Building the header of the website

1. Start from the started kit

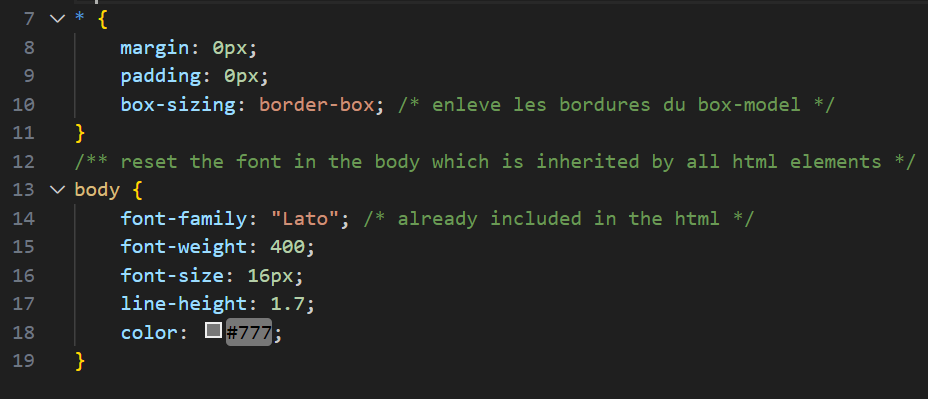
Colors are as follows



1. Start coding
   * Perform basic reset of the unversal selector
   * Set the font for the project
   * Clip part of éléments using clip-path (css)
2. Commencer par écrire le code pour l’en-tête de la page

<header> avec un contenu text au départ.

Et faire le code de reset basique dans le fichier css, comme suit (qui réinitialise les marges, les paddings, le box model ainsi que la police par défaut.



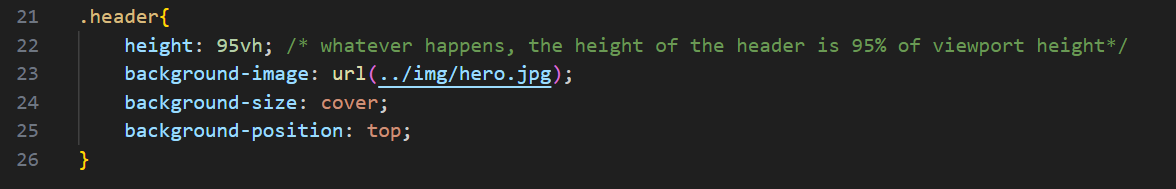
1. Exercice :

Définir la taille du header à 95% de la hauteur du viewport (indice : vh en css )

Définir une image en tant que fond du header.

Définir background-position et size pour que l’image couvre tout et que le top ne change jamais.

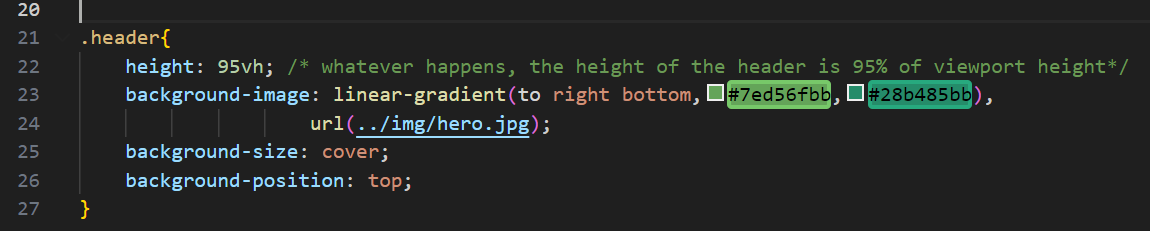
Solution :

