**Nom :** Joets

**Prénom :** Ethan

**Nom du TD/~~TP~~  :** VueJs

**Instructions:**

* **Format :** Le bilan peut être rédigé sous forme de document écrit ou présenté sous forme de diaporama si cela est plus adapté.
* **Déposez un fichier PDF**
* **Clarté et Précision :** Assurez-vous que toutes les sections sont bien complétées et que les informations sont présentées de manière claire et structurée.
* **Respect des délais :** Remettez votre bilan dans les délais impartis pour permettre une évaluation en temps voulu.

# I. Étape Terminée

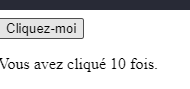
## Indiquez la dernière étape terminée :

* + **Numéro de l'étape :** Exercice N°12.
  + **Description de l'étape :** v-bind Création d'un formulaire de réservation d'hôtel

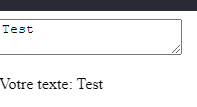
# II. Copies d’Écran

## Présentez des copies d’écran de votre application finale :

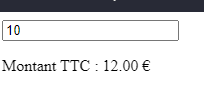
* + Exercice 1: Créer un compteur de clics



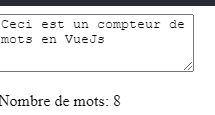
* + Exercice 2: Créer un éditeur de texte en temps réel



