|  |  |  |
| --- | --- | --- |
| Généralité  * Les règle de style sont appliqué à tous les enfants (héritage) * La règle la plus spécifique l’emporte (ex : choisir une classe sera plus spécifique qu’une balise de type) * La dernière règle de style l’emporte  SélecteurSélecteur simpleSélecteur de type *nomElement*Sélecteur de classe *.nomClasse*Sélecteur d’identifiant *#nomID*Sélecteur universel *\*, ns|\*, \*|\*, |\**  * Sélecteur d’attribut *[attr=valeur]*  Sélecteur de groupe (= et)  * Sélecteur de conjonction *A, B*  Sélecteur Combinateurs  * Combinateur de voisin direct A + B   Sélectionne les premier élmts B qui sont **directement** après A   * Combinateur de voisin général A ~ B   Sélectionne les élément B qui sont situé après A   * Combinateur d’enfant A > B   Sélectionne B qui est un enfant direct de A   * Combinateur de descendant A B   Sélectionne tous les B descendant de A (pas nécessairement enfant direct) Pseudo-classes Ex : :hover, :visited, :first-child, :focus, :nth-child(xn + y) tous les xn à partir de y (y de base vaut xn) ,. Pseudo-éléments Ex : ::after, ::before, ::first-letter, ::first-line, etc. | Modèle de boiteStyle des boites  * Border-radius : 50% // gerent l’arrondis des angles * Box-shadow : 6px 6px 0px black ; // ombre des boites * Border : 1px solid black ; // bordure * Overflow : hidden ; // cache ce qui dépasse du bloc  Display  * Bloc : les éléments se place les un en dessous des autres * Inline : les éléments se place les un a côté des autres * Inline-bloc, comme inline mais peut être dimensionné  Box-sizing Permet de définir comment est définie la taille de la box :   * Border-box : prend en compte les bordure + padding * Content-box : ne prend en compte que la bordure (défaut)  Police & textes  * Color : définie la couleur du texte * Font-size : définie la taille * Line-height : hauteur de ligne * Letter-spacing : espace entre les lettres * Font-family : police (ont peut en mettre plusieurs pour assurer la compatibilité, serif (texte) ou sans-serif (titre) à la fin * text-align : alignement, ex : justify (ne marche que sur les éléments inline et inline-bloc) * hyphens et word-break : césure des mots * text-decoration : none, underline, blink, etc… * font-weight : bold, normal etc. * font-style : italic, oblique, normal etc. * font-display : swap | optionnal ; (gère chargement police) | Couleur & opacité  * color : rgba(30, 30, 30, 0.6) (ou 0.6 représente l’opacité) * color : #009900aa (ou aa représente l’opacité)  Les unités de mesuresValeur absolu Le pixel est une valeur absolu de dimension, avec les écran haute résolution et les mobile ce n’est pas forcément la meilleur solution pour des design adaptatif Valeur relatif  * Pourcentage, dépendra de la largeur du conteneur * Vw et vh, est un pourcentage de l’espace visible * Em, définie une taille relative par rapport par rapport à la taille de police de l’élément parent * Rem, idem a em mais par rapport a la taille définie dans body  Le positionnement  * Position : relative ;   Comme le static mais permet de déplacer l’élément   * Position : absolute ;   se positionne de façon absolu par rapport à la page, attention cette position fait sortir le conteneur du flux   * Position : fixed ;   Restera toujours à la même position sur la page   * Position : static ;   Valeur par defaut, ne peut etre déplacer   * Position : sticky ;   Comme static jusqu’au moment ou il sort du cadre, il sera alors comme en fixed la ou le conteneur parent va « buter »   * Z-index : 1 ;   Définie la place dans l’espace z des éléments (profondeur)  On peut placer un élément en absolute dans un conteneur qui est en relative, sa position ne se fera plus relativement à la page entière mais a son conteneur |

|  |  |  |
| --- | --- | --- |
