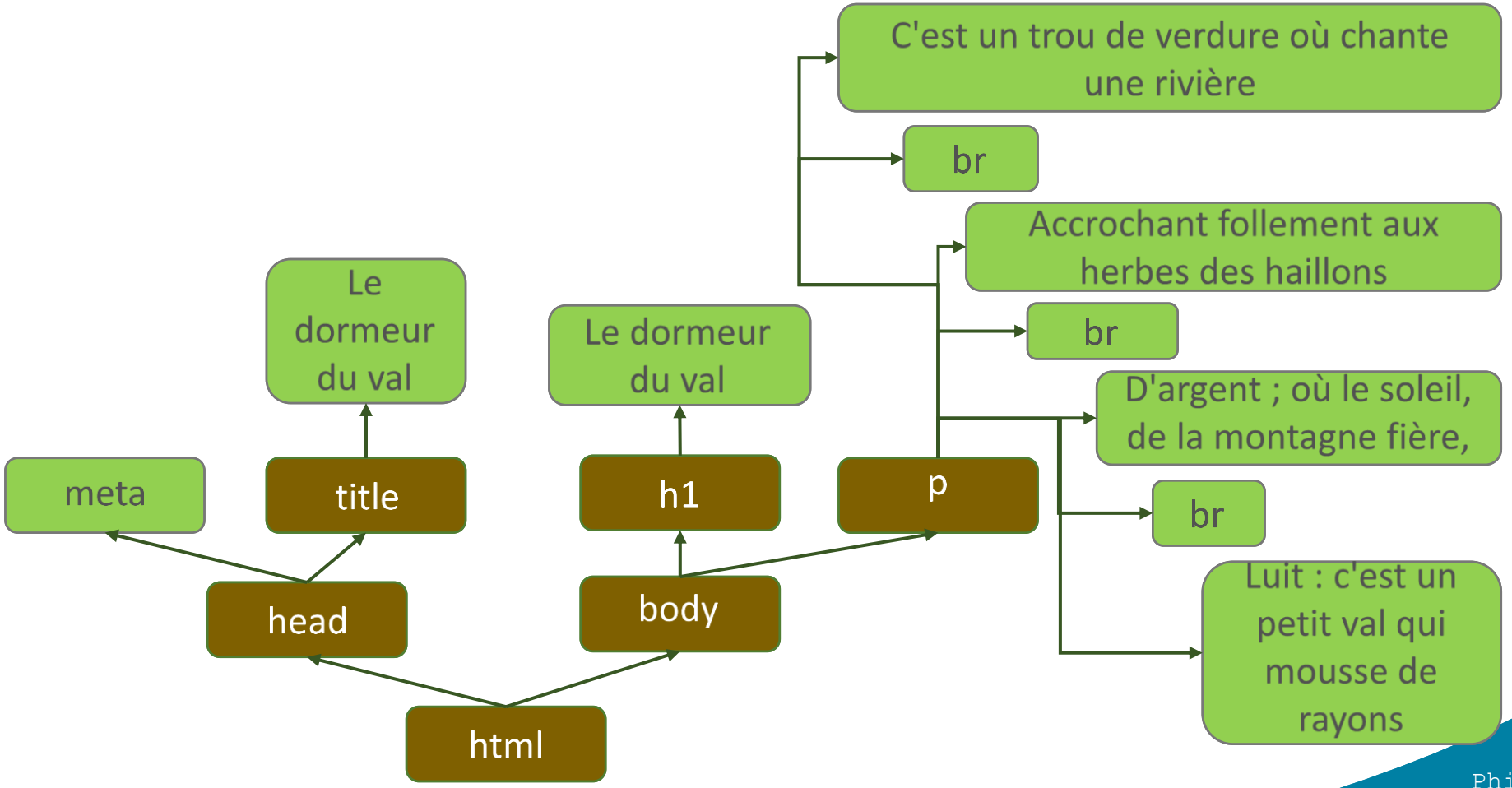
CSS: mise en page

Restructurer une page

Ressources: spécifications CSS, c.f.
http://www.w3.org/Style/CSS/current-work

Rappel : affichage par défaut

Arbre : 1re strophe du « dormeur du val »

