13 386 mots

Préparez-vous aux tests techniques pour devenir développeur

Vous êtes à la recherche d’un emploi dans un métier lié au développement ? Préparez-vous aux tests techniques (discussions techniques, tests d’algorithmique ou mini-projets) dans ce cours !

Vous êtes étudiant ou salarié et vous êtes à la recherche d’un stage ou d’un emploi ? Si c'est le cas, il y a fort à parier que vous allez avoir devoir passer par la case "test technique".

Le problème, c’est que vous ne savez pas vraiment ce que c'est, de quoi ils retournent et comment vous y préparer ?

Rassurez-vous, vous êtes au bon endroit. Tout comme vous, je suis aussi passé par l'étape des tests techniques et je me suis aussi posé ces questions. C'est d'ailleurs pourquoi j'ai décidé de réaliser un cours avec OpenClassrooms.

Ensemble, nous verrons **quand ont lieu les tests techniques** dans le processus de recrutement, les **différents types de tests techniques**, et les **attitudes à adopter** pour être un super candidat. Je partagerai beaucoup de conseils avec vous et de nombreux ressources complémentaires pour vous aider à les préparer.

À la fin de cours, vous serez capables de :

* Lister les étapes classiques du recrutement d'un développeur
* Décrire les bonnes pratiques pour réussir une discussion technique
* Décrire les bonnes pratiques pour résoudre un test d'algorithmique
* Réaliser un mini-projet de développement
* Défendre ses choix de développement

Prérequis :

* Pour suivre ce cours, il est important que vous ayez des bases en programmation : connaître le HTML/CSS/JavaScript et/ou PHP vous aidera à mieux comprendre les concepts.
* Il est aussi important que vous ayez pratiqué un peu le développement.
* Si vous suivez un [**parcours de développement sur OpenClassrooms**](https://openclassrooms.com/fr/paths?t=18-D%C3%A9veloppement), je vous invite à faire ce cours plusieurs fois : un première fois au début de votre formation puis une deuxième fois au milieu. Cela vous aidera à mieux retenir les informations.

Prêt à devenir un pro des tests techniques ? Alors, c’est parti ! 👉

Préparez-vous aux tests techniques

Tirez le maximum de ce cours

Bienvenue dans mon cours sur les tests techniques pour les développeurs :). Je m’appelle Thomas Dimnet, je suis développeur chez Sens Critique et mentor chez OpenClassrooms.

Ce cours vise à vous donner une meilleure **connaissance du processus de recrutement** pour les développeurs et en se focalisant sur la préparation de la partie**tests techniques**.

**Familiarisez-vous avec le format de ce cours**

Dans ce cours, je vais vous partager mon expérience en tant qu’ancien candidat et qu’actuel recruteur. Vous trouverez dans ce cours :

* de courtes**vidéos de cours**durant entre 1:30 minutes et 3 minutes. Je parlerais dedans des points fondamentaux à connaître sur les différents tests techniques ;
* du **texte**sous la vidéome permettant de
  + récapituler l'ensemble de la vidéo pour que vous puissiez vous y référer plus tard ;
  + approfondir les notions dont je parle dans les vidéos ;
  + partager des liens vers des ressources complémentaires telles que des banques d’exercices d’algorithmiques ;
* des **vidéos complémentaires** durant lesquelles je partagerais mon écran pour partager des exemples de tests, avec leur énoncé et une proposition de résolution ;
* un **quiz** vous permettant de valider les connaissances acquises autour des étapes de recrutement, la préparation d'une discussion technique et le respect de bonnes pratiques pour résoudre un test algorithmique ;
* et enfin une **activité** vous permettant de réaliser un mini-projet en conditions réelles (ou presque !), qui sera évalué par des pairs.

Il est important que vous pratiquiez, ne soyez donc pas des utilisateurs passifs. Je pense notamment :

* aux tests d’algorithmiques. Rendez-vous sur les sites que je vous partage pour travailler vos compétences ;
* aux mini-projets. Ne vous contentez pas de regarder ma solution. Essayez de faire la vôtre !

Dernier conseil : **refaites ce cours plusieurs fois** :). Si vous suivez un parcours de développement sur OpenClassrooms, vous pouvez le faire une fois en début et/ou milieu de votre formation ainsi qu'une fois en fin de formation.

**Profitez des tests techniques pour progresser**

Notez bien que la partie technique des entretiens n'est pas qu'une étape douloureuse et difficile à passer pour décrocher un peu, elle peut aussi **vous être utile à vous en tant que développeur**. Elle peut permettre de développer vos compétences et découvrir l'entreprise.

Les tests techniques, c’est le moment où vous allez pouvoir :

* **vous faire une réelle idée de votre niveau**. Que ce soit au travers de la discussion technique, des tests d’algorythmiques ou du mini-projet, votre capacité à développer va être jugée et c’est une bonne chose. Cela va vous permettre de découvrir des solutions auxquelles vous n’auriez pas pensé ou de découvrir de nouvelles librairies ;
* **savoir quelles sont vos points forts et vos faiblesses**. Alors, je ne sais pas pour vous, mais j’aime la critique et j’aime qu’on me dise ce qui ne va pas dans mon code ou dans mon travail de manière générale. Si vous êtes comme moi, vous allez aimer les entretiens . On va mettre en avant ce que vous comprenez très bien et là vous devez vous améliorer. Je trouve ça inestimable ;
* **renforcer votre capacité à argumenter**. Pourquoi ? Parce que vous allez devoir défendre l’implémentation de vos solutions. En développement, il y a souvent plusieurs solutions à un même problème. Si en tant que junior, on a souvent tendance à n’avoir qu’une seule solution. **Plus vous allez gagner de l’expérience, vous allez voir les différentes approches possibles pour résoudre un problème** et vous devrez vous justifier : c’est une partie intégrante du métier de développeur.

Je sais que cette idée peut sembler un peu naïve mais là encore, sachez que pour un recruteur, il n’y a **rien de mieux qu’un candidat/développeur qui cherche à s’améliorer** !

Quand je fais passer des entretiens de recrutement, c’est quelque chose sur lequel je suis particulièrement sensible.

**Demandez du feedback**

En parallèle aux retours que vous allez pouvoir obtenir, je vous invite, je devrais même je vous encourage, à demander du feedback et aussi bien si l’entretien s’est bien passé que s’il s’est mal passé !

Puisque le recruteur va juger votre travail ainsi que votre personnalité, vous allez peut-être, je l’espère, avoir envie de vous améliorer pour réussir un futur entretien et/ou devenir un meilleur développeur.

Si vous voyez que vous avez des difficultés à comprendre certains concepts techniques, des design patterns par exemple, **posez des questions et rebondissez**. Une telle attitude peut réellement faire passer quelqu’un d’encore un peu fragile techniquement à quelqu’un qu’on a envie de recruter pour former.

Pour vous donner une idée, vous trouverez ci-dessus une réponse que nous avons envoyée un candidat que nous n’avons pas retenu mais qui nous avons fait un feedback :

Bonjour [...],

C’était également un plaisir d’avoir pu te rencontrer pour parler de ta formation et ton expérience.

Avant toute chose, je t’annonce que nous ne souhaitons pas faire suite à ta candidature. Nous cherchons quelqu’un ayant des compétences techniques plus confirmées au vu des missions que nous souhaitons donner.

Cependant, nous avons beaucoup apprécié discuter avec toi et tu t’intégreras facilement dans n’importe quelle équipe grâce à ta force de savoir-être.

Pour tes prochains entretiens, il serait pertinent que tu rajoutes un lien vers ton Github dans ton CV et essayer de l’alimenter un peu plus. Pour Twitter, il est plus ou moins utile de l’inscrire sur ton CV en fonction du contenu présent.

Il faudrait également que tu arrives à mieux défendre la partie technique de tes projets, car de l’autre côté on sent que tu es réellement passionnée et que tu prends plaisir à les réaliser.

Je te souhaite bonne chance pour la suite.

Nous n’avons pas décidé de donner suite à sa candidature mais, néanmoins, nous lui avons donné des conseils pour qu’il progresse et pour que cette personne puisse réussir son prochain entretien.

Je pense que nous sommes réellement dans l’une des seules industries qui encouragent le feedback négatif et comme positif. Sachez donc en profiter !

**En bref**

* Ce cours inclut des vidéos, des textes, des vidéos complémentaires où je partage mon écran.
* Vos compétences seront évaluées par un quiz et une activité de mini projet évaluée par les pairs.
* Je vous encourage à le faire plusieurs fois.
* Profitez des tests techniques pour progresser, notamment en récoltant du feedback.

Situez les tests techniques dans le processus de recrutement

Les tests techniques : c’est souvent la partie où tout se joue. C’est une source de stress (pour moi, ça l’était en tout cas) où on ne sait pas forcément à quelle sauce on va être mangé.

Les tests techniques, est-ce la première étape du processus de recrutement pour un développeur ?

Excellente question ! En effet, les **tests techniques ne sont qu’une étape du processus de recrutement**. Il y a des étapes entre le moment où vous envoyez votre candidature et le moment où vous réalisez vos tests techniques. Et il y en aussi après !

Bien que dans ce cours, nous nous focalisons sur les tests techniques, il m’a semblé important de vous redonner du contexte. Autrement dit, de vous donner des détails sur le processus de recrutement dans sa globalité.

### Découvrez les étapes avant les tests techniques

#### Vous envoyez votre CV

C’est officiel, vous avez trouvé l’entreprise de vos rêves ! Après plusieurs jours à consulter les offres d’emploi et à parcourir différents sites webs, vous avez retenu une annonce particulièrement intéressante et vous venez d’envoyer votre CV.

PLACEHOLDER ICONE CV

Ce dernier va être reçu par les **ressources humaines de l’entreprise** ou alors par un cabinet en recrutement (si vous passez par un service tel que talent.io). Il va d’abord être étudié pour voir si votre profil et vos compétences correspondent à l’offre d’emploi.

#### L’entreprise vous retient

Bingo, votre profil a tapé dans l’oeil de la personne chargée du recrutement et cette dernière va rentrer en contact avec vous pour vous qualifier.

Le terme qualifié ne signifie pas que vous allez recevoir un diplôme ;). Il signifie simplement que l’on va vous contacter pour voir si vous avez bien compris l’offre, l’entreprise dans laquelle vous avez postulé et si vous avez des questions.

PLACEHOLDER ICONE TELEPHONE

Généralement**par téléphone,** cette étape permet à l’entreprise de se faire une première idée sur le candidat qu’elle souhaite recruter. Il n’y aura donc pas vraiment de partie technique dans cet appel. L’idée sera plus d’avoir un échange sur :

* votre profil (qui vous êtes) ;
* vos motivations (pourquoi avez-vous postulé) ;
* votre expérience (d’où venez-vous) ;
* vos compétences (que faites-vous).

