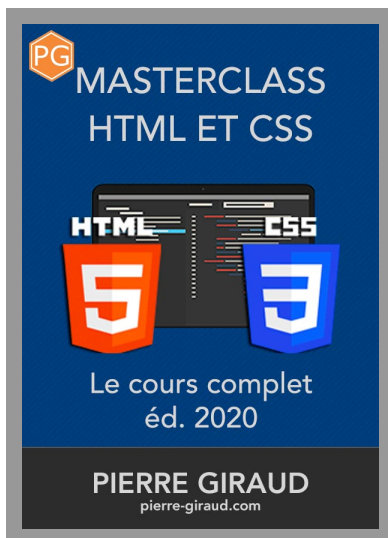




≡ Menu

Gérer le positionnement avec la propriété CSS position

Télécharger
le PDF du
cours



Cours
complet
HTML et
CSS

INTRODUCTION AU
COURS HTML ET
CSS

1. Introduction au
cours HTML et CSS

2. Définition et usage
du HTML et du CSS

La propriété **position** est une propriété CSS très puissante qui va nous permettre de définir un type de positionnement pour nos éléments.

On va ainsi pouvoir positionner un élément relativement à partir de sa position par défaut ou de façon absolue par rapport à un point donné dans la page en utilisation **position** conjointement avec les propriétés **top**, **left**, **bottom** et **right**.

Dans cette leçon, nous allons découvrir les différentes valeurs qu'on va pouvoir donner à **position** et apprendre à les utiliser intelligemment en tentant de comprendre leurs implications.

Le fonctionnement et les valeurs de la propriété position

Nous allons pouvoir gérer et modifier le type de positionnement d'une élément HTML grâce à la propriété CSS **position**.

La propriété **position** ne va pas nous permettre de positionner un élément en soi dans une page mais simplement de définir un type de positionnement grâce aux valeurs suivantes :

- **position : static ;**

Privacy & Cookies Policy

3. Evolution de l'informatique et des langages Web

4. Travail en local et en production

5. Choix et installation d'un éditeur de texte

LES BASES DU HTML

6. Eléments, balises et attributs HTML

7. Structure minimale d'une page HTML valide

8. Enregistrement et affichage d'une page HTML

9. L'indentation et les commentaires en HTML

10. Les titres et les paragraphes en HTML

11. Les espaces et retours à la ligne en HTML

12. Définir le niveau d'importance des contenus en HTML

13. Créer des listes en HTML

14. Créer des liens en HTML

15. Envoi de mails et téléchargement de fichiers en HTML

- **position : relative ;**
- **position : absolute ;**
- **position : fixed ;**
- **position : sticky.**

Une fois le type de positionnement défini avec **position**, nous allons pouvoir effectivement positionner un élément à un endroit précis dans une page grâce aux propriétés **top**, **left**, **bottom** et **right**.

Ces quatre propriétés vont pouvoir prendre des valeurs absolue ou relative et vont servir à indiquer où le coin supérieur gauche de la boîte représentant un élément doit être positionné par rapport à un certain point de référence (50px à droite et 30px en dessous de ce point par exemple).

Le type de positionnement défini pour l'élément va servir à définir ce point de référence et va donc affecter le fonctionnement de ces propriétés qui vont produire des résultats différents.

Les types de positionnement d'un élément HTML dans une page

Il existe trois types de positionnement en CSS. Il est très intéressant de les connaître et de les comprendre afin de mieux comprendre comment fonctionne le CSS et comment les différents éléments vont venir se positionner les uns par rapport aux autres.

Connaître et comprendre les types de positionnement va également nous permettre de comprendre comment fonctionne la propriété CSS **position** puisque selon la valeur donnée à **position**, un élément HTML va se conformer à un type de positionnement plutôt qu'un autre.