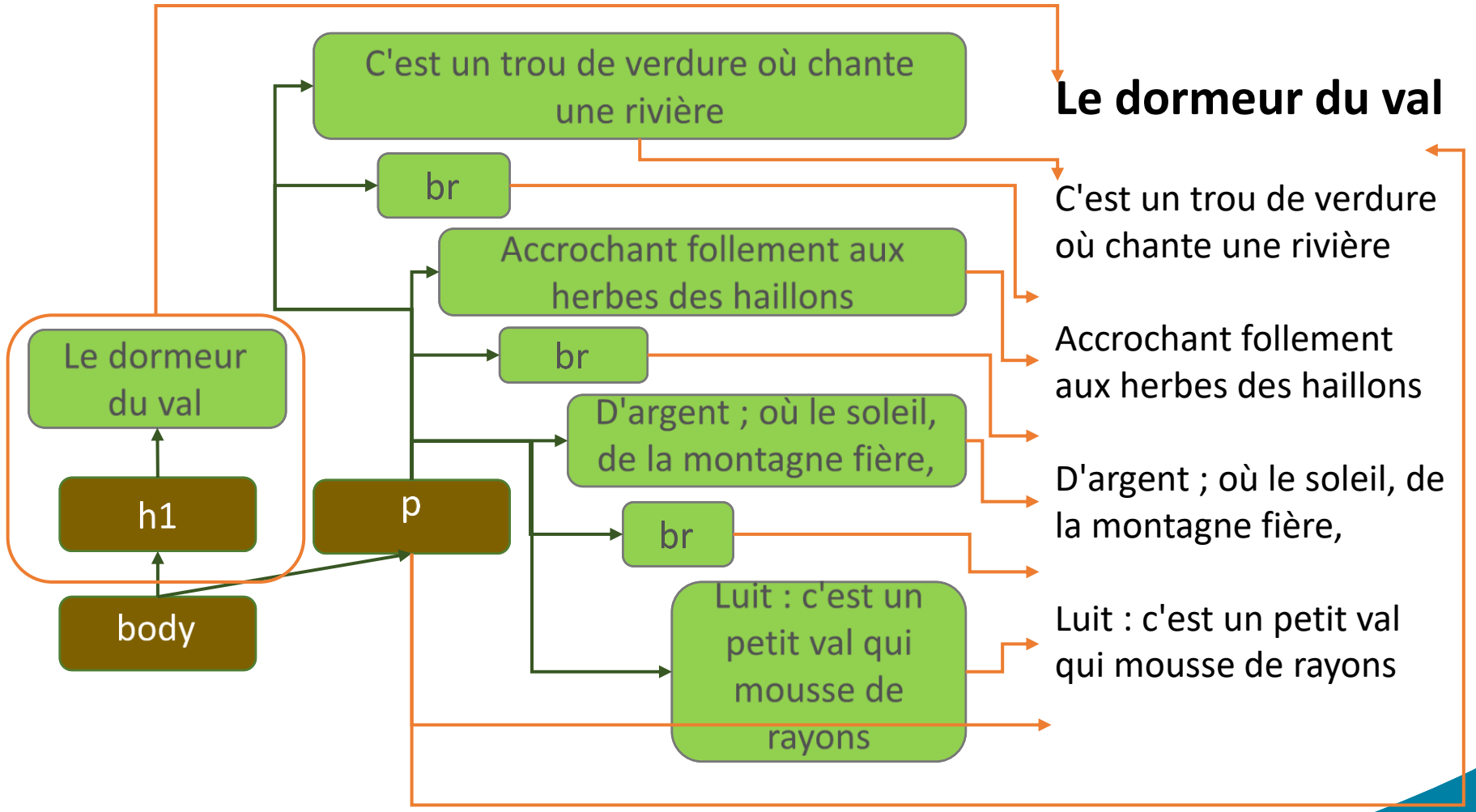


Arbre : représentation et structure

- L'arbre est une représentation pour un humain
- La structure est utilisée par les programmes (navigateur)
  - Pour rechercher du contenu
  - Calculer le rendu

- Pour du html sans css, le rendu se fait en lisant l'arbre (racine en bas) de bas en haut et de gauche à droite
  - On finit une branche avant de passer à la suivante (parcours en profondeur)
  - Un élément est « par-dessus » les éléments affichés avant lui, mais c'est sans conséquence car il n'y a pas de chevauchement sans CSS

De l'arbre vers le rendu (linéaire)



