**Recherches avancées sur le Css**

1. Box-sizing

La propriété CSS **box-sizing** définit la façon dont la hauteur et la largeur totale d'un élément est calculée (avec le modèle des boites).

En CSS, la largeur et la hauteur affectées à un élément s'appliquent par défaut à la boîte de contenu (content box) de l'élément. Si l'élément possède une bordure (border) ou du remplissage (Padding), celui-ci est ajouté à la largeur et/ou à la hauteur de la boîte affichée à l'écran. Cela signifie qu'il faut ajuster les valeurs de hauteur et de largeur afin qu'elles permettent d'ajouter n'importe quelle bordure ou n'importe quel remplissage qui serait ajouté par la suite.

La propriété box-sizing peut être utilisée afin d'ajuster ce comportement :

* content-box est la valeur par défaut et correspond au comportement par défaut décrit ci-avant. Si on définit un élément avec une largeur de 100 pixels, la boîte de contenu de cet élément mesurera 100 pixels de large et la largeur de la bordure et/ou du remplissage sera alors ajoutée pour constituer la largeur finalement affichée.
* border-box indique au navigateur de prendre en compte la bordure et le remplissage dans la valeur définie pour la largeur et la hauteur. Autrement dit, si on définit un élément avec une largeur de 100 pixels, ces 100 pixels incluront la bordure et le remplissage éventuellement ajoutés et c'est le contenu de la boîte qui sera compressé pour absorber cette largeur supplémentaire. Cela permet généralement de simplifier le dimensionnement des éléments.

[Valeurs](https://developer.mozilla.org/fr/docs/Web/CSS/box-sizing#valeurs)

**content-box**

C'est la valeur initiale et la valeur par défaut, spécifiée dans le standard CSS. Les propriétés [Width](https://developer.mozilla.org/fr/docs/Web/CSS/width) et [height](https://developer.mozilla.org/fr/docs/Web/CSS/height) sont mesurées en incluant le contenu, mais pas le remplissage, la bordure, la marge extérieure ou intérieure. Les dimensions de l'élément sont donc calculées à partir de la hauteur et de la largeur du contenu (on ne tient pas compte de la bordure ou du remplissage (*Padding*)). Ainsi, .box {Width: 350px; border:10px solid black;} fournira une boîte dont la largeur vaut  370px.

**border-box**

Les propriétés [width](https://developer.mozilla.org/fr/docs/Web/CSS/width) et [height](https://developer.mozilla.org/fr/docs/Web/CSS/height) incluent le contenu, le remplissage (*Padding*), la bordure, mais pas la marge. C'est le [modèle de boîte](https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/The_box_model) utilisé par Internet Explorer lorsque le document est en mode de compatibilité (Quirks). La boîte de contenu ne peut pas être négative et est ramenée à 0 si nécessaire, il est donc impossible d'utiliser border-box pour faire disparaître l'élément. Les dimensions de l'élément sont calculées comme la somme de la bordure, du remplissage (*Padding*) et du contenu.

1. Les ombres

Les ombres font partie des nouveautés récentes proposées par CSS3. Aujourd'hui, il suffit d'une seule ligne de CSS pour ajouter des ombres dans une page !

Nous allons ici découvrir deux types d'ombres :

les ombres des boîtes ;

les ombres du texte.

**box-shadow : les ombres des boîtes**

La propriété **box-shadow** s'applique à tout le bloc et prend quatre valeurs dans l'ordre suivant :

1. le décalage horizontal de l'ombre ;

2. le décalage vertical de l'ombre ;

3. l'adoucissement du dégradé ;

4. la couleur de l'ombre.

**p**{

Box-shadow: 6px 6px 0px black;

}

On peut aussi rajouter une cinquième valeur facultative : inset. Dans ce cas, l'ombre sera placée à l'intérieur du bloc, pour donner un effet enfoncé :

**p**{

Box-shadow: 6px 6px 6px black **inset**;

}

**text-shadow : l'ombre du texte**

Avec **Text-shadow**, vous pouvez ajouter une ombre directement sur les lettres de votre texte ! Les valeurs fonctionnent

exactement de la même façon que **box-shadow** : décalage, adoucissement et couleur.

p{

**text-shadow**: 2px 2px 4px black;

}

1. Opacité

La propriété **opacity** définit la transparence d'un élément. Autrement dit, elle permet de définir le degré de visibilité de l'arrière-plan sur lequel est placé l'élément. Sa valeur varie entre 0 et 1.