Vous pourrez aussi profiter de ce moment pour poser toutes les questions qui pourraient vous passer par la tête.

Auriez-vous des bons conseils pour bien se préparer à l’appel ?

Cette dernière demande avant tout beaucoup de bon sens, mais voici néanmoins quelques conseils  :

* **Montrez que vous avez bien compris le poste** :
  + L’activité de l’entreprise dans laquelle vous avez postulée ;
  + Les technologies utilisées ;
  + L’environnement de travail.
* C’est aussi le moment de**montrer votre motivation, vos ambitions et votre passion**. Je ne pourrais que vous conseiller de vous montrer curieux et intéressé à ce que votre interlocuteur.

D’ailleurs, petit conseil. Dans le milieu du développement, on dit souvent qu’il y a quatre postes pour un candidat. Nous, développeurs, sommes constamment sollicités par les entreprises qui ont dû mal à pouvoir les offres.

**C’est donc le moment d’être sympa, souriant et agréable**. Trop de développeurs ont tendance à être blasé au téléphone. Autrement dit, c’est le moment de vous distinguer et de sortir de la masse en étant sympa !

Rassurez-vous, cette étape est loin d’être compliquée. Comme vous pouvez le constater, ce n’est pas la partie la plus complexe du processus de recrutement. Elle demande avant tout du bon sens et du pragmatisme.

Une fois cette dernière réalisée,**la personne chargée de recrutement vous proposera généralement un entretien physique** pour démarrer la partie technique du processus de recrutement.

### Découvrez les tests techniques

À quoi servent les tests techniques ?

Les tests techniques permet à l’entreprise de se faire une idée :

* des compétences et du niveau technique d’un candidat ;
* de ses facultés d’apprentissage ;
* de sa capacité à résoudre et surmonter les difficultés techniques éventuelles.

Vous allez devoir montrer à l’entreprise que vous êtes la bonne personne et que vous êtes capable de résoudre les problématiques que vous rencontrerez.

Est-ce uniquement une étape pour les développeurs sans expérience ?

Non ! Certes l’entreprise ne demandera pas la même chose à un développeur junior qu’à un développeur senior, mais les seniors doivent également souvent passer des tests pour être recrutés.

En tant que junior, vous ne pouvez pas avoir la réponse à tout parce que vous manquez encore d’expérience. Vous êtes un développeur senior en devenir :).

Cette étape vise à répondre à la question : est-ce que ce développeur est la bonne personne pour l’entreprise et pour les difficultés qu’elle peut rencontrer ?

Quand on démarre, on a souvent tendance à se dire qu’une entreprise ne devrait recruter que des seniors. Et bien, c’est faux !

**Une bonne équipe technique se doit d’être composée de personnes de tout niveau**. De seniors qui vont pouvoir appréhender des problématiques complexes tout en formant des développeurs juniors.

C’est une des beautés du monde de l’informatique : la transmission des connaissances est quelque chose d’appréciée et est encouragée.

À quoi ressemblent un test technique ?

Justement, parlons-en ! Voyons brièvement une définition des trois types de test techniques, avant de les voir dans le détail dans la suite du cours.

#### La discussion technique

La discussion technique est le moment où vous, candidat, allez discuter de vos connaissances, votre expérience ainsi que de concepts généraux avec un ou plusieurs développeurs.

PLACEHOLDER ICONE DISCUSSION TECHNIQUE

Cela se rapproche fortement d’une discussion que vous pourriez avoir avec des collègues dans votre entreprise ou des collègues étudiants.

#### ****Les tests d’algorithmiques****

Généralement composée de 2 à 5 exercices, ce test a pour but de **vous tester sur des problématiques spécifiques de votre métier**. C’est souvent le moment de travailler sur des chaînes de caractère, des tableaux et autres concepts essentiels de la programmation.

PLACEHOLDER ICONE TEST ALGO

#### Le mini projet

Dernier type de test, un mini projet vous demande de coder un projet complet tel qu’une page de recherche. Plus complexe mais aussi plus intéressant, **cette étape vous permet de mettre en avant vos talents de développeur sur quelque chose de concret**.

PLACEHOLDER ICONE MINI PROJET

### Découvrez les étapes après les tests techniques

Que se passe-t-il si je réussis les tests ?

Si vous regardez le schéma vu ci-dessus, vous verrez qu’après les tests techniques, vient l’entretien final : la rencontre avec les fondateurs et/ou votre futur manager.

PLACEHOLDER ICONE DISCUSSION AVEC FONDATEURS

**Cette étape clôt généralement le processus de recrutement** et vous serez généralement informé quelques jours après de la suite des événements : si vous êtes embauché, si oui, quand allez-vous signer votre contrat, etc.

#### L’entretien avec les managers/fondateurs

Alors que se passe-t-il durant cette entretien ? De quoi va-t-on discuter avec les fondateurs ou le manager ?

Assez souvent, vous allez parler de deux choses : **de la vision de l’entreprise et de rémunération**.

* La vision : ce sont les **objectifs moyen et long terme de l’entreprise**. Que souhaite-t-elle faire sur les prochains mois, les prochaines années et êtes-vous en accord avec cette dernière. On peut vous parler des futures évolutions (le développement d’une nouvelle application mobile ou d’un nouveau site) ou de s’étendre sur un nouveau marché. L’idée est de voir que vous, candidat, et les fondateurs/managers êtes bien en phase avec les objectifs de l’entreprise.
* La rémunération : parce qu’il faut bien parler argent à un moment. Sur l’offre de poste, on vous donnera souvent une grille de salaire. **L’idée ici est de se mettre d’accord sur votre future rémunération**. Attention, on parle souvent en salaire brut et non en salaire net ;-).

Il arrive même parfois qu’on vous fasse une proposition directement pendant l’entretien. Ce dernier se déroulant bien et les fondateurs ayant vu que votre profil correspond bien à leur besoin. Ces derniers vont vous faire une proposition et vous pourrez soit répondre directement soit dire que vous souhaitez réfléchir avant de prendre une décision :-).

Quelles attitudes à adopter pour que l'entretien se passe bien ?

Comme pour l’entretien RH, nous ne sommes pas ici sur un échange technique sur vos compétences mais plus sur un échange assez généraliste.

Soyez donc honnête dans vos échanges et donnez simplement vos attentes.

* Soyez précis dans la rémunération que vous demandez. Vous pouvez donner une fourchette de salaire mais au final vous allez vous arrêter sur une somme !
* **Posez des questions** ! Que ce soit, sur le cadre et l’ambiance de travail, sur la formation des salariés, sur les jours de télétravail autant que sur la conduite du projet et de l’entreprise. **Ne vous focalisez pas uniquement sur le salaire mais aussi sur ce que l’entreprise est susceptible de vous apporter.**

En bref, essayez de vous projeter dans l’entreprise :

* Allez-vous vous y sentir bien ?
* Sentez-vous qu’ils vont vous aider à évoluer ?
* Comment vont se passer vos premiers jours en entreprise ?

**Le plus vous posez et vous vous posez des questions**, le mieux c’est. Il n’y a rien de pire que de commencer un nouveau poste et de sentir qu’on a peut-être fait le mauvais choix.

Pour finir sur ce chapitre, j’ai décidé de vous parler d’un sujet qui me tient à coeur : **comment s’habiller pour des entretiens quand on est développeur**. En effet, on se fait souvent l’idée qu’un développeur est un barbu avec des cheveux longs et que sa façon de s’habiller n’influe pas sur son recrutement.

En toute franchise, cela va dépendre de la structure dans laquelle vous avez postulé :

* Pour une startup : sweat à capuche, jean, baskets, quitte à reprendre un slogan assez connu “venez comme vous êtes”. Cela dit, essayez de vous renseigner sur le dress code - certains startups peuvent être plus formelles que d’autres. Je pense, par exemple, à l'ingénierie médicale et à la biomédicale.
* Pour une agence web ou de transformation digitale : le dress code est souvent assez casual. Autrement dit, rien ne vous empêche de venir en entretien en mode “jean, baskets, t-shirts”.
* Pour une ESN : les ESN ou SSII ont souvent un dress code plus formel. Je vous invite donc à venir en chemise et chaussures de ville. Le costume et la cravate ne sont pas obligatoires.

### En bref

* Le processus de recrutement d’un développeur comprend l’entretien RH, les tests techniques et l’entretien avec les fondateurs/managers.
* Le contact RH a pour but de qualifier votre candidature et ce n’est pas une discussion technique. Montrez-vous aimable, passionné par ce que vous faites et motivé.
* L’objectif de l’entretien avec les fondateurs/managers est de parler de vision long terme de l’entreprise et de salaire.
* Les softs skills sont au moins aussi importants que les compétences techniques, sachez les mettre en avant !

Préparez-vous aux discussions techniques

PLACEHOLDER VIDEO 1

Première étape de la phase des tests technique, la discussion technique se veut être un échange entre le candidat et un ou deux développeurs de l’entreprise.

Proche d’une discussion que vous pourriez avoir avec des collègues, elle a pour but principalement de briser la glace et de **vous découvrir en tant que candidat**. C’est souvent le premier contact que les développeurs avec le candidat potentiel.

Bien qu’assez peu formalisée, elle permet aux recruteurs de**rapidement former un avis sur le candidat**. La préparer et anticiper les questions vous permettra d’appréhender cette dernière plus sereinement.

### Gardez en tête les objectifs de la discussion technique

De manière générale, les **discussions techniques abordent tous les mêmes concepts**. Autrement dit, une fois que vous en avez réalisé une ou deux, vous commencerez à comprendre ce qui est attendu tant en terme d’attitude que de sujets de conversation.

Cela dit, quand c’est la première fois, la discussion technique est souvent un moment stressant. On ne sait pas forcément à quelle sauce on va être mangé ou alors comment vont être les personnes en face de vous.

Si ça peut vous rassurer, je suis passé par là, comme tous les développeurs et au final, cette partie n’est pas si terrible, ni si difficile. En somme, c’est comme si vous faisiez une discussion avec d’autres étudiants ou collègues sur des concepts techniques.

L’idée est de la discussion technique est donc de parler :

* de concepts généraux d’informatique ;
* de votre langage (JavaScript, Python, etc.) et de votre spécialité (front-end, back-end, etc.) ;
* de vos expériences ;
* des problèmes que vous avez rencontrés.

Pour faire simple, c’est un peu comme si vous faisiez le point sur votre vie de développeur :). Qu’est-ce qui vous plaît dans ce métier, qu’est-ce qui vous plaît moins, qu'est-ce que vous avez compris, etc.

Là encore, dans le monde de l’informatique, on est un peu à part du reste du monde du travail. Les développeurs qui vous recrutent sont souvent vraiment intéressés par votre personnalité et votre parcours.