16. Compatibilité, support et validation du code HTML et CSS

LES BASES DU CSS

17. Sélecteurs et propriétés CSS

18. Où écrire le code CSS ?

19. Commentaires et indentation en CSS

20. Sélecteurs CSS simples et combinateurs

21. Les attributs HTML class et id et les sélecteurs CSS associés

22. Ordre d'application (cascade) et héritage des règles en CSS

23. Les éléments HTML div et span (conteneurs génériques)

24. Les niveaux ou "types" d'éléments HTML block et inline

25. Notations complètes "long hand" et raccourcies "short hand" CSS

MISE EN FORME DE TEXTES EN CSS

26. La propriété CSS font-family et les Google Fonts

1. Le premier type de positionnement est ce qu'on pourrait appeler le positionnement « normal » ou par défaut des éléments. Ici, les éléments vont respecter le flux normal de la page et s'y intégrer sans le casser. Ainsi, un élément de type **block** sera formaté comme tel (c'est-à-dire qu'il occupera tout l'espace possible et se placera à la ligne), un élément de type **inline** n'occupera que l'espace nécessaire et etc. ;
2. Ensuite, on va également pouvoir faire flotter des éléments HTML avec la propriété **float**. Ce type de positionnement est particulier puisque l'élément flotté va être retiré du flux normal de la page pour être repositionné ailleurs (généralement à gauche ou à droite) et va également permettre à d'autres éléments de type **inline** de se positionner à côté de notre élément flotté ;
3. Finalement, on va pouvoir positionner un élément de manière absolue dans notre page. Avec le type de positionnement absolu, un élément est complètement retiré du flux normal de la page pour être placé absolument par rapport à son élément parent direct et va ainsi pouvoir potentiellement passer au-dessus d'autres contenus.

Position : static

La valeur **static** est la valeur par défaut de la propriété **position**. Ainsi, par défaut, tous les éléments HTML sont positionnés de manière **static**. Un élément HTML positionné avec **position : static** sera positionné selon le flux normal de la page.

Notez ici que les propriétés **top**, **left**, **bottom** et **right** n'auront aucun effet sur les éléments positionnés avec **position : static**.

27. Les autres propriétés CSS liées à la police
28. Les propriétés CSS liées au texte
29. Gérer la taille des interlignes et des espaces dans les textes en CSS
30. Gérer la couleur et l'opacité des textes
LE MODELE DES BOITES
31. Le modèle des boîtes
32. Largeur (width) et hauteur (height) de la boîte de contenu des éléments HTML
33. Gestion des marges internes ou padding en CSS
34. Gestion des bordures en CSS
35. Gestion des marges externes (margin) en CSS
36. La propriété CSS box-sizing
37. Créer des bordures arrondies en CSS
POSITION ET AFFICHAGE DES ELEMENTS
38. La propriété CSS display

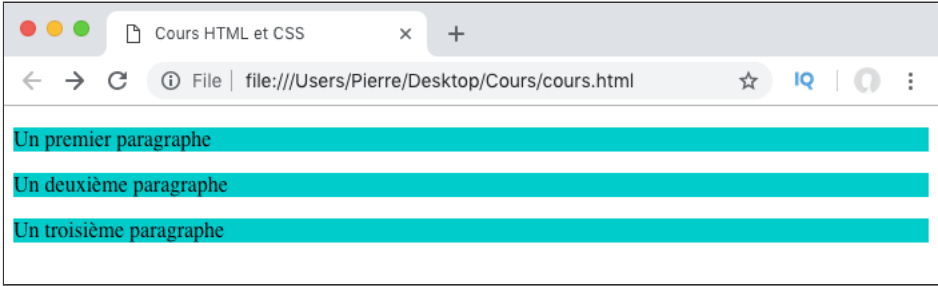
```
<!DOCTYPE html>
<html>
  <head>
    <title>Cours HTML et CSS</title>
    <meta charset= "utf-8">
    <link rel="stylesheet" href="cours.css">
  </head>

  <body>
    <p>Un premier paragraphe</p>
    <p class="statique">Un deuxième paragraphe</p>
    <p class="statique gauche">Un troisième paragraphe</p>
  </body>
</html>
```

```
p{
  background-color: #0CC;
}

.static{
  position: static; /*Valeur par défaut*/
}

.gauche{
  left: 100px; /*La propriété ne s'applique pas sur un élément static*/
}
```



HTML CSS Result

```
p{
  background-color: #0CC;
}

.static{
  position:
  static; /*Valeur par
  défaut*/
}

.gauche{
  left: 100px; /*La
  propriété ne s'applique
```

⚠ Do not enter passwords or personal information on this page.

⚠ This is a code demo posted by a web developer on [codepen.io](#). A referer from CodePen is required to render this page view, and your browser is not sending one ([more details](#)).