La valeur s'applique à l'ensemble de l'élément et à ce qu'il contient même si la valeur n'est pas héritée par les éléments fils. Ainsi, un élément et les fils qu'il contient auront tous la même opacité relative à l'arrière-plan, même si l'élément et ses descendants ont des opacités différentes. Si on souhaite utiliser différentes opacités pour les différents éléments enfants, plutôt que d'utiliser opacity, on pourra utiliser la propriété [background](https://developer.mozilla.org/fr/docs/Web/CSS/background) avec une composante alpha différente de 1 (par exemple : background: rgba(0, 0, 0, 0.4);).

1. Les angles arrondis

La propriété **border-radius** va nous permettre d'arrondir facilement les angles de n'importe quel élément. Il suffit d'indiquera taille (« l'importance ») de l'arrondi en pixels :L'arrondi se voit notamment si l'élément a des bordures.

On peut aussi préciser la forme de l'arrondi pour chaque coin. Dans ce cas, indiquez quatre valeurs :

**Code : CSS**

**P** {

**Border-**radius: 10px 5px 10px 5px;

}

Les valeurs correspondent aux angles suivants dans cet ordre :

1. en haut à gauche ;

2. en haut à droite ;

3. en bas à droite ;

4. en bas à gauche.

1. Les pseudo classes

Les pseudo- classes vont nous permettre d’appliquer des styles à des éléments HTML uniquement lorsque ceux-ci sont dans un certain état (cliqués, activés, etc.).

Les pseudo-classes vont nous permettre de cibler des éléments HTML en fonction de leur état, ou plus exactement d’appliquer des règles CSS à des éléments HTML uniquement dans un certain contexte (lorsqu’ils sont dans un certain état).

Les pseudo-classes : :hover , :visited, :active, :Link.

La pseudo-classe :hover va nous permettre d’appliquer des styles à un élément HTML uniquement lorsque celui-ci est survolé par la souris d’un utilisateur. Cette pseudo-classe peut être utilisée avec la majorité des éléments HTML mais on l’utilisera généralement pour appliquer des styles différents à des liens lorsqu’un utilisateur passe sa souris dessus.

La pseudo-classe CSS :visited va nous permettre d’appliquer des styles à un élément déjà visité, c’est-à-dire déjà cliqué. En pratique, cette pseudo-classe va une nouvelle fois surtout être utilisée avec des éléments de lien. Ainsi, on va pouvoir changer l’apparence d’un lien après que celui-ci ait été cliqué, lorsque l’utilisateur revient sur notre page de départ.

La pseudo-classe CSS :active va elle nous permettre de modifier les styles d’un élément HTML lors d’un état très particulier qui est le moment où l’on clique sur l’élément. Pour bien visualiser cet état, je vous conseille de rester cliqué sur l’élément en question le temps de voir les changements de style. Une nouvelle fois, en pratique, cette pseudo-classe va généralement être utilisée pour modifier l’apparence des liens au moment du clic.

La pseudo-classe :link va elle nous permettre au contraire de cibler tous les liens non visités et de leur appliquer des styles particuliers en CSS.