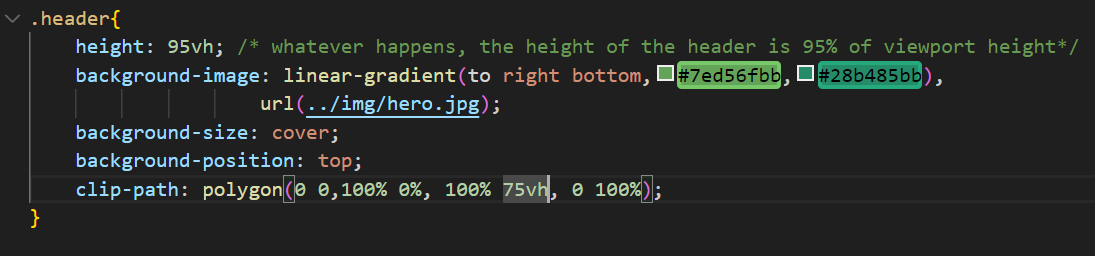


1. Rajout du fond vert :

Pour cela, il faut un deuxième background-image (mais cette fois-ci le définir en tant que gradient).

On ne peut pas définir deux images de background sinon ça écrase le premier mais on peut définir 2 valeurs dans l’instruction « background-image » et les 2 valeurs sont séparés par une virgule. Nous n’allons pas inclure une nouvelle image mais un linear-gradient avec une transparence.



1. Rajouter une bordure de padding de 30px autour de toute la page.
2. Clip-path pour avoir la bonne forme du header.



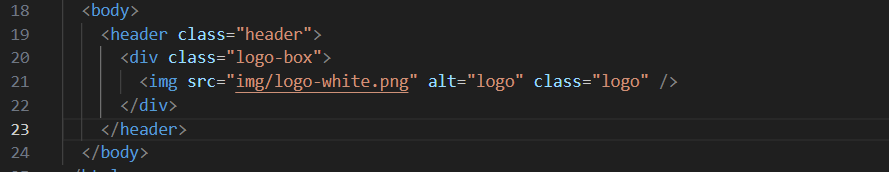
1. Add a logo :

Une image est en général un élement inline (qui s’affiche en général sur une ligne), il est donc de bonne pratique d’utiliser un container (un div par exemple) pour le manipuler.

Il est important de pouvoir manipuler le container « logo-box » et le contenu « logo »

Exercice :

* + - Rajouter une image comme logo : img/logo-white.png

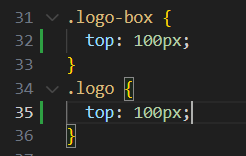


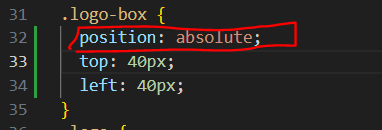
* + - Formatter l’image en définissant les propriétés suivantes
      * Top, bottom, left, right ⬄ au départ donnez des valeurs aléatoires et regardez le rendu. Que remarque-t-on ?

*Que les propriétés « top », « left », « right », « bottom » ne fonctionnent pas pour positionner le’image. Pourtant il s’affiche bien par défaut en haut et à gauche de la page*.



Rajouter les css.

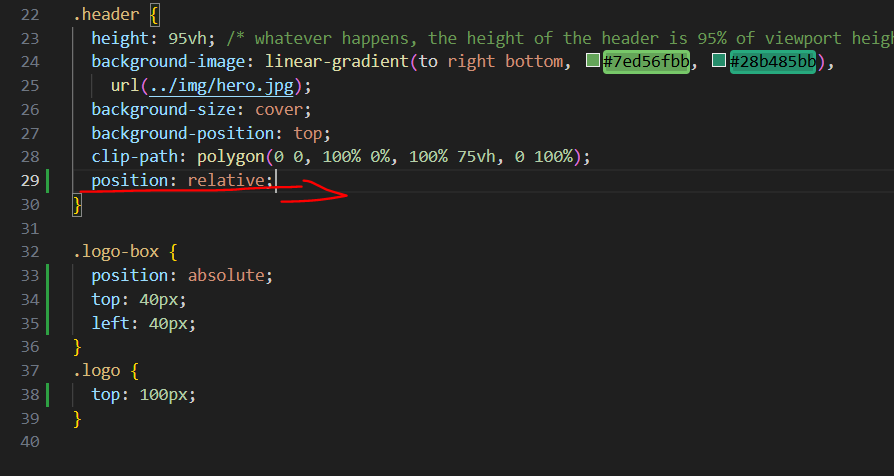
 🡺 aucun effet.  
  