Le premier objectif de la discussion technique est donc simplement d’en apprendre un peu plus sur vous :).

#### Parler d'ingénierie informatique (concepts généraux d’informatique)

Le premier point abordé lors d’une discussion technique est souvent en relation avec les **bases de l’informatique**. On ne parle donc ici pas de langage spécifique ou de spécialité mais plus de **concepts généraux aux sciences de l’informatique**.

Comme toujours, la complexité de la discussion dépendra de votre séniorité : en fonction du poste visé, les développeurs qui vous recrutent n’attendent pas tout à fait la réponse. À la fin de ce chapitre, je vous donne quelques ressources pour vous aider à préparer la discussion technique et pour en apprendre un peu plus sur votre fascinant métier :-).

Pour cette partie donc, les questions peuvent aussi bien porter sur l’éditeur de texte que vous utilisez que sur l’allocation de mémoire que sur des design patterns. Ça peut sembler simple comme sujet mais là encore, ça permet d’en apprendre un peu plus sur le candidat.

#### Parler de son langage/sa spécialité

L’idée va être de savoir si vous maitrisez les éléments essentiels de votre métier.

##### Son langage

Pour un développeur JavaScript, on peut lui demander :

* Qu’est-ce sont les gestionnaires de librairie il utilise ?
* Lui demander d’expliquer ce que sont que les task runners et leur mode de fonctionnement.
* Qu’est-ce que l’ES6 et qu’est-ce qui change par rapport à l’ES5 ?

Autre exemple, pour un développeur Python, on pourra lui demander :

* Pourquoi est-il si important de créer un environnement virtuel en python ?
* Qu’est-ce qu’anaconda ? Qu’est-ce que panda ?
* Comment bien gérer les dates en Python ?

Si vous n’avez pas la réponse à ces questions, c’est normal. Comme je vous l’expliquais précédemment, en fonction de votre niveau, les réponses ne sont pas les mêmes.

Cela dit, certaines réponses sont plus importants à connaître que d’autres. Je pense, par exemple, à l’ES6 en JavaScript qui est aujourd’hui un incontournable en entreprise.

##### **Sa spécialité**

Puis, une fois parlé de votre langage, vient le moment où vous aurez des questions sur votre spécialité.

* Pour un développeur front-end, par exemple, on pourra lui poser des questions sur l’optimisation (pour qu’un site web se charge plus vite), sur la gestion des compatibilités navigateurs ou sur des bibliothèques graphiques en JavaScript.
* Pour un développeur back-end, ce sera l’occasion de lui parler de versionnage d’API et d’architectures à mettre en place pour créer une back end robuste.
* Pour un développeur mobile, enfin, ce sera l’occasion de parler d’optimisation du cache et de persistance des données.

Le plus important ici est de comprendre les problématiques de chacune des spécialités et de savoir en parler. Un développeur front-end ne rencontrant pas les mêmes problèmes qu’un développeur back-end et n’utilisant souvent pas les mêmes librairies.

#### Partager ses expériences

Parler de ses expériences en entretien me semble quelque chose d’assez logique. Après tout, vous êtes là pour parler de vous et de que vous avez fait avant de postuler dans l’entreprise de vos rêves :).

Sachez parler aussi bien de **vos expériences professionnelles**, autrement dit d’un stage ou d’un emploi, que de vos expériences personnelles, par exemple, si vous avez travaillé sur des projets Open Source sur votre temps libre.

Vous pouvez, par exemple, décider de parler de votre projet personnel : quelles technologies avez-vous utilisé ? Pourquoi ? Quelles problèmes avez vous rencontré ? etc.

L’idée de cette partie va être de parler des choix technologiques que vous avez faits mais aussi de donner un retour d’expérience par rapport à ces derniers.

#### Échanger autour de problèmes rencontrés et les solutions mises en place

Liée à la partie ci-dessus sur les expériences du candidat, il va être ici d’en savoir un peu plus sur les difficultés que vous avez rencontrées lors de vos projets et les solutions que vous avez mises en place.

Elle est certainement **une des parties les plus importantes de la discussion technique**, notamment parce qu’elle permet aux recruteurs de :

* se faire une idée de ce que vous avez réellement vu que ce soit durant votre formation, votre stage ou vos expériences professionnelles différentes ;
* voir comment vous arrivez à prendre du recul sur ce que vous produisez ;
* savoir ce que vous en avez retenu et donc ce que vous referez et surtout ce que vous ne referez pas.

C’est une des questions que je pose le plus quand je fais passer des soutenances aux étudiants : “Qu’est-ce qui a bien marché sur votre projet, qu’est-ce que vous referiez si vous deviez refaire votre projet et qu’est-ce que vous ne referiez pas ?”

Si vous vous rappelez du mail que nous avons envoyé à la personne que nous n’avons pas retenu, c’est justement ce qui a pêché dans son entretien : cette personne n’a pas réussi à expliquer ce qu’elle avait et le problème grand challenge qu’elle avait rencontré. Elle est restée évasive et c’est quelque chose qui se voit très vite en entretien. Pensez donc à être factuel.

### Projetez-vous dans les conditions réelles d'une discussion technique

Maintenant que nous avons vu les objectifs de cette discussion, je vais vous présenter son cadre de réalisation, à savoir où la discussion technique se passe et dans quelles conditions.

Où se passe cette discussion ?

Elle se déroule soit dans les locaux de l’entreprise soit en visioconférence. Cela dépend de votre éloignement de l’entreprise.

Par exemple, si vous habitez sur Paris et que vous souhaitez rejoindre une entreprise au Canada, il y a fort à parier que vous allez faire une visioconférence. Comme toute bonne visioconférence, pensez à être dans une pièce calme et éclairée.

Pour les étudiants des parcours OpenClassrooms, vous partez normalement avec un avantage. Vous avez déjà l’habitude de réaliser ce type d’exercice quand vous passez vos soutenances.

Qui participe ?

Vous l’avez peut-être déjà compris. Cette discussion se déroule généralement avec deux développeurs de l’entreprise. Souvent un développeur lead ou senior accompagné d’un développeur soit junior, soit “midnior”.

Gardez en tête que vous allez devoir expliquer et vulgariser vos connaissances aux deux personnes :).

Combien de temps dure une discussion technique en général ?

Entre 30 minutes et 1h30. J’y reviendrais plus tard, mais sachez que plus la discussion technique dure dans le temps, plus c’est bon signe pour vous.

### Respectez des bonnes pratiques

PLACEHOLDER VIDEO 2

#### Faites des schémas pour expliquer des notions complexes

Savoir faire des schémas sont, selon moi, l’une des compétences de base à avoir pour un développeur. Réaliser des schémas, que ce soit en entretien ou sur votre lieu de travail, vous **permet d’expliquer** des notions complexes.

Ils permettent de montrer que vous avez bien compris de quoi on parle et permettent aussi d’éviter les quiproquos. On vous demandera assez souvent de réaliser un schéma en entretien (pour expliquer Redux par exemple).

L’avantage, c’est que vous pouvez travailler ça lors de votre préparation : Internet fourmille de schémas intéressants que vous pouvez utiliser, en entretien ou pour votre connaissance personnelle.

#### Admettez quand vous ne savez pas

**On se dit souvent qu’en entretien, on doit tout savoir sur tout**. On doit connaître sur les bouts des doigts les technologies utilisées et qu’on ne peut pas ne pas savoir quelque chose. **C’est faux !**

Savoir ne pas savoir vous met souvent dans une posture d’ouverture vis à vis des recruteurs. Vous acceptez de ne pas connaître telle ou telle notion mais vous souhaitez l’apprendre.

Faisons le point rapidement, ce que l’on demande le plus souvent à un stagiaire ou à un développeur junior, c’est :

* d’être là pour apprendre et pour progresser ;
* de connaître les notions “de base” de sa spécialité ;
* de poser des questions.

Typiquement, lors de mon premier stage, je ne pense pas avoir assez posé de questions. Je suis resté sur le principe que c’était à moi de tout savoir et de tout comprendre alors qu’au contraire j’étais là pour apprendre et engranger de la connaissance.

S’il vous plait, ne faîtes pas la même erreur que moi ;).

**Dès l’entretien, n’hésitez donc pas à dire que vous ne savez ou ne comprenez pas**. Ce sera souvent une porte ouverte vers une digression qui pourra vous apprendre de nouvelles choses.

#### Communiquez votre passion

Commençons par une petite mise en situation :).

Vous êtes recruteur et vous êtes actuellement en train de faire passer un entretien à un futur stagiaire. Votre société est, en effet, en train de grossir et elle aimerait qu’un stagiaire intègre votre équipe pour renforcer vos effectifs.

Le matin même, vous avez fait un entretien avec un autre stagiaire. Ce dernier n’était pas mauvais, plutôt bon techniquement et à l’aise avec un certain nombre de notions.

Problème : il ne semblait pas habiter par une passion et ne semblait pas avoir envie d’apprendre.

Cette après-midi, vous faites un entretien avec un autre stagiaire : il est moins que le premier mais semble “en vouloir” et montre bien sa passion.

Qui choisissez-vous ?

Personnellement, je préfèrerais le stagiaire passionné ! Je sens que je vais pouvoir lui apprendre plus de choses et dans de meilleures conditions. Je sens aussi qu’il ira plus loin et que je vais avoir envie de travailler avec lui et de le faire progresser.

Pour avoir été confronté à cette situation, soit dans le cadre où j’étais le stagiaire moins bon mais plus passionné, soit dans le cadre où je devais choisir entre deux candidats, comprenez bien que la passion est quelque chose de palpable et qui a une réelle importance lors d’un entretien.

**La capacité à communiquer une passion dépend certes des individus mais essayez de faire un effort dessus.**Cela peut vraiment vous permettre de changer la donne.

#### Adoptez une posture d’apprentissage

Corollaire du point ci-dessus, **essayez de montrer que vous êtes dans un processus et une posture d’apprentissage**. Montrez que la critique n’est pas quelque chose qui vous gêne plus qu’elle vise à vous faire progresser en tant que développeur.

Posez des questions, intéressez à ce que fait l’entreprise, aux technologies qu’elle utilise et demandez des feedbacks même en entretien.

Par exemple, si on vous parle d’une technologie pour améliorer le référencement, le Server Side Rendering par exemple, et que vous ne savez pas ce que c’est. Posez des questions et essayez de rebondir dessus : c’est certainement l’un des meilleurs moyens pour que l’entretien se passe bien.

#### Mettez en avant vos soft skills et votre parcours

Pour les personnes qui n’ont pas suivi un cursus classique type école d’ingénieur, et j’en fais parti, sachez mettre en avant ce qu’on appelle les soft skills.

Qu'est-ce qu'un soft skill ?