| Mise en pageFloat L’élément sera placé sur le côté et le reste du flux entoure l’élément flottant.   * Float : left|right ; * Clear : both ; // à placer sur le conteneur qui doit retourner à la ligne   Un élément flottant peu perturbé l’affichage de son conteneur, on utilisera alors la propriété overflow :hidden ; ou encore une position autre que static (par défaut) FlexBox *Idéal pour mettre en page des modules* Sur le conteneur  * Display : flex ; // A mettre sur le conteneur principal * Flex-wrap : wrap ; Les éléments qui sorte vont à la ligne * Flex-direction : row | column | row-reverse ; * Flex-flow : row wrap ; // combine wrap et direction * Justify-content : flex-end ; // axe principal * Align-items : flex-start ; // axe secondaire * Align-content : flex-start ; // alignement entre n ligne  Sur les éléments  * Flex-grow :10 ;//augmente l’élément par rapport au autre * Flex-shrink : 10 ; // reduit l’élément par rapport au autre * Flex-basis : 50px ; // l’elmts fera 50px * Flex : 0 0 0 ; / regroupe grow, shrink et basis * Align-self : flex-end ; // align un élément particulier * Order : 2 ; // permet de changer l’ordre | GridSur le conteneur  * Display : grid ; // sur mon conteneur principal * Grid-template-columns : 1fr 5fr 1fr ; | grid-template-rows   Ici ont créer 3 colonnes, le fr représente une fraction. On peut également utiliser les % ou les px ou encore minmax(x,y). On créer 12 colonne qui auront 1fr en taille avec repeat(12,1fr),   * Grid-template-areas :   « header header »  « sidebar body » ;  Permet de facilement gérer les zones, Il suffira alors dans les éléments d’ajouter grid-area : body| header etc.   * Grid-column-gap : 10% ;   Gèrent les espacements entre les colonnes   * Grid-auto-flow : column | row ;   Définie si les nouveaux éléments rentrent d’abord colonne par colonne ou ligne par ligne   * Grid-auto-rows : 100px | minmax (100px, auto) ;   Si des nouvelles lignes sont créer, définie leurs valeurs par défaut, fonctionne également avec les columns.   * Justify-items  et align-items pour disposition par défaut  Sur les éléments *Par défaut les éléments prennent une « case » de la grille*   * Grid-column-start : 2 ;   Indique que l’élément doit commencer à la 2em position sur la grille, on peut aussi utiliser grid-row-start.   * Grid-column-end : 4 | span 2 ;   S’arretera à la 4em colonne, on peut aussi dire span 2 ce qui veut dire qu’elle s’étendra de 2colonne à partir de sa position de départ   * Grid-column : 1 / 3 | 1 / span 2 ;   Combine le start et end, ici démarre à la colonne 1 et fin a 3   * Justify-self  et align-self pour disposition dans la case | Background  * Background-color : rgb | #FFF | etc ; * Background-image : url(«chemin.jpg ») ; * Background-image : linear-gradient(to bottom, #000, rgba(0,0,0,0.5) ;   Créer un dégradé linéaire, on peut aussi utiliser le radial-gradient.   * Background-repeat : repeat-y | repeat | no-repeat ; * Background-position : left | center | right 35% top 10px ;   On peut indiquer 2 valeurs, d’abord x (horizontal) puis y   * Background-size : cover | contain | 30% | 200px 150px; * Background-attachment : scroll | fixed | local | scroll, local;   On peut utiliser plusieurs valeurs si on a plusieurs background   * Background-clip : text ; color : transparent ;   Permet de faire une transparence sur les lettre pour qu’elle laisse apparaitre le background   * Background : url(« chemin.png) no-repeat ;   Combine toute les propriétés possibles de background Media Querry *Permet de créer des éléments responsifs, qui s’adapte en fonction notamment de la définition de l’écran du client*   * @media screen and(max-widht : 1280px)   Ciblera les écran d’une définition maximale de 1280px   * @media all and(min-widht :1020px) and (max-widht : 1280px) ;   Ciblera tout les ecran entre 1020 au mini et 1280px au max Transformation, animation et transitionTransformation 2D  * Transform : rotate(10deg) // rotation de 10degré   Vautre valeur : scale(1.5) | translate(30px, 30px) | translateX   * Transform-origin : top left ; // modifie l’origine de rotation du transform |