**AFCI Formation**

**Guide pour la scénarisation pédagogique des modules digitaux**

1. **Mode d’emploi**

**Objet du présent fichier**

Concevoir un module digital, c'est beaucoup écrire ! Les formations sont d'abord écrites sur des documents extérieurs aux outils de digitalisation. Les jeux de relecture sont facilités, vous gardez la maîtrise sur le contenu que vous proposez et la médiatisation peut se faire sur une base sûre.

Cette trame se compose de plusieurs parties (présentation + X sections) formant la base d'un module digitalisé standard.

**Mode d'emploi**

1. Déterminez le nombre de sections dans votre module

2. Si vous avez + de 3 sections, copiez-collez autant de descriptif de section qu’il vous manque.

3. Adaptez la partie 3 de l'onglet présentation de votre formation pour qu'il puisse accueillir le nombre de sections voulues

4. Commencez à remplir l'onglet présentation :

* Auteur (a savoir…vous !)
* Domaine (Par exemple : Sanitaire et social, numérique,...)
* Certification concernées (les diplômes qui peuvent être concernés par ce module)
* Compétences ciblées
* Référence technique : laissez à Thomas ou à l’ambassadeur le soin de le compléter 😉

5.Continuez par l’en-tête :

* Titre (titre du module, quoi d’autre ?)
* Image/GIF/Vidéo (choisissez des images libres de droit, par exemple sur les sites “FREEPIK” ou “PIXABAY”, pour les vidéos youtube, copiez-collez simplement le lien)
* Introduction/résumé : Présentez en quelques lignes le module aux stagiaires, comme vous pourriez le faire en présentiel, une touche d’humour ou de sympathie est fortement appréciée !
* Objectif : ce sont les objectifs pédagogiques, ce que le stagiaire doit apprendre, ils commencent toujours par : “A la fin de ce module de formation, vous serez capable de…” suivit d’un verbe d’action, au besoin, fiez-vous à la taxonomie de Bloom
* Consignes spécifiques : s’il y en a, par exemple : pendant tout le module, ayez avec vous votre carnet XXX…” Sinon laissez cette case vide
* Durée indicative : combien de temps faut-il, à peu près, pour terminer le module

6. Remplissez le tableau “Plan du module”, cela vous permettra de voir plus clair dans les différents chapitres, au besoin, ajoutez des sections

7. Remplissez les sections

Complétez la présentation de la section, si besoin, référez-vous au point 5

8. Par section, déterminez le nombre d'activités.

Attention, une section peut très bien ne contenir qu'une seule activité

9. Commencez à remplir vos activités en rédigeant le contenu.

10. Si vous avez déjà une idée, vous pouvez choisir dès à présent le type d'activité

11. A chaque section son évaluation, elle n’est pas forcément là pour sanctionner le stagiaire, mais plutôt pour lui permettre de s’assurer qu’il à bien compris. Vous devez proposer à minima un exercice par section, choisissez s’il s’agira d’un texte ou d’un devoir, ne remplissez que la partie concernée.

12. Lorsque toutes les sections sont créées, compléter la dernière partie : Section finale

13. Dans la section finale, rédigez le résumé du module, il reprend les axes essentiels à retenir, ajoutez l’image si vous le souhaitez

14. Créer l’évaluation finale du module

15. Une fois l'ensemble des chapitres sections terminés, retournez sur la présentation et vérifiez l'exactitude des informations du plan

**Félicitations ! Votre module de formation est prêt à être digitalisé !**

1. **Lexique**

Activité :

Consigne :

Consigne pédagogique :

Consigne opératoire/opérationnelle :

Devoir :

Evaluation :

Module :

Objectif :

Section :

Test :

1. **Trame d’écriture**
2. **Présentation du module :**

**Auteur : Didier ROLLAND**

**Domaine :** Numérique

**Certifications concernées : TP DWWM**

**Compétences ciblées :**

**Références techniques :**

1. **En-tête**

**Titre : Le ccs3**

**Image/ GIF/ Vidéo d’introduction :**

**Introduction/ Résumé du module :**

**Objectif :** A l’issue de ce module de formation, vous serez capable de

**Consignes spécifiques :**

**Durée indicative :** 38 h

1. **Plan du module**

|  |  |
| --- | --- |
|  | **Titre** |
| **Section 1 - présentation** | Présentation du CSS et illustration |
| **Section 2 - présentation** | Où positionner le code CSS |
| **Section 3 - le sélecteur** | Les sélecteurs, comment les utiliser et résultat |
| **Section 4 - le sélecteur** | Priorité des sélecteurs et fonctionnement |
| **Section 5 - les propriétés** | Les propriétés basiques du CSS et démonstration |

|  |  |
| --- | --- |
| **Section 6 - les propriétés** | Le Responsive en CSS et démonstration avec les média queries |
| **Section 7 - les propriétés** | Transformation/animation en CSS et démonstration |
| **Section 8 - les propriétés** | Division et import en CSS et démonstration |
| **Section 9 - Framework** | Présentation et démonstration de quelques Library et Framework |

1. **Section 1**
   1. **Présentation de la section**

**Titre :** Présentation du CSS et illustration

**Image/Gif/Vidéo d’introduction :**

**Introduction/Résumé de la section :** Nous allons voir ce

**Objectifs :** Comprendre à quoi

**Consignes spécifiques :**

**Durée indicative :** 15 mn

* 1. **Contenu**

|  |  |  |
| --- | --- | --- |
|  | **Contenu** | **Type d’activité** |
| **Activité 1** | Le CSS (Cascading Style Sheets) est un langage de feuilles de style utilisé pour décrire la présentation visuelle d'une page web écrite en HTML (HyperText Markup Language). Son but principal est de séparer la structure du contenu d'une page web de sa présentation visuelle, ce qui permet une gestion plus efficace et flexible du design et de la mise en page d'un site web. La finalité du CSS peut être résumée comme suit :  **Séparation des préoccupations** : Le CSS permet de séparer la structure (HTML) du contenu d'une page web de sa présentation visuelle. Cette séparation des préoccupations rend le code plus modulaire, maintenable et évolutif en permettant aux développeurs de travailler sur le contenu et le style indépendamment les uns des autres.  **Contrôle de la mise en page** : Le CSS offre un contrôle précis sur la mise en page d'une page web, y compris le positionnement, la taille, la couleur, la police de caractères et d'autres aspects visuels des éléments HTML. Cela permet aux concepteurs de créer des mises en page complexes et esthétiquement attrayantes.  **Réutilisabilité** : Le CSS permet de définir des règles de style qui peuvent être appliquées à plusieurs éléments HTML. Cela favorise la réutilisabilité du code en évitant la duplication et en centralisant les styles communs dans un fichier CSS externe.  **Flexibilité** : Le CSS offre une grande flexibilité dans la conception de l'interface utilisateur, permettant aux développeurs et aux concepteurs de créer une variété de mises en page et de styles pour s'adapter aux besoins et aux préférences des utilisateurs.  **Accessibilité** : Le CSS permet d'améliorer l'accessibilité des sites web en séparant le contenu structuré des informations de présentation visuelle. Cela facilite la création de sites web adaptés aux technologies d'assistance et aux utilisateurs ayant des besoins spécifiques en matière d'accessibilité.  **Performance** : En séparant la présentation visuelle du contenu, le CSS permet de réduire la taille des fichiers HTML, ce qui contribue à améliorer les performances de chargement des pages web. De plus, en utilisant des règles de style optimisées, on peut réduire le temps de rendu des pages et améliorer l'expérience utilisateur.  Le fonctionnement du CSS (Cascading Style Sheets) repose sur un ensemble de principes qui définissent comment les styles sont appliqués aux éléments HTML d'une page web. Voici les principes de base du fonctionnement du CSS :   * Sélecteurs * Propriétés et valeurs * Hiérarchie et cascade * Feuilles de style * Priorité des styles | Leçon |
| **Activité 2** | Site avec du CSS | Type d'activité |
| **Activité 3** | Même site sans CSS | Type d'activité |
| **Activité 4** |  | Type d'activité |
| **Activité 5** |  | Type d'activité |