Ce sont toutes les compétences non techniques que peut avoir une personne. On peut, par exemple, citer l’empathie, la gestion du temps ou l’adaptabilité.

De plus en plus, le recrutement de développeurs se fait sur les soft skills. On ne veut plus simplement un expert technique, sauf pour des postes de senior et non de lead, on veut des personnes qui soient capables de communiquer avec les autres équipes de l’entreprise.

Il est loin le cliché du développeur qui code dans sa cave. Aujourd’hui, être développeur, c’est être capable de vulgariser ses connaissances tant aux autres développeurs qu’autres personnes non techniques.

### Évitez certains comportements rédhibitoires

Maintenant que nous avons vu ce qu’il faut faire pendant la discussion technique, voyons ensemble ce que je vous déconseille de faire :).

#### Ne soyez pas prétentieux

En top 1 des choses à ne pas faire en entretien, être prétentieux. Je pense que certainement l’une des choses les plus éliminatoires.

Que ce soit pour un développeur junior ou un développeur senior, être prétentieux nuit très souvent à la vie et aux projets de l’entreprise. Bien que je ne doute pas que vous ayez de l’expérience, sachez rester humble en entretien (tout comme en entreprise d’ailleurs ;-)).

Il y a un véritable équilibre à trouver entre ce que vous avez été capable de faire durant votre précédent emploi ou durant votre stage et comment est-ce que vous le présentez.

Comme je vous le disais plus haut, sachez rester factuel tout en ne dénigrant pas les équipes avec lesquelles vous avez travaillées (tant pour les équipes techniques que non techniques).

#### No soyez pas sur la défensive

Autre point important, essayez au maximum de ne pas être sur la défensive. Je sais que vous pouvez penser que votre travail et que votre niveau est jugé lors d’un entretien. Dans un sens, c’est vrai.

Cela dit, voyez plutôt ça comme un bilan de compétences. Vous réalisez l’entretien, certes pour être recruté, mais aussi pour avoir une idée de votre niveau et ce qui vous manque peut-être.

C’est donc le moment d’être totalement honnête avec vous (et vous pouvez le faire dès la préparation) :

* Qu’est-ce que vous savez faire ?
* Quels sont vos softs skills ?
* Qu’est-ce qui vous différencie d’un autre candidat ?

Posez vous ces questions lors de la préparation de l’entretien. Cela vous permettra de ne pas être déstabilisé et sur la défensive le jour J.

#### Ne manquez pas de curiosité

Il arrive parfois qu’en entretien, on s'aperçoit qu’un développeur n’est pas très intéressé par qu’il fait et le pire, c’est que cela se voit très vite.

Ne pas avoir un GitHub actif, ne pas lire régulièrement des blog posts, ne pas être inscrit à des newsletters. Autant de signes qui ne trompent pas. Gardez ça en tête avant et pendant votre entretien !

Sachez donc vous montrer curieux : lisez des blogs posts (sur medium par exemple), regardez des conférences sur Youtube (la chaîne freecodecamp est très bien pour ça), posez des questions et suivez des cours sur OpenClassrooms.

En bref, montrez que devenir développeur vous intéresse !

#### Ne soyez pas trop vague sur vos expériences

En tant que développeur, nous apprenons constamment que ce soit via des cours (comme celui-ci ;-)), des projets en étude (comme ce que vous réalisez peut-être avec OpenClassrooms) ou des projets en entreprise. C’est le ciment de votre apprentissage et vous devez être capable de prendre du recul là-dessus.

En prenant du recul, vous allez être capable d’expliquer ce que vous avez fait en étant factuel.

Il n’y a donc rien de pire qu’un candidat restant dans le vague vis à vis de ce qu’il a fait précédemment. Car, pour le recruteur, cela veut peut-être dire que le candidat n’a pas “fait grand chose”.

Attention, je ne dis pas ici que le candidat n’a rien fait. Je dis plutôt qu’il échoue à l’expliquer et à le raconter.

Soyez factuel, donnez des exemples précis, dites ce qui a marché, ce qui n’a pas marché et pourquoi.

J’ai passé 1h30 en discussion, est-ce mauvais signe ?

**En fait, c’est généralement un très bon signe**! Je vous disais plus haut qu’une discussion technique dure généralement entre 30 et 45 minutes.

Si vous commencez à voir que le temps est en train de s’allonger et que vous êtes plus sur une durée de 1 heure voir 1 heure et demi, c’est très bon signe. Cela veut dire que les développeurs qui vous font passer la discussion prennent du plaisir à parler avec vous et cela veut potentiellement dire qu’ils se voient bien travailler avec vous.

Deux autres bons signes sont :

* **si les développeurs essayent de vous apprendre des choses**. Autrement dit, vous avez suivi les conseils dont je vous ai parlés plus haut concernant l’importance du “je ne sais pas” et vos interlocuteurs y ont été sensibles. Et bien, il se peut très bien que ces derniers soient déjà en train de vous former en vous expliquant de nouvelles notions, et c’est un excellent signe pour vous !
* **Si les développeurs sont réceptifs à vos questions techniques**. Si vous posez des questions techniques quant à l’architecture du produit : pourquoi avoir choisi React Native plutôt qu’une solution Swift/Kotlin par exemple.Si ces derniers répondent de façon complète à vos questions, il y a fort à parier que cette discussion se passe très bien. Vous montrez que vous vous intéressez à l’entreprise et que vous vous projetez dedans si bien que les développeurs ont envie d’en discuter avec vous.

### Bien se préparer à la discussion

Comme pour tout, la préparation est souvent l’étape essentielle à la réussite. Une préparation régulière à ce type d’entretien peut vous permettre de passer cette étape sereinement.

Voici une liste de conseils pour vous aider à la préparer :

#### Parlez développement à d’autres développeurs

Que vous soyez encore en étude, chez OpenClassrooms par exemple, ou que vous soyez en poste. Il est important que vous parliez régulièrement à d’autres développeurs. Utilisez tous les outils à votre disposition pour le faire : mails, chats, Facebook, etc.

Vous pouvez parlez avec eux de sujets très variés tels que des technologies que vous utilisez ou que vous avez envie d’utiliser que des cours que vous avez suivis (comme le mien :-)).

#### Allez à des meetups

Suivant votre lieu de résidence, vous pouvez avoir plus ou moins de facilité à vous rendre à des meetups locaux. Pour les personnes habitants de grande ville comme Paris, Marseille, Lyon, Nantes, etc., des meetups sont organisés chaque semaine dans toutes les spécialités possibles.

Ne vous rendez pas forcément à tous les meetups mais essayez d’en faire un par mois et sur une technologie qui vous intéresse.

Petit bonus : des ateliers de recrutement sont généralement organisés à la fin de chaque meetup. Le bon moment pour découvrir de nouvelles choses et peut-être trouver votre nouveau job.

#### Renseignez-vous sur les technologies que l’entreprise utilise

Alors, attention, histoire vraie ! Nous avons récemment eu besoin de nouveaux développeurs dans notre équipe au travail. Nous avons naturellement rédigé une fiche de poste précisant que nous cherchions un développeur NodeJS et nous avons commencé à accueillir les candidats.

Et bien, figurez-vous que de nombreux candidats n’avaient même pas pris la peine de regarder la fiche de poste et ne connaissaient pas les technologies que nous utilisons. Pire encore, un candidat nous a même dit détester le JavaScript (alors que la fiche de poste mentionnait qu’on recherchait un développeur JavaScript). Carton rouge !

Détester le JavaScript, je peux le comprendre (quoique ;) ). Mais dans ce cas-là, ne venez pas en entretien pour un poste de développeur NodeJS :).

Ne pas tout connaître sur les technologies qu’une entreprise utilise c’est une chose mais ne pas faire l’effort de connaître la base de l’annonce en est une autre. Alors, avant d’aller en entretien, renseignez-vous !

#### Entraînez-vous à expliquer des notions complexes à vos proches

Savoir expliquer une notion complexe à quelqu’un de non technique est certainement l’un des meilleurs moyens de briller en entretien ! C’est d’ailleurs souvent le moment de faire des schémas ;).

Si vous n’avez pas d’exemples en tête, vous pouvez penser à :

* Expliquez ce qu’est une API à votre grand-mère ;
* Présentez Redux à votre copain ou à votre copine ;
* Expliquez la différence entre html, css et javascript à votre petit frère.

Vulgariser des notions complexes et**faire l'exercice de les expliquer** est aussi un bon moyen de mieux retenir les notions et connaissances que vous avez abordées pendant votre formation. C’est donc une méthode gagnant/gagnant.

#### Suivez des blogs

Internet est, aujourd’hui, rempli de blogs et de posts techniques de développeurs cherchant à partager leur expérience. C’est une excellente nouvelle pour vous, pour préparer les entretiens mais aussi pour continuer à vous former.

Si certains articles sont plutôt dédiés à la découverte ou à l’explication d’une technologie particulière, de nombreux ont aussi été écrit pour vous aider à préparer vos entretiens de recrutement :).

Si vous ne savez pas par où commencer, voici une liste de blogs à suivre :

* [Chaîne medium du FreeCodeCamp](https://medium.freecodecamp.org/) - ressource anglophone. De nombreux blog posts sur tous les concepts et langages. Le niveau va de débutant à intermédiaire.
* [Blog d'Eric Elliott](https://medium.com/@_ericelliott) - ressource anglophone. Important nom de la scène JavaScript. Le niveau est de intermédiaire à avancé.
* [Blog de Trey Huffine](https://medium.com/@treyhuffine) - ressource anglophone. Principalement du JavaScript. Tous niveau.
* [Blog de Dan Abramov](https://overreacted.io/) - ressource anglophone - Blog du créateur de Redux. Là encore principalement JavaScript - Tous niveau.
* [AlsaCréations](https://www.alsacreations.com/) - ressource francophone - principalement sur de l’intégration et du développement front-end.

Petit rappel, si vous ne comprenez pas forcément tout ce qui se dit dans les articles, c’est normal : il y en a pour tous les niveaux. Sachez donc comprendre le sens général de ce qui se dit dans ces articles et, par la suite, au fil de vos lectures, vous allez en comprendre de plus en plus.

#### Regardez des conférences sur Youtube

Si vous êtes plutôt vidéo, Youtube regorge de vidéos dédiées au développement informatique.

Voici une liste de chaînes Youtube que vous pouvez suivre :

* [Chaîne Youtube du FreeCodeCamp](https://www.youtube.com/channel/UC8butISFwT-Wl7EV0hUK0BQ) - anglophone - la chaine Youtube de FreeCodeCamp - tous niveau.
* [Chaîne Youtube de LevelUpTuts](https://www.youtube.com/channel/UCyU5wkjgQYGRB0hIHMwm2Sg) - anglophone - une chaîne donnant pas mal de tutos sur une grande variété de sujets.
* [Chaîne Youtube de Thoughtbot](https://www.youtube.com/user/ThoughtbotVideo/videos) - anglophone - tutos sur une grande variété de sujets.

