Javascript

Basics

```
//il faut mettre un ";" à la fin de chaque ligne de code

console.log("Hello world!");   //afficher un message dans la console
console.log("Hello"+"world!"); //same

let a;     //déclaration variable
a = 3;     //initialisation
a++;       //a = 4
a += 1;    //a = 5
let b = (1 + 1)**2; //déclaration + initialisation ; b = 4

let a;
let a;     //erreur

let a = 3;
a = String(a); //Typecast
a = Number(a); //Typecast

const a = 3; //constant variable
a = 4;           //erreur
let const b;     //erreur

const country = "France";
console.log("Je vis en"+country+".");

{ ... }          //code block (acts on the scope of variables)

let num1 = 0;
{
  num1 = 1;      //ok : num1 est déclarée dans le bloc parent
  const num2 = 0;
}

const country = "France";
console.log(`Je vis en ${country}`);  //this is called a "template literal"
                                      //altgr+µ puis espace to type a backtit `
                                      //les template literals permettent de repérer les ${...}

console.log(`Je vis en ${"France"}`); //works too

const x = 3;
const y = 7;
console.log(`${x} + ${y} = ${x + y}`); //"3 + 7 = 10"

const f = 100;
console.log("f contient " + f); //conversion implicite

const g = "cinq" * 2; //ne lance pas d'erreur même si ca n'a pas de sens g = Nan
console.log(g); //NaN

const h = "5";
console.log(h + 1); //51

const h = 5;
console.log(h + 1); //6

const h = 5;
console.log(h + "1"); //51
```

Interactions avec l'utilisateur

```
const prenom = prompt("Entrez votre prénom :"); //affiche une boîte de dialogue.
                                                //la variable prénom est de type String. Par défaut ce
qui est renvoyé par prompt est de type string

const nb = Number(prompt("Entrez un nombre : ")); // nb est de type
nombre
```

```
alert(`Bonjour, ${prenom}`); //ouvre une nouvelle boîte

//Les instructions  prompt()  et  alert()  fonctionnent uniquement dans le contexte d'un navigateur web.
//Les exemples de code interactifs de ce cours utilisent la plate-forme Node.js, dans laquelle ces instructions ne sont pas supportées.

const temp1 = 36.9;
const temp2 = 37.6;
const temp3 = 37.1;
console.log(temp1, temp2, temp3); // La virgule tient lieu d'espace: "36.9 37.6 37.1"
```

Boucles IF et opérateurs logiques

```
const nombre = Number(prompt("Entrez un nombre :"));
if (nombre > 0) {
  console.log(nombre + " est positif");
} else if (nombre < 0) {
  console.log(nombre + " est négatif");
} else {
  console.log(nombre + " est nul");
}
```

ATTENTION LES PARENTHESES SONT OBLIGATOIRES POUR UN IF!!!

```
=== // égal. On peut également utiliser "==" mais c'est déconseillé

!== // différent. On peut également utiliser "!=" mais c'est déconseillé
```

L''égalité faible (==) effectuera une conversion des deux éléments à comparer avant d''effectuer la comparaison
L''égalité stricte (===) effectuera la même comparaison mais sans conversion préalable (elle renverra toujours false si les types des deux valeurs comparées sont différents)

```
<  // strictement plus petit

<= // plus petit

&& // "et" logique

|| // "ou" logique

!  // "non" logique
```

```
 const meteo = prompt("Quel temps fait-il dehors ?");
switch (meteo) {
  case "soleil":
    console.log("Sortez en t-shirt.");
    break;
  case "vent":
    console.log("Sortez en pull.");
    break;
  case "pluie":
    console.log("Sortez en blouson.");
    break;
  case "neige":
    console.log("Restez au chaud à la maison.");
    break;
  default: //ceci est optionnel
    console.log("Je n'ai pas compris !");
}



const x = "abc";
switch (x) {
  case "abc":
    console.log("x vaut abc");
    // pas de break : on passe au bloc suivant !
  case "def":
    console.log("x vaut def");
    break;
}
//L''exécution de cet exemple affiche deux messages :"x vaut abc" (résultat attendu) mais aussi"x vaut
```

```
def".

=> les "break" sont indispensables!!
```

Boucles WHILE et FOR

```
console.log("Début du programme");
let nombre = 1;
while (nombre <= 5) {
  console.log(nombre);
  nombre++;
}
console.log("Fin du programme");


let compteur; //pas indispensable de déclarer avant
for (compteur = 1; compteur <= 5; compteur++) {
  console.log(compteur);
}

for (let i = 1; i < 5; i++) {

   console.log("J'ai faim !");

}
```