* 1. **Section 2**
  2. **Présentation de la section**

**Titre :** Où positionner le code CSS

**Image/Gif/Vidéo d’introduction :**

**Introduction/Résumé de la section :** Nous allons

**Objectifs :**

**Consignes spécifiques :**

**Durée indicative :** 15 mn

* 1. **Contenu**

|  |  |  |
| --- | --- | --- |
|  | **Contenu** | **Type d’activité** |
| **Activité 1** | A l’intérieur d’une balise HTML avec ‘style=’’propriété : valeur ;’’ on peut mettre plusieurs ‘propriété :valeur’ en les séparant par ‘ ; ‘  <!DOCTYPE html>  <html lang="fr">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>Document</title>  </head>  <body>      <h1 ***style="color:red;"***>Le CSS a l'intérieur d'une balise HTML</h1>      <p ***style="border:1px solid black;"***>un paragraphe</p>      <div ***style="background-color:red; color:yellow"***>une division</div>  </body>  </html>  Résultat :    Dans cette méthode, l’avantage est que l’on peut viser directement la balise dans le fichier HTML s’il on n’a qu’une page avec peu d’informations.  L’inconvénient, s’il y a plusieurs pages avec beaucoup d’informations et que l’on veut changer par exemple le style des paragraphes, il va falloir le changer dans toutes les balises paragraphes, donc long et fastidieux et il y a aussi le problème de surcharge de code dans les balises HTML. | Leçon |
| **Activité 2** | Autre solutions, dans une balise ‘**style’** dans un fichier HTML et plus précisément dans le ‘**HEAD’** comme dans l’exemple ci-dessous :  <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>Document</title>  ***<style>***  ***h1 {***  ***color: red;***  ***}***    ***p {***  ***border: 1px solid black;***  ***}***    ***div {***  ***background-color: red;***  ***color: yellow;***  ***}***  ***</style>***  </head>  <body>      <h1>Le CSS a l'intérieur d'une page HTML dans le head</h1>      <p>un paragraphe</p>      <div>une division</div>  </body>  </html>  Dans cette pratique, l’avantage est que si je veux changer la couleur de tous mes paragraphes, je n’ai qu’à aller à un seul endroit pour faire ma modification et cela sera pareil pour toutes les autres balises.  L’inconvénient est le même que la solution précédente si j’ai plusieurs pages, je devrais changer mes propriétés dans chacune d’elles. De plus mon fichier HTML sera plus lourd car le CSS prend beaucoup de ligne par rapport au HTML. | Leçon |
| **Activité 3** | Dernière solution, qui sera la seule à suivre car le CSS aura ses propres fichiers qui nous apportera une maintenabilité importante, voyons la mise en place. Pour cela nous mettrons un lien dans un fichier HTML pour indiquer quel fichier CSS sera utilisé et son chemin.  ***<link rel="stylesheet" href="./style.css">***  Attribut ‘**REL’** indique la relation entre le document ici HTML et la ressource lié ici CSS donc ‘stylesheet’  D’autre type de lien existe, la liste : <https://developer.mozilla.org/fr/docs/Web/HTML/Attributes/rel>  Attribut ‘**HREF’** défini l’URL de la ressource lié, peut être absolue ou relatif  Relatif : ./style.css  Absolu : C:\Users\RLDeveloppement\Documents\GitHub\digital-dwwm\style.css  D’autre attributs existent, voir le site : <https://developer.mozilla.org/fr/docs/Web/HTML/Element/link>  Le fichier HTML (index.html)  <!DOCTYPE html>  <html lang="fr">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>Document</title>  *<!-- Voici le lien pour accéder à notre fichier style.css -->*  ***<link rel="stylesheet" href="./style.css">***    </head>  <body>      <h1>Le CSS a l'intérieur d'une balise HTML</h1>      <p>un paragraphe</p>      <div>une division</div>  </body>  </html>    Et le fichier CSS (style.css)  h1 {      color: red;  }  p {      border: 1px solid black;  }  div {      background-color: red;      color: yellow;  }  De ce cas, si je veux changer un style dans les balises <p>, j’irai dans le fichier css et j’apporterez la modification et toutes mes pages qui sont concernées pas se fichier css auront la modification. Autre avantage, je pourrai créer plusieurs fichiers css pour être plus précis dans les parties visées, c’est-à-dire que je pourrais créer par exemple :  Pour ma barre de navigation : navigation.css  Pour mon body : style.css  Pour mon footer : footer.css  Pour la partie administration : style\_admin.css  Dans ce cas, si j’ai un problème dans une de ces parties, je peux ou un autre développeur, aller rapidement dans le fichier concerné. | Type d'activité |
| **Activité 4** | **ATTENTION :**  Dans les deux dernières solutions, il y a la priorité dans l’exécution du code CSS. En effet le code CSS est lu dans l’ordre c’est-à-dire que si au débit je défini la balise H1 de couleur rouge et que quelque ligne plus bas je donne une couleur jaune, c’est cette couleur qui sera défini par défaut. Voyons ça en code :  h1 {      color: red;  }  p {      border: 1px solid black;  }  div {      background-color: red;      color: yellow;  }  h1 {      color: blueviolet;  }  Le résultat :    Nous voyons sur notre exemple j’ai défini :  <h1> => color : red  Et en dernière ligne :  <h1> => color : blue  Donc c’est bien le bleu qui l’emporte car c’est la dernière propriété qui a été mise en place.  De même, quand vous utiliserez plusieurs fichiers CSS, le dernier fichier que vous aurez mis dans votre HEAD, ce sera les propriétés de ce fichier qui seront prioritaire.  Démonstration  Fichier HTML (index.html)  <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>Document</title>  *<!-- Voiçi le lien pour accéder a mon premier fichier style.css -->*      <link rel="stylesheet" href="./style.css">  *<!-- Voiçi le lien pour accéder a mon second fichier style2.css -->*      <link rel="stylesheet" href="./style2.css">  </head>  <body>      <h1>Le CSS a l'intérieur d'une balise HTML</h1>      <p>un paragraphe</p>      <div>une division</div>  </body>  </html>  Premier fichier CSS (style.css)  h1 {      color: red;  }  p {      border: 1px solid black;  }  div {      background-color: red;      color: yellow;  }  Second fichier CSS (style2.css)  h1 {      color: blue;  }  Et notre résultat    Donc notre H1 à bien pris le style du fichier style2.css.  **DONC ATTENTION AUX POSITIONS DE VOS FICHIERS CSS !!!** | Type d'activité |
| **Activité 5** |  | Type d'activité |

* 1. **Section 3**

**Titre :** Les sélecteurs, comment les utiliser et résultat

**Image/Gif/Vidéo d’introduction :**

**Introduction/Résumé de la section :** Nous allons

**Objectifs :** Vous

**Consignes spécifiques :**

**Durée indicative :** 2 h 15 mn

**Contenu**

|  |  |  |
| --- | --- | --- |
|  | **Contenu** | **Type d’activité** |
| **Activité 1** | En CSS (Cascading Style Sheets), un sélecteur est un motif qui permet de cibler spécifiquement des éléments HTML auxquels appliquer des styles. Les sélecteurs permettent aux développeurs de définir de manière précise quelles parties du document HTML doivent être stylisées.  Avant de commencer à savoir comment viser un élément HTML, voyons déjà les bases de ce qu’est un ID (identifiant) et une class (classe) :  ID : le nom d’un ID est unique dans une même page  Pour définir un ID en HTML : **id=’’nomID’’**  Pour l’appeler en CSS : **#nomID**  Class : le nom d’une class peut être utilisé plusieurs fois dans une même page  Pour définir une class en HTML : **class= ‘’nomClass’’**  Pour l’appeler en CSS : ***(point)*.nomClass**  **Une class ou un ID se mettra toujours dans la balise ouvrante.** | Leçon |
| **Activité 2** | Voyons maintenant comment sélectionner une balise. Pour sélectionner une balise, il suffit de l’appeler par son nom, exemple un paragraphe se sera p, un titre de niveau 1 se sera h1 et comme ça pour chaque balise, la n’a pas d’importance, c’est-à-dire vous pouvez mettre ‘**h1**’ ou ‘**H1**’, a l’inverse un ID ou une Class, la casse a son importance, **id=’’monId’’** et **id=’’MonId’’** seront différents. Voyons ce que cela nous donne en code !!!  **p**  {  Propriété : valeur ;  Propriété2 : valeur ;  } | Type d'activité |
| **Activité 3** | Sélectionner un élément via une classe :  **.nomClass**  {  Propriété : valeur ;  Propriété : valeur ;  } | Type d'activité |
| **Activité 4** | Sélectionner un élément via un ID  **#nomId**  {  Propriété : valeur ;  Propriété : valeur ;  } | Type d'activité |
| **Activité 5** | Sélectionner via une pseudo-class  Les pseudo-class sont là : <https://developer.mozilla.org/fr/docs/Web/CSS/Pseudo-classes#liste_des_pseudo-classes_standards>    Avec le **:hover**, au survole de la souris le changement va s’effectuer. | Type d'activité |

|  |  |  |  |
| --- | --- | --- | --- |
| **Activité 6** | Sélectionner via une pseudo-éléments  Les pseudo-éléments sont là : <https://developer.mozilla.org/fr/docs/Web/CSS/Pseudo-elements>    Avec le **::first-line**, juste la première ligne change de couleur. | | |
| **Activité 7** | | Sélectionner via un attribut  Les attributs sont là <https://developer.mozilla.org/fr/docs/Web/CSS/Attribute_selectors>    Avec le **a[title]**, on va appliquer le style seulement sur les lien dont on a mis un ‘title’ | Type d'activité |
| **Activité 8** | |  | Type d'activité |