Resources 1x 0.5x 0.25x Rerun

Position : relative

39. La propriété CSS position

40. La propriété CSS float

41. Gestion des conflits entre display, position et float

CREATION DE TABLEAUX HTML

42. Créer des tableaux en HTML

43. Structurer un tableau HTML

44. Mettre en forme un tableau HTML avec du CSS

INSERTION DE MEDIAS EN HTML

45. Insérer des images dans des pages HTML

46. Insérer de la musique avec l'élément HTML audio

47. Insérer des vidéos avec l'élément HTML video

48. L'élément HTML iframe

FOND, DEGRADES ET OMBRES CSS

49. Gérer la couleur de fond des éléments HTML

50. Ajouter des images en fond des

Attribuer une **position : relative** à un élément va positionner l'élément dans le flux normal de la page tout comme la valeur **static** de la propriété **position : static**.

Cependant, à l'inverse d'un élément HTML positionné avec **position : static**, un élément positionné avec **position : relative** va ensuite pouvoir être décalé par rapport à sa position initiale grâce aux propriétés **top**, **left**, **bottom** et **right**.

Ces propriétés vont prendre comme origine la position initiale de l'élément. Nous allons ainsi pouvoir positionner un élément relativement à sa position de départ.

Notez qu'ici l'espace occupé initialement par l'élément va continuer à lui appartenir : les autres éléments ne seront pas affectés par le décalage de notre élément et ne vont pas se repositionner en fonction de celui-ci.

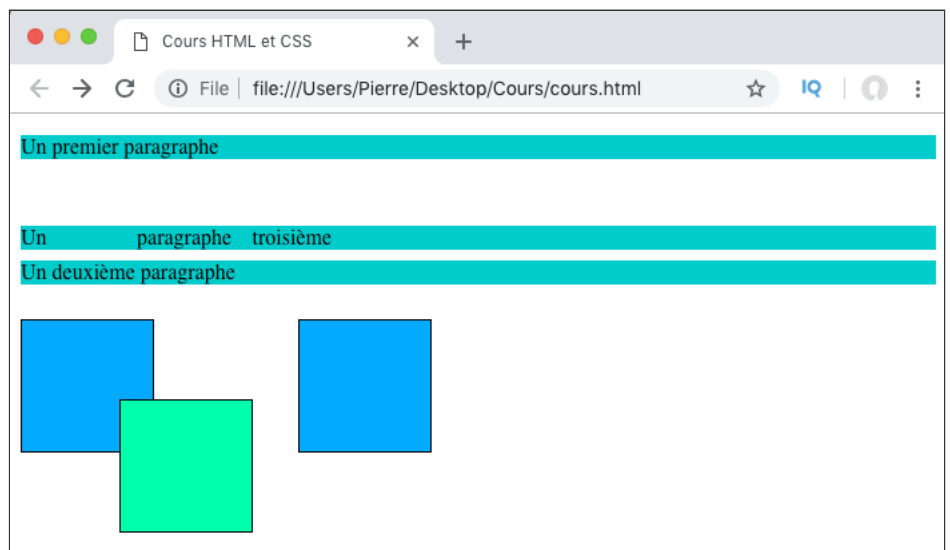
Cela implique également que l'élément décalé va pouvoir être à cheval par-dessus d'autres éléments puisque la position de ces autres éléments ne va pas changer en fonction de l'élément décalé possédant une **position : relative**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Cours HTML et CSS</title>
    <meta charset="utf-8">
    <link rel="stylesheet" href="cours.css">
  </head>

  <body>
    <p>Un premier paragraphe</p>
    <p class="relatif haut">Un deuxième paragraphe</p>
    <p>Un <span class="relatif gauche">troisième</span> paragraphe</p>
    <br><br>
    <div class="carre bleu"></div>
    <div class="carre vert relatif haut droite"></div>
    <div class="carre bleu"></div>
  </body>
</html>
```

éléments HTML	
51. Création de dégradés linéaires en CSS	
52. Création de dégradés radiaux en CSS	
53. Ajouter des ombres aux éléments en CSS	
SELECTEURS COMPLEXES	CSS
54. Les sélecteurs CSS d'attributs	
55. Les pseudo-classes CSS	
56. Les pseudo-éléments CSS	
57. EXERCICE – Création d'un menu horizontal sticky en HTML et CSS	
58. EXERCICE – Création d'un menu déroulant en HTML et CSS	
FORMULAIRES HTML	
59. Présentation des formulaires HTML	
60. Les éléments des formulaires	
61. Attributs des formulaires et sécurité	