SNT

Comment modifier l’affichage

Les propriétés sont « nommées » ici, le détail est donné après

Philippe Renevier-Gonin

Web : css

Avec CSS, on peut modifier le rendu exemple : un menu

- Donner des dimensions souhaitées (et non celles par défaut)
  - **width** (largeur – dimension horizontale)
  - **height** (hauteur – dimension verticale)
- Cacher ou montrer ou mettre des *scrollbars* (barres de défilement) ce qui dépasse (puisqu'on a redimensionné)
  - **overflow**
  - **opacity**
- Animer (en cas de changement)
  - Pseudo classe **:hover**
  - **transition**
- Mettre un « encadré » sur la droite ou sur la gauche
  - **float** (avec margin)
- Illustration : https://dabblet.com/gist/e3b0d010c2613502237 7a8803452b42b
- Notons : pour modifier qui s'affiche par-dessus qui il faut :
  - **position** et **z-index**

Avec CSS, on peut modifier le rendu exemple : changer la façon dont c'est affiché

- Mettre un élément sur le côté
  - float
- Un morceau de phrase (*inline* - ex: a) s'affiche comme un paragraphe (*block* - ex: p, div)
  - display
  - display a d'autre usage, avec les valeurs flex et grid
- Déplacer un élément
  - position
  - Valeur : static (on ne change rien, c'est la valeur par défaut)
  - Valeur : relative (pour un changement local)
  - Valeur : fixed / absolute pour un positionnement général
  - Valeur : absolute dans un fixed / relative / absolute : pour position un élément dans un élément
  - Fonctionne avec top | right | bottom | left

display

- Indique le type d'affiche de l'élément

- S'applique à tous les éléments
- Valeur par défaut : inline
- Pas d'héritage
- Cette propriété n'est pas prise en compte dans les animations CSS

- Valeurs possibles, 1 parmi :
  - inline — comme un morceau de phrase
  - block — comme un "block" (p, h1, etc.)
  - inline-block — à l'intérieur comme un block, à l'extérieur traité comme un inline
  - list-item — comme un item de liste
  - none — l'élément n'apparait pas, ne prend pas de place, comme s'il n'existait pas
  - Etc.
- Pour render invisible mais "present" (il prend de la place) dans la page : "visibility : hidden;"

Impact de l'utilisation de display

- Ouvre des propriétés non possibles à des propriétés qui ne s'appliquaient pas (changement de catégorie)
- display: inline-block sur des « blocks »
  => ils s'aligneront sur une ligne (c.f. vertical-align, normalement pas pour les blocks), en gardant leur structure interne
- display: block sur des « inlines »
  => change complétement l'organisation initiale
- http://dabblet.com/gist/006222ff322fb02ed96e

Position

- Couplé avec top, left, right, bottom
  - Mais aussi z-index
  - Mais aussi width, height
- relative , absolute (peut disparaître en scrollant) , fixed (toujours visible)
- Le plus
  - Un élément B en position: absolute
  - Dans un élément A en position: {relative , absolute ou fixed }
  - Placement de B dans A et non plus par rapport au document
  - Permet de faire des hiérarchies contenant / contenu et de déplacer des blocks
- http://dabblet.com/gist/65e5c53205592ae46085

Avec CSS, on peut modifier le rendu exemple : faire des alignements adaptatifs

- display: flex
- En ligne : https://dabblet.com/gist/fc3daa5052f4514a44e3
  - Variante : https://dabblet.com/gist/011f764c855144ff5bf1
  - Variante : https://dabblet.com/gist/21f54aa77edc8dd04885
  - Variante : http://dabblet.com/gist/93463d0d73aae793edf6
  - Variante : http://dabblet.com/gist/5c6f1b951653fc895aa8
- Gestion de l'espacement : https://dabblet.com/gist/2276cc05e685b027a729
- Astuce : alignement vertical : https://dabblet.com/gist/511e6552b47e560f185b9f1b4498db08

Avec CSS, on peut modifier le rendu exemple : faire des grilles

- https://developer.mozilla.org/fr/docs/Web/CSS/CSS_grid_layout
- Exemple modifié : https://developer.mozilla.org/fr/play?id=YSZnqMKaDgGzuy4M58cMVb4e6Af1Tqp7%2Fs2ZMnhmlgLwcZS1YL806QXKxXVGl%2F7XoecrhGA62Wz0mdPm
- Pour le contenant :
  - display : grid
  - grid-template-columns et grid-template-rows : Définissent le nombre et la taille des colonnes et des lignes.
    - Exemple :
      grid-template-columns: 1fr 2fr 1fr; /* Trois colonnes : 1re et la 3e ont la même taille, la 2e est 2x plus grande */
      grid-template-rows: auto; /* Hauteur automatique pour les lignes */
    - Unités fr : Représentent une fraction d'espace. Sinon, on peut utiliser des mesures.
  - gap : Définit l'espacement entre les éléments de la grille.
  - align-items et justify-items : Contrôlent l'alignement des éléments dans leurs cellules.
- Pour les éléments contenus
  - grid-column et grid-row : Positionnent les éléments dans des colonnes ou lignes spécifiques.
  - Exemple : grid-column : 2 / span 3 ; -> l'élément commence dans la 2e colonne et s'étend sur 3 colonnes