Pour commencer à avoir des effets, il faut commencer par :



On verra un peu plus loin sur la différence entre absolute et relative et le flow normal des éléments dans un HTML.

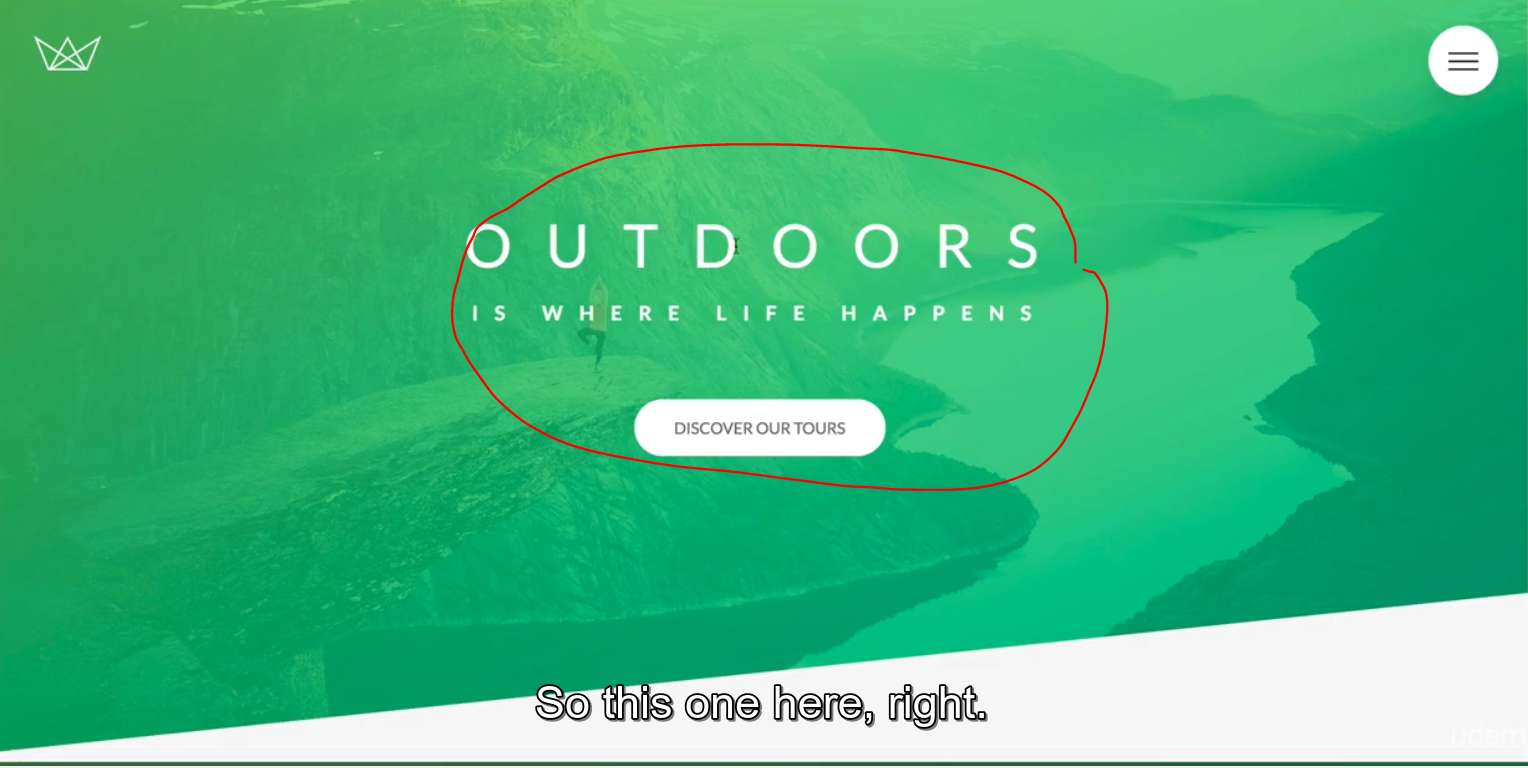
Mais on se pose des questions sur le positionnement. Par rapport à quoi est-ce que ces 40px sont comptés ?

