice 1**

## **Manipulation de chaînes**

16

## **./exercice-1/string.js**

### **ucfirst**

1ère lettre en MAJ

hello world => Hello world

### **capitalize**

1ère lettre de chaque mot en MAJ

hello world => Hello World

### **camelCase**

Capitalize + coller les mots

hello world => HelloWorld

### **snake\_case**

Joindre les mots par des underscores en MIN

### **leet** - Cryptage (uniquement les voyelles)

anaconda => 4n4c0nd4

A=>4, E=>3, I=>1, O=>0, U=>(\_), Y=>7

### **prop\_access**

"animal.type.name" => animal["type"]["name"]

prop\_access(prairie, "animal.type.name")

=> prairie.animal.type.name => "chien"

Si attribut non existant, afficher le chemin

jusqu'à l'attribut => "animal.gender not exist"

### **verlan**

Inverser chaque mot d'une phrase

Hello world => olleH dlrow

### **yoda**

Inverser la position des mots d'une phrase

Hello world => world Hello

### **vig** - Cryptage => Chiffre de Vigenère

wikipedia + crypto => yzixisfzy

Commit : [DONE] exercice 1