* + Exercice 3 :Créer une calculatrice de TVA



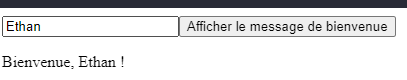
* + Exercice 4 :Compteur de mots



* + Exercice 5 :v-if



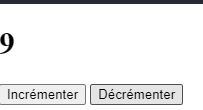
* + Exercice 6 :v-bind



* + Exercice 7 :Case à cocher



* + Exercice 8 :Conversion



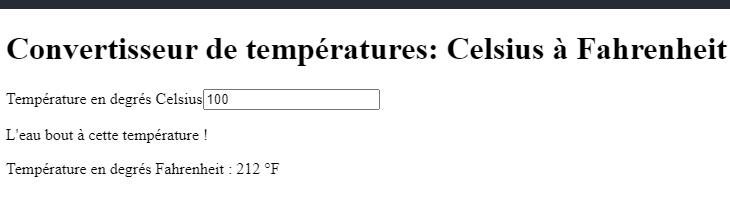
* + Exercice 9 :v-for



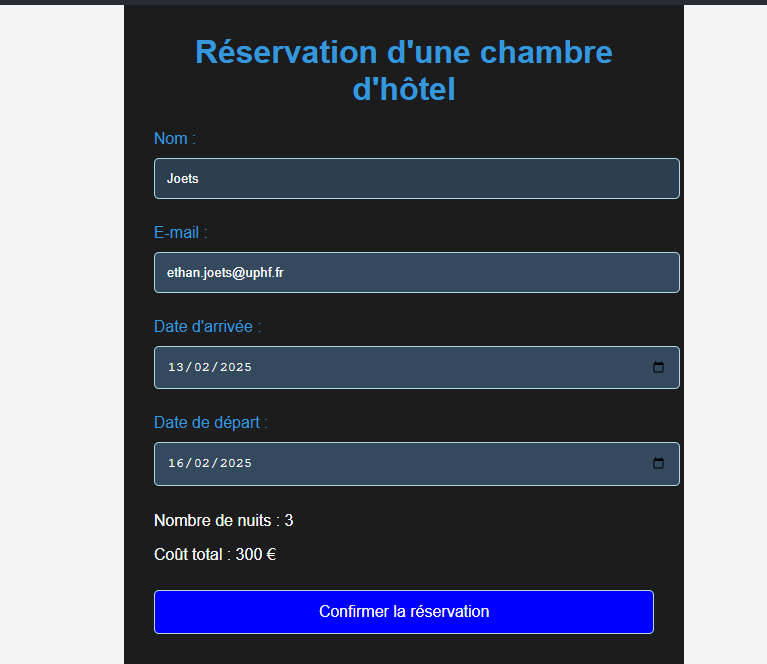
* + Exercice 10 :v-if bienvenue



* + Exercice 11 :v-if conversion de température



* + Exercice 12 :v-bind création d’un formulaire



## Commentaires sur les copies d’écran :

* + Décrivez les fonctionnalités visibles sur chaque capture d’écran.
    - Pour l’exercice 1, le but est de créer un bouton qui incrémente un compteur et affiche le nombre total de clics.
    - Quant à l’exercice 2, on cherchait à créer une zone de texte qui pouvait être affiché en temps réel à côté.
    - Pour l’exercice 3, on voulait créer un champ de saisie pour un montant hors taxe et afficher son équivalent TTC.
    - Pour l’exercice 4, on crée une zone de saisie de texte et un compteur qui affiche le nombre de mots saisis en temps réel.
    - Pour l’exercice 5, on crée une application qui affiche un message différent en fonction d'une condition.
    - Pour l’exercice 6, on crée une application Vue 3 qui affiche un message de bienvenue personnalisé.
    - Pour l’exercice 7, on crée une application qui permet de cocher et décocher une case, et modifier le texte en fonction de cela.
    - Pour l’exercice 8, on convertie un fichier vue3 en vue2.
    - Pour l’exercice 9, on crée une application Vue3 qui affiche une liste de noms de fruits. Pour chaque fruit, on affiche un message personnalisé qui inclut le nom du fruit et sa longueur.
    - Pour l’exercice 10, on crée une application Vue qui affiche un message de bienvenue en fonction de la langue sélectionnée par l'utilisateur.
    - Pour l’exercice 11, on crée une application Vue qui permettra aux utilisateurs de convertir une température de degrés Celsius à Fahrenheit.
    - Pour l’exercice 12, on crée un formulaire interactif qui permet à l'utilisateur de saisir ses informations personnelles et de réserver une chambre d'hôtel.
  + Le design de l’interface reste simple et suit ce qui est présent dans le sujet de manière globale.

# III. Fonctionnalités Implémentées

## Liste des fonctionnalités développées :

* 1. Ajout d’un compteur qui incrémente à l’aide d’un bouton,d’une instance et de v-on
  2. Ajout d’une application qui permet un affichage en temps réel à partir d’une zone de texte grâce à v-model
  3. Ajout d’une application permettant la conversion d’un montant sans taxe vers TTC à l’aide de v-model
  4. Ajout d’une application comptant le nombre de mots en direct dans une zone de texte à l’aide de v-model et newText
  5. Ajout d’une application qui fait des affichages en fonction d’une condition à l’aide de v-if et la condition isTrue
  6. Ajout d’une application qui affiche un message de bienvenue selon le nom entré dans la zone de texte à l’aide de v-bind
  7. Ajout d’une application qui permet de cocher ou décocher une case grâce à v-model
  8. Modification d’une application faite en vue3 vers vue2
  9. Ajout d’une application qui permet d’afficher une liste de fruits et le nombre de lettres composant chacun d’eux à l’aide de data()
  10. Ajout d’une application qui affiche un message de bienvenue en fonction de la langue souhaitée grâce à v-if
  11. Ajout d’une application qui convertie des degrés vers des Fahrenheit et fait un affichage si la température dépasse les 100°c à l’aide de v-if
  12. Ajout d’une application qui donne un formulaire de réservation détaillée à l’aide de v-bind