Fonctions

```
function direBonjour() {
  console.log("Bonjour !");
}
direBonjour();

////////////////////////////////////////

function direBonjour() {
  return "Bonjour !";
}

const resultat = direBonjour();
console.log(resultat); // "Bonjour !"

////////////////////////////////////////

function direBonjour() {
  const message = "Bonjour !";
  return message;
}

console.log(direBonjour()); // "Bonjour !"

////////////////////////////////////////

function direBonjour(prenom) {
  const message = `Bonjour, ${prenom} !`;
  return message;
}

console.log(direBonjour("Baptiste")); // "Bonjour, Baptiste !"
console.log(direBonjour("Sophie")); // "Bonjour, Sophie !"

////////////////////////////////////////

const bonjour = function(prenom) {

  const message = `Bonjour, ${prenom} !`;

  return message;

}

console.log(bonjour("Thomas")); // "Bonjour, Thomas !"
```

La fonction créée ci-dessus est anonyme et directement affectée à la variable bonjour. "Fonction

```
anonyme"
La valeur de cette variable est donc une fonction. Cette manière de créer une fonction est appelée
expression de fonction.

//////////////////////////////////////////

const bonjour = (prenom) => {
  const message = `Bonjour, ${prenom} !`;
  return message;
}

console.log(bonjour("Thomas")); // "Bonjour, Thomas !"

Syntaxe plus concise que ci-dessus: "Fonction fléchée"

//////////////////////////////////////////

const bonjour = prenom => `Bonjour, ${prenom} !`;

console.log(bonjour("Thomas")); // "Bonjour, Thomas !"

Encore plus concis. On ne peut faire ca que si la fonction n''a qu''un seul argument
```

Objets

```
const stylo = {
  type: "bille",
  couleur: "bleu",
  marque: "Bic"
};

// "J'écris avec un stylo bille bleu de marque Bic"
console.log(`Jécris avec un stylo ${stylo.type} ${stylo.couleur} de marque ${stylo.marque}`);

Cette manière de créer des objets est appelée syntaxe littérale.
Le point-virgule après l''accolade fermante est optionnel
(mais il vaut mieux prendre l''habitude de l''ajouter systématiquement pour éviter certaines mauvaises
surprises dans des cas particuliers)

"ON DECLARE TOUS LES OBJETS COMME "CONST", CECI SIGNIFIE QUON NE PEUT PAS FAIRE "stylo = {};" MAIS ON
PEUT MANIPULER LOBJET EN LUI AJOUTANT DES ELEMENTS ETC."
//////////////////////////////////////////

const stylo = {
  type: "bille",
  couleur: "bleu",
  marque: "Bic"
};

// Modification de la propriété "couleur"
stylo.couleur = "rouge";

// "J'écris avec un stylo bille rouge de marque Bic"
console.log(`J'écris avec un stylo ${stylo.type} ${stylo.couleur} de marque ${stylo.marque}`);

//////////////////////////////////////////

const stylo = {
  type: "bille",
  couleur: "bleu",
  marque: "Bic"
};

// Ajout de la propriété "prix"
stylo.prix = "2.5";

// "Mon stylo coûte 2.5 euros"
console.log(`Mon stylo coûte ${stylo.prix} euros`);

//////////////////////////////////////////

const aurora = {
  nom: "Aurora",
  sante: 150,
  force: 25,
```

```
  // Renvoie la description du personnage
  decrire() {
    return `${this.nom} a ${this.sante} points de vie et ${this.force} en force`;
  }
};

// "Aurora a 150 points de vie et 25 en force"
console.log(aurora.decrire());

////////////////////////////////////////
```

Objets prédéfinis

console et Math sont deux objets prédéfinis

```
////////////////////////////////////////////////////////////////////

console.log(Math.min(4.5, 5)); // 4.5
console.log(Math.min(19, 9));  // 9
console.log(Math.min(1, 1));   // 1

console.log(Math.random()); // Un nombre aléatoire entre 0 et 1
```

