



PROGRAMMATION I

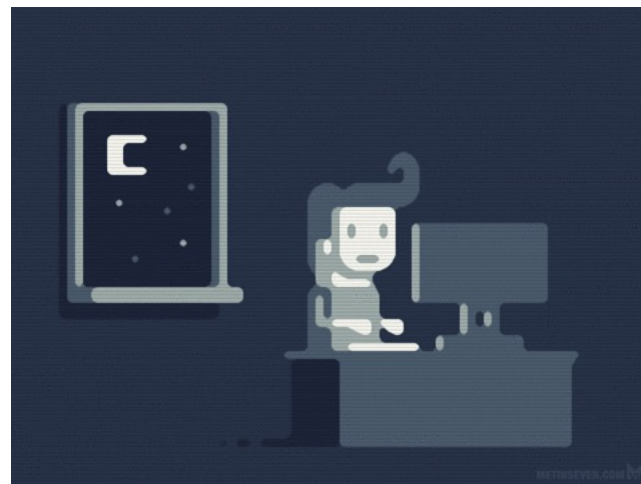
Session 4





Au programme :

- Opérateurs d'incrémentation
- Boucles (*for / while*)
- TurtleJS et les boucles



C'est parti !



Écris un programme qui va afficher tous les nombres entiers de 1 à 20



Opérateur	La valeur de a/b
a++	11
++a	11
b--	9
--b	9

a = 10, b = 10



```
let m = 10;  
m++;  
console.log(m);
```

```
let z = 5;  
console.log(z++);  
console.log(z);
```

```
let k = 5;  
console.log(++k);  
console.log(k);
```



z++ utilise la valeur précédente de la variable, puis augmente le z par 1.



++k augmente la valeur de la variable par 1, puis utilise la nouvelle valeur.

Boucles



```
for (let i = 1; i <=20; i++) {  
  console.log(i)  
}
```



script.js



```
let i = 1;
console.log(i++);
console.log(i++);
console.log(i++);
console.log(i++);
console.log(i++);
console.log(i++);
console.log(i++);
console.log(i++);
console.log(i++);
console.log(i++);
console.log(i++);
```



```
for (let i = 1; i <= 20; i++) {
  console.log(i);
}
```



Les boucles permettent d'exécuter une séquence de commandes plusieurs fois.



Les boucles permettent de raccourcir le code pour le rendre plus performant et plus clair.

Boucle for

Avec l'itération



```
for (let i = 1; i <= 20; i++) {  
  console.log(i);  
}
```



La boucle for est souvent appelée boucle d'**itération**.



Dans cet exemple la variable `i` est l'itérateur.



Initialisation

Condition

Itération



Ce qui est écrit dans le corps sera répété pour chaque i , où i

- Initialise l'itérateur
- Le modifie à chaque étape
- Tant que la condition est vraie

```
for (let i = 1; i <= 20; i++) {  
  console.log("Prochain numéro");  
  console.log(i);  
}
```

Scope du for



Écris des programmes qui affichent les nombres suivantes :

1. Tous les nombres entiers de 5 à 60
2. Tous les nombres entiers de 100 à 5
3. Tous les nombres entiers impairs entre 20 à 200
4. Tous les nombres entiers impairs entre 90 à 30



Écris des programmes qui réalisent les calculs suivants :

1. La somme de tous les nombres entiers de 5 à 60
2. Le produit de tous les nombres entre 5 et 30 divisibles par 7

Boucle while

Avec la précondition



```
let i = 1;  
  
console.log(i++);  
console.log(i++);  
console.log(i++);  
console.log(i++);  
console.log(i++);  
console.log(i++);
```



```
let i = 1;  
  
while (i <= 20) {  
    console.log(i);  
    i++;  
}
```

[Python Tutors: boucle while\(\)](#)



Condition

```
while (i <= 20) {  
  console.log("next number");  
  console.log(i);  
  i++;  
}
```



Ce qui est écrit dans le scope
sera répété tant que la **condition** est
vraie.

