



# CPFPP - Sciences Mat: La pensée computationnelle avec Scratch Junior

## Description

---

Ce **plan de cours** est destiné aux débutants en matière de codage. Aucune connaissance au préalable n'est requise. Il vise spécifiquement à donner confiance aux enseignants et aux élèves. La plateforme [Scratch Junior](#) est entièrement gratuite, disponible dans de nombreuses langues et considérée comme une "norme industrielle" pour l'enseignement du codage à ce niveau.

Les termes "pensée computationnelle" et "codage" peuvent faire peur à certains, mais en réalité, ils font simplement référence à la résolution méthodique de problèmes. Jane Krauss et Kiki Prottsman parlent de curiosité, de résolution de problèmes, de tests, de création, de rêve, d'innovation et de pensée critique dans leur livre "Computational thinking {and coding} for Every Student" (La pensée informatique {et le codage} pour chaque élève). Ils affirment également que "la pensée informatique est un ensemble de techniques de résolution de problèmes qui sont utiles dans de nombreux domaines de notre vie".

Pour ces raisons, la pensée informatique (et le codage) s'adresse à tout le monde. Le but ces "micro-leçons" est de bâtir des compétences en pensée computationnelle. Ces leçons abordent plusieurs composantes de Scratch JR. Il ne faut pas se sentir obligé de tous les faire. Les liens au programme d'études se répètent plusieurs fois dans ces leçons.

Si les leçons sont correctement planifiées et exécutées, vous verrez que tous les élèves réussiront. Vous constaterez également qu'il s'agit de l'une des matières les plus faciles à motiver et à différencier. Il est généralement assez facile de les impliquer.

[Légende \(types de ressources\)](#)

## Matières

---

Sciences

# Niveaux scolaires

---

Maternelle

**Créé par :** nla1 nla1

**Dernière modification le :** 21 octobre 2024

**Créé le :** 11 janvier 2026

# Bienvenue à Scratch Junior! Votre plateforme de codage

---

Scratch est la plus grande communauté de codage pour enfants au monde et un langage de codage doté d'une interface visuelle simple qui permet aux jeunes de créer des histoires, des jeux et des animations numériques. Scratch est conçu, développé et animé par la [Fondation Scratch](#), un organisme à but non lucratif.

Scratch encourage la pensée computationnelle et la résolution de problèmes, l'enseignement et l'apprentissage créatifs, l'expression personnelle et la collaboration, ainsi que l'équité dans le domaine de l'informatique. Scratch est toujours gratuit et disponible dans plus de 70 langues.

Les leçons suivantes (en français!) suivent plus ou moins les mêmes grands objectifs que les "cartes coding" produites par Scratch JR (seulement disponibles en anglais). Les liens aux habiletés et compétences dans le programme d'études sont nombreuses et reviennent dans plusieurs leçons.

# Le codage visuel - pourquoi ?

---

Le codage visuel présente deux avantages très importants (par rapport au codage textuel) :

- Facile à saisir et à comprendre
- Motivant

Le codage visuel vous permettra d'impliquer et de retenir tous vos élèves. Certains deviendront peut-être des programmeurs professionnels, d'autres non, mais tous en sortiront avec des compétences en pensée computationnelle !

La réalité du codage textuel est qu'il n'attire qu'un très faible pourcentage d'étudiants. La plupart se découragent et/ou s'ennuient dès les premières étapes. Le résultat a été, historiquement, un profil très particulier de codeurs.

Il existe un troisième avantage pour quiconque enseigne dans une langue autre que l'anglais. Le codage visuel est le seul moyen de faire coder les étudiants dans la langue d'enseignement. Il devient donc une nécessité si vous enseignez en français, en espagnol, etc.

# Installation du logiciel (gratuit)

---

## Accès basé sur le web

Il existe une nouvelle version de Scratch Junior qui est basée sur le web. C'est infiniment plus facile à gérer. Essentiellement, la seule différence est que le chat est remplacé par une baleine bleue. C'est pour des raisons légales, car il ne s'agit pas d'une version officielle de Scratch Junior. Vous pouvez y accéder ici : [codejr.org](http://codejr.org). **C'est ce que nous recommandons.**

\*\*\*\*\*

**Ci-dessous, nous énumérons les autres options au cas où vous souhaiteriez installer la version officielle.**

La plateforme officiel de Scratch JR est une "[App](#)" alors elle fonctionne via installation sur les tablettes **Android**, **Ipad** et **Amazon** ainsi que les **Chromebook** qui ont été configurés pour les Apps Android: (communiquez avec votre équipe technique).

Scratch JR est aussi [disponible sur Mac et Windows](#). N.B. Pour installer sur Windows ou MAC, vous aurez probablement un avertissement que ce site n'est pas sécurisé. Voici comment procéder:

## Windows

Locate and right-click on the app's executable and select Properties.

In the Properties dialog, open the General tab.

In the Secure section, check the Unblock option.

Click Apply and OK to save the changes.

Launch the app again to see if the error is resolved.

## MAC

Open "System Preferences" by clicking on the Apple icon and choosing that option.

Select "Security & Privacy" from the "System Preferences" window.

Select the "General" tab, and select the lock in the lower left corner to allow changes.

# Pratiques recommandées

---

- Avant de demander aux élèves de créer leurs propres projets, il est très important d'acquérir quelques compétences de base.
- L'achèvement de ces leçons ouvrira la voie à la réussite des leçons et des projets ultérieurs, car elles couvrent méthodiquement les compétences de base en matière de pensée informatique (codage).
- Bien que cela ne soit pas nécessairement essentiel, il est recommandé de parcourir ces leçons à l'avance avant de les présenter à la classe.
- Lors de l'enseignement, il est préférable d'avoir un appareil par élève afin de s'assurer que chacun acquiert les compétences de base présentées.
- Demander à un élève de vous aider à démontrer la leçon (via l'appareil projeté). Cela vous permet d'expliquer les étapes tout en faisant face à vos élèves. C'est aussi une façon d'assurer que vous n'allez pas trop vite en expliquant les étapes.
- Il faut généralement entre cinq à dix minutes pour expliquer les leçons et les faire exécuter. Veillez à leur poser des questions, à leur demander comment vous pourriez faire ceci, etc. pendant la présentation.
- Proposez un "complément" aux élèves qui terminent avant les autres. Il s'agit généralement de leur demander d'améliorer leur code en y apportant des modifications pour voir comment il se comporte. C'est l'occasion pour eux d'améliorer leurs compétences et, en prime, de ne pas vous déranger pendant que vous aidez les élèves qui n'avancent pas aussi vite que les autres.

# Curriculum

---

Suggestion pour l'année implantation (2023/2024)

- Maternelle: Leçons #1 - #10, ensuite à eux d'innover, explorer et créer leurs propres projets selon leurs nouvelles habiletés. Les possibilités de création qui suivront sont illimitées!
- Les habiletés et procédures du programme d'études sont abordés plusieurs fois dans les leçons. Ils sont listés ici: [Maternelle](#)

# Micro Leçons en pensée computationnelle - #1 à #10

---

- [Minet, dis bonjour!](#)
- [Minet cherche un fruit](#)
- [Minet se promène](#)
- [Minet monte à bord d'une fusée](#)
- [Minet rentre à la maison](#)
- [Minet se présente à un extraterrestre](#)
- [Minet plante un arbre](#)
- [Minet marche vers la rivière](#)
- [Minet et le lapin bondissant](#)
- [Minet écoute aux animaux](#)