Tableaux

```
const films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];

console.log(films[0]); // "Le loup de Wall Street"
console.log(films[1]); // "Vice-Versa"
console.log(films[2]); // "Babysitting"

console.log(films.length); // 3

//Utiliser un indice invalide pour accéder à un élément d'un tableau JavaScript renvoie la valeur
spéciale undefined.

console.log(films[3]); // undefined : le dernier indice valide est 2. Pas d'erreur explicite!

//tableau

const tableau = ["Bonjour", 7, { message: "Coucou maman" }, true];

//objet dans un tableau

////////////////////////////////////////////////////

//BOUCLE FOR SUR UN TABLEAU

const films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];

for (let i = 0; i < films.length; i++) {
  console.log(films[i]);
}

////////////////////////////////////////////////////

//METHODE forEach

const films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];

films.forEach(film => {
  console.log(film);
});

C''est pareil qu'avec une boucle for

////////////////////////////////////////////////////

//BOUCLE FOR ... OF

const films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];
```

```
for (const film of films) {
  console.log(film);
}

C''est pareil qu'avec une boucle for

////////////////////////////////////////////////

//METHODE PUSH

const films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];

films.push("Les Bronzés"); // Ajoute le film à la fin du tableau
console.log(films[3]); // "Les Bronzés"

////////////////////////////////////////////////

//METHODE UNSHIFT

const films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];

films.unshift("Les Bronzés"); // Ajoute le film au début du tableau
console.log(films[0]); // "Les Bronzés"

Ceci permet d''ajouter un élément au début du tableau

////////////////////////////////////////////////

//METHODE POP

const films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];

films.pop(); // Supprime le dernier élément
console.log(films.length); // 2
console.log(films[2]); // undefined

////////////////////////////////////////////////

//METHODE SPLICE

const films = ["Le loup de Wall Street", "Vice-Versa", "Babysitting"];

films.splice(0, 1); // Supprime 1 element à partir de l'indice 0
console.log(films.length); // 2
console.log(films[0]); // "Vice-Versa"
console.log(films[1]); // "Babysitting"

Ceci permet de supprimer un range dans le tableau
```

Strings

```
//LONGUEUR

console.log("ABC".length); // 3
console.log("Je suis une chaîne".length); // 18

////////////////////////////////////////////////////////

//METHODES toLowerCase, toUpperCase

const motInitial = "Bora-Bora";
console.log(motInitial.toLowerCase());  // "bora-bora"
console.log(motInitial.toUpperCase());  // "BORA-BORA"
console.log(motInitial);  // "Bora-Bora"

-> La chaîne de caractère n''est modifiée que comme R-value et pas comme L-value

////////////////////////////////////////////////////////

//COMPARAISON

const chaine = "azerty";
console.log(chaine === "azerty"); // true
console.log(chaine === "qwerty"); // false

IL EST DECONSEILLE DUTILISER "=="
```

```
//////////////////////////////////////////////////////

//ACCEDER A DES ELEMENTS

const sport = "Tennis-ballon"; // 13 caractères
console.log(sport[0]); // "T"
console.log(sport[6]); // "-"
console.log(sport[13]); // undefined (longueur dépassée).. PAS DERREUR LANCEE!!!

//////////////////////////////////////////////////////

//BOUCLER SUR LES ELEMENTS

const prenom = "Odile";
for (let i = 0; i < prenom.length; i++) {
  console.log(prenom[i]);
}

const prenom = "Odile";
for (const lettre of prenom) {
  console.log(lettre);
}

//////////////////////////////////////////////////////

//METHODE Array.from()

-> permet de transformer un string en tableau

const prenom = "Odile";
const tabPrenom = Array.from(prenom);
tabPrenom.forEach(lettre => {
  console.log(lettre);
});

//////////////////////////////////////////////////////

//METHODE indexOf()

const chanson = "Honky Tonk Women";
console.log(chanson.indexOf("onk")); // 1
console.log(chanson.indexOf("Onk")); // -1 (à cause du O)

//////////////////////////////////////////////////////

//METHODES startsWith() et endsWith()

const chanson = "Honky Tonk Women";

console.log(chanson.startsWith("Honk")); // true
console.log(chanson.startsWith("honk")); // false

console.log(chanson.endsWith("men")); // true
console.log(chanson.endsWith("Men")); // false

//////////////////////////////////////////////////////

//METHODE SPLIT()

const listeMois = "Jan,Fev,Mar,Avr,Mai,Jun,Jul,Aou,Sep,Oct,Nov,Dec";
const mois = listeMois.split(",");
console.log(mois[0]); // "Jan"
console.log(mois[11]); // "Dec"
```