Tableau récapitulatif des pseudo-classes

| **Sélecteur** | **Description** | **Version CSS** |
| --- | --- | --- |
| E:link | Sélectionne tout élément E représentant l’ancre d’un lien non visité jusqu’à présent | CSS1 |
| E:visited | Sélectionne tout élément E représentant l’ancre d’un lien déjà visité | CSS1 |
| E:active | Sélectionne un élément E au moment où il est cliqué | CSS1 |
| E:hover | Sélectionne un élément E lorsque le curseur de la souris passe dessus | CSS2 |
| E:focus | Sélectionne un élément E qui a le focus (dans lequel le curseur de la souris est placé) | CSS2 |
| E:first-child | Sélectionne tout élément E étant le premier enfant de son parent | CSS2 |
| E:lang(fr) | Sélectionne tout élément E dont l’attribut langage possède la valeur « fr » | CSS2 |
|  |  |  |
| E:target | Sélectionne un élément E contenant une ancre qui vient d’être cliquée à partir d’un lien ancre | CSS3 |
| E:enabled | Sélectionne tout élément E avec lequel l’utilisateur peut interagir et qui est activé | CSS3 |
| E:disabled | Sélectionne tout élément E avec lequel l’utilisateur peut interagir et qui est desactivé | CSS3 |
| E:root | Sélectionne un élément E racine du document | CSS3 |
| E:empty | Sélectionne tout élément E qui ne possède pas d’enfant (ni de noeud de type texte) | CSS3 |
| E:nth-child(n) | Sélectionne tout élément E étant le n-ième enfant de son parent | CSS3 |
| E:nth-last-child(n) | Sélectionne tout élément E étant le n-ième enfant de son parent en comptant les enfants à partir du dernier | CSS3 |
| E:checked | Sélectionne tout élément E de type input coché au sens large (checked ou selected) | CSS3 |
| E:last-child | Sélectionne tout élément E étant le dernier enfant de son parent | CSS3 |
| E:only-child | Sélectionne tout élément E qui est le seul enfant de son parent | CSS3 |
| E:nth-of-type(n) | Sélectionne tout élément E étant le n-ième enfant d’un certain type par rapport à son parent | CSS3 |
| E:nth-last-of-type(n) | Sélectionne tout élément E étant le n-ième enfant d’un certain type par rapport à son parent en comptant à partir de la fin | CSS3 |
| E:first-of-type | Sélectionne tout élément E premier enfant de son type par rapport à son parent | CSS3 |
| E:last-of-type | Sélectionne tout élément E dernier enfant de son type par rapport à son parent | CSS3 |
| E:only-of-type | Sélectionne tout élément E seul enfant de son type par rapport à son parent | CSS3 |
| E:read-write | Sélectionne tout élément E de type input avec lequel l’utilisateur peut interagir (comme un champ dans lequel il peut écrire par exemple) | 3-UI/4 |
| E:read-only | Sélectionne tout élément E de type input avec lequel l’utilisateur ne peut pas interagir (éléments possédant un attribut disabled par exemple) | 3-UI/4 |
| E:placeholder-shown | Sélectionne tout élément E qui affiche actuellement la valeur de son attribut placeholder | 3-UI/4 |
| E:default | Sélectionne un élément E dans une liste ou un groupe qui est l’élément défini par défaut | 3-UI/4 |
| E:valid | Sélectionne tout élément E de type input dont la valeur est évaluée comme valide (dont la valeur possède une forme correspondant à ce qui est attendu) | 3-UI/4 |
| E:invalid | Sélectionne tout élément E de type input dont la valeur est évaluée comme invalide (valeur ne correspondant pas à ce qui est attendu) | 3-UI/4 |
| E:in-range | Sélectionne tout élément E de type input dont la valeur fournie se situe dans une fourchette de valeurs prédéfinies | 3-UI/4 |
| E:out-of-range | Sélectionne tout élément E de type input dont la valeur fournie se situe en dehors d’une certaine fourchette de valeurs prédéfinies | 3-UI/4 |
| E:required | Sélectionne tout élément E de type input dont la valeur doit être renseignée (élément possédant un attribut required) | 3-UI/4 |
| E:optional | Sélectionne tout élément E de type input dont la valeur ne doit pas obligatoirement être renseignée | 3-UI/4 |
| E:not(E1, .c1) | Sélectionne tout élément E qui n’est pas de type E1 et qui ne possède pas d’attribut class= »c1″ | 3/4 |

1. Les Transitions

Pour contrôler les modification de style ou les transformations Etat de départ vers un état d'arrivée avec un timing.les transitions en css permettent de passer d’un etat a un autre.

*Transition :* regroupe plusieurs paramètres :

● *transition-property :* la(les) propriété(s) CSS à modifier.