**4.3 Section 4**

**Titre :** Priorité des sélecteurs et fonctionnement

**Image/Gif/Vidéo d’introduction :**

**Introduction/Résumé de la section :**

**Objectifs :** Savoir

**Consignes spécifiques :**

**Durée indicative :** 15 mn

* 1. **Contenu**

|  |  |  |
| --- | --- | --- |
|  | **Contenu** | **Type d’activité** |
| **Activité 1** | Tous les sélecteurs n'ont pas la même priorité. En cas de contradiction entre deux règles, celle qui s'applique est celle qui a le sélecteur le plus prioritaire. De façon générale, les sélecteurs ont une priorité d'autant plus grande qu'ils sont plus précis. Le sélecteur très général \* a une priorité de 0. A l'inverse, un sélecteur sur un identifiant, qui en principe ne concerne qu'un seul élément de la page, à une priorité de 100.  Pour déterminer la priorité d'un sélecteur, il faut considérer la logique suivante :   * Une règle avec la mention **!important** a une priorité de **10000**. * Une règle écrite dans **l'attribut style d'une balise HTML** a une priorité de **1000**. * Une règle avec un sélecteur sur **un identifiant ( # )** a une priorité de **100**. * Une règle avec un sélecteur sur **une classe ( . )** ou des **pseudo-class ( : )** a une priorité de **10**. * Une règle avec un sélecteur sur **un type d'élément ( p )** ou des **pseudo-éléments ( :: )** a une priorité de **1**. * Le sélecteur **étoile ( \* )** a une priorité de **0**. | Atelier |
| **Activité 2** | Démonstrations    Si on suit la logique, que c’est ce qui a été demandé en dernier dans notre CSS, le titre h1 doit être de couleur rouge, alors regardons ça !    Eh non, il est jaune car l’ID est d’une priorité plus élevé comme indiqué juste avant. | Fichier |
| **Activité 3** | Autre test avec le ‘ !important’, voyons le code    Cela nous donne    Donc sa priorité est encore plus élevé que l’ID. | Type d'activité |
| **Activité 4** |  | Type d'activité |
| **Activité 5** |  | Type d'activité |

* 1. **Section 5**

**Titre :** Les propriétés basiques du CSS et démonstration

**Image/Gif/Vidéo d’introduction :**

**Introduction/Résumé de la section :** Maintenant que vous avez vu où positionner le CSS, les différents sélecteurs que vous pouvez utiliser, vous allez maintenant passer à des choses plus intéressantes, à savoir designer votre site avec les nombreuses propriétés CSS qui s’offre à vous, comme les couleurs, modifier les polices, faire des cadres, positionner, animer vos éléments et bien autres choses encore… Allez lancez-vous !!!