## Description détaillée :

* 1. Chaque fonctionnalité fait appel au Framework VueJs.
  2. Dans l’exercice 1, on crée une application Vue avec un bouton qui incrémente un compteur à chaque clic. Tout d'abord, un conteneur avec l'ID "app" est ajouté dans le fichier HTML, servant de point d'ancrage pour l'application Vue. À l'intérieur de ce conteneur, un bouton avec le texte "Cliquez-moi" est placé, accompagné d'un élément <p> qui affichera le nombre total de clics. Une instance Vue est ensuite créée et montée sur cet élément, ce qui permet de lier la logique Vue au DOM. Dans cette instance, une donnée compteur est initialisée à 0 pour stocker le nombre de clics. Une méthode est également définie dans la section methods pour incrémenter cette donnée chaque fois que le bouton est cliqué. L'événement de clic du bouton est lié à cette méthode via la directive v-on:click, ce qui garantit qu'à chaque clic, la méthode sera exécutée, augmentant le compteur et mettant à jour dynamiquement l'affichage dans l'élément <p>.
  3. Dans l’exercice 2, on va créer une application Vue qui permet de saisir du texte et de l'afficher en temps réel. Tout d'abord, un conteneur avec l'ID "app" est créé dans le fichier HTML pour servir de point d'ancrage à l'application Vue. À l'intérieur de ce conteneur, un élément <textarea> est ajouté pour permettre à l'utilisateur de saisir du texte. En parallèle, un élément <p> est ajouté pour afficher dynamiquement le texte saisi. Une nouvelle instance Vue est ensuite créée et montée sur l'élément avec l'ID "app", ce qui permet à Vue de gérer et de lier les données au DOM. Dans cette instance Vue, une donnée est définie pour stocker le texte saisi, initialisée avec une chaîne vide. La liaison bidirectionnelle entre le champ de saisie et la donnée est établie à l'aide de v-model, ce qui permet que chaque modification du texte dans le <textarea> soit immédiatement reflétée dans la donnée.
  4. Dans l’exercice 3, on crée une application Vue permettant de calculer le montant TTC à partir d'un montant hors taxe en utilisant un taux de TVA fixe de 20%. Tout d'abord, un conteneur avec l'ID "app" est ajouté dans le fichier HTML pour servir de point d'ancrage à l'application Vue. À l'intérieur de ce conteneur, un champ de saisie est placé pour permettre à l'utilisateur d'entrer un montant hors taxe. Un élément <p> est également ajouté pour afficher le montant TTC calculé. Une instance Vue est ensuite créée et montée sur l'élément avec l'ID "app". Dans cette instance Vue, une donnée est définie pour stocker le montant HT, initialisée à 0, et v-model est utilisé pour établir une liaison bidirectionnelle entre cette donnée et le champ de saisie. Une propriété calculée est ensuite définie pour calculer le montant TTC en appliquant un taux de TVA de 20% au montant HT. Cette propriété utilise la méthode toFixed(2) pour arrondir le résultat à deux décimales, assurant un affichage précis du montant TTC. Enfin, l'affichage du montant TTC est réalisé via l'interpolation Vue, garantissant que le montant est mis à jour automatiquement à chaque modification du montant HT.
  5. Dans l’exercice 4, on crée une application Vue qui permet de saisir du texte et d'afficher en temps réel le nombre de mots du texte saisi. Tout d'abord, un conteneur avec l'ID "app" est ajouté dans le fichier HTML, servant de point d'ancrage pour l'application Vue. À l'intérieur de ce conteneur, un élément <textarea> est placé pour que l'utilisateur puisse entrer du texte. Un élément <p> est également ajouté pour afficher dynamiquement le nombre de mots du texte. Une nouvelle instance Vue est ensuite créée et montée sur l'élément avec l'ID "app", ce qui permet à Vue de lier les données et les méthodes au DOM. Dans l'instance Vue, deux données sont créées : une pour stocker le texte saisi, initialisée avec une chaîne vide, et une autre pour le nombre de mots, initialisée à zéro. On utilise la directive v-model pour établir une liaison bidirectionnelle entre la donnée de texte et la zone de saisie. Ensuite, une fonction watch est définie pour observer les changements dans le texte saisi et mettre à jour automatiquement la donnée du nombre de mots. À chaque modification du texte, la fonction watch utilise l'instruction newText.split(' ').filter(n => n != '').length pour compter le nombre de mots, en divisant le texte par des espaces et en filtrant les éléments vides, puis en calculant la longueur du tableau résultant. Enfin, le nombre de mots est affiché en temps réel dans l'élément <p>.