La référence de base à ces positionnements est l’élément parent de la classe logo-box pour qui la position définie à relative.   
Dans notre cas, le header est l’élémént parent qui doit avoir sa position à relative (à rajouter donc comme suit.



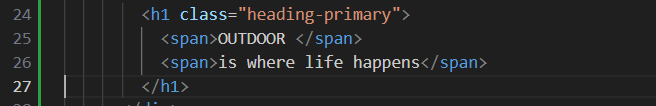
Définir la taille de l’image à une hauteur de 35px ;  


Le texte du milieu du background.



Nous allons le mettre juste dans un <h1> (important pour les moteurs) mais la taille se doit de diviser le texte en 2 lignes (pour le SEO, il nous faut le garder sur 2 lignes).

Pour le diviser en 2, nous allons utiliser <span> .



Pour le formattage, nous allons utiliser 2 classes différentes pour les 2 spans

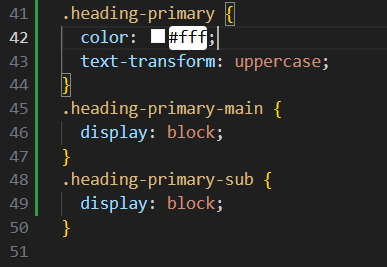
<span class="heading-primary-main"> OUTDOORS </span>

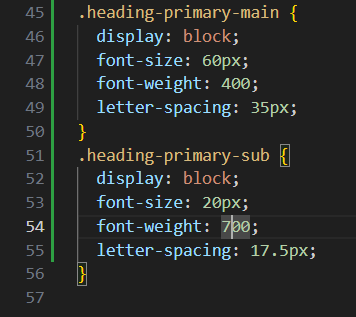
          <span class="heading-primary-sub">is where life happens</span>

Et nous allons le faire en CSS.

* + - La couleur du texte entier doit être blanc
    - Le texte « outdoors » doit être en majuscule : utiliser l’instruction css ‘text-transform’
    - Mettre « OUTDOORS » en dessus et le reste du texte en dessous ⬄ une bonne approche est de les afficher comme des blocks car les éléments « block » occupent toute la largeur disponible

Voici le code qui fait tout ce qu’on vient de dire



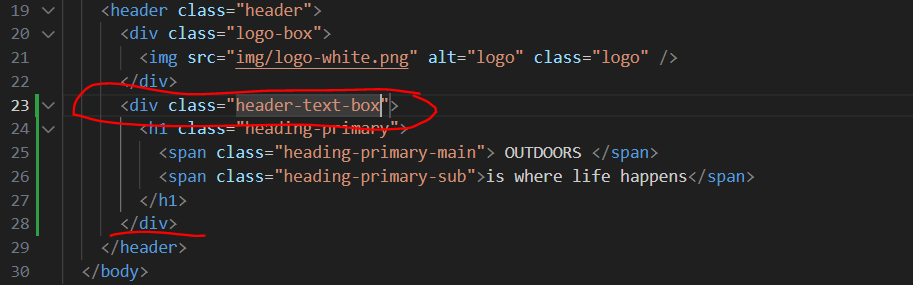
Le reste est facile pour le font et la taille de police. On remarque l’espacement entre les caractères de OUTDOOR (letter-spacing :35px en CSS)   
  
Ajuster la taille avec l’inspecteur d’objet pour avoir la bonne taille.  
  
cela me donne ceci :  
  