**Objectifs :** Savoir

**Consignes spécifiques :**

**Durée indicative :** 7 h

**Contenu**

|  |  |  |
| --- | --- | --- |
|  | **Contenu** | **Type d’activité** |
| **Activité 1** | Vous allez voir les différentes propriétés basiques du CSS pour commencer à apporter le style dans vos page HTML.  Pour commencer, on a :  **Color** : indique la couleur d’écriture du texte  p  {  Color : Red ou #125487 ou rvb(125, 125 , 110) ou rvba(125, 125, 110 , 0.5);  }  Soit nous mettons le nom de la couleur (en anglais) : Red, Black, White, Yellow  Soit en Hexadécimal avec # devant : #124578, #1b253c  Soit en RVB (Rouge – Vert – Bleu) suivi d’un chiffre compris en 000 et 255 : rvb(110, 045, 125)  Soit RVBa (a : Alpha) pour de l’opacité compris entre 0 et 1 : rvba(125, 045, 110, 0.5)  .c1 {      color: blue;  */\* couleur direct en anglais \*/*  }  .c2 {      color: #d51919;  */\* couleur en hexadécimal \*/*  }  .c3 {      color: rgb(125, 125, 150);  */\* couleur en Red Green Blue \*/*  }  .c4 {      color: rgba(125, 125, 150, 0.3);  */\* couleur en Red Green Blue Alpha \*/*  }  **Font-size** : taille du texte  P  {  Font-size : 15px ou 15% ou small ;  }  Les possibilités sont :  En pixel : 10px, 20px  En % : 15%, 5% (100% = normal)  En em : 5em, 1.5em (taille relative, 1.0 = normal)  En ex : 1.2ex, 1ex ( taille relative à la hauteur de la lettre ‘x’, 1.0 = normal)  Prédéfini : xx-small -> très très petit  x-small -> très petit  small -> petit  medium -> moyen  large -> grand  x-large -> très grand  xx-large -> très très grand  .t1 {      font-size: 25px;  }  .t2 {      font-size: 55%;  }  .t3 {      font-size: 2.5em;  }  .t4 {      font-size: x-large;  }    **font-weight** : mise en gras du texte  P  {  Font-weight : bold ;  }  bold : gras  bolder : plus gras  normal : par défaut (non gras)  .g1 {      font-weight: bold;  }  .g2 {      font-weight: bolder;  }  **font-style** : mise en italique  P  {  Font-style : italic ;  }  italic : italique  oblique : autre façon de mettre en italique  normal : par défaut (pas en italique)  .i1 {      font-style: italic;  }  .i2 {      font-style: oblique;  }  **font-family** : nom de la police  P  {  Font-family : police1, police2, policee3 ;  }  Indiquer les noms de polices possibles par ordre de préférence : police1, police2, police3;  Utilisez des guillemets si le nom de la police comporte des espaces. Essayez de toujours mettre comme  dernière police possible "serif" ou "sans-serif".  Exemple : font-family: "Arial Black", Arial, Verdana, serif;  .p1 {      font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;  }  .p2 {      font-family: Arial, Helvetica, sans-serif;  }  **Border** : permet de mettre des bordures de différents types  P  {  Border : 1px solid black ;  }  Le premier paramètre est l’épaisseur du trait en pixel : 1px, 2px  Le second paramètre est le type de trait : solid, dotted, dashed…  Le dernier paramètre est la couleur soit en utilisant son nom, héxa, rvb ou rvba  On peut aussi choisir quelle bordure on veut mettre comme celle de gauche, haut, droite ou bas avec   * Border-left * Border-top * Border-right * Border-bottom   P  {  Border-top : 1px dashed #12584f ;  }  .b1 {      border: 1px solid black;  }  .b2 {      border: 1px dashed black;  }  .b3 {      border: 3px dashed black;  }  .b4 {      border: 3px dotted blue;  }  .b5 {      border-left: 3px dotted blue;      border-top: 3px dotted blue;  }  .b6 {      border-right: 3px solid blue;      border-bottom: 3px solid blue;  }  **Float** : pour le positionnement d’un objet  P  {  Float : left ;  }  On peut avoir : left (droite), right (gauche) ou none (pas de flottant, valeur par défaut)  **Clear** : pour stopper un flottant  P  {  Clear : left ;  }  On peut avoir :   * **left** : supprime l'effet d'un flottant à gauche précédent * **right** : supprime l'effet d'un flottant à droite précédent * **both** : supprime l'effet d'un flottant précédent, qu'il soit à gauche ou à droite * **none** : pas de suppression de l'effet du flottant (par défaut)   **position** : le type de positionnement pour pouvoir utiliser les propriétés **float** et **clear**   * **absolute** : position absolue par rapport au coin en haut à gauche * **fixed** : position fixe (fonctionne comme la position absolue). L'élément reste à sa position même quand on descend plus bas dans la page. * **relative** : position relative, par rapport à la position "normale" de l'élément * **static** : positionnement normal (par défaut)   exemple :  */\* on définit d'abord la position relative*  *par rapport à ca position de départ \*/*      position: relative;  */\* et on lui donne une position \*/*      float: right;  Autres positionnements possibles : Valeur en px, %, em... A utiliser pour un positionnement absolu, fixe ou relatif.   * **Top** : par rapport au haut * **Bottom** : par rapport au bas * **Left** : par rapport à droite * **Right** : par rapport à gauche   **Image** : Permet d’afficher une image avec différents paramètres, balise mise en place en HTML avec :    <img src= ‘’./image.jpg’’ alt=’’description de l’image’’ />  Côté CSS voyons les propriétés :  Width : largeur de l’image  Height : hauteur de l’image  Si width :50% et height : auto => on réduit sa taille et on reste proportionnel sur la réduction  Border, border-radius, opacity et d’autre propriétés peuvent être utilisé, même combiné  img {      width: 30%; /\* largeur \*/      height: auto; /\* hauteur \*/      opacity: 0.5; /\* opacité \*/      border: 3px dashed blue; /\* bordure bleu 3px et tirait \*/      border-radius: 25px; /\* on arrondi les coins \*/  }    Astuce : pour une image ronde ?  On applique un width et height de même dimension (250px) puis on applique un border-radius de 50%  img {      width: 250px;      height: 250px;      border-radius: 50%;  }  **Background** : pour définir le fond que l’on pourra mettre sur notre page (couleur, image), vous avez plusieurs propriétés possible.  Mettre une couleur unie en fond  body {      background: green;  }  Mettre du dégradé  body {      background: content-box radial-gradient(crimson, skyblue);  }  Mettre une image  body {      background: url("./html.jpg");  }  Vous avez aussi d’autre propriété lié au background (<https://developer.mozilla.org/en-US/docs/Web/CSS/background>)  On va voir les plus courantes :  **Background-repeat**: no-repeat / repeat / repeat-x / repeat-y  body {      background: url("./html.jpg");      background-repeat: no-repeat;  }  **Background-size**: auto / content / cover  body {      background: url("./html.jpg");      background-repeat: no-repeat; /\* pour ne pas répéter l’image si trop petite \*/      background-size: cover; /\* s’ajuste à votre page, le plus utilisé \*/  }  **Background-image** : pratique s’il on veut mettre deux images ou image + fond en couleur  body{      background-image: url(./html.jpg);  }  Image + fond de couleur avec effet  body {      background-image: linear-gradient(rgba(0, 0, 255, 0.5), rgba(255, 255, 0, 0.5)), url("./lezard.png");  }  Avec 2 images superposées  body {      background-image: url(./html.jpg), url(./lezard.png);  }  Attention, la première image sera au premier plan et la seconde en arrière-plan donc faites le bon choix !!!  Voyons les ombres avec la propriété : **box-shadow**  Box-shadow : offset-x | offset-y | color  .box {      width: 150px; /\* taille de notre p \*/      color: blanchedalmond; /\* couleur de l’écriture \*/      background-color: blue; /\* couleur de fond de notre p \*/      box-shadow: 5px 10px red; /\* notre ombre 5px en X, 10px en Y et couleur rouge \*/  }  Box-shadow : offset-x | offset-y | adoucis-x-y | color  .box {      width: 150px; /\* taille de notre p \*/      color: blanchedalmond; /\* couleur de l’écriture \*/      background-color: blue; /\* couleur de fond de notre p \*/      box-shadow: 5px 10px 6px red; /\* notre ombre 5px en X, 10px en Y, adoucis en X Y de  6px et couleur rouge \*/  }  On peut aussi adoucir notre couleur d’ombre avec **rgba**  .box {      width: 150px; /\* taille de notre p \*/      color: blanchedalmond; /\* couleur de l’écriture \*/      background-color: blue; /\* couleur de fond de notre p \*/      box-shadow: 5px 10px 6px rgba(245,33,33,0.5) ; /\* notre ombre 5px en X, 10px en Y,  adoucis en X Y de 6px et couleur  rouge adoucis de 0.5 \*/  }  On peut aussi ajouter des ombres a du texte avec text-shadow  H1 {      color: blue;      text-shadow: 5px 10px 6px rgba(245, 6, 6, 0.5);  }  La propriété **display** définit le type d'affichage utilisée pour le rendu d'un élément (de bloc ou en ligne) et la disposition utilisée pour ses éléments fils : grille ou boîtes flexibles.  Fichier HTML   <p class="text">Mon texte</p>   <p class="secret">Mot de passe</p>  Fichier CSS  .secret {      display: none;  }  Dans ce cas, le **display : none ;** va s’appliquer à la classe **secret**, ce qui a pour effet de ne pas voir **Mot de passe** dans notre page.    Je peux aussi transformer une balise **‘inline’** en balise **‘block’** et vis-vers-ça, voyons ça !!!  Fichier html      <p>mettre un span dans un paragraphe <span>'Balise span'</span>, bien intégré </p>  Fichier CSS  span {      display: block;  }  Le fait de faire un **display : block ;** va transformer notre balise **span** (inline) en balise **block**    Le résultat parle de lui-même, notre balise **span** (inline) c’est bien transformée en balise **block**, on voit bien le **saut de ligne**.  Et donc l’inverse :  Fichier HTML      <p>premier paragraphe</p>      <p class="inline">second paragraphe</p>      <p class="inline">troisième paragraphe</p>  Fichier CSS  .inline {      display: inline;  }  Donc on aura    On a bien le deuxième et troisième paragraphe en inline  Mettre élément **block** en **inline** avec **flex**  Fichier HTML  <ul>          <li>accueil</li>          <li>apropos</li>          <li>contact</li>      </ul>  Fichier CSS  ul {      display: flex;  list-style-type : none ; /\* on retire les puces \*/  }  Ça nous donne    Bon d’’accord pas cool !!! on modifie un peu  li {      padding: 5px;  }    C’est mieux !!! c’est le début d’un menu en CSS. | Atelier |
| **Activité 2** |  | Fichier |
| **Activité 3** |  | Type d'activité |
| **Activité 4** |  | Type d'activité |
| **Activité 5** |  | Type d'activité |