Dernier point avant de conclure. Comme vous pouvez le constater, je donne beaucoup de ressources anglophones. Ce n’est pas forcément parce que je n’aime pas les ressources francophones, c’est simplement parce que l’anglais est la langue de l’informatique.

De nombreux auteurs et/ou experts techniques sont anglophones et ne parlent pas un mot de français. Vous allez donc devoir apprendre à lire l’anglais technique :honte:.

### En bref

* La discussion technique est souvent la première étape du processus de recrutement. Elle s’effectue souvent avec un ou deux développeurs de l’entreprise.
* Généralement de 30 à 45 minutes, elle peut parfois durer 1h/1h30 et c’est souvent un très bon signe.
* Elle porte sur des concepts généraux d’informatique que sur des concepts relatifs à votre spécialité.
* Pour la préparer le mieux possible, lisez régulièrement des articles et regardez régulièrement des vidéos.

Entrainez-vous aux tests d’algorithmiques

Vous venez de réaliser votre discussion technique et cette dernière s’est très bien passée ? Félicitations à vous ! Nul doute que vous avez été capable durant cette dernière de montrer vos compétences et vos connaissances.

Vous avez suffisamment bon pour que l’entreprise désire passer à l’étape d’après avec vous et vous demande de réaliser des tests d’algorithmiques.

* Dans ce chapitre, je vais vous : expliquer ce que sont les tests d’algorithmiques ;
* donner l’objectif de ces derniers ;
* proposer des aides à la préparation et à la résolution de ces derniers.

Bouclez votre ceinture, on passe la seconde et je vais faire de vous des experts en test d’algorithmique !

PLACEHOLDER VIDEO

### Projetez-vous dans la réalisation de tests d’algorithmiques

#### Définition de l’algorithmique et des tests d’algorithmiques

Selon Wikipédia, l’algorithmique, c’est :

l’étude et la production de règles et techniques qui sont impliquées dans la définition et la conception d’algorithmes, c’est-à-dire de processus systématiques de résolution d’un problème permettant de décrire précisément des étapes pour résoudre un problème algorithmique.

Comme vous pouvez le constater, c’est une définition un peu complexe et compliquée pour expliquer un concept qui est, en fait, assez simple.

Pour nous, développeurs, l’algorithmique, c’est **résoudre un problème spécifique via l'exécution d’un programme, grâce à du code** que nous aurons préalablement écrit.

Autrement dit, à la différence d’un projet plus complet comme la création ou la consommation d’une API, un problème d’algorithmique nous fait travailler sur quelque chose de beaucoup plus spécifique.

C’est un peu comme la différence entre la macroéconomie et la microéconomie. D’un côté, en macroéconomie, on étudie les grands agrégats économiques (le revenu, l’investissement, l’inflation) et d’un autre côté, en microéconomie, on étudie des éléments plus précis tels que les consommateurs, les ménages, etc.

Autrement dit, l’algorithmique se rapproche de la microéconomie alors que les projets, tels que vos projets OpenClassrooms, sont, eux, plus proches de la macroéconomie.

D’ailleurs, petit point vocabulaire :-). Il semblerait qu’on puisse aussi bien dire “algorithmique” que “algorithmie”. Pendant très longtemps, j’ai personnellement utilisé “algorithmie”. Sachez donc que ce n’est pas forcément choquant d'utiliser l’un ou l’autre.

#### Intérêt des tests d’algorithimiques

Pourquoi me demandera-t-on de réaliser des tests d’algorithmiques ?

Tout d'abord, ces tests permettent d'**estimer votre connaissance des types et fonctions natives de votre langage**.

À la différence de la discussion technique et des mini projets, les tests d’algorithmiques vous demandent de travailler sur des éléments précis de votre langage.

Ces éléments peuvent être les suivants :

* manipuler des chaînes de caractère, ou strings, dans votre langage. En JavaScript, il existe de nombreuses méthodes vous permettant de convertir une chaîne de caractères en un tableau, de filtrer et de le retransformer ;
* chercher et supprimer des occurrences via des expressions régulières ;
* convertir en tableau des données sous forme d’objets litéraux ;
* etc...

En fonction des langages informatiques, la manipulation, le traitement et la transformation des données se font de manière différente. Ces tests permettent donc de voir si vous connaissez bien les bases de votre langage ainsi que votre logique.

Deuxièmement, ces testes permettent de **voir votre capacité à appréhender des problèmes et à les résoudre**.

Dans ces tests, à la différence des mini projets, le cadre de réalisation est beaucoup plus restreint. On ne vous demande pas de monter une architecture ou de travailler avec un framework. On vous demande simplement de résoudre un problème précis.

Vous aurez souvent des tests automatisés écrits au préalable qui vous permettront de tester si votre solution est viable. En fait, faire des tests d’algorithmiques, c’est un peu comme travailler avec des tests unitaires.

Les tests unitaires permettent de tester une fonction ou une classe spécifique, par exemple, faire la somme des prix pour une commande alors qu’un test d’intégration ou fonctionnelle permet lui de tester l’intégralité d’un parcours utilisateur.

#### Le cadre de réalisation

Où et comment réalise-t-on les tests d’algorithmiques ?

En fonction de l’entreprise pour laquelle vous postulez, si c’est plutôt une startup ou une ESN, vous allez soit réaliser les tests de **chez vous** (souvent dans le cadre des ESN), soit les réaliser **dans les locaux de l’entreprise** (dans le cadre des startups).

Vous allez pouvoir réaliser ces derniers soit sur des **plateformes dédiées**, soit via **un ordinateur de la société** qui vous sera mis à disposition pour que vous les réalisez.

Les tests d’algorithmiques donnent parfois lieu à une discussion technique après ou pendant. Il est donc important que vous soyez capable d’expliquer ce que vous avez fait.

### Évitez le copier/coller de solutions

Quand on réalise ces tests, il peut parfois être tentant d’aller chercher la solution directement sur Internet. En effet, en tant que développeur, nous passons une bonne partie de notre temps à lire des bouts de code d’autres développeurs.

Il peut parfois arriver que nous soyons bloqué sur un script et que nous cherchions sur Internet une aide pour nous débloquer sur notre problème. C’est une situation tout à fait courante en entreprise.

Cela dit, **en entretien, copier/coller des bouts de code peut s’avérer être une mauvaise idée**. D’une part, parce que vous risquez de copier une solution qui marche ou qui ne marche qu’à moitié.

Pire encore, vous risquez de copier une solution que vous n’êtes pas capable d’expliquer si le recruteur vous des questions dessus. Et croyez-moi, vous ne voulez pas vous faire coincer la dessus en entretien.

Autre point gênant avec le copier/coller, de nos jours, les plateformes sur lesquelles vous réalisez votre test, peuvent analyser quand vous faites des copier/coller. Ne vous tirez donc pas une balle dans le pied, comprenez les solutions que vous trouvez sur Internet et sachez les expliquer.

### Exemples de tests d’algorithmiques

Bien que les tests d’algorithmiques soient légion, le concept général est souvent le même. Vous allez devoir manipuler les données en, par exemple, changeant leur type (de nombres à des chaînes de caractère), devoir les trier, sortir les erreurs, etc.

Ces exemples, ne sont pas spécifiques à un langage informatique. Vous pourrez donc les retrouver aussi en JavaScript qu’en Python ou qu’en Swift. L’implémentation et les solutions seront peut-être un peu différentes mais l’approche restera globalement la même.

#### Transformation d'un nombre décimal en nombre romain

Bienvenue dans ce premier screencast dédié aux tests d’algorithmique :).

Dans ce premier test, je vais vous demander de créer une fonction appelé parseToRoman qui pour un nombre donné va retourner sa valeur en chiffres romains.

*function* parseToRoman(*number*) {

}

parseToRoman(4); // === "IV"

parseToRoman(37); // === "XXXVII"

parseToRoman(143); // === "CXLIII"

parseToRoman(1234); // === "MCCXXXIV"

Souvenez-vous que si vous rencontrez des difficultés, pensez à décomposer votre problème en problèmes plus simple et n’hésitez pas à utiliser la méthode des 4P de la résolution de problèmes.

**PLACEHOLDER SCREENCAST**

#### Algorithme de decriptage de ROT13

Dans ce deuxième test, on va faire notre petite machine de Turing et on va craquer un algorithme de chiffrements ;).

Nous allons donc essayer de craquer ce qu’on appelle le ROT13 (ou Rotate By 13 Places). L’idée est de décaller chacune des lettres de l’alphabet de 13 places. Autrement dit, le H devient le U, le E devient le R, etc.

Si vous voulez en savoir plus sur l'histoire du ROT13, vous pouvez vous rendre sur sa page Wikipédia : [**https://fr.wikipedia.org/wiki/ROT13**](https://fr.wikipedia.org/wiki/ROT13)

C’est un test que je trouve assez intéressant parce que vous y manipulez avant tout des nombres et des chaînes de caractère.

*function* transformToRot13(*stringToTransform*) {

}

toRot13('URYYB JBEYQ'); // HELLO WORLD

toRot13('BCRAPYNFFEBBZF'); // OPENCLASSROOMS

toRot13('PRPV RFG ZBA PBQR FRPERG'); // CECI EST MON CODE SECRET

Bon courage et amusez-vous bien :).

PLACEHOLDER SCREENCAST

### Préparez-vous à ces tests

Jusqu’à présent, je vous ai présenté l’objectif et le contexte de réalisation des tests ainsi que quelques exemples. Comme pour la discussion technique, **la réussite de cette étape se joue dans la préparation**.

Dans cette partie, nous allons voir des outils de préparation possibles pour que vous ne soyez pas surpris le jour J.

#### Approfondissez votre connaissance de votre langage

Bien connaître le framework que vous utilisez au quotidien est une chose, mais connaître les bases de son langage en est une autre.

En effet, le principe des tests d’algorithmiques est que vous travaillez sur votre langage sans aucune librairie ou framework.

Pour en savoir découvrir plus sur votre langage, il n’y a rien d’autres à faire que de **suivre des cours et tuto** (sur la manipulation de dates en python, par exemple) et de le dire la documentation technique de votre langage. Celle de PHP par exemple est plutôt bien réalisé :).

#### Utilisez des méthodes de résolution de problème

“The Four P’s to Problem Solving” est l’une des méthodes les plus connues pour résoudre des problèmes. Cette dernière a pour but de vous aider à**décomposer un problème complexe en de petits problèmes plus simples à gérer**. Elle repose donc sur quatre grandes étapes :

* **Prep** - c’est l’étape où vous allez essayer de comprendre le problème et de le décomposer des parties plus simples.
* **Plan** - c’est le moment où vous allez écrire du “pseudcode” qui correspond aux instructions que vous souhaitez réaliser.
* **Perform** - c’est l’étape où vous transformez votre “pseudocode” en vrai code.
* **Perfect** - enfin, vous essayez d’améliorer le code écrit, en le refactorisant, et en traquant d’éventuels bugs.