Scope du while



Pour relever ce challenge, le capteur de couleur peut être utilisé selon deux modes différents (mode couleur ou mode intensité de la lumière réfléchié).

Le mode intensité apportera une précision maximale.

Il existe donc deux méthodes :

- Détecter un changement de couleur
- Détecter une variation d'intensité

Par exemple :





Écris un programme qui affiche les nombres suivants :

1. Tous les nombres entiers de 5 à 60
2. Tous les nombres entiers de 100 à 5
3. Tous les nombres entiers impairs de 20 à 200
4. Tous les nombres entiers impairs de 90 à 30



Écris des programmes qui réalisent les calculs suivants :

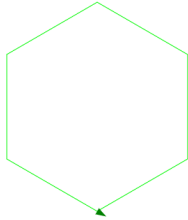
1. La somme de tous les nombres entiers de 5 à 60
2. Le produit de tous les nombres entre 5 et 30 divisibles par 7

Géométrie (Turtle)

Dessine un hexagone



Canvas



```
forward(150);  
left(60);  
forward(150);  
left(60);  
forward(150);  
left(60);  
forward(150);  
left(60);  
forward(150);  
left(60);
```

```
for (let i = 1; i <= 6; i++)  
{ forward(150);  
  left(60);  
}
```

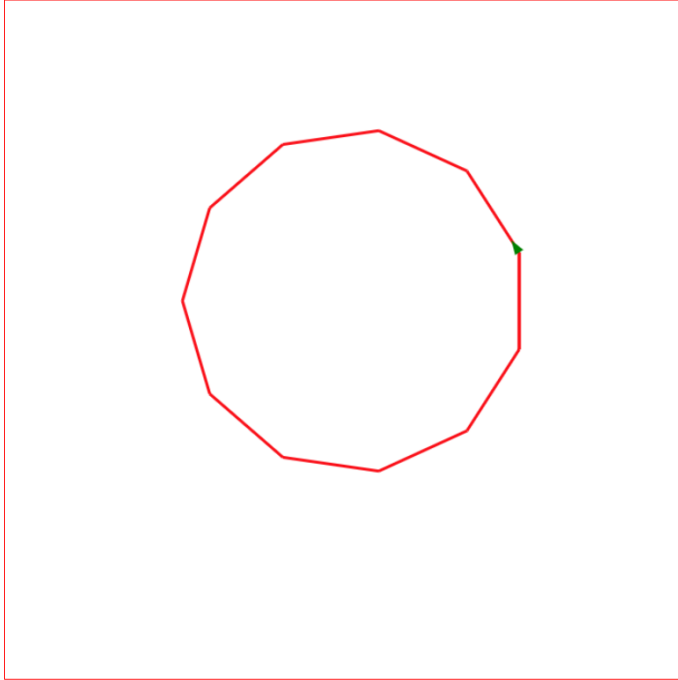


```
for (let i = 1; i <= 6; i++)  
{ forward(150);  
  left(60);  
}
```

```
let i = 0;  
  
while (i <= 6) {  
  forward(150);  
  left(60);  
  i++;  
}
```