```
p{
  background-color: #0CC;
}
div{
  width: 100px;
  height: 100px;
  display: inline-block;
  border: 1px solid black;
  box-sizing: border-box;
}
.relatif{
  position: relative;
}
.haut{
  top: 60px; /*Élément décalé de 50px vers le bas p/r à une origine*/
}
.gauche{
  left: 150px; /*Element décalé de 150px vers la droite p/r à une origine*/
}
.droite{
  right: 30px; /*Element décalé de 30px vers la gauchr p/r à une origine*/
}
.bleu{
  background-color: #0AF;
}
.vert{
  background-color: #0FA;
}
```



TRANSITIONS, ANIMATIONS ET TRANSFORMATIONS CSS

62. Créer des transitions en CSS

63. Créer des animations en CSS

64. Créer des transformations en CSS

65. EXERCICE – Créer un diaporama en HTML et CSS

MODELE DES BOITES FLEXIBLES - FLEXBOX CSS

66. Introduction au modèle des boites flexibles ou flexbox

67. Gérer la direction des éléments flexibles ou flex items

68. Gérer l'alignement des éléments flexibles

69. Gérer la taille et la flexibilité des éléments flexibles

70. Cas d'utilisation et limites du flexbox

71. EXERCICE – Création d'un menu HTML et CSS en utilisant le flexbox

RESPONSIVE DESIGN CSS

HTML

CSS

Result

```
<p>Un premier paragraphe</p>
<p class="relatif haut">Un deuxième paragraphe</p>
<p>Un <span class="relatif gauche">troisième</span>
paragraphe</p>
<br><br>
<div class="carre bleu"></div>
<div class="carre vert relatif haut droite"></div>
<div class="carre bleu"></div>
```

Resources

Position : absolute

Un élément positionné avec **position: absolute** va être positionné par rapport à son parent le plus proche positionné (avec une valeur de position différente de **static**).

Si aucun parent positionné n'est trouvé, alors l'élément sera positionné par rapport à l'élément racine représentant la page en soi.

72. Introduction au responsive design
73. La balise meta viewport
74. Créer un design responsive avec les media queries
75. Images responsive
76. EXERCICE - Création d'un menu déroulant responsive HTML et CSS
SEMANTIQUE ET ELEMENTS HTML STRUCTURANTS
77. Sémantique et éléments structurants du HTML
78. EXERCICE - Création d'un site de CV responsive en HTML et CSS
MODELE DES GRILLES CSS
79. Introduction au modèle des grilles CSS
80. Créer une grille et définir des pistes de grille
81. Positionner des éléments dans une grille
82. Aligner et espacer des éléments dans une grille

Le point de référence pour les propriétés **top**, **left**, **bottom** et **right** va ainsi être le côté de l'élément parent liée à la propriété (côté gauche pour **left**, supérieur pour **top**, etc.).

De plus, un élément positionné avec **position : absolute** va être retiré du flux normal de la page. Cela signifie et implique que l'espace initialement attribué à un élément au positionnement absolu (espace attribué selon le flux normal de la page) va être occupé par les éléments suivants.

Un élément positionné avec **position: absolute** va ainsi pouvoir se placer par-dessus d'autres éléments.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Cours HTML et CSS</title>
    <meta charset= "utf-8">
    <link rel="stylesheet" href="cours.css">
  </head>

  <body>
    <p>Un premier paragraphe</p>
    <p class="absolu haut">Un deuxième paragraphe</p>
    <p>Un troisième paragraphe</p>
    <br><br>
    <div class="conteneur relative">
      <div class="carre bleu"></div>
      <div class="carre vert absolu bas droite"></div>
      <div class="carre bleu"></div>
    </div>
  </body>
</html>
```


83. EXERCICE –
Création d'une page à
3 colonnes contenant
des éléments flexibles

EVOLUTION ET
FUTUR DU CSS

84. Le CSS, un quasi
langage de
programmation

85. Les fonctions CSS

86. Les propriétés
personnalisées ou
variables CSS

87. Les règles arobase
CSS

88. Imbrication et
héritage étendu

CONCLUSION DU
COURS

89. Conclusion du
cours HTML et CSS

HTMLCSSResult

```
p{
  background-color: #0CC;
}
div{
  width: 100px;
  height: 100px;
  display: inline-block;
  border: 1px solid
black;
  box-sizing: border-box;
}
.conteneur{
  width: 100%;
  height: 200px;
  display: block;;
  background-color:
#DD6; /*Jaune*/
}

.absolu{position:
absolute;}
.relative{position:
relative}

.haut{top: 70px;}
.bas{bottom: 20px;}
.droite{right: 30px;}
```

⚠ Do not enter
passwords or personal
information on this page.