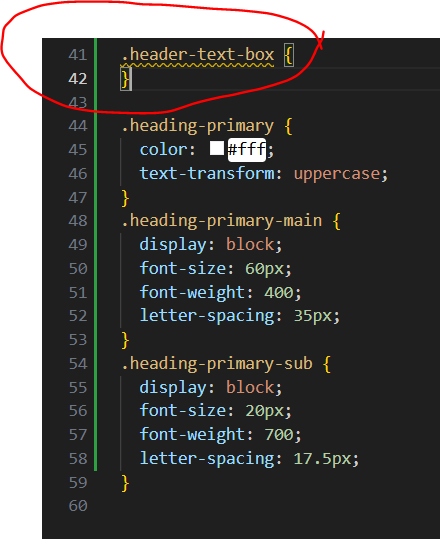
Pour ça :



Prochaine étape : positionner correctement le texte au centre

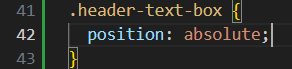
Comment faire ?

l’idée serait de nommer le div comme container et nous allons centrer tout le div  




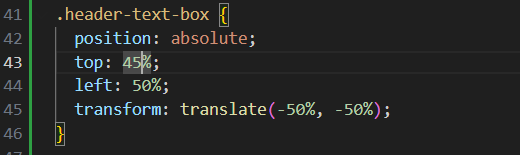
L’ordre est important car cela suit l’ordre naturelle du HTML.

Je veux pouvoir positionner librement le text au milieu sans affecter les autres donc position : absolute ;



Pour commencer, nous allons mettre top :50% et left :50 pour centrer. Ce ne sera pas idéal mais ce sera un début.

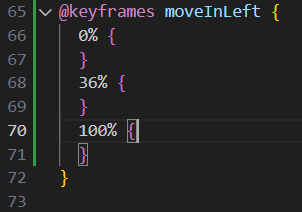
Cela positionne la pointe gauche de la boite à 50%, 50%.  
Il faut encore translater la boite de -50% -50%



Ajuster ensuite si trop bas ou trop haut.

**CREATION D’ANIMATION EN CSS.**Les textes du header entrer en fade-in et se déplacent de gauche à droite (ou vice versa).

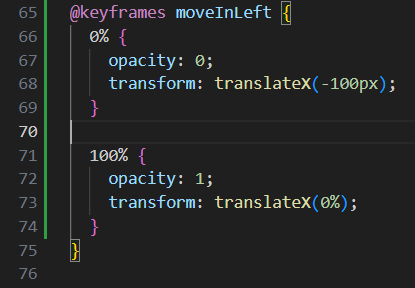
Pour créer une animation, on va créer un keyframe dans le css



On peut définir les éléments de l’animation à des pourcetages que nous définissions.

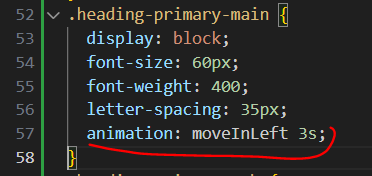
Pour faire le fade-in, on peut définir au départ ‘opacity : 0’ (à 0%) et à 100%, on peut définir ‘opacity : 1 ‘.

Comme ‘OUTDOORS’ commence son départ sur la gauche, on voudrai qu’il soit sur la gauche alors. On va commencer à 0% à translater l’objet à -100px de l’axe des x (attention aux axes infographie).

Attention, pour l’instant, on n’a pas dit que ce keyframe est pour OUTDOORS, on ne fait que le définir  
  


Voici le keyframe qu’on a définit.  
Maintenant, il faudra définir à quel élément, il va s’appliquer ?

Pour cela, on appelle le keyframe dans l’élement (ex : pour notre cas on veut le faire pour ‘OUTBOORS’ donc pour la classe .heading-primary-main .  
  
Pour l’appel, on doit spéciifer dans l’objet à animer, le nom de l’animation et sa durée comme suit :



On peut aussi les définir un à un.

Pour avoir un petit mouvement de dépassement de la position d’origine et ensuite revenir à la position, normale, nous allons définir dans l’animation que la position est +20 de X à 80% avant de revenir à 0 (position originale).  
  
Ajuster le temps de l’animation à 1s et ajuster le décalage vers la droite et voir ce que ça donne.