Classes

```
//RAPPEL OBJETS

const aurora = {
  nom: "Aurora",
  sante: 150,
  force: 25,
  xp: 0,
```

```
  // Renvoie la description du personnage
  decrire() {
    return `${this.nom} a ${this.sante} points de vie, ${
      this.force
    } en force et ${this.xp} points d'expérience`;
  }
};

// "Aurora a 150 points de vie, 25 en force et 0 points d'expérience"
console.log(aurora.decrire());

console.log("Aurora apprend une nouvelle compétence");
aurora.xp += 15;

// "Aurora a 150 points de vie, 25 en force et 15 points d'expérience"
console.log(aurora.decrire());


////////////////////////////////////////////////////////////////////////////////////////////

//CLASSE

//syntaxe générale

class MaClasse {
  constructor(param1, param2 /* ... */) { //METHODE SPECIALE!! MOT RESERVE
    this.propriete1 = param1;
    this.propriete2 = param2;
    // ...
  }
  methode1(/* ... */) {
    // ...
  }
  methode2(/* ... */) {
    // ...
  }
  // ...
}

const monObjet = new MaClasse(arg1, arg2, ...);

//exemple

class Personnage {
  constructor(nom, sante, force) { //METHODE SPECIALE!! MOT RESERVE
    this.nom = nom;
    this.sante = sante;
    this.force = force;
    this.xp = 0; // Toujours 0 au début
  }
  // Renvoie la description du personnage
  decrire() {
    return `${this.nom} a ${this.sante} points de vie, ${
      this.force
    } en force et ${this.xp} points d'expérience`;
  }
}

const aurora = new Personnage("Aurora", 150, 25); //const  signifie qu'on ne peut pas réassigner la
variable aurora; cependant on peut changer ses attributs etc!
const glacius = new Personnage("Glacius", 130, 30);

// "Aurora a 150 points de vie, 25 en force et 0 points d'expérience"
console.log(aurora.decrire());
// "Glacius a 130 points de vie, 30 en force et 0 points d'expérience"
console.log(glacius.decrire());
```

Prototypes

```
//CREATION DUN OBJET A PARTIR DUN AUTRE

const unObjet = { a: 2 };

// Crée unAutreObjet avec unObjet comme prototype
const unAutreObjet = Object.create(unObjet);
```

```
console.log(unAutreObjet.a); // 2
```

Dans cet exemple, l''instruction JavaScriptObject.create() est utilisée pour créer l''objetunAutreObjet en lui donnant comme prototype l''objetunObjet.
Lors de l''appel à unAutreObjet.a, c''est la propriétéa de unObjet qui est utilisée puisque la propriété a n''existe pas dansunAutreObjet!

```
////////////////////////////////////////////////////////////////////////////////////////////////////

// RECURSIVEMENT

const unObjet = { a: 2 };

// Crée unAutreObjet avec unObjet comme prototype
const unAutreObjet = Object.create(unObjet);

console.log(unAutreObjet.a); // 2

// Crée encoreUnObjet avec unAutreObjet comme prototype
const encoreUnObjet = Object.create(unAutreObjet);

console.log(encoreUnObjet.a); // 2
console.log(encoreUnObjet.b); // undefined
```

Ce mode de relation entre les objets JavaScript est appelé délégation : un objet délègue une partie de son fonctionnement à son prototype.
Si le prototype d''un objet ne possède pas une propriété recherchée, alors c''est dans son propre prototype que la recherche **continue**, jusqu''à arriver à la fin de chaîne des prototypes.
Si la propriété n''a été trouvée dans aucun objet, son accès renvoie la valeurundefined.

Le modèle objet de JavaScript est basé sur les prototypes et non sur les classes: une classe JavaScript est elle-même un objet, pas un modèle statique.
"Instancier" une classe crée un nouvel objet lié à un objet prototype.

La syntaxe des classes et le mot-clé **class** ont été introduites par la norme ES2015 afin de faciliter l''utilisation de JavaScript par des développeurs habitués à d''autres langages.
Cependant, le modèle objet de JavaScript reste basé sur les prototypes.

p5.js

Basics

```
//REFERENCES
p5js.org
p5js.org/reference
editor.p5js.org

//////////////////////////////////////////

function setup() {
  createCanvas(400, 400); //taille du rectangle principal
}

function draw() {

  background(5,100,0); //couleur du fond en rgb. Si un seul argument ca va du blanc au noir

  rect(10,20,0,40);    //rect(x,y,largeur,hauteur) -> x et y partent d'en haut à gauche et vont de 0 à 399.
                       //on peut dépasser les indexes auxquels cas ce qui dépasse n'est pas affiché

  rectMode(CENTER);    //to center the rectangle

  line(30, 20, 85, 75); //line(x1, y1, x2, y2) ligne de (x1,y1) à (x2,y2)


}
```