

# Javascript – les bases



Bailly Benjamin



Qu'est ce que c'est ?

# Rajouter du JS



**01** Directement dans l'html

**02** Dans les balises scripts

**03** Inclure comme pour le  
css notre fichier JS





TP-1

# TP-1 console.log

- Installez Node JS
- Installez sur vs code les extensions qui pourrez vous aider aux développement en JS.  
(javascript code snippet)
- Créez un fichier js avec le code pour afficher un message dans la console.
- Lancez le débbugger pour voir si le code marche bien.

# TP-1

```
JS 1.js  
1 console.log("hello men ! ");  
2
```

Console.log('son texte')  
permet d'afficher un message dans la  
console.

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE  
  
D:\install prog\node.js\node.exe .\1.js  
hello men !
```

# TP-2 variables et concaténation

- Créez une variable contenant votre nom
- Créez une variable contenant votre prénom
- Faites un `console.log` des deux variable afin d'afficher votre nom et prénom à la suite (attention à bien rajouter un espace entre les deux).

# TP-2 variables et concaténation

```
JS 2.js > ...  
1  let nom = "Bailly";  
2  
3  let prenom = "Benjamin";  
4  
5  console.log(nom + " " + prenom)
```

Déclaration de  
variable

Ajoute de l'espace

Il est possible de déclarer des variables à l'aide de  
« let » ou « var ».  
« const » permet de déclarer des constantes.

Pour concaténer il suffit d'ajouter un + entre chaque  
variables.

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

```
D:\install prog\node.js\node.exe .\2.js  
Bailly Benjamin
```



# TP-2 variables et concaténation

- Créez une variable contenant votre nom
- Créez une variable contenant votre prénom
- Faites un `console.log` des deux variable afin d'afficher votre nom et prénom à la suite (attention à bien rajouter un espace entre les deux).



# TP-2-2 Debugger

Prenons ensemble 5 minute pour regarder comment marche le Débugger de vs code.

# TP-3 Opérateurs

Il y a plusieurs opérateurs en js :

- « + » addition
- « - » soustraction
- « \* » multiplication
- « / » division
- « % » modulo
- « \*\* » exponentielle
- Je vous invite à tester chacun d'eux en déclarant 2 variables contenant chacune un nombre et de stocker le résultat de chaque opération dans une autre variable.
- Console.log les résultat.

# TP-3 Opérateurs

```
JS 3.js  X
JS 3.js > ...
1  let nombre1 = 5;
2  let nombre2 = 2;
3
4  let result = nombre1+nombre2; //7
5  console.log(result);
6
7  //ajouter 5 au résultat
8  result += 5;
9  console.log(result);
10 // rajouter 1 au résultat
11 result++;
12 console.log(result);
13 //enlever 1 au résultat
14 result--;
15 console.log(result);
16 |

TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE
D:\install prog\node.js\node.exe .\3.js
7
12
13
12
```

Il est possible de modifier un nombre à l'aide de :

- « += » permet d'additionner avec le nombre de notre choix
- « -= » permet de soustraire avec le nombre de notre choix
- « ++ » ajoute 1
- « -- » enlève 1

# TP-3 Opérateurs

Il y a plusieurs opérateurs en js :

- « + » addition
- « - » soustraction
- « \* » multiplication
- « / » division
- « % » modulo
- « \*\* » exponentielle
- Je vous invite à tester chacun d'eux en déclarant 2 variables contenant chacune un nombre et de stocker le résultat de chaque opération dans une autre variable.
- Console.log les résultat.

# Inclure un script dans un fichier html

Il est important comme pour le css de séparer nos fichiers, nous allons donc créer notre premier script lié à notre page html dans un dossier à part.

Ou incluons nous nos scripts et pourquoi ? Je vous invite à faire la recherche.



# TP-4 Boites de dialogues

- Créez un fichier html et faites appel à un script afin d'afficher un message en boite de dialogue à l'aide de la fonction `alert()`.
- Essayez d'afficher un message de confirmation à l'aide de la fonction `confirm()`.

# TP-4 Boites de dialogues / condition

- Maintenant que vous savez afficher une demande de confirmation et une alerte, nous allons afficher une alerte différente selon si l'utilisateur à confirmer ou pas.
- Utilisez la condition « if / else » pour demander quelques chose à l'utilisateur
- Afficher une alerte différente selon si l'utilisateur confirme ou non.

# TP-4 Boites de dialogues / condition

JS 4.js

```
1  if(confirm("Est tu fort ? ")){
2      alert("Wow ces muscles ! ")
3  }
4  else{
5      alert("Tu es faible !")
6  }
```

# TP-4-2 Boites de dialogues

- A l'aide de la fonction « `prompt()` » demandez une informations à l'utilisateurs.
- Stockez cette information dans une variable et affichez la .

# TP-4-2 Boites de dialogues

JS 4-2.js > ...