● *transition-duration :* durée de la transition (s ou ms)

● *transition-timing-function :* méthode d’interpolation

● *transition-delay :* durée avant le début de la transition

1. Les animations

Enchaînement de plusieurs transitions. Elles Consistent à modifier

Automatiquement les propriétés CSS et à appliquer des transformations 2D ou 3D selon un scénario. Du a la recense des animations les navigateurs ont du mal a la supporter ainsi on utilise alors des préfixes vendeurs pour forcer leurs compatibilités.

Pour créer une animation en css nous devons utiliser la règle @keyframes avec la propriété css animation.

La propriété animation est la notation short-end de differentes propriétés relatives aux animations notamment :

*animation-name* : nom de l’animation à utiliser.

*animation-duration* : durée totale de l’animation.

*animation-timing-function* : méthode d’interpolation

*animation-itération-count :* nombre de répétition de l’animation.

Utilisez *infinité* pour une animation en continu.

*animation-direction :* permet de jouer une animation en sens inverse

*animation-Play-state :* mettre en pause l’animation.

*animation-delay :* Temps avant que l’animation ne démarre.

*animation-fill-mode :* conserver l’état de l’animation

1. Transformations

Le CSS va également nous permettre d’appliquer des transformations à nos éléments : on va pouvoir incliner nos éléments, les déformer, les translater, etc.

Cette propriété va accepter différents mots clefs ou plus exactement différentes fonctions qui vont définir le type de transformation qui va être appliqué à un élément.

Les transformations 2D

* *rotate (angle)* rotation
* *scale (x,y), scale X(x) et scale Y(y)* mise à l’échelle
* *skewX (angle) et skewY (angle)* pour incliner un élément
* *translate(x,y), translateX(x) et translateY(y)* translation
* matrix (a, c, b, d, x, y) matrice de transformation

Les transformations 3D

● RotateX (angle), rotateY (angle) ET rotateZ (angle)

● rotate3d(x, y, z, angle) pour une rotation autour du vecteur 3D x,y,z.

● translatez (z) pour des translations sur l’axe Z (profondeur).

● translate3d(x, y, z) pour des translations en 3D.

● ScaleZ (z) pour des changements d’échelle sur l’axe Z

● scale3d(x, y, z) pour des changements d’échelle en 3D.

● matrix3d () spécifie une matrice de transformation 4x4.

● perspective distance entre l'observateur et la scène

● perspective-origin point de fuite des éléments.

● backface-visibility: visibilité des faces arrières.

1. Les pseudo-éléments

Les pseudo-éléments CSS vont nous permettre de ne cibler qu’une partie d’un élément HTML pour lui appliquer des styles. Il convient de ne pas les confondre avec les pseudo-classes qui elles servent à cibler un élément selon son état.

Les pseudo-éléments ont une syntaxe particulière puisqu’il faudra utiliser un double deux-points :: entre le sélecteur élément et le mot clef désignant la partie de l’élément qui va être ciblé.

Il existe aujourd’hui 4 pseudo-éléments qui sont des recommandations du W3C :

* ::first-letter ;
* ::first-line ;
* ::before ;
* ::after.

Il existe également un cinquième pseudo-élément ::selection qu’il convient de connaitre car celui-ci devait faire partie des recommandations du CSS3 puis a finalement été abandonné et est de nouveau candidat aujourd’hui.

Les pseudo-éléments ::before et ::after vont nous permettre respectivement de cibler l’emplacement avant et après le contenu d’un élément HTML.

Le pseudo-élément ::first-letter va nous permettre de sélectionner uniquement la première lettre du contenu d’éléments HTML.

De manière analogue à ::first-letter, nous allons pouvoir sélectionner uniquement la première ligne d’un élément HTML grâce au pseudo-élément ::first-line pour lui appliquer des styles.

Finalement, nous allons pouvoir appliquer des styles particuliers à certaines parties de nos éléments HTML en fonction de ce qui est déjà présélectionné par l’utilisateur grâce au pseudo-élément ::selection.

1. Les sélecteurs avances

Le CSS va nous permettre de mettre en forme nos contenus HTML en appliquant des styles aux différents éléments. Cependant, pour appliquer un style particulier à un ou plusieurs éléments HTML en CSS, il va avant tout falloir les cibler, c’est-à-dire indiquer avec précision à quels éléments doivent s’appliquer les styles créés en CSS.