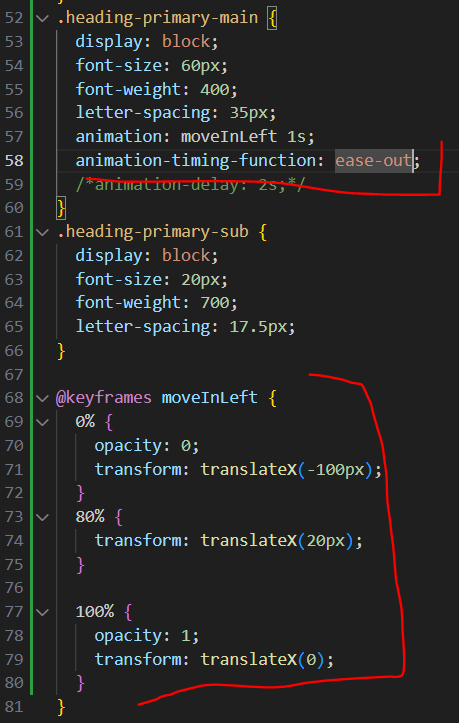
Pensez aussi à « animation-delay : xs » pour attendre un moment avant que l’animation ne se déclenche ⬄ définir par exemple que l’animation ne se déclenche qu’après 3s.

« animation-iteration-count : x» permet de rejouer l’animation x fois.

Voir la doc de toutes les variantes des fonctions ici :  
<https://developer.mozilla.org/en-US/docs/Web/CSS/animation-timing-function>

Pour notre cas nous allons utiliser « ease-out » (ça commence lentement et accélère)

Voici le code résultant :



A partir de ces exemples : faire le mouvement du texte en dessous de la droite vers la gauche.

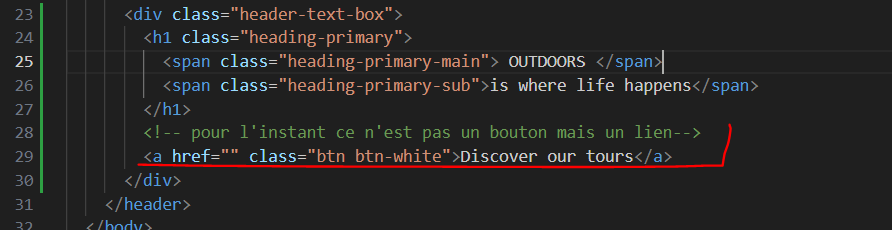
je ne sais pas pourquoi (apparemment, un effet indésirable de translatex mais il faut juste mettre « backface-visibilit :hidden dans heading-primary tout entier)

Apparemment ça permet d’enlver un tremblement de l’animation sur certains navigateur.

**Animation du bouton.**

**Description de l’animation :**  le bouton grossit légèrement pour ensuite trembloter et reprendre sa taille réelle.

* + - Créer d’abord le bouton dans le HTML

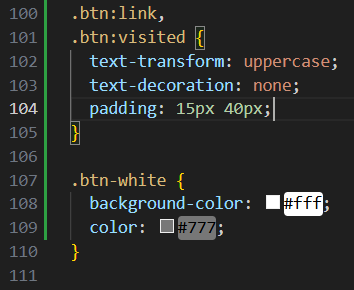


* + - Rappel des pseudo éléments et des pseudo classes en CSS ( ::after par exemple)

Pour formatter la classe btn, on va utiliser un pseudo classe (exemple nom:link ou nom :visited pour un lien déjà visité).   
  
Nous allons déjà commencer par :

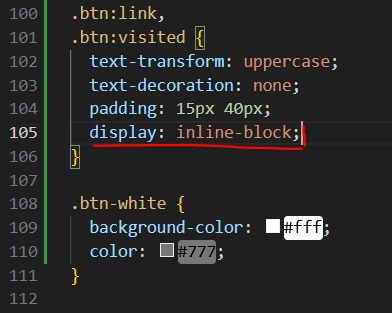
* + - * Mettre le lien en majuscule en utilisant text-transform
      * On ne veut pas que le lien contienne une ligne (utiliser text-decoration)

Tout cela, nous permet déjà d’avoir ce code



Mais cela pose un problème d’affichage car le lien va occuper une partie de la ligne d’en dessus.  