Détails des propriétés

width / height (15)
overflow (22)
transition (23)
opacity (29)

position (30)
z-index (32)
float / clear (33)
display (36)

vertical-align (39)
flex à partir de 40

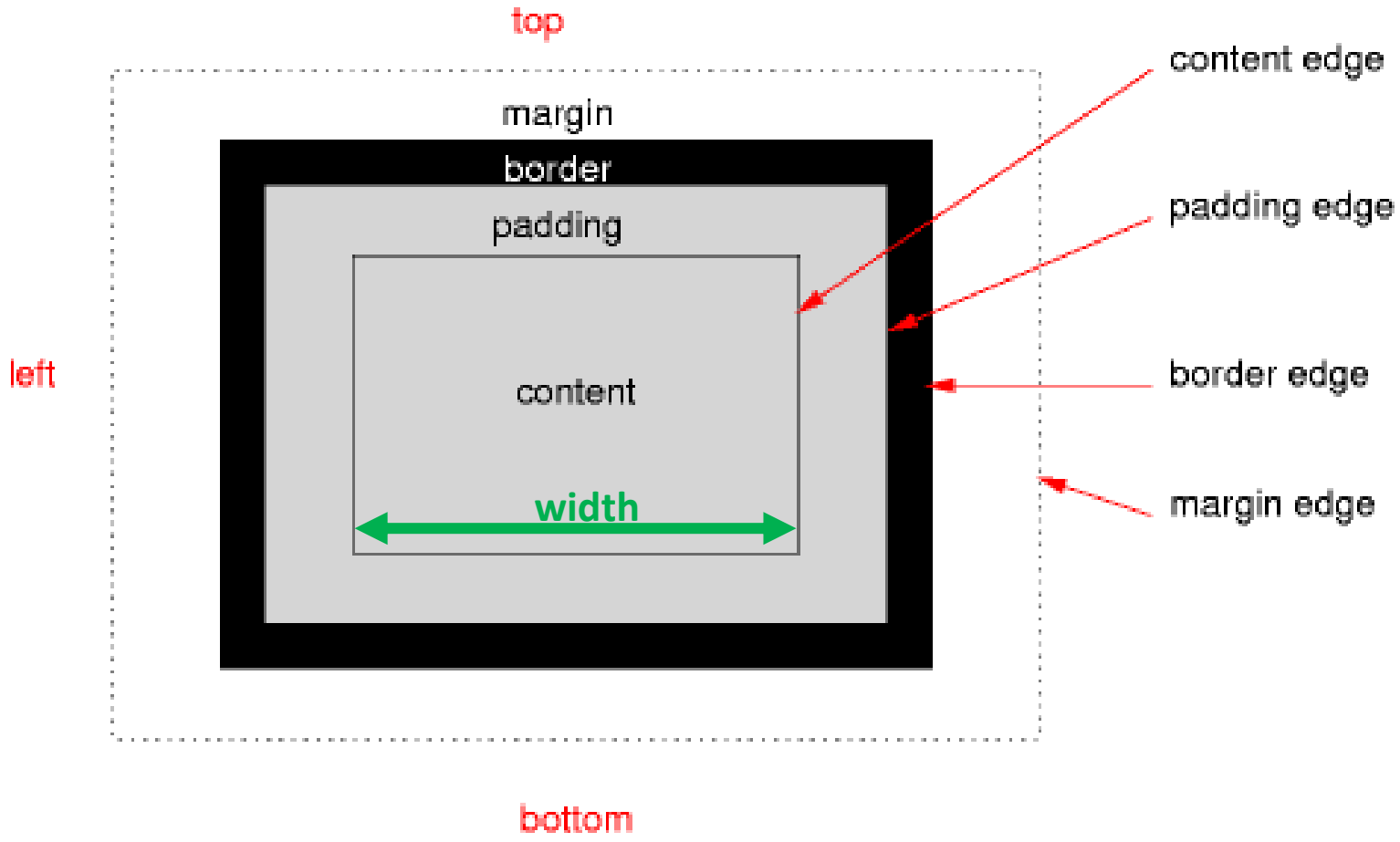
Définir la largeur d'un élément : width

- Indique la largeur du contenu d'un élément

- S'applique à tous les éléments sauf les lignes des tableaux et les éléments inline qui ne sont pas remplacés (i.e., les balises inclues dans le texte, comme a, em, etc.)
- Valeur par défaut : auto
- Chaque élément n'hérite pas de la largeur de l'élément parent (pas systématiquement)
- Cette propriété est prise en compte dans les animations CSS (sauf si la valeur d'arrivée ou de départ est auto)