Les sélecteurs de type « éléments HTML » sont appelés des sélecteurs CSS simples car il suffit d’indiquer le nom d’un élément pour lui attribuer des styles.

En revanche, l’utilisation de caractères spéciaux pour naviguer dans notre page HTML et cibler un élément en particulier va créer ce qu’on appelle des sélecteurs CSS complexes car un sélecteur complexe va être composé de différents sélecteurs juxtaposés.

**Liste complète de tous les sélecteurs CSS**

|  |  |  |
| --- | --- | --- |
| E | Sélectionne tous les élément de type E | 1 |
| E .maclasse | Sélectionne les éléments E possédant un attribut class="maclasse" | 1 |
| E #monid | Sélectionne un élément E possédant un attribut id="monid" | 1 |
| E:link | Sélectionne tout élément E représentant l’ancre d’un lien non visité jusqu’à présent | 1 |
| E:visited | Sélectionne tout élément E représentant l’ancre d’un lien déjà visité | 1 |
| E, F | Sélectionne tous les éléments de type E et de type F | 1 |
| E F | Sélectionne tous les éléments F à l’intérieur des éléments E | 1 |
| E:active | Sélectionne un élément E qui est actuellement cliqué | 1 |
| E::first-letter | Sélectionne la première lettre de tout élément E | 1 |
| E::first-line | Sélectionne la première ligne (dans le résultat final) de tout élément E | 1 |
| \* | Sélectionne tous les éléments | 2 |
| E > F | Sélectionne les éléments F enfants directs des éléments E | 2 |
| E + F | Sélectionne tout élément F placé directement après un élément E | 2 |
| E[foo] | Sélectionne tout élément E possédant un attribut foo | 2 |
| E[foo="bar"] | Sélectionne tout élément E possédant un attribut foo dont la valeur est exactement « bar » | 2 |
| E[foo~="bar"] | Sélectionne tout élément E possédant un attribut foo dont la valeur contient distinctement « bar » (c’est-à-dire dont la valeur contient le mot « bar » séparé du reste par des espaces) | 2 |
| E[foo|="en"] | Sélectionne tout élément E possédant un attribut foo dont la valeur commence par « en » séparé du reste par un tiret (ou hyphen en anglais) | 2 |
| E:hover | Sélectionne un élément E qui se situe sous le curseur de la souris | 2 |
| E:focus | Sélectionne un élément E qui a le focus | 2 |
| E:first-child | Sélectionne tout élément E étant le premier enfant de son parent | 2 |
| E:lang(fr) | Sélectionne tout élément E dont l’attribut langage possède la valeur « fr » | 2 |
| E::after | Ce sélecteur permet d’intégrer du contenu après un élément E | 2 |
| E::before | Ce sélecteur permet d’intégrer du contenu avant un élément E | 2 |
| E ~ F | Sélectionne tout élément F placé après un élément E dans la page | 3 |
| E[foo^="bar"] | Sélectionne tout élément E possédant un attribut foo dont la valeur commence exactement par « bar » | 3 |
| E[foo$="bar"] | Sélectionne tout élément E possédant un attribut foo dont la valeur se termine exactement par « bar » | 3 |
| E[foo\*="bar"] | Sélectionne tout élément E possédant un attribut foo dont la valeur contient la valeur « bar » | 3 |
| E:target | Sélectionne un élément E contenant une ancre qui vient d’être cliquée à partir d’un lien ancre | 3 |
| E:enabled | Sélectionne tout élément E avec lequel l’utilisateur peut interagir et qui est activé | 3 |
| E:disabled | Sélectionne tout élément E avec lequel l’utilisateur peut interagir et qui est desactivé | 3 |
| E:root | Sélectionne un élément E racine du document | 3 |
| E:empty | Sélectionne tout élément E qui ne possède pas d’enfant (ni de noeud de type texte) | 3 |
| E:nth-child(n) | Sélectionne tout élément E étant le n-ième enfant de son parent | 3 |
| E:nth-last-child(n) | Sélectionne tout élément E étant le n-ième enfant de son parent en comptant les enfants à partir du dernier | 3 |
| E:checked | Sélectionne tout élément E de type input coché au sens large (checked ou selected) | 3 |
| E:last-child | Sélectionne tout élément E étant le dernier enfant de son parent | 3 |