```
1  let name = prompt("Comment vous appelez vous ? ");  
2  
3  alert("Bonjour " + name);
```



# TP-5 Fonctions

- Créez une fonctions permettant de `console.log` un message.





# TP-5 Fonctions avec paramètre

- Créez une fonction avec un nom en paramètre permettant d'afficher un message avec le nom de la personne.

# TP-5 Fonctions

JS 5-fonctions.js > ...

```
1  function hello(){
2    |   console.log("hello ! ");
3  }
4  function helloName(name){
5    |   console.log("hello " + name);
6  }
7  hello();
8  helloName("Benjamin");
```



# Fonctions

Attention ! Les variables déclarer dans une fonction, ne seront accessible que dans celle-ci, idem avec les variables déclarées dans les conditions.

# Fonctions

Il est possible de mettre un paramètre par défaut à une fonction en l'assignant directement dans sa déclaration, comme ceci :

```
function boyOrGirl(sexe="Fille"){  
    console.log("Félicitation votre bébé est de  
sexe " + sexe);  
}
```


Une fonction ne sert pas qu'à afficher un message, vous verrez il sera souvent utile de stocker le résultat d'une fonction à l'aide d'un « return » comme ceci :

```
function addition(nombre1, nombre2){  
    result = nombre1 + nombre2;  
    return result;  
}
```

# Tp-6 fonctions

- Créez une fonction permettant de demander l'âge de l'utilisateur.
- Créez une alerte dans la fonction affichant le message disant que l'utilisateur aura bientôt son âge + 1 an (en faisant le calcul dans la fonction alert()).

# Tp-6 fonctions

```
JS 6.js >  nextAge
1  function nextAge(){
2      let age = prompt("quel âge avez vous");
3      age = Number(age);
4      alert("Vous aurez bientôt " + (age +1));
5  }
6
7  nextAge();
```



# Tp-7 fonctions

Créez une fonction permettant d'afficher l'IMC d'une personne, en rentrant comme paramètre son poids et sa taille.

Pour info : La formule de l'IMC est  $\text{poids}(\text{en kg}) / \text{taille}^2(\text{en mètre})$ .

L'utilisateur devra rentrer son poids en kg, donc à vous de faire la conversion cm->mètre, de faire le calcul nécessaire pour l'IMC et d'afficher le résultat arrondi à 1 chiffre après la virgule.

```
let poids = prompt("Quel est votre poids en kg ?");  
let taille = prompt("Quel est votre taille en cm ?");
```

# Tp-7 fonctions

```
function imc(poids, taille){  
    taille = taille/100;  
    taille = Math.pow(taille,2);  
    let result = poids / taille;  
    result = result.toFixed(1);  
    return result;  
}
```

```
let poids = prompt("Quel est votre poids en kg ?");  
let taille = prompt("Quel est votre taille en cm ?");  
  
alert(imc(poids,taille));
```

# Les fonctions fléchées

Pour les « petites fonctions » nous pouvons faire des fonctions fléchées à la volé sur une seule ligne, stocké dans une variable.

Syntaxe :

```
let fonctionFlechee = (message) => console.log(  
message);
```

```
fonctionFlechee('hello');
```

# Tp 8 conditions / opérateurs

== : égal à la valeur

=== : égal à la valeur et au type

!= : différent de la valeur

!== : différent de la valeur et du type

> : supérieur

< : inférieur

>= : supérieur ou égal

<= : inférieur ou égal

! : Pas

Pour vérifier plusieurs conditions : && (et) || (ou).

If (si)

If else (et si)

Else (sinon)

Créez une fonction qui affiche « c'est le matin » si l'heure est inférieur à midi, « c'est l'après midi » si l'heure est entre 12h et 18h et « c'est le soir » si l'heure est au dessus.

# TP 8 conditions / opérateurs