* 1. **Section 6**

**Titre :** Le Responsive en CSS et démonstration avec les média queries

**Image/Gif/Vidéo d’introduction :**

**Introduction/Résumé de la section :**

**Objectifs :** Savoir

**Consignes spécifiques :**

**Durée indicative :** 7 h

**Contenu**

|  |  |  |
| --- | --- | --- |
|  | **Contenu** | **Type d’activité** |
| **Activité 1** | QU’EST-CE QUE LE RESPONSIVE DESIGN ?  Le Responsive design a été traduit en français par Design réactif/adapté/réceptif. Ce design permet de modifier la mise en page d’un site afin que le contenu s’adapte à l’écran quel que soit le terminal utilisé (smartphone, tablette, ordinateur de bureau, TV…). Il s’agit d’adapter le site à toutes les résolutions. Autrement dit, les blocs de contenus (textes et images) se redimensionnent et se réorganisent en fonction de l’espace disponible sur l’écran. Exemple ci-dessous.  (voir gif responsive design)  Principe Responsive WebDesign  RESPONSIVE DESIGN VS ADAPTIVE DESIGN : QUELLE DIFFÉRENCE ?  Le Responsive Design se définit comme un contenu qui « s’adapte » à l’écran, d’où la confusion fréquente avec l’Adaptive Design. Le Responsive Web Design doit plutôt être vu comme un sous-ensemble de l’Adaptive Design (souvent traduit par « conception évolutive »).  <https://www.usabilis.com/wp-content/uploads/2018/05/10-Difference-affichage-Responsive-design-VS-Adaptative-Design.gif>  <https://www.usabilis.com/wp-content/uploads/2018/05/09-RWD-vs-Adapt-example.gif>  TROIS POSSIBILITÉS POUR CRÉER LA VERSION MOBILE DE SON SITE WEB  Le responsive Design est l’une des différentes solutions pour créer une version mobile d’un site web. Cette vidéo de vulgarisation pose clairement les enjeux et solutions. Trois sont présentées ici :  Site en responsive Design  Site mobile dédié  Application mobile  En vidéo : <https://youtu.be/jn1OQ1tKDmQ>  Pour le responsive design, vous allez voir les MEDIA QUERIES, mais c’est quoi et comment ça marche ?  Les media queries, ou « requêtes média » en français, sont des modules CSS conditionnels qui permettent de créer un site responsive, c'est-à-dire d'optimiser le contenu d'une page internet ou d'une application en fonction de l'écran sur lequel il est affiché. Le contenu s'adapte ainsi à la taille de l'écran, à son orientation et à sa résolution. Concrètement, la police de caractères, les couleurs du site et la taille des images sont modifiables pour rester lisibles sur tous les supports, sans agresser l'œil ni occuper trop ou trop peu d'espace.  Les media queries sont, en d'autres termes, des modèles d'apparence décrits dans la feuille CSS qui déterminent la mise en forme d'une application ou d'un site internet sur le média utilisé (téléphone, tablette, ordinateur, etc.). Ils s'exécutent lorsqu'une requête correspond à la situation vécue. Chaque media query doit donc préciser le média concerné et les critères qui le conditionnent.  En plus de la précision du média, 9 autres critères permettent de caractériser les media queries :   * La largeur de la fenêtre d'affichage (width). * La hauteur de la fenêtre d'affichage (height). * La hauteur de l'écran (device-height). * La largeur de l'écran (device-width). * Le ratio de l'écran (device-aspect-ratio). * Le ratio de la fenêtre d'affichage (aspect-ratio). * L'orientation de la zone d'affichage (orientation). * La gestion de la couleur (color). * Le type d'écran de sortie (media).   Les media queries peuvent être complexifiés en combinant les caractéristiques à l'aide des opérateurs logiques « not », « and » et « only ». Ainsi, un même style peut être appliqué à plusieurs hauteurs d'écran par exemple avec « and ». Quant à « not », il active les media queries lorsque la situation d'utilisation correspond à l'inverse de la requête. Il nécessite par ailleurs la désignation d'un type de média pour fonctionner. Enfin, « only » ne s'applique que si l'intégralité de la requête est vérifiée.  On a deux façons d’intégrer les média queries, soit dans le fichier CSS, soit dans le fichier HTML.  Exemple d’utilisation dans le fichier CSS avec la propriété **@media** : @media screen and (max-width :1100px){le code css}  Media queries écrit dans le style CSS  Exemple d’utilisation dans le fichier HTML qui appellera le fichier CSS concerné en fonction de la taille de l’écran via la balise **link** et en incluant **media** : <link rel="stylesheet" href="./css/style-576.css" media="(max-width:576px)">  <!DOCTYPE html>  <html lang="fr">      <head>          <meta charset="UTF-8">          <meta name="viewport" content="width=device-width, initial-scale=1.0">          <meta http-equiv="X-UA-Compatible" content="ie=edge">          <link rel="stylesheet" href="./css/style-576.css" media="(max-width:576px)">          <link rel="stylesheet" href="./css/style-576-1024.css" media="(min-width:577px) and (max-width:1024px)">          <link rel="stylesheet" href="./css/style-1024.css" media="(min-width:1024px)">          <link rel="stylesheet" href="./css/style.scss">          <title>CV</title>      </head>      <body>          <main>  L’avantage de cette solution et que si vous avez besoin de modifier un paramètre d’affichage pour la version mobile, rien de plus simple d’aller dans le fichier concerné. Pour la première solution, il va falloir chercher dans le fichier CSS la ligne qui correspond à la propriété du mobile. En effet s’il n’y a qu’un fichier, cela implique plus de ligne de code car on aura le code du desktop, tablette, mobile et peut être d’autre.  Les différentes tailles pour le responsive :  Media queries pour ordinateur de bureau, tablette, smartphone  Points de rupture responsives communs :  Pour travailler avec les media queries, vous devez décider des « points de rupture responsive » ou des points d’arrêt de la taille de l’écran. Un point de rupture est la largeur de l’écran où vous utilisez une media query pour implémenter de nouveaux styles CSS.  Les résolutions à adopter pour le responsive sont   * Mobile portrait : 576 px * Tablette : 768 px * Ordinateur portable : 992 px * Ordinateur portable ou de bureau grands écrans : 1200 px   Si vous voulez que votre site soit visible via des mobiles, pensez à « **mobile first** ».  Pour augmenter vos chanche de faire un site le plus responsive, pensez à bien position vos « container » et utilisez les « grid » ou les « flexbox » pour une meilleur souplesse. | Atelier |
| **Activité 2** | <https://developer.chrome.com/docs/devtools?hl=fr>  Les outils pour les développeurs Chrome sont un ensemble d'outils pour les développeurs Web intégrés directement au navigateur Google Chrome. Les outils de développement vous permettent de modifier des pages à la volée et de diagnostiquer les problèmes rapidement, ce qui vous aide à créer des sites Web de meilleure qualité, plus rapidement. | URL |
| **Activité 3** |  | Type d'activité |
| **Activité 4** |  | Type d'activité |
| **Activité 5** |  | Type d'activité |