| E:only-child | Sélectionne tout élément E qui est le seul enfant de son parent | 3 |
| E:nth-of-type(n) | Sélectionne tout élément E étant le n-ième enfant d’un certain type par rapport à son parent | 3 |
| E:nth-last-of-type(n) | Sélectionne tout élément E étant le n-ième enfant d’un certain type par rapport à son parent en comptant à partir de la fin | 3 |
| E:first-of-type | Sélectionne tout élément E premier enfant de son type par rapport à son parent | 3 |
| E:last-of-type | Sélectionne tout élément E dernier enfant de son type par rapport à son parent | 3 |
| E:only-of-type | Sélectionne tout élément E seul enfant de son type par rapport à son parent | 3 |
| E:read-write | Sélectionne tout élément E de type input avec lequel l’utilisateur peut interagir (comme un champ dans lequel il peut écrire par exemple) | 3-UI/4 |
| E:read-only | Sélectionne tout élément E de type input avec lequel l’utilisateur ne peut pas interagir (éléments possédant un attribut disabled par exemple) | 3-UI/4 |
| E:placeholder-shown | Sélectionne tout élément E qui affiche actuellement la valeur de son attribut placeholder | 3-UI/4 |
| E:default | Sélectionne un élément E dans une liste ou un groupe qui est l’élément défini par défaut | 3-UI/4 |
| E:valid | Sélectionne tout élément E de type input dont la valeur est évaluée comme valide (dont la valeur possède une forme correspondant à ce qui est attendu) | 3-UI/4 |
| E:invalid | Sélectionne tout élément E de type input dont la valeur est évaluée comme invalide (valeur ne correspondant pas à ce qui est attendu) | 3-UI/4 |
| E:in-range | Sélectionne tout élément E de type input dont la valeur fournie se situe dans une fourchette de valeurs prédéfinies | 3-UI/4 |
| E:out-of-range | Sélectionne tout élément E de type input dont la valeur fournie se situe en dehors d’une certaine fourchette de valeurs prédéfinies | 3-UI/4 |
| E:required | Sélectionne tout élément E de type input dont la valeur doit être renseignée (élément possédant un attribut required) | 3-UI/4 |
| E:optional | Sélectionne tout élément E de type input dont la valeur ne doit pas obligatoirement être renseignée | 3-UI/4 |
| E:not(E1, .c1) | Sélectionne tout élément E qui n’est pas de type E1 et qui ne possède pas d’attribut class= »c1″ | 3/4 |
| E::selection | Permet de sélectionner une partie active d’un élément E déjà sélectionnée par l’utilisateur | 3/4 |
| E >> F | Sélectionne tous les éléments F à l’intérieur des éléments E. Ce sélecteur est strictement équivalent au précédent mais représente la nouvelle syntaxe de celui-ci qui sera rendue officielle avec le CSS4. | 4 |
| F || E | Sélectionne un élément E qui représente une cellule dans un tableau ou dans une grille appartenant à une colonne représentée par un élément F | 4 |
| E[foo="bar" i] | Sélectionne tout élément E possédant un attribut foo dont la valeur est exactement « bar » sans tenir compte de la casse (l’utilisation de majuscules / minuscules). Le « i » représente ici « case insensitive » c’est-à-dire « insensible à la casse » en français | 4 |
| E:any-link | Sélectionne tout élément E étant l’ancre d’un lien, que celui-ci ait été cliqué ou non. Utiliser cette pseudo classe est équivalent à utiliser :link et :visited | 4 |
| E:scope | Sélectionne un élément E étant l’élément de référence désigné. | 4 |
| E:current | Sélectionne un élément E actuellement affiché ou lu au sein d’un canvas comportant une dimension temporelle ou l’un de ses parents | 4 |
| E:current() | Alternative à E:current. On va pouvoir passer une liste de sélecteurs en argument de :current(). L’élément sélectionné sera l’élément :current qui correspond au(x) sélecteur(s) passé(s) | 4 |
| E:past | Sélectionne un élément E dont l’affichage précède immédiatement celui de l’élément :current | 4 |
| E:future | Sélectionne un élément E dont l’affichage suit immédiatement celui de l’élément :current | 4 |