Si vous ne savez pas ce qu’est le pseudocode, c’est, selon Wikipédia, une façon de décrire un algorithme en langage presque naturel, sans référence à un langage de programmation en particulier.

# Un exemple de pseudo code

Fonction factorielle (n)

r = 1

Pour i de 1 jusqu'à n avec un pas de 1

r = r\*i

Fin pour

Retourner r

Fin Fonction

Ce que vous devez retenir quand vous bloquez sur un problème d’algorithmique, c’est que vous devez décomposer votre problème complexe en de multiples problèmes plus simples à réaliser.

En fait, c’est un peu comme quand vous jouez aux legos. Vous créez de petites structures à droite, à gauche, puis vous les assemblez pour en faire quelque chose cohérent et de stable.

PLACEHOLDER IMAGE LEGOS

#### Entraînez-vous sur des plateformes

Pour vous aider à préparer ces tests, je vous ai regroupé plusieurs plateformes que vous pouvez utiliser pour vous entraîner.

|  |  |  |  |
| --- | --- | --- | --- |
| Plateforme | Langage | Langage | Niveau |
| [Free Code Camp - Basic algorithm scripting](https://learn.freecodecamp.org/javascript-algorithms-and-data-structures/basic-algorithm-scripting/) | Anglais | JavaScript | Principalement niveau débutant |
| [Free Code Camp - Intermediate algorithm scripting](https://learn.freecodecamp.org/javascript-algorithms-and-data-structures/intermediate-algorithm-scripting/) | Anglais | JavaScript | Débutant à intermédiaire |
| [Code wars](https://www.codewars.com/) | Anglais | Nombreux langages | Tout niveau |

### En bref

* Les tests d’algorithmiques vous font travailler sur votre langage et non sur un framework ou une architecture. Ils sont là pour montrer votre logique et votre capacité à résoudre des problèmes.
* Ils sont réalisés soit dans les locaux de l’entreprise, soit chez vous sur une plateforme informatique dédiée.
* Résoudre le problème est une chose, savoir l’expliquer en est une autre. Evitez les copiers/collers de code que vous ne comprenez pas.
* Pour les résoudre, essayez de décomposer votre problème en des problèmes plus petits et plus simples. C’est comme quand vous jouez aux legos.

## Préparez-vous aux mini-projets de développement

PLACEHOLDER VIDEO 1

Alors cette fois, on rentre dans la cour des grands et on va vous demander de réaliser un vrai projet. Généralement en dernière phase des entretiens techniques, c’est le test qui peut être à la fois le plus stressant mais aussi le plus intéressant tant techniquement qu’humainement.

Il est effectivement stressant et difficile car c’est votre travail qui est jugé et donc, dans un sens, votre esprit critique, votre logique ainsi que votre raisonnement.

Cela dit, ne paniquez pas ! Les développeurs voudrontsimplement pour la plupart**apprendre à vous connaître et savoir comment vous réfléchissez** :).

Si le terme mini-projet ne vous parle pas, vous pouvez aussi utiliser le mot feature ou fonctionnalité. Généralement, ce test technique est très proche de ce qui pourrait vous être demandé en entreprise.

Parfois, l'entreprise vous fournira une base déjà existante et vous devrez intégrer la fonctionnalité dessus. Parfois, ce sera à vous de monter toute l'architecture, en choisissant, par exemple les librairies, l'environnement de développement, etc.

Si le terme vous semble encore un peu abstrait et que vous avez du mal à distinguer ce qui pourrait vous être demandé, voici quelques exemples.

* Une page telle qu'une landing page à intégrer ;
* un moteur de recherche à implémenter ;
* aller chercher des données auprès d'une API pour les afficher.

### Découvrez l’utilité des minis projets pour un recruteur

Mais pourquoi dois-je réaliser ce test ? Je pensais que la discussion technique ou les tests d’algorithmiques étaient suffisants pour donner un avis sur mon niveau :euh:.

Et bien, c'est, en parti vrai. Comme je vous le disais dans les chapitres précédents, vous aurez rarement tous les tests dont je vous parle ici lors du processus de recrutement (même si pour mon recrutement chez Sens Critique, je les ai tous eus ;) ).

Autre point, **c'est le test le plus important et aussi le plus plus intéressant**. En tant que candidat, vous allez avoir un feedback et des retours sur comment vous développez et comment vous améliorez. Vous allez d’ailleurs souvent apprendre de nouvelles choses (une résolution différente d’un problème, une nouvelle librairie, etc.). Pour l’entreprise, c’est le moment de vous évaluer sur votre capacité à développer.

Faisons une petite pause et prenons un instant un peu de hauteur par rapport à ce que je viens de vous dire sur les tests techniques .

* D’un côté, la discussion technique permet au recruteur de connaître un peu plus le candidat et de parler avec lui de concepts généraux.
* Ensuite, les tests d’algorithmiques permettent de voir et de jauger le raisonnement logique et la capacité à appréhender un problème du candidat sur un cas bien précis ;
* Enfin, **le mini-projet permet de vous faire travailler sur un projet concret**. Il va donc montrer au recruteur comment vous travaillez et comment vous réfléchissez. Il va aussi très souvent amener le recruteur à vous poser des questions (Pourquoi as-tu implémenté cette solution ? Est-elle meilleure qu'une autre ? etc.)

En somme, ce test donne un aperçu à vous, candidat, ainsi qu’au recruteur de comment peut se passer la collaboration au jour le jour une fois dans l’entreprise.

Et comment se déroule ce type tests ?

Généralement, ce test se déroule en deux parties.

1. **la première partie est celle où vous allez travailler sur le test**. Vous pouvez le réaliser aussi bien dans les locaux de l'entreprise que chez vous dans votre chambre ou votre bureau ;
2. **la deuxième partie quant à elle se fera plutôt au sein de l'entreprise**. Vous aurez une discussion avec un ou deux développeurs de l'entreprise. Cette partie vous permettra de défendre vos idées et de parler de notions liées à ce que vous avez mis en place.

D’ailleurs, **Il peut aussi arriver que la deuxième partie se fasse en peer programming**. Autrement dit, vous allez développer votre feature avec un autre développeur de l’entreprise et discuter de vos solutions/implémentations.

### Projetez-vous dans l’exercice

Maintenant que je vous ai présenté l’utilité de ce test et ses conditions de réalisation, parlons ensemble des projets potentiels que vous pourriez avoir et pourquoi.

En fonction de votre spécialité, intégrateur web, développeur front-end, back-end, les compétences évaluées ne sont pas forcément les mêmes. Ce qui est finalement assez logique.

En effet, un développeur back-end implémente ce qu’on appelle la “logique métier” autrement dit les règles qui régissent le bon fonctionnement de l’application.

Pour la logique métier, on peut, par exemple, penser aux différents types d’utilisateur d’un site (utilisateur, admin, super admin), récupération, agrégation et traitement des données, etc.

Un développeur back-end peut donc se voir poser des questions de ce type en entretien -comment optimises-tu tes requêtes en BDD ? Comment crées tu et maintiens-tu une API REST ? etc.

Le développeur front-end aura, quant à lui, des questions portées sur l’intégration CSS ainsi que la gestion de la partie dynamique du site (récupération des données via des requêtes AJAX et utilisation d’API, gestion et sécurisation des sessions, etc.)

Pour l’intégrateur web, les questions posées seront avant tout des questions de CSS et de sémantiques HTML ainsi que des questions sur des CMS type WordPress ou PrestaShop.

Si c’est encore trop abstrait pour vous, voici une liste de mini-projets potentiels par spécialité et avec quelques explications complémentaires.

#### Intégrateur Web

* intégrer un dashboard ;
* réaliser une intégration d’une page, page de profil par exemple, en utilisant des préprocesseurs CSS type Sass ;
* réaliser un WordPress ou un Prestashop ;
* intégrer une formulaire de contact ou de newsletter avec des selecteurs CSS avancés.

#### Développeur front-end

* intégrer une maquette, type page de profil, avec un framework front tel que ReactJS/VueJS ;
* réaliser une jeu, de morpion par exemple, en utilisant des design patterns (type module pattern) ;
* réaliser une PWA ;
* récupérer des données auprès d’une API (de météo par exemple) et afficher les données de cette dernière ;
* faire une galerie interactive d’images ;
* mettre en place un player Vidéo Interactif.

#### Développeur back-end

* faire une API Rest ;
* réaliser un blog ou un gestionnaire quelconque (pour une libraire par exemple) et mettre en place le CRUD ;
* réaliser un bot type Scraper.

#### Développeur mobile

* faire un jeu ;
* utiliser une API pour afficher des données (ce qui vous fera travailler les appels réseaux) ;
* réaliser un logiciel type gestion de parcs d’attraction (avec différents droits utilisateurs et zones d’accès possible)

Comme vous pouvez le constater, il existe de très nombreux sujets possibles et je ne pourrais pas forcément tous vous les citer surtout que la plupart se base sur mon expérience professionnelle tant comme candidat que comme recruteur. De plus, les vidéos complémentaires en bas de page vous permettront d’aller plus dans la préparation de ce test.

### Adoptez la bonne attitude pendant le développement

PLACEHOLDER VIDEO 2

Durant la phase de développement du mini-projet, vous allez devoir mettre en avant toutes vos connaissances et compétences acquises au sein de votre formation.

Voici quelques conseils qui vont vous permettre de réaliser cette étape sereinement.

#### Indentez proprement votre code et suivez les bonnes pratiques de développement

Quand on réalise ce genre de tests, le stress aidant, on peut parfois être tenté d’aller un peu vite et d’écrire du code un mode un peu “crade”.

render() {

return (

<View style={styles.onboarding}>

<Swiper

activeDotColor="#FFF"

dotStyle={{

backgroundColor'#17B96A'

}}

loop={false}

style={styles.wrapper}

>

<View style={styles.slide}>

<Image source={logo} style={styles.screen1Logo} />

<Image source={screen1} style={styles.screen1Text} />

</View>

<View style={styles.slide}>

<Image source={screen2} style={styles.onBoardingImg} />

</View>

<View style={styles.slide}>

<Image source={screen3} style={styles.onBoardingImg} />

</View>

</Swiper>

<View style={styles.footer}>

<TouchableOpacity

onPress={() => {}}

style={styles.button}

>

<Text style={styles.buttonText}>

C'est parti

</Text>

</TouchableOpacity>

</View>

</View>

)

}

Voici clairement le genre de code à éviter. Il n’est pas convenablement indenter, il n’y a pas de commentaires. En bref, il est très peu lisible.