- Les valeurs possibles, une valeur qui peut être :
  - Une longueur positive (10rem ou 150px etc.)
  - auto (calcul par le navigateur, en fonction de la nature de l'élément, de son contenu, des autres propriétés et du parent)
  - un mot-clef (c.f. dans 5 transparents)
  - Un pourcentage par rapport à la largeur de la boite englobante (i.e., width « normale » + padding + épaisseur des contours gauche et droit + marge)
- http://dabblet.com/gist/4e44eeca6cd8afec3d49

Impact de width



pour connaitre les cas très particuliers de calcul des largeur (position absolue, etc.) :
http://www.w3.org/TR/CSS21/visudet.html#Computing_widths_and_margins

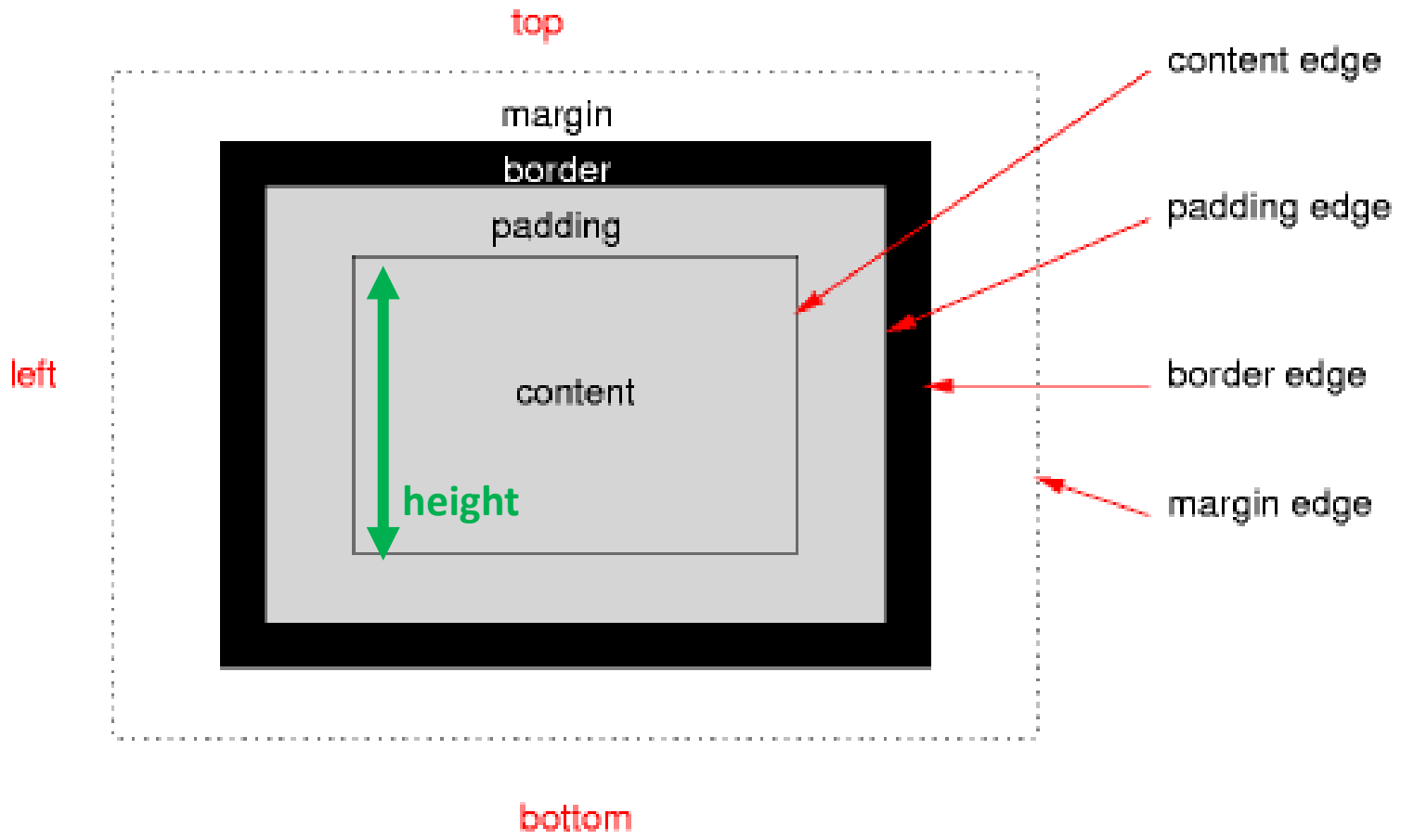
Définir la hauteur d'un élément : height