Le texte est naturellement inline et ne peut être redimensionné. Pour qu’on puisse contrôler sa taille ou sa bordure et padding, on le met à inline-block.



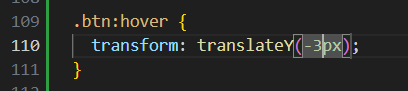
Cette astuce fonctionne pour tout élément qu’on veut pouvoir redimensionner.

Ensuite, on attend que le block soit au centre.   
pour cela, le plus facile est de mettre tout le block .header-text-box

à un alignement au centre.

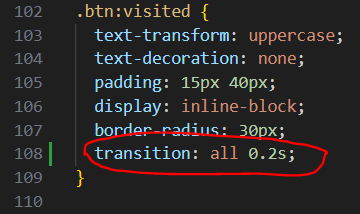
On otbient alors ce résultat :  
  


* + - Animation sur l’event « hover » du bouton en utilisant la propriété « transition »
      * Quand la souris passe (hover), la boite se déplace léègrement vers le haut : (transform : translateY)



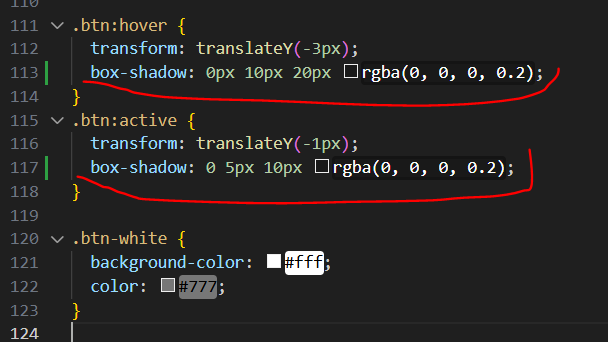
* + - * Sur l’event click ( :active), lors du click le bouton se déplace légèrement vers le haut de 1px
      * Et enfin le formatage du bouton pour avoir des bordures arrondies (border-radius)
      * Animation du bouton :

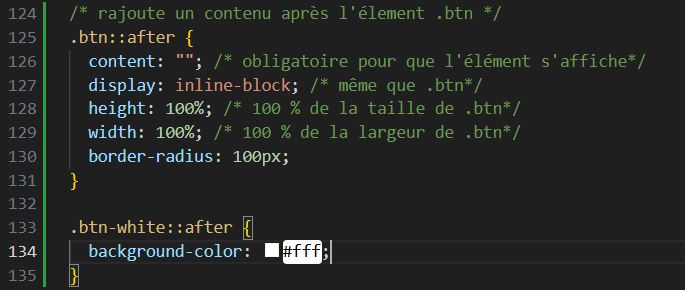
**Rappel :** on peut animer une propriété avec transition ou utiliser un keyframe.



Rien que ça va faire en sorte que les transformations que nous avons défini pour cet élement « .btn » soient animés.  
  
Transition est toujours dans l’état initial (ici c’est .btn :visited)

* + - * Un autre aspect est l’animation que nous avons lorsqu’on passe la souris sur le bouton (elle explose légèrement). Propriété « box-shadow ».