D’ailleurs, je vous conseille d’**utiliser unlinter** et **de suivre des conventions établies** pour votre langage. Cela vous permettra d’avoir un code conforme au standard :).

Qu’est-ce qu’un linter ?

C’est un outil qui vous permet d’améliorer la qualité de votre code source. En enlevant les console.log de votre code par exemple. Voici [un article de blog d'un développeur français](https://blog.nathanaelcherrier.com/fr/linting-good-practices/), qui explique bien l’utilité d’un linter.

Concernant les conventions, elles peuvent s’appliquer en complément de votre linter. Elles vont permettre d’enrichir les bonnes pratiques lors de votre développement.

Voici quelques exemples de conventions.

* Le guide JavaScript d'Airbnb - [github.com/airbnb/javascript](https://github.com/airbnb/javascript)
* Le guide ReactJS d’Airbnb - [github.com/airbnb/javascript/tree/master/react](https://github.com/airbnb/javascript/tree/master/react)
* Un autre exemple de guide JavaScript - [standardjs.com/](https://standardjs.com/)

#### Soignez vos nommages et vos commentaires

En plus des conventions, essayez de bien soigner vos nommages tant au niveau de vos variables que de vos fonctions.

Qu’est-ce que le nommage ?

Quand vous créez une fonction ou une variable, ou même une classe, vous lui donnez un nom, non ? Idéalement, vous allez essayer de lui donner un nom représentatif, qui donne un indice sur ce que fait la fonction.

Le nommage, c’est ce qui permet de bien comprendre ce que fait une fonction ou une variable sans que vous ayez à plonger le nez dans le code. Un bon nommage vous permet de ne pas trop commenter votre code.

Si vous souhaitez aller plus loin, voici quelques articles.

* [Un article de blog francophone](https://blog.cellenza.com/dotnet/clean-code-nommer-vos-variables/) pour comprendre pourquoi il faut bien nommer ses variables ou fonctions.
* [Un article de blog anglophone](https://hackernoon.com/the-art-of-naming-variables-52f44de00aad) vous aidant à trouver de bons noms de variables.

Comme vous pouvez le constater, **le nommage est quelque chose de très important**. Cela permet de comprendre facilement ce que fait une fonction ou alors quel est le type d’une variable utilisée.

**Concernant les commentaires, essayez de trouver un équilibre**. Vous ne pouvez pas commenter chacune de vos lignes mais vous ne devez pas non plus rien commenter.

Là, encore, vous pouvez vous rapprocher de conventions tels que la JSDoc pour le JavaScript - [site officiel de la JS Doc](http://usejsdoc.org/). Cette convention vous permettra d’avoir des commentaires utiles et compréhensibles par le plus grand nombre  :).

#### Ne faites pas de copier/coller

Dernier point, comme je vous le disais avec les tests d’algorithmiques, ne réalisez pas de copier/coller lors que vous développez votre mini-projet.

En effet, une fois le mini-projet terminé, vous allez souvent devoir en discuter avec un ou deux des développeurs de l’entreprise. Vous devez donc être capable d’expliquer ce que vous avez fait.

### Mettez en avant vos compétences et argumentez sur vos choix techniques

Pour la phase de discussion/argumentation, les choses ici sont un peu différentes. Il va surtout vous être demandé d'expliquer ce que vous avez fait et pourquoi. Autrement dit, c'est le moment de défendre votre travail !

* **Sachez expliquer au mieux pourquoi vous avez pris telle ou telle décision**. Vous n'aurez pas forcément tout le temps la réponse mais ce n’est pas bien grave. Vous n'avez pas à tout savoir sur tout. C'est ce dont je vous parlais dans une dans une précédente partie, sachez dire "je ne sais pas". C'est souvent une bonne ouverture au dialogue.
* **Ne soyez pas sensible à la critique !** J'aurais aussi pu dire soyez ouvert aux améliorations. Il n'y a rien de pire, d'après moi, qu'un développeur n'acceptant pas la critique ! Songez-y, quand on critique votre travail, on vous aide à devenir un meilleur développeur !
* Soyez humble !
* Soyez naturel. Après tout, vous êtes là pour montrer que vous aimez développer et résoudre des problèmes :).

D'ailleurs, lors d'une séance en peer programming, vous réaliserez la partie discussion/argumentation en même temps.

Ne soyez pas dérouté par cette méthode - c'est souvent assez ludique et cela vous donne une vraie idée de l'ambiance de travail.

### Entraînez-vous !

Prêt à coder ? Tant mieux !

Pour aller plus loin, **j'ai préparé deux potentiels mini-projets sur lesquels travailler.**Le premier axé développeur front-end et le deuxième axé développeur back-end.

Je vous invite à regarder celui qui correspond le mieux à votre profil, ou bien les deux si vous êtes curieux :)

Chacune de ces deux sous-parties comprendra une présentation et une analyse du mini-projet. Si tous les concepts abordés ne vous parlent pas, ne paniquez pas, c'est normal !

#### Le mini-projet front

PLACEHOLDER SCREENCAST

#### Le mini-projet back

PLACEHOLDER SCREENCAST

### En bref

* Le mini projet, c’est souvent la conclusion des tests techniques. Vous allez travailler sur un exemple concret et réel qui pourrait d’ailleurs être une feature.
* Ce test permet au recruteur d’évaluer votre niveau et vos compétences. Il peut être réalisé chez vous ou en entreprise.
* Il est composé de deux parties - une partie développement et une partie présentation/argumentation.
* Soyez le plus rigoureux possible durant la phase de développement et soigner votre code.
* Pour la partie présentation/argumentation, défendez vos choix tout en acceptant la critique.

## Faites le point sur les tests techniques

Vous voilà arrivé au terme de ce cours, merci de l’avoir suivi. Pour que vous puissiez valider vos connaissances, je vous ai prévu un quiz et une activité.

Ce quiz va vous permettre de vérifier que vous avez bien compris le cours et ce qui est attendu. L’activité, quant-à-elle, vous permettra de mettre le tout en pratique.

Happy coding :).

### Question 1

**Dans quelle ordre se déroulent les étapes du processus de recrutement ?**

* + 

L’entretien avec les managers/fondateurs, la discussion technique, les tests d’algorithmiques, le mini projet, le contact RH.

* + 

Le mini projet, le contact RH, L’entretien avec les managers/fondateurs, la discussion technique, les tests d’algorithmiques.

* + 

Le contact RH, la discussion technique, les tests d’algorithmiques, le mini projet, L’entretien avec les managers/fondateurs.

* + 

La discussion technique, L’entretien avec les managers/fondateurs, les tests d’algorithmiques, le mini projet, le contact RH

*Comme nous l’avons vu dans le cours, les étapes du processus de recrutement se situent dans cet ordre : le contact RH, la discussion technique, les tests d’algorithmiques, le mini projet et enfin l’entretien avec les managers/fondateurs.*

### Question 2

**À quoi sert le contact RH ?**

* + 

À parler prétentions salariales et congés payées ?

* + 

À me présenter et à savoir ce que j’ai compris du poste ?

* + 

À parler de concepts de programmation ?

* + 

À parler de la dernière saison de Game of Thrones ?

*le contact RH est souvent un contact de présentation. Il permet de vous présenter et de savoir si vous avez bien compris le poste.*

*Petit conseil : soyez dynamique et intéressé durant cette partie :).*

### Question 3

**La discussion technique permet au recruteur**

* + 

D’en apprendre plus sur vous.

* + 

De connaître vos précédentes expériences.

* + 

De parler de votre salaire.

* + 

De parler de votre spécialité.

* + 

De parler de vos congés payés.

* + 

De parler du dernier framework ou de la dernière librairie à la mode.

*durant cette discussion, vous allez avant tout parler de technique. Autrement dit, cela peut aussi bien être de parler de la dernière librairie JavaScript à la mode que de votre éditeur de textes.*

*L’idée est de mieux comprendre vos expériences et vos connaissances théoriques et techniques en programmation.*

### Question 4

**Quelles sont les bonnes pratiques à respecter durant la discussion ?**

* + 

Être sur la défensive.

* + 

Être dans une posture d’apprentissage.

* + 

Poser des questions sur l’entreprise et les technologies qu’elle utilise.

* + 

Se montrer curieux sur les choix technologies.

* + 

Faire des schémas.

* + 

Être vague.

*pour bien réussir cette étape, essayez de suivre ces indications :*

* + *montrez-vous curieux et intéressé par les technologies de l’entreprise ;*
  + *faites des schémas pour expliquer des notions complexes ;*
  + *soyez dans une posture d’apprentissage.*

### Question 5

**Comment vous préparer à la discussion technique ?**

* + 

en vous rendant à des meetups et en discutant avec vos collègues développeurs.

* + 

en parlant de notions très techniques à vos proches sans les vulgariser.

* + 

en regardant des vidéos sur le développement personnel.

* + 

en copiant/collant du code.

*comme nous l’avons vu durant le cours, pour bien préparer la discussion technique, vous devez baigner dans un environnement propice à l’apprentissage.*

*Autrement dit, vous devez essayer de regarder des conférences et de lire des blog posts techniques régulièrement. Essayez aussi parler régulièrement avec d’autres développeurs et d’expliquer des notions complexes à vos proches.*

### Question 6

**Choisir les bonnes ressources pour vous aider à vous préparer à la discussion technique ?**

* + 

[CSS Tricks](https://css-tricks.com/)

* + 

[Blog Medium d'Eric Elliott](https://medium.com/@_ericelliott)

* + 

[A List Apart](https://alistapart.com/)

* + 

[Facebook](https://www.facebook.com/)

* + 

[Alsacreations](https://www.alsacreations.com/)

* + 

[Overreacted](https://overreacted.io/)

* + 

[Documentation officielle de Symfony](https://symfony.com/)

* + 

[Gamekult](https://www.gamekult.com/)

*Essayez d’avoir de nombreuses sources d’apprentissages, aussi bien francophone qu’international. Sachez consulter des blogs d’expert mais aussi des documentations de framework.*

*Vous ne comprendrez pas tout tout de suite mais plus vous allez pratiquer et lire de la documentation, plus vous allez progresser et être prêt pour les entretiens techniques.*

### Question 7

**Parmi les exemples suivants, lesquels sont des exemples de tests d’algorithmiques.**

* + 

Établir la complexité d’un mot de passe.

* + 

Établir si une adresse e-mail est correcte ou non.

* + 

Créer un gestionnaire de librairie.

* + 

Créer un jeu en JavaScript.

* + 

Transformer des chiffres en lettres.

* + 

Réaliser un thermomètre.

* + 

Convertir des degrés celsius en degrés fahrenheit.