  6. Dans l’exercice 5, on crée une application Vue.js qui affiche un message différent en fonction de la valeur d'une condition. Tout d'abord, on crée une instance de Vue.js, et dans cette instance, une propriété isTrue est définie dans les données (data) et initialisée avec la valeur true. Ensuite, dans le template HTML, on utilise la directive v-if pour conditionner l'affichage du message. Si la valeur de isTrue est true, Vue affichera le message "La condition est vraie", sinon il affichera "La condition est fausse". Cette logique permet d’afficher dynamiquement un message en fonction de l'état de la variable. Enfin, pour modifier la valeur de isTrue depuis la console, on peut utiliser la méthode this.isTrue = !this.isTrue dans la console du navigateur, ce qui alterne la valeur de isTrue et met à jour l'affichage en fonction de la nouvelle valeur de la variable.
  7. Dans l’exercice 6, on crée une application Vue 3 qui affiche un message de bienvenue personnalisé en utilisant la string interpolation avec la directive v-bind. Tout d'abord, un champ de saisie de texte est ajouté dans le template pour permettre à l'utilisateur d'entrer son nom. Un bouton intitulé "Afficher le message de bienvenue" est ensuite ajouté. Lorsque l'utilisateur clique sur ce bouton, une méthode est déclenchée pour mettre à jour le message de bienvenue. La donnée qui stocke le nom de l'utilisateur est liée à la zone de saisie à l'aide de la directive v-model, permettant une liaison bidirectionnelle. Ensuite, dans la méthode liée au bouton, un message de bienvenue est affiché sous la forme "Bienvenue, [nom de l'utilisateur] !" en utilisant la string interpolation (par exemple, {{ message }}). Le message est dynamique et change en fonction du nom que l'utilisateur saisit. Pour afficher ce message, on utilise la directive v-bind pour lier l'élément à la donnée contenant le message de bienvenue, permettant ainsi d’afficher le texte personnalisé en temps réel.
  8. Dans l’exercice 7, on crée une application Vue.js qui utilise une case à cocher et met à jour un texte affiché à côté de celle-ci en fonction de son état. Dans le fichier HTML, un conteneur avec l'ID "app" est créé, où l'application Vue sera montée. À l'intérieur de ce conteneur, un élément <input> de type "checkbox" est utilisé pour afficher la case à cocher. La directive v-model="isChecked" est appliquée à cet élément pour créer une liaison bidirectionnelle entre l'état de la case à cocher et la propriété isChecked dans l'instance Vue. Lorsque l'utilisateur coche ou décoche la case, la valeur de isChecked sera automatiquement mise à jour. À côté de la case, un élément <label> est utilisé pour afficher un texte dynamique. Grâce à l'interpolation {{ isChecked ? 'Coché' : 'Décoché' }}, le texte affiché sera "Coché" si la case est cochée (c'est-à-dire si isChecked est true), et "Décoché" si la case est décochée (si isChecked est false). L'application Vue est ensuite montée sur l'élément avec l'ID "app", ce qui permet d'actualiser automatiquement l'affichage à chaque changement de l'état de la case à cocher.
  9. Dans l’exercice 9, on crée une application Vue 3 qui affiche une liste de fruits et un message personnalisé pour chaque fruit, en indiquant son nom et sa longueur. Tout d'abord, dans l'objet data() de l'instance Vue, un tableau de fruits est déclaré, chaque élément étant une chaîne représentant un nom de fruit (par exemple, "Pomme", "Banane", "Orange"). Ensuite, une boucle v-for est utilisée pour parcourir ce tableau et afficher chaque fruit dans un élément <li>. À l'intérieur de chaque <li>, on utilise l'interpolation de chaînes pour générer un message personnalisé, qui inclut le nom du fruit et sa longueur. Ce message est affiché sous la forme de {{ Le ${fruit} a une longueur de ${fruit.length} caractères. }}, où ${fruit} représente le nom du fruit et ${fruit.length} donne sa longueur en caractères. Le tableau est parcouru, et pour chaque fruit, le message correspondant est affiché, vérifiant ainsi que l’interpolation fonctionne correctement.