* 1. **Section 7**

**Titre :** Transformation/animation en CSS et démonstration

**Image/Gif/Vidéo d’introduction :**

**Introduction/Résumé de la section :**

**Objectifs :** Savoir

**Consignes spécifiques :**

**Durée indicative :** 7 h

**Contenu**

|  |  |  |
| --- | --- | --- |
|  | **Contenu** | **Type d’activité** |
| **Activité 1** | Transition CSS :  Une transition permet de créer des effets de transition fluides lorsqu'un élément change d'état, comme lorsqu'il passe d'une couleur à une autre, ou lorsqu'il change de taille ou de position.  Voici les différentes propriétés pour les transitions :   * transition * transition-delay * transition-duration * transition-property * transition-timing-function   La propriété **transition** :  Vous allez créer un fichier index.html et syte2.html (n’oubliez pas le lien CSS dans le fichier HTML)  Index.html (fichier simplifié)  <!DOCTYPE html>  <html>  <head>      <link rel="stylesheet" href="style2.css">  </head>  <body>      <h1>La propriété transition</h1>      <p>Passez la souris sur l'élément div ci-dessous pour voir l'effet de transition :</p>      <div></div>  </body>  </html>  Style2.css  div {      width: 100px;      height: 100px;      background: red;  */\* transition de 2s sur la largeur de l'objet \*/*      transition: width 2s;  }  div:hover {  */\* au survol, l'objet va prendre une largeur de 300px dans un délais de 2 s \*/*      width: 300px;  }  Vous pouvez aussi cumuler les propriétés de la façon suivante, en les séparant par une **virgule**.  Index.html (reste inchangé)  Style2.css  div {      width: 100px;      height: 100px;      background: red;  */\* transition de 2s sur la largeur et 4s sur la hauteur de l'objet \*/*      transition: width 2s, height 4s;  }  div:hover {  */\* au survol, l'objet va prendre une largeur de 300px dans un délais de 2 s \*/*  */\* et en même temps, l'objet va prendre une hauteur de 300px dans un délais de 4 s \*/*      width: 300px;      height: 300px;  }  A toutes les transitions vous pouvez ajouter des vitesses de transition, voyons ça avec **transition-timing-function** :   * **ease**- spécifie un effet de transition avec un démarrage lent, puis rapide, puis une fin lente (c'est la valeur par défaut) * **linear**- spécifie un effet de transition avec la même vitesse du début à la fin * **ease-in**- spécifie un effet de transition avec un démarrage lent * **ease-out**- spécifie un effet de transition avec une fin lente * **ease-in-out**- spécifie un effet de transition avec un début et une fin lents * **cubic-bezier(n,n,n,n)**- vous permet de définir vos propres valeurs dans une fonction cubique-bézier   Vous allez modifier votre fichier HTML comme ci-dessous  Index.html  <!DOCTYPE html>  <html>  <head>      <link rel="stylesheet" href="style2.css">  </head>  <body>      <h1>La propriété transition-timing-function </h1>      <p>Passez la souris sur l'élément div ci-dessous pour voir l'effet de transition :</p>      <div id="div1">          <bold>linear</bold>      </div><br>      <div id="div2">          <bold>ease</bold>      </div><br>      <div id="div3">          <bold>ease-in</bold>      </div><br>      <div id="div4">          <bold>ease-out</bold>      </div><br>      <div id="div5">          <bold>ease-in-out</bold>      </div><br>  </body>  </html>  Style2.css  div {      text-align: center;      width: 100px;      height: 100px;      background: red;      transition: width 2s;  }  #div1 {      transition-timing-function: linear;  }  #div2 {      transition-timing-function: ease;  }  #div3 {      transition-timing-function: ease-in;  }  #div4 {      transition-timing-function: ease-out;  }  #div5 {      transition-timing-function: ease-in-out;  }  div:hover {      width: 300px;  }  La propriété **transition-delay** : cette propriété va nous donner du retard au démarrage de la transition.  Index.html (à modifier)  <!DOCTYPE html>  <html>  <head>      <link rel="stylesheet" href="style2.css">  </head>  <body>      <h1>La propriété transition-timing-function </h1>      <p>Passez la souris sur l'élément div ci-dessous pour voir l'effet de transition :</p>      <div>retard 1s<br>transition 2s</div>  </body>  </html>  Style2.css  div {      text-align: center;      color: blanchedalmond;      width: 100px;      height: 100px;      background: red;      transition: width 3s;      transition-delay: 1s;  }  div:hover {      width: 300px;  }  Vous allez voir que l’on peut combiner plusieurs propriété en même temps.  La **transition** et **transform** :  Index.html  <!DOCTYPE html>  <html>  <head>      <link rel="stylesheet" href="style2.css">  </head>  <body>      <h1>La propriété transition-timing-function </h1>      <p>Passez la souris sur l'élément div ci-dessous pour voir l'effet de transition :</p>      <div>transition 2s<br>transform 2s</div>  </body>  </html>  Style2.css  div {      text-align: center;      color: blanchedalmond;      width: 100px;      height: 100px;      background: red;  */\* d'abord une transition qui va agir sur la taille de la div \*/*  */\* et la propriété transform qui va aussi agir sur la div \*/*      transition: width 2s, height 2s, transform 2s;  }  div:hover {      width: 300px;      height: 300px;  */\* le transform associé ici sera la propriété rotate de 180 degré \*/*      transform: rotate(180deg);  }  Vous pouvez aussi agir sur chaque propriété une à une en appelant les propriétés avec plus de précision. Voyons ça en code :  Index.html (code reste inchangé)  Style2.css  div {      text-align: center;      color: blanchedalmond;      width: 100px;      height: 100px;      background: red;      transition-property: width;  */\* property pour agir sur la largeur \*/*      transition-duration: 2s;  */\* duration pour le délais de transition \*/*      transition-timing-function: linear;  */\* timing-fuction pour le type de départ/arrivé \*/*      transition-delay: 1s;  */\* delay pour le temps du début de la transition \*/*  }  div:hover {      width: 300px;  }  Ou avec un code raccourci sur la propriété **transition :**  Style2.css  div {      text-align: center;      color: blanchedalmond;      width: 100px;      height: 100px;      background: red;  */\* property pour agir sur la largeur \*/*  */\* duration pour le délais de transition \*/*  */\* timing-fuction pour le type de départ/arrivé \*/*  */\* delay pour le temps du début de la transition \*/*      transition: width 2s linear 1s;  }  div:hover {      width: 300px;  } | Atelier |
| **Activité 2** | Animation CSS :  Une animation permet de créer des mouvements ou des changements d'apparence plus complexes et plus contrôlés que les transitions.  Les différentes propriétés possible avec les animations :   * @keyframes * animation-name * animation-duration * animation-delay * animation-iteration-count * animation-direction * animation-timing-function * animation-fill-mode * animation   Les **@keyframes** :  Lorsque vous spécifiez des styles CSS dans la règle @keyframes, l'animation passera progressivement du style actuel au nouveau style à certains moments.  Pour faire fonctionner une animation, vous devez lier l'animation à un élément.  L'exemple suivant lie l'animation « exemple » à l'élément <div>. L'animation durera 4 secondes et changera progressivement la couleur d'arrière-plan de l'élément <div> de "rouge" à "jaune" :  Index.html  <!DOCTYPE html>  <html>  <head>      <link rel="stylesheet" href="style2.css">  </head>  <body>      <h1>CSS Animation</h1>      <div></div>      <p><b>Note : </b>Lorsque l'animation est terminée, elle retrouve son style d'origine.</p>  </body>  </html>  Style2.css  div {      width: 100px;      height: 100px;      background-color: red;      animation-name: example;  */\* nom de l'animation qui sera utilisé pour le @keyframes \*/*      animation-duration: 4s;  */\* durée de l'animation \*/*  }  @keyframes example {      from {          background-color: red;  */\* point de départ \*/*      }      to {          background-color: yellow;  */\* point d'arrivé \*/*      }  }    Autre exemple, on peut aussi utiliser les pourcentages à la place de ‘from’ et ‘to’ :  Index.html (reste inchangé)  Style2.css  div {      width: 100px;      height: 100px;      background-color: red;      animation-name: example;  */\* nom de l'animation qui sera utilisé pour le @keyframes \*/*      animation-duration: 4s;  */\* durée de l'animation \*/*  }  @keyframes example {      0% {          background-color: red;      }      25% {          background-color: yellow;      }      50% {          background-color: blue;      }      100% {          background-color: green;      }  }  Voyons comment lier le changement de couleur et en même temps faire un déplacement :  Index.html (reste inchangé)  Style2.css  div {      width: 100px;      height: 100px;      background-color: red;      position: relative;  */\* nous utilisons la propriété 'position' pour la position de départ \*/*      animation-name: example;      animation-duration: 4s;  }  @keyframes example {      0% {          background-color: red;          left: 0px;  */\* position gauche de départ \*/*          top: 0px;  */\* position en haut de départ \*/*      }      25% {          background-color: yellow;          left: 200px;  */\* on se déplace de 200px vers la droite \*/*          top: 0px;  */\* on reste en haut \*/*      }      50% {          background-color: blue;          left: 200px;  */\* on reste a notre position de 200 px \*/*          top: 200px;  */\* on descend de 200px \*/*      }      75% {          background-color: green;          left: 0px;  */\* on se repositionne à gauche \*/*          top: 200px;  */\* on reste à 200px du haut \*/*      }      100% {          background-color: red;          left: 0px;  */\* on reste a gauche, position d'origine \*/*          top: 0px;  */\* on revient à notre position d'origine \*/*      }  }  Vous pouvez aussi définir le nombre de cycle de cette animation avec **animation-iteration-count** :  Index.html (reste inchangé)  Style2.css  div {      width: 100px;      height: 100px;      background-color: red;      position: relative;  */\* nous utilisons la propriété 'position' pour la position de départ \*/*      animation-name: example;      animation-duration: 4s;      animation-iteration-count: 3; */\* va exécuter notre @keyframes 3 fois \*/*  }  @keyframes example {      0% {          background-color: red;          left: 0px;  */\* position gauche de départ \*/*          top: 0px;  */\* position en haut de départ \*/*      }      25% {          background-color: yellow;          left: 200px;  */\* on se déplace de 200px vers la droite \*/*          top: 0px;  */\* on reste en haut \*/*      }      50% {          background-color: blue;          left: 200px;  */\* on reste a notre position de 200 px \*/*          top: 200px;  */\* on descend de 200px \*/*      }      75% {          background-color: green;          left: 0px;  */\* on se repositionne à gauche \*/*          top: 200px;  */\* on reste à 200px du haut \*/*      }      100% {          background-color: red;          left: 0px;  */\* on reste a gauche, position d'origine \*/*          top: 0px;  */\* on revient à notre position d'origine \*/*      }  }  Pour cette propriété vous pouvez mettre « **infinite** » pour que l’exécution soit infinie.  Vous avez aussi a votre disposition d’autre propriété comme agir sur le sens de l’animation avec **animation-direction**, voici les valeurs possibles :   * **normal**- L'animation est jouée normalement (avants). C'est par défaut * **reverse**- L'animation est jouée en sens inverse (en arrière) * **alternate** - L'animation est jouée d'abord en avant, puis en arrière * **alternate-reverse**- L'animation est jouée d'abord à l'envers, puis en avant   voyons en une comme **alternate** :  index.html (reste inchangé)  style2.css  div {      width: 100px;      height: 100px;      background-color: red;      position: relative;  */\* nous utilisons la propriété 'position' pour la position de départ \*/*      animation-name: example;      animation-duration: 4s;      animation-iteration-count: 3;  */\* va exécuter notre @keyframes 3 fois \*/*      animation-direction: alternate;  }  @keyframes example {      0% {          background-color: red;          left: 0px;  */\* position gauche de départ \*/*          top: 0px;  */\* position en haut de départ \*/*      }      25% {          background-color: yellow;          left: 200px;  */\* on se déplace de 200px vers la droite \*/*          top: 0px;  */\* on reste en haut \*/*      }      50% {          background-color: blue;          left: 200px;  */\* on reste a notre position de 200 px \*/*          top: 200px;  */\* on descend de 200px \*/*      }      75% {          background-color: green;          left: 0px;  */\* on se repositionne à gauche \*/*          top: 200px;  */\* on reste à 200px du haut \*/*      }      100% {          background-color: red;          left: 0px;  */\* on reste a gauche, position d'origine \*/*          top: 0px;  */\* on revient à notre position d'origine \*/*      }  }  Vous pouvez aussi spécifier le mode de remplissage pour une animation avec **animation-fill-mode** et les valeurs suivantes :   * **none**- Valeur par défaut. L'animation n'appliquera aucun style à l'élément avant ou après son exécution * **forwards**- L'élément conservera les valeurs de style définies par la dernière image clé (dépend de la direction de l'animation et du nombre d'itérations d'animation) * **backwards**- L'élément obtiendra les valeurs de style définies par la première image clé (dépend de la direction de l'animation) et les conservera pendant la période de retard de l'animation. * **both**- L'animation suivra les règles pour l'avant et l'arrière, étendant les propriétés de l'animation dans les deux sens   voyons ensemble l’une de ces valeurs, **both** :  index.html (reste inchangé)  style2.css  div {      width: 100px;      height: 100px;      background: red;      position: relative;      animation-name: example;      animation-duration: 3s;      animation-delay: 2s;      animation-fill-mode: both;  }  @keyframes example {      from {          top: 0px;          background-color: yellow;      }      to {          top: 200px;          background-color: blue;      }  }  Vous avez aussi les propriétés de raccourci d’animation :  Non raccourci  div {      animation-name: example;      animation-duration: 5s;      animation-timing-function: linear;      animation-delay: 2s;      animation-iteration-count: infinite;      animation-direction: alternate;    }  Raccourci    div {      animation: example 5s linear 2s infinite alternate;    } | URL |
| **Activité 3** |  | Type d'activité |
| **Activité 4** |  | Type d'activité |
| **Activité 5** |  | Type d'activité |