*le jeu en JavaScript ou même le thermomètre sont des tests beaucoup trop complexes pour être des exercices d’algorithmiques.*

*Le but d’un test d’algorithmique est de vous faire travailler sur quelque chose de spécifique.*

### Question 8

**Trouvez la bonne solution pour un test d’algorithmique.**

**On vous demande de réaliser le test d’algorithmique suivant : vous devez changer le sens d’un mot, autrement dit, “bonjour” devient “ruojnob”. Quel serait le script JavaScript utilisé ? (si le JavaScript n’est pas votre langage de prédilection, vous pouvez essayer de réaliser cet exercice dans le langage de votre choix).**

*function* reverseString(*str*) {

}

reverseString("hello"); // -> "olleh"

* + 
  + *function* reverseString(*str*) {
  + str = str.replace(/[\W\_]/g, '').toLowerCase();
  + *var* strReverse = str
  + .replace(/[\W\_]/g, '')
  + .toLowerCase()
  + .split('')
  + .reverse()
  + .join('');
  + if (str === strReverse) {
  + return true;
  + } else {
  + return false;
  + }
  + }
  + 
  + *function* reverseString(*str*) {
  + str = str.toLowerCase().split(' ');
  + for(*var* i = 0; i < str.length; i++){
  + str[i] = str[i].split('');
  + str[i][0] = str[i][0].toUpperCase();
  + str[i] = str[i].join('');
  + }
  + return str.join(' ');
  + }
  + 
  + *function* reverseString(*str*) {
  + return str.split('').reverse().join('');
  + }
  + 
  + *function* reverseString(*str*) {
  + if (num > 0) {
  + *var* arrOne = [];
  + *var* arrTwo = [];
  + *var* i = 0;
  + for (i = 0; i < num; i++) {
  + arrTwo = str.split('');
  + arrOne.push(arrTwo);
  + }
  + arrTwo = [];
  + for (i = 0; i < arrOne.length; i++) {
  + arrTwo.push(arrOne[i].join(''));
  + }
  + arrOne = '';
  + arrOne = arrTwo.join('');
  + return arrOne;
  + } else {
  + return '';
  + }
  + }

*La réponse 1 permet de vérifier si le mot est un palindrome. Autrement dit que le mot, ou le groupe de mots, peut s’écrire dans les deux sens, par exemple Kayak.*

*La réponse 2 permet d’écrire une phrase en lettres capitales. Autrement dit, chaque première lettre d’un mot va être écrit en capital.*

*La réponse 4 permet de répéter une phrase. On voit d’ailleurs qu’il manque l’argument num à la fonction*

### Question 9

**Quel est l’ordre des 4 P de la résolution de problèmes, utile pour résoudre les tests d’algorithmiques.**

* + 

Prep, Plan, Perform, Perfect.

* + 

Perform, Plan, Prep, Perfect.

* + 

Prep, Perform, Perfect, Plan.

* + 

Plan, Prep, Perfom, Perfect.

* + *Prep correspond à la partie “préparation” du projet. Vous essayez de comprendre le problème et de le diviser en des problèmes plus simples à résoudre.*
  + *Plan correspond à la planification. Vous écrivez du pseudo-code comprenant les instructions/étapes à réaliser.*
  + *Perform est la partie où vous transformez votre pseudo-code en vrai code.*
  + *Perfect est le moment où vous vignollez votre code.*

### Question 10

**Comment bien appréhender un test d’algorithmique complexe ?**

* + 

en divisant le problème en de multiples petits problèmes.

* + 

en planifiant la résolution du problèmes via du pseudocode.

* + 

en essayant de tout résoudre en même temps.

* + 

en utilisant des librairies qui pourraient faire le travail pour vous.

* + 

en réalisant des copiers/collers de solutions trouvés sur internet.

* + 

en ajoutant des commentaires explicatifs vous aidant à décomposer le problème.

*réaliser des tests d’algorithmiques en entretien peut s’avérer assez stressant. On vous demande souvent d’en réaliser entre 2 et 5 dans un temps limité.*

*Essayez de toujours garder en tête que vous devez décomposer vos problèmes en de petits problèmes plus simples à gérer. N’hésitez pas à vous entraîner régulièrement sur des sites tels que CodeWars :).*

Bas du formulaire

Réalisez un mini-projet

# Consignes

Bienvenue dans l’activité relative au cours sur les techniques techniques pour les développeurs.

Pour cette activité, je vous ai préparé un mini-projet typique que je pourrais donner à un candidat pour un entretien de recrutement. Ce dernier s’adresse avant tout aux développeurs web (vs des développeurs Android ou iOS) et vous devez avoir quelques bases en HTML/CSS pour le réaliser.

Vous devez réaliser l’activité en deux temps :

* Dans un premier temps, intégrez [cette maquette](https://s3-eu-west-1.amazonaws.com/course.oc-static.com/courses/6045521/maquette-mini-projet.png) à partir de ce code source (**LIEN A AJOUTER ?**).
  + Notez bien : n’étant pas graphiste, je vous ai fourni une maquette au format PNG. Ce n’est donc pas grave si votre version intégrée ne ressemble pas totalement à l’image.
* Puis, dans un second temps, listez des questions que l’on pourra vous poser avec leur réponses possibles. Vous devez trouver un minimum de 5 questions : ces dernières peuvent porter sur le html, le css et la programmation de manière générale (par exemple, les requêtes HTTP et ou les boucles).

# Livrables

* Le lien du repository GitHub avec la maquette intégrée ;
* Un document texte avec votre liste de questions et réponses.

# Conseils

Gardez bien en tête que ce mini-projet est là pour vous faire pratiquer :). L’idée est donc que vous fassiez le maximum pour rendre un projet de qualité professionnelle.

Faites donc attention à vos nommages, à vos commentaires, à la façon dont vous organisez votre CSS, etc. Vous êtes, par ailleurs, libre d’utiliser un framework CSS tel que Bootstrap.

Pour les étudiants OpenClassrooms, vous pouvez aussi demander à votre mentor de regarder votre code. C’est en quelque sorte un projet bonus :).

Codez bien :-)

Bienvenue dans la correction de l’activité. J’espère que vous avez bien travaillé et que vous êtes fier du travail que vous avez réalisé pour cette dernière.

Pour vous aider dans votre travail de correction, vous avez à votre disposition le code source que j’ai réalisé pour le mini-projet.

Inspirez-vous de cette solution pour réaliser votre correction mais gardez bien en tête que c’est la cohérence et la logique du travail qui comptent.

Il y a deux documents dans le corrigé :

* un premier se concentre sur la partie code du projet.
* un deuxième donne des exemples de questions et de réponses possibles.

### La partie HTML

Pour cette partie, essayez de voir si l’étudiant a bien compris le HTML. Pour vous aider, voici des questions que vous pouvez vous poser :

* Le code HTML est-il compréhensible ? Est-il, par exemple, bien indenté et commenté ?
* Le code génère-t-il des erreurs lors que vous utilisez un validateur (<https://validator.w3.org/>). Si oui, tenez-en compte lors de votre correction.
* Si le projet utilise Bootstrap, est-ce qu’il a bien été intégré au projet ?

Si les textes ou les images sont différents de la maquette, ce n’est pas grave. L’important est ici de se concentrer sur l’organisation du HTML et sur le sémantique du projet :).

### La partie CSS

Cette partie va dépendre si l’étudiant a utilisé ou non un framework CSS. En effet, si un framework CSS a été utilisé, le code CSS va être fortement allégé. Néanmoins, posez-vous les questions suivantes :

* Le code CSS est bien indenté et commenté ?
* Genère-t-il des erreurs via un validateur ? - <https://jigsaw.w3.org/css-validator/>
* Si un framework CSS a été utilisé, a-t-il été bien utilisé ? Autrement dit, n’y a-t-il pas du CSS en trop ?
* Est-ce que l’étudiant utilise des media querries ? Si oui, sont-ils utilisés de façon pertinentes ?
* Enfin, le code CSS est-il ordonné ? Autrement dit, est-ce qu’il vise les points les plus génériques (comme la couleur ou la police de fond) avant de s’attarder sur les points les plus précis ?

### Les questions possibles et leur réponses

Pour cette partie là, les étudiants doivent proposer 5 questions. Ces questions vont devoir être pertinents, par exemple, qu’est-ce que le responsive web design ou le mobile first ?

L’idée de cette partie est de pouvoir prendre du recul par rapport à ce qui est réalisé. Les questions doivent donc être les plus précises possible. Elles doivent aborder des notions importantes au développement web.

Concernant les réponses, ces derniers doivent faire quelques lignes et expliquer le concept de façon générale. Demandez-vous si ces questions pourraient tomber lors d’un entretien technique et comment vous y répondriez.

**Réaliser un mini-projet de développement**

La compétence est validée, si l'étudiant atteint au moins 6 sur 10 en appliquant le barème ci-dessous.

### HTML

0 - le code html n’est pas correctement indenté, il n’y a pas non plus de commentaires. Il n’utilise pas les balises HTML 5 (section, article).   
2 - le code est correctement indenté.  
3 - le code est correctement indenté et comprend des commentaires explicatifs.  
5 - le code est correctement indenté, comprendre des commentaires explicatifs et il utilise le html 5.

### CSS

0 - Le code CSS n’est pas présent et s’il y l’est, il n’est pas correctement indenté ou commenté.  
2 - Il y a un peu de code CSS mais le projet s’affiche mal sur mobile, tablette et desktop.  
3 - Le code CSS est présent mais il y a des bugs à l’affichage. Si le projet utilise Bootstrap, il y a trop de CSS utilisé et on se sert trop peu de Bootstrap.  
5 - Le CSS est correctement indenté et organisé. Si le projet utilise Bootstrap, peu de CSS a été écrit pour utiliser au maximum les fonctionnalités de Bootstrap.

### Compétence 2

**Défendre ses choix de développement**

La compétence est validée, si l'étudiant atteint au moins 6 sur 10 en appliquant le barème ci-dessous.

### Questions

0 - Aucune question n’a été rédigée ou alors ces derniers ne sont pas pertinentes.  
1 - Il y a un ou deux questions possibles sur le projet. Cela dit, ces dernières ne sont pas assez complètes et/ou précises.  
3 - Il n’y a pas les 5 questions possibles. Cela dit, ces dernières sont précises et pertinentes.   
5 - Il y a les 5 questions possibles. Les questions sont précises et pertinentes.

### Réponses

0 - Il n’y a aucune réponse aux questions.  
1 - Il y a quelques réponses mais ces dernières ne sont pas claires ou pas assez explicatives.  
3 - Il manque une partie des réponses ou les réponses proposées ne sont pas assez claires.  
5 - Toutes les réponses aux questions ont été réalisées et ces dernières sont claires et expliquent bien les notions.