⚠

This is a code demo posted by a
web developer on [codepen.io](#).
A referer from CodePen is
required to render this page
view, and your browser is not
sending one ([more details](#)).

Un premier paragraphe

Un troisième paragraphe

Un deuxième paragraphe

Resources

1x0.5x0.25x

Rerun

Ici, nous avons un paragraphe et un `div` positionnés de manière absolue. Notre paragraphe ne possède pas de parent positionné. Il va donc être positionné par rapport à l'élément racine de la page, c'est-à-dire par rapport à la page en soi. Ici `top: 70px` signifie donc que

notre paragraphe sera placé à 70px du point le plus haut de la page auquel il aurait pu être placé (en tenant compte des marges appliquées).

Notre `div` possède lui un parent positionné qui est le `div` jaune « conteneur ». Ce dernier est positionné de manière relative. Notez qu'on lui a déclaré une `position: relative` mais qu'on n'a pas modifié sa position avec une propriété `top`, `left`, etc. Cela n'empêche pas au `div` conteneur d'être positionné.

Notre `div` vert va donc être positionné par rapport au `div` jaune. Ici, `bottom: 20px` et `right: 30px` signifie que notre `div` vert sera positionné à 20px du bord inférieur et à 30px du bord droit de son parent.

Vous pouvez également observer que les éléments positionnés de manière absolue sont bien retirés du flux normal de la page et les autres éléments vont pouvoir venir se positionner à la place de ces éléments, comme s'ils n'existaient pas. On voit bien cela avec notre deuxième `div` bleu qui vient se coller au premier et vient donc prendre la place du `div` vert ainsi qu'avec notre troisième paragraphe qui prend la place initiale du deuxième.

Position : fixed

Le positionnement fixe est très proche du positionnement absolu. Un élément positionné avec `position: fixed` va également être retiré du flux de la page et l'espace qui lui était attribué selon le flux normal de la page va également pouvoir être utilisé par d'autres éléments.

La seule différence entre `position: fixed` et `position: absolute` est que l'élément ne va plus être positionné par rapport à son parent le plus proche mais par rapport au viewport, c'est-à-dire à la

fenêtre visible à moins que l'un de ses parents possède une propriété `transform`, `filter` ou `perspective` dont la valeur est différente de `none`.

En dehors de ces cas particuliers, un élément positionné avec `position: fixed` apparaîtra toujours à la même place même dans la fenêtre si on descend ou on monte dans la page : il sera fixe par rapport à la fenêtre. En effet, sa position va être calculée par rapport à la fenêtre visible.

A noter ici une exception pour les contenus paginés : dans ce cas-là, l'élément possédant une `position: fixed` sera répété dans chaque page.

HTML

CSS

Result

```
body,div,p{
  margin: 0px;
}
p{
  text-align: justify;
}
.carre{
  width: 75px;
  height: 75px;
  border: 1px solid
black;
}
.bleu{
  background-color: #0AF;
}
.vert{
  background-color: #0FA;
}
```

⚠ Do not enter passwords or personal information on this page.

⚠

This is a code demo posted by a web developer on [codepen.io](#). A referer from CodePen is required to render this page view, and your browser is not sending one ([more details](#)).

m dolor sit amet, ea
riri, ut fugit erroribus
ipta eripuit ius, eos
; noster nostrud perfe
Alia suscipit concludaturque duo
sumo volutpat dissentiunt. pri in iu

Resources1x0.5x0.25xRerun

Position : sticky

La dernière valeur de la propriété CSS `position` est la valeur `sticky`. Un élément positionné avec `position: sticky` sera d'abord positionné selon le flux normal de la page puis va pouvoir être décalé de manière similaire à un élément positionné de n

suivants ne verront pas leur position changée : ils seront toujours placé « comme si » l'élément positionné avec `position: sticky` occupait sa place d'origine.

La différence ici entre un élément positionné avec `position: sticky` et `position: relative` est que la position d'un élément sticky va être calculée par rapport à son parent possédant un mécanisme de défilement (scrolling) le plus proche.