* 1. **Section 8**

**Titre :** Division et import en CSS et démonstration

**Image/Gif/Vidéo d’introduction :**

**Introduction/Résumé de la section :**

**Objectifs :** Savoir

**Consignes spécifiques :**

..

**Durée indicative :** 2 h

**Contenu**

|  |  |  |
| --- | --- | --- |
|  | **Contenu** | **Type d’activité** |
| **Activité 1** | La règle @ **@import** est utilisée afin d'importer des règles à partir d'autres feuilles de style. Ces règles @ doivent être utilisées avant toutes les autres règles, à l'exception de @charset. **@import** n'est pas une instruction imbriquée et ne peut donc pas être utilisée à l'intérieur de groupe de règles conditionnelles. | Atelier |
| **Activité 2** | Exemple de syntaxe CSS @import{} :  **@import url("testimport.css");**  La règle de feuille de style @import CSS accepte comme valeur :  url(url), l'url d'un fichier CSS, l'url peut être relative ou absolue.  nom de médias, un ou des médias séparé par un virgule pour lesquels on veut appliquer la feuille de style importée.  Par défaut all.  Exemple d'écriture CSS de @import avec médias :  **@import url("testimport.css") screen, print;**  Exemple de code @import{} CSS  Conditions d'application de la CSS @import{}.  La règle CSS @import permet de cacher certaines propriétés CSS qui ne seraient pas compatibles avec les vieux navigateurs.  La règle de feuille de style @import CSS n'a pas d'équivalence en HTML ou XHTML.  Problème d'interprétation de @import{} en Css.  La règle CSS @import pose des problèmes car elle n'est interprétée par les vieux navigateurs, Netscape 4 par exemple. Opéra semble accepter à partir de la version 7.11. Evitez seulement de mettre des " ' ".  Sur Blackberry, il est déconseillé d'utiliser la feuille de style @import CSS.  Evolutions de la syntaxe CSS de @import{}.  En CSS 3, la règle CSS @import CSS accepte une "Media Query" qui permet de mettre des règles supplémentaires autre que les médias (tailles, résolutions, orientation, etc...).  Exemple en code :  Index.html  <!DOCTYPE html>  <html>  <head>      <link rel="stylesheet" href="style2.css">  </head>  <body>      <h1>CSS Animation</h1>      <div></div>      <p><b>Note : </b>Lorsque l'animation est terminée, elle retrouve son style d'origine.</p>  </body>  </html>  Style2.css  @import url("./style3.css");  div {      width: 100px;      height: 100px;      background: red;      position: relative;      animation-name: example;      animation-duration: 3s;      animation-delay: 2s;      animation-fill-mode: both;  }  @keyframes example {      from {          top: 0px;          background-color: yellow;      }      to {          top: 200px;          background-color: blue;      }  }  Style3.css  h1 {      color: blue;      border: 1px solid blue;  }  On pourrait même lier un troisième fichier css au second  Style4.css  p {      font-size: x-large;      border: 1px solid black;  }  Et dans style3.css, on ajoute un @import comme suivant  @import url("./style4.css");  h1 {      color: blue;      border: 1px solid blue;  } | URL |
| **Activité 3** |  | Type d'activité |
| **Activité 4** |  | Type d'activité |
| **Activité 5** |  | Type d'activité |