| E:drop | Sélectionne un élément E qui peut potentiellement recevoir un objet HTML dans le cadre d’un glissé-déposé (drag and drop) | 4 |
| E:drop(active) | Version alternative de E:drop. Sélectionne l’élément E qui est actuellement la cible d’un drag and drop pour l’élément que l’on est en train de faire glisser | 4 |
| E:drop(valid) | Version alternative de E:drop. Sélectionne l’élément E qui est actuellement la cible d’un drag and drop pour l’élément que l’on est en train de faire glisser uniquement si cet élément peut recevoir l’objet qui va y être déposé | 4 |
| E:drop(invalid) | Sélectionne tout élément E qui ne peut pas recevoir l’objet actuellement glissé mais qui pourrait recevoir un autre type d’objet | 4 |
| E:indeterminate | Sélectionne tout élément E de type input qui est dans un état indéterminé, c’est-à-dire ni coché ni non-coché | 4 |
| E:user-error | Sélectionne tout élément E de type input altéré par l’utilisateur et dont la valeur passée est invalide | 4 |
| E:blank | Sélectionne tout élément E ne possédant pas de contenu à part éventuellement des espaces | 4 |
| E:nth-column(n) | Sélectionne tout élément E représentant une cellule appartenant à la n-ième colonne d’un tableau ou d’une grille | 4 |
| E:nth-last-column(n) | Sélectionne tout élément E représentant une cellule appartenant à la n-ième colonne d’un tableau ou d’une grille en comptant à partir de la fin | 4 |
| :playing | Sélectionne tout média capable d’être joué (audio, vidéo…) lorsque l’élément en question est en train d’être joué (ou lu) | 4 |
| :paused | Sélectionne tout média capable d’être joué (audio, vidéo…) lorsque l’élément en question n’est pas en cours de lecture (en pause, en cours de chargement etc.) | 4 |
| E:matches(E1, .c1) | Sélectionne tout élément E qui est soit de type E1 ou qui possède un attribut class= »c1″ | 4 |
| E:has(E1, .c1) | Sélectionne un élément E si au moins un des sélecteurs passés en paramètre correspond à l’élément (selon la portée, :scope, de l’élément) | 4 |
| E:dir(ltr) | Sélectionne tout élément E dont on à spécifié une direction de lecture « de gauche à droite » (la direction est à priori déterminée par la langue déclarée dans le document. Par exemple, pour le français, ce sera de gauche à droite par défaut) | 4 |
| E::inactive-selection | Permet de sélectionner une partie inactive d’un élément E déjà sélectionnée par l’utilisateur | 4 |
| E::spelling-error | Sélectionne une partie d’un élément E qui a été identifiée par le user agent comme mal orthographiée | 4 |
| E::grammar-error | Sélectionne une partie d’un élément E qui a été identifiée par le user agent comme grammaticalement incorrecte | 4 |
| E::marker | Permet de sélectionner le marqueur de tout élément E dont le type d’affichage est display : list-item (par exemple la puce ou le numéro de l’élément d’un élément li) | 4 |
| E::placeholder | Sélectionne le texte de l’attribut placeholder d’un élément E de type input | 4 |

1. L’approche Normalizing css et resetting css

## Reset

L'objectif de cette approche est de contrer l'apparence que le navigateur impose par défaut à la plupart des élément HTML. Ceci permet ensuite, lorsque l'on travaille sur le CSS, de ne pas avoir à penser aux styles qui pourraient être appliqués défaut aux éléments. Si par exemple vous souhaitez styliser un titre, vous ne serez pas obligés de retirer les marges avant de travailler. il existe plusieurs feuilles de styles vous proposant une réinitialisation :

## Normalize

Cet outil propose une approche différente qui consiste à essayer de normaliser l'apparence des éléments HTML à travers les différents navigateurs. L'objectif est d'avoir le comportement le plus prévisible possible. Cette feuille de style va redéfinir certaines propriétés (par exemple spécifier les marges sur les titres, paragraphes ou autre) afin d'offrir une expérience consistante pour les utilisateurs quel que soit le navigateur utilisé.

Jocelyn yemaleu