Ainsi, un élément positionné avec `position: sticky` va avoir une position relative au départ puis son positionnement va devenir fixe dès qu'un certain point sera franchi, c'est-à-dire à partir d'un certain niveau de défilement de la page. Les propriétés `top`, `left`, `bottom` et `right` vont nous permettre de pouvoir préciser à partir de quel moment l'élément positionné avec `position: sticky` va devoir être fixe.

Notez que la valeur `sticky` est une valeur assez récente de la propriété `position` et est à ce titre toujours en développement. On évitera donc de l'utiliser pour le moment sur un site « live ».

Un mot sur l'accessibilité du contenu

Lorsqu'on code, il faut toujours s'efforcer de réfléchir en termes d'accessibilité à tous, et notamment pour les personnes souffrant de déficiences comme des déficiences visuelles.

En effet, un des principes de base du web est d'être accessible à tous ou du moins c'est l'une des valeurs fondamentales vers laquelle tendre.

Ici, il faudra donc faire bien attention à ce que les contenus positionnés

contenus de façon non désirée lorsqu'un utilisateur par exemple zoome sur la page pour augmenter la taille du texte.

Définir l'ordre d'affichage des éléments en cas de chevauchement avec la propriété z-index

Les éléments HTML vont pouvoir être positionné dans une page en CSS selon 3 dimensions : selon la largeur (axe horizontal ou axe des X en math), la hauteur (axe vertical ou axe des Y) et également selon une épaisseur ou un ordre d'empilement (axe des Z).

En effet, vous avez pu remarquer dans les exemples précédents que lorsque deux éléments se chevauchaient, il y en avait toujours un au-dessus de l'autre : il y a donc une notion d'ordre d'empilement selon cet axe 3D qui est l'axe des Z.

La propriété **z-index** va nous permettre de choisir quel élément doit apparaitre au-dessus de quel ordre en donnant un index sous forme de nombre à un ou plusieurs éléments. Ainsi, lorsque deux éléments se chevauchent, celui possédant la plus grande valeur pour son **z-index** apparaitra au-dessus de l'autre.

Notez que la propriété CSS **z-index** ne va fonctionner (et n'a de sens) qu'avec des éléments HTML positionnés, c'est-à-dire qu'avec des éléments possédant une propriété **position** dont la valeur est différente de **static** en CSS.

[Précédent](#)

[Suivant](#)

5 réflexions au sujet de “Gérer le positionnement avec la propriété CSS position”

Youssef75

13 septembre 2019 à 11 h 34 min

Bonjour Pierre,

Je me permets de signaler que les captures d'écran des différents codes sont décalés.

En effet, les captures d'écran correspondant à la valeur static sont dans la partie liée à la valeur relative et ainsi de suite.

Je sais pas si cela est intentionnel, mais je trouve que ça peut prêter à

confusion.
Cordialement.

PS: vos cours sont magnifiques.

Connectez-vous pour répondre

Pierre GIRAUD

13 septembre 2019 à 21 h 05 min

Bonjour,

Oui en effet je me suis complètement embrouillé en nommant les captures apparemment. Merci beaucoup c'est corrigé !

Connectez-vous pour répondre

povejan493@imail5.net

29 décembre 2019 à 14 h 11 min

Bonjour Pierre,

En effet, vos cours sont magnifiques.

Je ne comprend pas pourquoi, avec votre niveau d'expertise dont vos cours sont la preuve, vous vous rendez la tâche plus difficile en utilisant des images pour montrer le code alors qu'il est si facile d'utiliser des balises :

Votre exemple

Avec : Highlight.js, PrismJs ou Rainbow, de plus WordPress à ce qu'il faut pour :

<https://alex.blog/wordpress-plugins/syntaxhighlighter/>

Connectez-vous pour répondre

povejan493@imail5.net

29 décembre 2019 à 14 h 17 min

je voulais écrire les balises « pre » « code »...

Connectez-vous pour répondre

Pierre GIRAUD

9 janvier 2020 à 6 h 29 min

Bonjour,

J'utilise déjà un mix de tout cela sur le site en fonction des cours. Parfois, un format de présentation me semble plus adapté qu'un autre (les captures d'écran par exemple permettent au débutant de bien comprendre qu'on travaille sur tel ou tel fichier et les « oblige » à recopier le code à la main plutôt que d'être tenté de le copier coller.

Cordialement,

Connectez-vous pour répondre

Laisser un commentaire

Vous devez être connecté pour publier un commentaire.

