Javascript

Variables let ou var (privilégier let)

```
const name = "Fabien" //constante
let age = 40 //privilégier le let
var place = "Lyon"
```

Conditions

```
 if(test===true){
   console.log("Hello 🙋‍♂️") //Affiche dans la console un
message
 }else{
   console.error("Error 🙈") // Affiche un message d'erreur
dans la console
 }

 //inliner
 if (testVar)console.log("Hello 🙋‍♂️");

//opération ternaire
const displayMessage = testVar ? "Hello 🙋‍♂️" : "Error 🙈";
```

Tableaux

```
const myArray=["toto",148,"Hello"]

 //Taille du tableau
 console.log(myArray.length); //3

 //Boucle for
 for (const arrayElement of myArray) {
   console.log(arrayElement);
 }

 //Boucle foreach
 myArray.forEach((arrayElement) =>
console.log(arrayElement));
```

Javascript

Fonctions

```
//Définition d'une fonction
function displayMessage(msg) {
 console.log(`Message ${msg}`);
 return true;
}

//Appel d'une fonction
displayMessage("Hello world ! 🙋‍♂️");

//fonction fléchée
const maFonctionFlechee = (maVariable) => {
 console.log(`Message ${msg}`);
 return true;
};

//fonction fléchée en une seule ligne (return implicite)
const autreFonctionFlechee = (a, b) => a + b;
console.log(autreFonctionFlechee(5, 5)); //10
```

Classes

```
class Person {
 lastname;
 firstname;

 constructor(firstname, lastname) {
   this.firstname = firstname;
   this.lastname = lastname;
 }

 sayHello() {
   console.log(`Bonjour je suis ${this.firstname}
${this.lastname} 👋`);
 }
}

//Initialisation et utilisation de la classe
const jd = new Person("Jean", "Dupont");
jd.sayHello(); //Bonjour je suis Jean Dupont 👋
```

Javascript

Héritage de classes

```
class Employee extends Person {
 job;

 constructor(firstName, lastName, job) {
   super(firstName, lastName);
   this.job = job;
 }

 sayYourJob() {
   console.log(`et je suis ${this.job}`);
 }
}
//Initialisation et utilisation de la classe
const ey = new Employee("Evan", "You", "developpeur");
ey.sayHello(); //Bonjour je suis Evan You 👋
ey.sayYourJob(); //et je suis developpeur
```

Modules

Export et import d'un seul élément

Fichier 1 func.js

```
const func = (message)=>console.log(message)
export default func
```

Fichier 2 main.js

```
import func from "./func"
func("Hello world")
```

Export et import de plusieurs éléments

Fichier 1 func.js

```
const func = (message) => console.log(message);
const person={firstname:"Léa",lastname:"Giorni"}
export {func,person};
```

Fichier 2 main.js

```
import {func,person} from "./func"
```

Javascript

Objets

```
//Définition d'un objet
const person={firstname:"Fabien",address:"Lyon",gender:"M"}

console.log(person.firstname); //Fabien

//Ajout d'une propriété
person.age=40

//Object destructuring (permet de "prendre" certains
éléments d'un objet)
const { firstname, address: city } = person;
console.log(`${firstname} habite ${city}`); //Lyon
```

Date

```
//Creation d'un objet de date
const dateObject = new Date("01/02/2023");
const dateFormat1 = dateObject.toLocaleDateString();
//1/2/2023
const dateFormat2 = dateObject.toISOString();
//20230101T23:00:00.000Z

//Ajout d'un jour
dateObject.setDate(dateObject.getDate() + 1);
const dateFormat3 = dateObject.toLocaleDateString(); //
1/3/2023
```

Javascript

Fonctions utiles ES2020

```
//map crée un nouveau tableau à partir d'un tableau
const numbers = [1, 2, 3, 4, 5];
const numbersDoubled = numbers.map((number) => number * 2);
console.log(numbersDoubled); //[2, 4, 6, 8, 10]

//filter permet de filtrer un tableau
const array = [1, "banane", 10, "bateau", 22];
const arrayNumbers = array.filter((element) => {
 if (typeof element === "number") return element;
});
console.log(arrayNumbers); //[1, 10, 22]

//reduce réduit un tableau en une valeur
const weightInKg=[10,20,500]
const totalWeight=weightInKg.reduce((accumulator,currentValue)=>accumulator+=currentValue,0)
console.log(totalWeight); //530
```

Typescript

```ts
// Définition d'un type
let lastname: string = "Léonard";
lastname = 42; //error

//Inférence de type
let unNombre=42
unNombre="coucou" // error car inférence de type

//Tableau d'un type
const arrayNames:string[] = ["Ahmed", "Léa", "Bernard"];
//tableau de string
arrayNames.push(42) //error

// Tableau de plusieurs types
const tableauMixte:(string|number)[]=["hello",42,"world"]

//Forcer un typage quand on ne connait pas le type (ex call http)
let result:string
result= "data" as string
```

Fonctions

```ts
// Il faut typer les arguments (ici 2 nombres) et le retour (ici nombre ou chaîne)
const myFunc = (a: number, b: number): number | string => {
 if (typeof a !== "number" || typeof a !== "number") {
   return a + b;
 } else {
   return "a et b doivent être des nombres";
 }
};
```

TS Typescript

Interface et types

```
//Création d'un type
type Gender = "H" | "F" | "NB";

//Création d'une interface
interface IPerson {
 id?: string; // facultatif grâce au ?
 gender: Gender;
 name: string;
 age?: number;
}

//Extension d'une interface existante
interface IEmployee extends IPerson {
 job: string;
}
```

Utilisation d'un type et d'une interface

```
//Implémentation d'un type
const sexe: Gender = "F";

//Implémentation d'une interface
const person: IEmployee = {
 name: "Fabien",
 gender: "H",
 job: "développeur",
};
```

Fonctions

```
// Il faut typer les arguments (ici 2 nombres) et le retour (ici nombre ou chaine)
const myFunc = (a: number, b: number): number | string => {
 if (typeof a !== "number" || typeof a !== "number") {
   return a + b;
 } else {
   return "a et b doivent être des nombres";
 }
};
```

TS Typescript

Classes

```
interface IPerson {
 firstname: string;
 lastname: string;
 sayHello(): void;
}

class Person implements IPerson {
 constructor(public firstname: string, public lastname:
string) {}

 sayHello() {
   console.log(`Bonjour je suis ${this.firstname}
${this.lastname} 👋`);
 }
}

//Initialisation et utilisation de la classe
const jd = new Person("Jean", "Dupont");
jd.sayHello(); //Bonjour je suis Jean Dupont 👋
```

Generics

```
//T est un paramètre de type
function displaySomething<T>(something: T) {
 console.log(
   "Affichage d'un variable dont le type est passé en
paramêtre (T)",
   something
 );
}

displaySomething<string>("Hello");
displaySomething<number>(42);
```