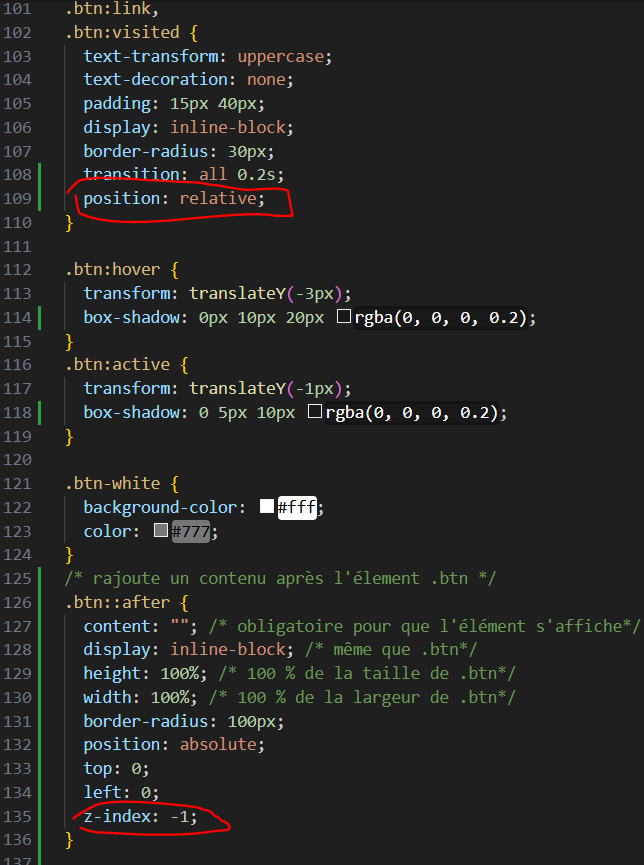


* + - * Nous allons utiliser le pseudo elemnet « ::after » . Il doit avoir un contenu « content »  
        

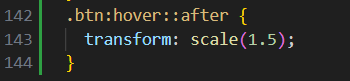
Et ça donne ceci :

Maintenant, il faut que l’élément after soir derrière notre élément (sinon il s’affichera tout le temps.   
Pour cela, il faut une position absolute ⬄ dans cette config, la positon est par rapport au plus proche ancêtre dans le html, mais dont la position est définie à relative ⬄ ici ce serait alors «header » (qui est le plus proche ancêtre avec position :relative, mais ce n’est pas ce que l’on souhaite, on veut qu’il se cache sous .btn ⬄ pour cela, on va fixer .btn à une position relative pour respecter cette condition).

Enfin, définir le z-index de l’élément ::after à -1 pour le mettre derrière.

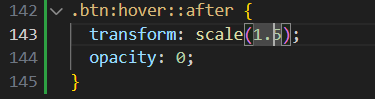


* + - La question est maintenant, quand est-ce que nous voulons voir cet élément cachée par after ? La réponse est quand nous passons la souris sur la souris …. ::hover

On veur rajouter un élément à .btn::hover, cela revient à faire .btn ::hover ::after.  
  
quand on passe la souris, on veut que l’élement .btn-white::after grandissent et ensuite s’estompe.  
  


Si on veut ensuite animer cette transformation, il faut rajouter « transition : all ; dans l’état initial qui est .btn ::after).

Si on veut à la fois que l’objet disparaisse progressivement, l’astuce est d’animer son opacity à 0.

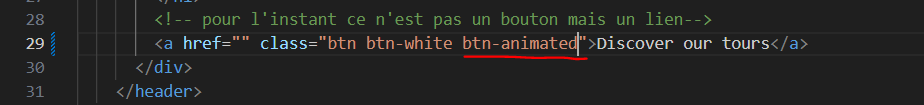


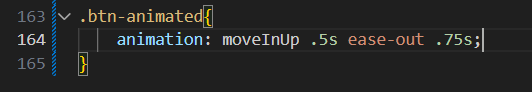


Agrandissement et disparition

On voit dans cet agrandissement que l’agrandissement vers le haut est petit.  
On peut changer la fonction de transformation (scale en scaleX et scaleY)

* + - Il nous reste à animer l’apparition du bouton. Au momenet du chargement de la page, nous allons animer le bouton pour qu’il se déplace du bas vers le haut. Rajouter aussi un petit délai pour que l’animation ne commence qu’après un petit moment. De préférence, utiliser une nouvelle classe comme .btn-animated pour éviter des effets de bords.  
        
      Pour cela, il nous faut un nouveau keyframe.





Mais le souci est que le bouton s’affiche avant d’être animé.  
  
Comment résoudre, utiliser ‘animation-fill-mode’ (Defines what values are applied by the animation outside the time it is executing.) Si **backwards** est la valeur alors « The beginning property value (as defined in the first @keyframes at-rule) is applied before the animation is displayed, during the period defined by 'animation-delay'. »

moveInUp est un keyframe.