```
function timeOfDay(heure){  
    if(heure >= 0 && heure < 24){  
        if(heure < 12){  
            console.log("c'est le matin");  
        }  
        else if(heure<18){  
            console.log("c'est l'après-midi");  
        }  
        else{  
            console.log("c'est la soirée");  
        }  
    }  
    else{  
        console.log("rentrez une heure valable !")  
    }  
}  
  
timeOfDay(12.22);
```

# TP 9 conditions / switch / ternaire

Dans certains cas il peut être plus utile de faire une condition avec un switch, plutôt qu'un if/else.

- Faites une comparaison en utilisant un switch une variable prénom.
- Dans le switch vous pouvez checker 3 prénoms filles, 3 garçon.
- Votre programme demande votre prénom (prompt).
- Si vous rentrez un des 3 prénoms garçon un console log « vous êtes un garçon ».
- Si vous rentre un des 3 prénoms filles, un console log « vous êtes une filles ».
- Sinon console.log « vous n'avez pas rentré de prénom ».
- Attention, la comparaison ne doit pas être sensible à la casse (case insensitive).
  
- Bonus : Créez une condition ternaire ( ex : si tu as moins de 18 ans tu est mineur sinon tu es majeur) .



# TP 9 conditions / switch / ternaire

```
let surname = prompt("Quel est votre prénom");
let men = "Tu est un gars";
let woman = "Tu est une femme";
let error = "Ceci n'est pas un prénom";
switch(surname.toLowerCase()){
  case "jean":
  case "bernard":
  case "benjamin":
    console.log(men);
    break;
  case "jeanne":
  case "maria":
  case "fatima":
  case "ophelie":
    console.log(woman);
    break;
  default:
    console.log(error);
    break;
}
age = prompt("Quel âge as tu ?")
age >= 18 ? console.log("tu est majeur") : console.log("tu est mineur")
```

# TP 10 boucles de base while/doWhile/for

Il existe plusieurs type de boucles :

- La boucle while, C'est la boucle de base qui permet de faire une action tant qu'une condition est vrai, ATTENTION aux boucles infinies, sous peine de faire planter votre page.
- La boucle do-while, permet de faire une action au moins une fois et sera répété tant que la condition sera vrai.
- La boucle for, permet de combiner la déclaration de variable, la condition et l'incrément.
- Il est possible de casser une boucle avec « break ».
- Il est possible d'ignorer une itération de boucle avec « continue ».

Je vous invite à faire une boucle de chaque.

Pour la boucle do while créez un programme qui demande le prénom tant que celui-ci est vide (prompt et alert)

# TP 10 boucles de base while/doWhile/for

```
let i = 1
while(i < 5){
  console.log(i);
  i++;
}
```

```
for(let i = 1; i < 5 ; i ++ ){
  console.log(i);
}
```

```
let prenom = "prenom"
do{
  prenom = prompt("quel est votre prénom ?");
}
while(
  prenom == "" || prenom==null
)
alert("bonjour " + prenom);
```

# TP 11 Exception

Les exceptions en Js vont permettre d'intercepter les erreurs. (try : code que l'on analyse, catch : code envoyé en cas d'erreur.)

Exemple :

```
try{
    console.log("coco")
}
catch{
    console.log("titi")
}
```

Ici le code dans le catch va être ignoré car aucune erreur avec ce code.

```
try{
    console.log(coco)
}
catch{
    console.log("titi")
}
```

Ici nous allons rentrer dans le « catch » car coco est une variable qui n'existe pas. A savoir que dans la console les erreurs lié aux try ne s'afficheront plus. Il est possible de les afficher avec un `console.error(error.stack)`

# TP 11 Exception

Pour ce petit tp vous allez créer votre propre erreur à l'aide de « `throw new Error()` »

- Créez un programme qui demande à l'utilisateur de choisir entre 3 sorts (boule de feu – trait de glace et chaine d'éclair).
- boule de feu = 40 dégats, trait de glace = 35 dégats et chaine d'éclair = 25 dégat.
- Si l'utilisateur choisit l'un de ses trois sorts, une alerte apparaît disant que vous avez choisit le « nom du sort » qui fait les dégats annoncé ci-dessus.
- Si l'utilisateur choisit un sort autre que les trois ci-dessus, vous créez une erreur et l'affichez.

« Switch case » vous sera utile pour cette exercice.