  10. Dans l’exercice 10, on crée une application Vue qui affiche un message de bienvenue en fonction de la langue sélectionnée par l'utilisateur. Tout d'abord, une instance Vue est créée avec un tableau de données, incluant une propriété language qui stocke la langue sélectionnée par l'utilisateur, initialisée par défaut à "anglais". Ensuite, des boutons sont ajoutés pour permettre à l'utilisateur de choisir la langue. Ces boutons, grâce à la directive v-on:click, modifient la valeur de la propriété language lorsqu'un bouton est cliqué. Pour afficher le message approprié, on utilise la directive v-if qui conditionne l'affichage du message de bienvenue en fonction de la langue choisie. Par exemple, si language vaut "français", le message affiché sera "Bienvenue sur mon site !", et si la langue est "anglais", le message sera "Welcome on my website !". De cette manière, le message de bienvenue change en temps réel, selon la sélection de l'utilisateur.
  11. Dans l’exercice 11, on crée une application Vue permettant aux utilisateurs de convertir une température de degrés Celsius à Fahrenheit. Tout d'abord, un champ de saisie est ajouté pour que l'utilisateur puisse entrer une température en degrés Celsius. Cette donnée est liée à la propriété celsius de l'instance Vue à l'aide de la directive v-model, permettant une liaison bidirectionnelle entre le champ de saisie et la donnée. Ensuite, la température en degrés Fahrenheit est calculée en utilisant la formule de conversion Fahrenheit = (Celsius \* 9/5) + 32 et affichée à l'aide de l'interpolation de chaîne dans le DOM. En parallèle, une directive v-if est utilisée pour afficher un message indiquant si l'eau bout à la température donnée. Si la température en degrés Celsius est supérieure ou égale à 100, le message "L'eau bout !" sera affiché. Sinon, il ne sera pas affiché.
  12. Dans l’exercice 12, on crée une application Vue interactive qui permet à l'utilisateur de saisir ses informations personnelles et de réserver une chambre d'hôtel. L'application commence par la création d'une instance Vue, avec un conteneur principal ayant l'ID "app". Dans l'objet data() de l'instance Vue, plusieurs données sont déclarées pour stocker les informations de l'utilisateur : nom, email, dateArrivee, dateDepart, nbNuits (calculé à partir des dates saisies), et tarif (tarif de la chambre par nuit). Un formulaire est créé, contenant des champs de saisie pour le nom, l'adresse e-mail, la date d'arrivée et la date de départ. Des validations sont ajoutées pour garantir que les informations sont correctement saisies, par exemple, vérifier le format de l'adresse e-mail et la validité des dates. La différence entre les dates d'arrivée et de départ est utilisée pour calculer le nombre de nuits (nbNuits), et si les dates sont incorrectes, une gestion des erreurs est effectuée. Le tarif de la chambre est défini dans la variable tarif (par exemple, 100€ par nuit). Ensuite, à l'aide de la directive v-bind et de la string interpolation, les détails de la réservation sont dynamiquement affichés, incluant le nom, l'e-mail, les dates d'arrivée et de départ, le nombre de nuits, ainsi que le coût total de la réservation. Un bouton de soumission permet de valider la réservation et de déclencher une action, par exemple, pour envoyer les informations à un serveur. On finit par styliser le tout grâce à du CSS pour rendre le formulaire plus attrayant.

# IV. Difficultés Rencontrées et Solutions Apportées

## Difficultés :

* + Prendre en main ce Framework et les fonctions associées et propres à VueJs.

## Solutions :

* + Utiliser le cours et la documentation

# V. Commentaires et Suggestions

## Remarques Générales :

* + A l’issue de ce TD, qui m’a pris environ 6 heures, j’ai pu constater que VueJs n’était pas un Framework difficile à prendre en main comme pourrait l’être React.

## Suggestions d’Amélioration :

* + Je pourrais éventuellement suggérer de pousser la découverte et l’utilisation de VueJs un peu plus loin afin d’en apprendre davantage.

# VI. Auto-évaluation

## Évaluation de votre travail :

* + Compréhension et réalisation du TP plus rapides que pour les autres Framework, donc plus efficace, le code est simple à comprendre pour tout le monde, autant grâce aux commentaires que grâce aux noms de variables, d’id et class de balises utilisées au cours de ce tp.

## Objectifs pour les futurs projets :

* + Intégrer ce Framework dans un travail de plus grande envergure

# VII. Code

Soit le lien ou alors, pour chaque fichier

## Lien vers le code source : https://github.com/Tag59/TpJquery