Canvas



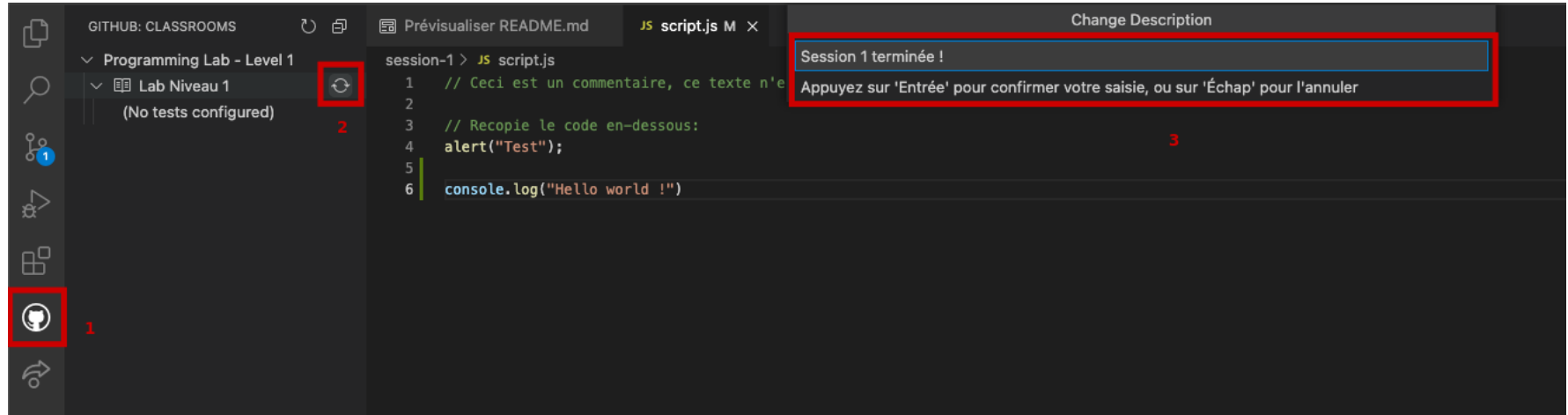
Dessine un polygone à 11 côtés égaux :

1. L'utilisateur saisit la longueur des côtés
2. Le programme doit dessiner la forme

Enregistrer ton travail



Retourne dans l'onglet GitHub et clique sur l'icone  correspondant à ce lab.



The screenshot shows the GitHub Classroom interface. On the left sidebar, under 'GITHUB: CLASSROOMS', there is a section 'Programming Lab - Level 1' containing 'Lab Niveau 1'. A red box labeled '1' highlights the GitHub icon in the sidebar. In the main area, the 'Lab Niveau 1' is selected, and a red box labeled '2' highlights the refresh icon. The code editor shows a file named 'JS script.js' with the following content:

```
1 // Ceci est un commentaire, ce texte n'e
2
3 // Recopie le code en-dessous:
4 alert("Test");
5
6 console.log("Hello world !")
```

On the right, a 'Change Description' dialog box is open, containing the text 'Session 1 terminée !' and 'Appuyez sur 'Entrée' pour confirmer votre saisie, ou sur 'Échap' pour l'annuler'. A red box labeled '3' highlights the 'Entrée' key instruction.

Apprends à apprendre !



- Notre mémoire conserve mieux les informations que nous utilisons souvent.
- Il est impossible de tout mémoriser



Si tu as oublié quelque chose, trouve-le dans :

- les présentations
- ton code
- **les manuels en ligne** surtout dans les liens que tu trouveras à la fin de chaque présentation

Ressources supplémentaires



+



COPIER



+



COUPER



+



COLLER



+



ENREGISTRER



+



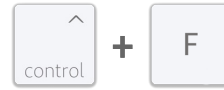
REVENIR EN ARRIERE



+



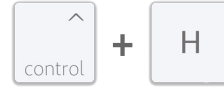
REVENIR EN AVANT



+



RECHERCHER



+



REEMPLACER



+



{ / }



+



[/]



Mac



=>





Tableaux et boucles (for, while)

1. http://eloquentjavascript.net/04_data.html (en)
2. http://eloquentjavascript.net/02_program_structure.html#h_rDxYNPd65Z (en)
3. http://eloquentjavascript.net/02_program_structure.html#h_oupMC+5FKN (en)



Labs et livres importants

1. <http://eloquentjavascript.net> (en)
2. <https://www.w3schools.com/js/default.asp> (en)

Information

- <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference> (en)
- <https://www.w3schools.com/jsref/default.asp> (en)
- <https://hanumanum.github.io/js-turtle/>

Fin de session

À la prochaine !