```
throw new Error("Les autres sorts ne sont pas de votre niveau")
```

# TP 11 Exception

```
try {
    let sort = prompt("Choisissez un nouveau sort parmi les suivant : boule de feu -  
trait de glace - chaine d'éclair");
    let degat;

    switch (sort) {
        case 'boule de feu':
            degat = 40;
            break
        case 'trait de glace':
            degat = 35;
            break
        case 'chaine d\'éclair':
            degat = 25;
        default:
            throw new Error("Les autres sorts ne sont pas de votre niveau")
    }
    alert("Félicitation vous avez choisis " + sort + " qui fait " + degat + " dégats. ")
}
catch(error){
    alert(error);
}
```

# Eval 12 calculateur boite de dialogue

Pour voir si tout vas bien jusque ici nous allons créer un calculateur en boite de dialogue (une sorte de calculatrice simplifiée).

Objectif :

- **Demandez de faire un choix entre addition – soustraction – multiplication – division**
- Indice : pour un retour à la ligne « \n » tant que l'utilisateur ne choisit pas une des quatre propositions il serait bien de répéter la question.
- **Demandez de rentrer un nombre, puis un deuxième**
- Indice : Répéter la demande si un nombre n'est pas rentré, je vous laisse chercher la fonction qui permet de checker cela.
- **Crée 4 fonctions correspond aux 4 méthodes de calculs**
- **Selon le choix de l'utilisateur appelé la fonction correspondante**
- **Affichez le résultat**
- **Proposez à l'utilisateur de recommencer**

isNaN

\n\n

# Les tableaux

Les tableaux sont super utile en langage de programmation et vous en croiserez tout au long de vos aventures de développeur. Il est important de savoir les gérer.

Comment créer un tableau exemple :

```
let tab = ['titi','toto','tata'];
```

Tableau dans un tableau :

```
let tab = [  
  [  
    'titi',  
    'toto',  
    'tata'  
  ],  
  [  
    'truc',  
    'muche',  
    'bidule'  
  ]  
];
```

Tableau associatif / objet :

```
let associatif = {  
  'nom': 'bailly',  
  'prenom': 'benjamin',  
  'age': '27 ans'  
};
```



# Les tableaux – accéder aux éléments

En js et dans les langages de programmation on commence à compter à 0 et non à 1.  
Pour accéder par exemple a titi ci-dessous, on ferait un `console.log(tab[0])` et titi serait affiché.

Tableau dans un tableau, accéder à titi :

```
let tab = [
  [
    'titi',
    'toto',
    'tata'
  ],
  [
    'truc',
    'muche',
    'bidule'
  ]
];
console.log(tab[0][0]);
```

Tableau associatif (objet) :

```
let associatif = {
  'nom': 'bailly',
  'prenom': 'benjamin',
  'age': '27 ans'
};

console.log(associatif['nom'])
```

# Les tableaux – Ajouter des valeurs

Il est possible et il peut être utile dans certains cas de modifier vos tableaux.

```
tab.push('tutu'); // ajout d'une valeur à la fin
```

```
tab.unshift('popo'); // ajout d'une valeur au début
```

```
associatif['sexe']='masculin'; // ajouter une valeur à un  
tableau associatif
```

# Les tableaux – Supprimer des valeurs

```
tab.pop(); // suppression de la dernière valeur du tableau  
tab.shift(); // suppression de la première valeur du tableau
```

```
delete(associatif.nom); // suppression de l'index sélectionné
```

# Les tableaux tp13 – Fonctions magique : splice

Je vous invite à récupérer le tableau ci-dessous et à y rajouter un 3eme tableau à l'aide de la fonction splice().