* 1. **Section 9**

**Titre :** Présentation et démonstration de quelques Library et Framework

**Image/Gif/Vidéo d’introduction :**

**Introduction/Résumé de la section :**

**Objectifs :** Savoir

**Consignes spécifiques :**

**Durée indicative :** 3 h 30 mn

**Contenu**

|  |  |  |
| --- | --- | --- |
|  | **Contenu** | **Type d’activité** |
| **Activité 1** | Un framework CSS, est une librairie qu’on importe dans un projet de développement, qui contient un ensemble de composants et fonctionnalités, censés aider au développement front-end.  Typiquement, il vient généralement avec un système de grid (pour du responsive), et tout un tas de composants et de thèmes (boutons, headers, containers, etc.), simplifiant la construction d’une interface front.  Bien qu’un framework CSS puisse être une aide intéressante dans un projet de développement, il n’est pas toujours utile d’en utiliser un. Au contraire, cela peut même parfois desservir un projet.  La question qui revient souvent est quelle est la différence entre librairie et framework. Cette question n’a pas toujours évidente à répondre car la différence est plutôt vague. Pour cela je voous propose de voir cette vidéo : <https://apprendre-a-coder.com/difference-librairie-framework/>  Voyons quelques librairies et framework.  Le premier est **Bootstrap**  Logo du framework CSS Boostrap <https://getbootstrap.com/>  **Bootstrap** a été l’un des premiers frameworks CSS. Créé par Twitter, la librairie a commencé à être utilisée par le grand public notamment parce qu’elle mettait à disposition un système de grid, permettant aux développeurs d’adapter facilement des composants aux différentes tailles d’écran.  Mais, comme déjà expliqué dans cet article, CSS propose aujourd’hui nativement cette fonctionnalité, via les grid ou flexbox. Bootstrap a donc perdu une partie de son intérêt, mais le fait qu’il propose un éventail complet de composants le rend toujours attractif aux yeux des développeurs.  Attention toutefois, Bootstrap est assez rigide, c’est à dire qu’une fois importé, il est difficile de facilement personnaliser les différents composants qu’il propose.  Pour l’utiliser, n’oubliez pas le lien CDN (réseau de diffusion de contenu) :  Dans le <Head>:  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">  Avant </body>  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous"></script>  Exemple :  <a href="#" class="btn btn-primary">Bouton</a> | Atelier |
| **Activité 2** | Logo du framework CSS Tailwindcss <https://tailwindcss.com/>  Un autre framework CSS qu’on peut citer, plus récent cette fois, c’est Tailwind CSS. Un des éléments qui différencie **Tailwind** d’autres frameworks comme Bootstrap, c’est l’utilisation de classes utilitaires à la place de classes sémantiques.  Qu’est ce que cela veut dire ? Eh bien, que les noms des classes, au lieu d’être sémantiques (définies par un mot ou un ensemble de mots), décrivent réellement ce que fait la classe, leur utilité.  Tailwind par rapport à Bootstrap, il s’installe, procédure ici : <https://tailwindcss.com/docs/installation>  Exemple : reprenons le code de **Bootstrap** et voyons la différence :  <a href="#" class="bg-blue-700 text-white px-5 py-3 font-bold rounded">Bouton</a>  Dans Boostrap, c’est du sémantique, le style appliqué sur le lien est celui défini dans les classes btn et btn-primary. Avec Tailwind, on retrouve le même style, mais cette fois défini de façon utilitaire. Cela peut sembler plus barbare, mais au moins, on sait exactement de quoi il s’agit quand on lit ces lignes, sans forcément connaître le framework. | URL |
| **Activité 3** | Vous avez aussi **Fontawesome**, qui lui nous permet d’intégrer des icones  <https://fontawesome.com/>  On peut utiliser le lien CDN : (<https://www.bootstrapcdn.com/fontawesome/>)  A mettre dans le <head>  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-free@6.4.2/css/fontawesome.min.css" integrity="sha384-BY+fdrpOd3gfeRvTSMT+VUZmA728cfF9Z2G42xpaRkUGu2i3DyzpTURDo5A6CaLK" crossorigin="anonymous">  Mais peut être lui aussi installé. | Type d'activité |
| **Activité 4** | <https://www.bootstrapcdn.com/bootswatch/>  Nous allons aussi parler de **Bootswatch** qui va nous permettre de choisir entre différents thèmes et récupérer directement le code soit d’un menu, bouton, typo ou encore des tables.  Pour l’utiliser, il vous faudra le CDN Bootstrap et le CDN JS que je vous mets :  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min.js" integrity="sha384-QJHtvGhmr9XOIpI6YVutG+2QOK9T+ZnN4kzFN1RtK3zEFEIsxhlmWl5/YESvpZ13" crossorigin="anonymous"></script> | Type d'activité |
| **Activité 5** | <https://sass-lang.com/>  Et enfin nous avons **Sass** / **SCSS**   * Le **Sass**, ou syntaxe indentée : il s’agit de la syntaxe originale reposant sur l’indentation (comme son nom l’indique) ; elle a pour extension .sass * Le **SCSS** : la syntaxe est proche de CSS et n’importe quel code CSS sera valide en SCSS, ce qui simplifie son utilisation ; elle a pour extension .scss   Avec le **Sass** ou **SCSS**, l’écriture est **encapsulée dans sa règle parente**, voyons ça :  En CSS :  header{      height: 60px;  }   header .logo{      float: left;      width: 80px;      height: 100px;   }  Maintenant en Sass/SCSS :  header{      height: 60px;        .logo{          float: left;          width: 80px;          height: 100px;      }   }  Dans notre second exemple , il nous sera plus facile de récupérer une propriété d’un élément ou section qu’en CSS. Il n’y a pas que ça, vous pouvez créer des variables ($variable : blue ;), encapsuler des pseudo-class ( :hover, :not…), des extensions c’est-à-dire un ensemble de règle que vous pourrait utiliser plus loin dans votre code, les inclusions utilisées comme les extensions mais en plus poussées comparable à des fonctions (avec ou sans paramètres). On peut faire des calculs avec différents opérateurs. Encore plus puissant, on peut faire des vrai fonctions avec des boucles, tests (même en dehors des fonctions). | Type d'activité |

* 1. **Evaluation de la section**

Au choix :

* + 1. **Test**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Question** | **Réponses possibles** | **Bonne réponse** |
| **Q1** |  |  |  |
| **Q2** |  |  |  |
| **Q3** |  |  |  |
| **Q4** |  |  |  |
| **Q5** |  |  |  |
| **Q6** |  |  |  |

* + 1. **Devoir**

**Consigne pédagogique :**

**Consigne opératoire :**

* + 1. **Autre activité**

**Renseignez**

1. **Section 2**
   1. **Présentation de la section**

**Titre :**

**Image/Gif/Vidéo d’introduction :**

**Introduction/Résumé de la section :**

**Objectifs :**

**Consignes spécifiques :**

**Durée indicative :**

* 1. **Contenu**

|  |  |  |
| --- | --- | --- |
|  | **Contenu** | **Type d’activité** |
| **Activité 1** |  | Type d'activité |
| **Activité 2** |  | Type d'activité |
| **Activité 3** |  | Type d'activité |
| **Activité 4** |  | Type d'activité |
| **Activité 5** |  | Type d'activité |
| **…** |  | Type d'activité |

* 1. **Evaluation de la section**

Au choix :

* + 1. **Test**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Question** | **Réponses possibles** | **Bonne réponse** |
| **Q1** |  |  |  |
| **Q2** |  |  |  |
| **Q3** |  |  |  |
| **…** |  |  |  |

* + 1. **Devoir**

**Consigne pédagogique :**

**Consigne opératoire :**

* + 1. **Autre activité**

**Renseignez**

1. **Section 3**
   1. **Présentation de la section**

**Titre :**

**Image/Gif/Vidéo d’introduction :**

**Introduction/Résumé de la section :**

**Objectifs :**

**Consignes spécifiques :**

**Durée indicative :**

* 1. **Contenu**

|  |  |  |
| --- | --- | --- |
|  | **Contenu** | **Type d’activité** |
| **Activité 1** |  | Type d'activité |
| **Activité 2** |  | Type d'activité |
| **Activité 3** |  | Type d'activité |
| **Activité 4** |  | Type d'activité |
| **Activité 5** |  | Type d'activité |
| **…** |  | Type d'activité |

* 1. **Evaluation de la section**

Au choix :

* + 1. **Test**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Question** | **Réponses possibles** | **Bonne réponse** |
| **Q1** |  |  |  |
| **Q2** |  |  |  |
| **Q3** |  |  |  |
| **…** |  |  |  |

* + 1. **Devoir**

**Consigne pédagogique :**

**Consigne opératoire :**

* + 1. **Autre activité**

**Renseignez**

1. **Section 4**
   1. **Présentation de la section**

**Titre :**

**Image/Gif/Vidéo d’introduction :**

**Introduction/Résumé de la section :**

**Objectifs :**

**Consignes spécifiques :**

**Durée indicative :**

* 1. **Contenu**

|  |  |  |
| --- | --- | --- |
|  | **Contenu** | **Type d’activité** |
| **Activité 1** |  | Type d'activité |
| **Activité 2** |  | Type d'activité |
| **Activité 3** |  | Type d'activité |
| **Activité 4** |  | Type d'activité |
| **Activité 5** |  | Type d'activité |
| **…** |  | Type d'activité |

* 1. **Evaluation de la section**

Au choix :

* + 1. **Test**

Pour les tests intermédiaires, privilégiez les tests courts, autour de 5 questions

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Question** | **Réponses possibles** | **Bonne réponse** |
| **Q1** |  |  |  |
| **Q2** |  |  |  |
| **Q3** |  |  |  |
| **…** |  |  |  |

* + 1. **Devoir**

**Consigne pédagogique :**

**Consigne opératoire :**

* + 1. **Autre activité**

**Renseignez**

1. **Section finale**
   1. **Evaluation du module**

Pour l’évaluation du module, l’exercice final portera sur l’ensemble du contenu des différentes sections, il ne devra pas se concentrer uniquement sur une section.

Au choix :

* + 1. **Test**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Question** | **Réponses possibles** | **Bonne réponse** |
| **Q1** |  |  |  |
| **Q2** |  |  |  |
| **Q3** |  |  |  |
| **…** |  |  |  |

* + 1. **Devoir**

**Consigne pédagogique :**

**Consigne opératoire :**

* + 1. **Autre activité**

**Renseignez**

* 1. **Résumé du module et conclusion**

**Résumé et conclusion** :