```
let tab2 = [  
  [  
    'titi',  
    'toto',  
    'tata'  
  ],  
  [  
    'truc',  
    'muche',  
    'bidule'  
  ]  
];
```

# Les tableaux 14 – Comment les parcourir

Il y a plusieurs possibilité pour parcourir les tableaux :

```
let classe = [  
  'jean',  
  'michel',  
  'bertrand',  
  'mamacita'  
]
```

For of

```
for(const nom of classe){  
  console.log(nom);  
}
```

For in

```
for(const nom in classe){  
  console.log(nom);  
}
```

Foreach

```
classe.forEach(prenom=>console.log(prenom))
```

# Les objets 14 – Comment les parcourir

Il y a plusieurs possibilité pour parcourir les tableaux :

```
let classe = {  
  'prenom': 'jean',  
  'nom': 'michel',  
  
  bouleDeFeu : () => console.log("boule de feu")  
};
```

Foreach

```
Object.values(classe).forEach(value=>{  
  console.log(value)  
})  
  
Object.keys(classe).forEach(key=>{  
  console.log(value)  
})
```

For in

```
for (const index in classe){  
  console.log(classe[index])  
};  
}
```

# Les tableaux Memento

```
let fruits = ['pomme', 'banane', 'poire', 'fraise']
```

- **fruits.length** : retourne le nombre d'éléments dans le tableau (ici retourne 4)
- **fruits[0]** : sélectionne le premier élément
- **fruits[length - 1]** : sélectionne le dernier élément
- **fruits.push('pamplemousse')** : ajoute un élément à la fin du tableau
- **fruits.unshift('pamplemousse')** : ajoute un élément au début du tableau
- **fruits.pop()** : supprime le dernier élément
- **fruits.shift()** : supprime le premier élément
- **fruits.indexOf('banane')** : retourne l'index d'un élément
- **fruits.join()** : concatène les éléments dans une chaîne de caractères avec virgules, mais il est possible de spécifier un autre séparateur dans les parenthèses
- **fruits.slice()** : crée une copie du tableau (à associer à une autre variable donc)
- **fruits.splice([début], [nbASupprimer], [élément(s)])** : retire, remplace ou ajoute des éléments.
  - **Début** : l'index à partir duquel commencer le changement, si négatif, part de la fin du tableau
  - **nbASupprimer** : un entier indiquant le nombre d'éléments à retirer ou remplacer
  - **Element(s)** : les éléments à ajouter à partir du début précisé. Si aucun élément n'est spécifié, alors n'en ajoutera pas.

# Tp-15- parcourir tableau

Consigne :

- Créez une fonction qui permet de lire parcourir et de console.log les deux tableaux ci-dessous.
- Le résultat attendu est celui-ci :

```
let formateur = ['Ophelie', 'Benjamin', 'Mathieu', 'Leo']
```

```
let benjamin = {  
  'nom': 'bailly',  
  'prenom': 'benjamin',  
  'force': 'extreme',  
  'ego': 'surdimensionné'  
}
```

```
0 : Ophelie  
1 : Benjamin  
2 : Mathieu, Leo  
  
nom : bailly  
prenom : benjamin  
force : extreme  
ego : surdimensionné
```



# Tp-15- parcourir tableau

```
function readTab(tab){  
    let data ="";  
    for (let index in tab){  
        data += (index + ' : ' + tab[index] + '\n');  
    }  
  
    console.log(data);  
}  
  
readTab(formateur);  
readTab(benjamin);
```

# Les closures / portée des variables

Une closure est quelque chose qui va être créé lorsque l'on va créer une fonction dans une fonction. C'est la mémoire associée à celles-ci.

Lorsque une variable est déclarée localement (dans un scope, comme une fonction) elle n'est utilisable que localement. Il est possible d'utiliser une closure pour sauvegarder la mémoire d'une variable locale au sein d'une fonction.

```
function timer(){  
    let num = 0;  
  
    return function closure(){  
        ++num;  
        return num;  
    }  
}
```

```
let compteur = timer();  
console.log(compteur());  
console.log(compteur());  
console.log(compteur());  
  
let compteur2 = timer();  
console.log(compteur2());  
console.log(compteur2());
```

# Les objets

En JavaScript, un objet est une entité à part entière qui possède des propriétés.

Si on effectue cette comparaison avec une tasse par exemple, on pourra dire qu'une tasse est un objet avec des propriétés. Ces propriétés pourront être la couleur, la forme, le poids, le matériau qui la constitue, etc.

De la même façon, un objet JavaScript possède des propriétés, chacune définissant une caractéristique.

Exemple :

```
let chien = {  
  race: 'caniche',  
  poil: 'long',  
  
  aboyer: () => console.log('Ouaaf ouaf')  
}  
  
chien.aboyer();
```