- Indique la hauteur du contenu d'un élément

- S'applique à tous les éléments sauf les éléments inline qui ne sont pas remplacés (i.e., les balises inclues dans le texte, comme a, em, etc.)
- Valeur par défaut : auto
- Chaque élément n'hérite pas de la hauteur de l'élément parent (pas systématiquement)
- Cette propriété est prise en compte dans les animations CSS (sauf si la valeur d'arrivée ou de départ est auto)

- Les valeurs possibles, une valeur qui peut être :
  - Une longueur positive (10rem ou 150px etc.)
  - auto (calcul par le navigateur, en fonction de la nature de l'élément, de son contenu, des autres propriétés et du parent)
  - un mot-clef (c.f. dans 3 transparents)
  - Un pourcentage par rapport à la hauteur de la boite englobante (i.e., height « normale » + padding + épaisseur des contours haut et bas + marge)
- http://dabblet.com/gist/9982b56b40c843050414

Impact de height



pour connaitre les cas très particuliers de calcul des hauteurs (position absolue, etc.) :
http://www.w3.org/TR/CSS21/visudet.html#Computing_heights_and_margins

largeur/hauteur minimale et maximale

- min-width, min-height permettent de définir une largeur et une hauteur minimale
  - Fonctionne comme width et height
  - Valeur par défaut : 0
- max-width, max-height permettent de définir une largeur et une hauteur maximale
  - Fonctionne comme width et height avec une valeur de plus, « none »
    - Valeur par défaut
    - Signifie qu'il n'y a pas de limite dans la dimension
- Dans le calcul des dimensions des éléments, voici l'impact des contraintes:
  - Calcul de la largeur et de la hauteur
  - 1re vérification : si ça dépasse, il faut tout recalculer en prenant la taille maximale concernée
  - 2e vérification : si c'est trop petit, il faut tout recalculer en prenant la taille minimale concernée
  - Donc si min-width > max-width, c'est min-width qui « l'emporte »

Mots-clefs pour les dimensions

- ‘fill-available’ | ‘max-content’ | ‘min-content’ | ‘fit-content’
  - Fonctionnent avec ‘width’, ‘min-width’, ‘max-width’, ‘height’, ‘min-height’, ‘max-height’
  - Ces valeurs sont préfixées (ex : -webkit-min-content)
  - pas pour ie ni opera-mini, pas pour height sur firefox
- min-content : taille minimale
- max-content : taille maximale, sans ajout de ligne (pas de « wrap »)
- fit-content : s’adapte au contenu (ni + ni -)
- fill-available : taille maximale, mais avec ajout de ligne.
- http://dabblet.com/gist/64b99efe26794234f26d

- Encore plus : http://www.w3.org/TR/css3-sizing/

Gérer ce qui déborde : overflow

- Comme avec width / height, il est possible de changer la taille « naturelle » (i.e., calculée pour le contenu), il peut y avoir des débordements
- overflow permet de définir comment cela sera afficher (ou non)
- overflow est un raccourci : permet de définir avec la même (et unique) valeur le traitemetn du dépassement sur l'horizontale (overflow-x) et sur la verticale (overflow-y)

overflow-x | overflow-y

- Indique comment gérer ce qui dépasse, dans la direction indiquée

- S'applique aux « containants » (niveau bloc, ou de display bloc ou flex)
- Valeur par défaut : visible (ça dépasse)
- Pas d'héritage
- Cette propriété n'est pas prise en compte dans les animations CSS

- Les valeurs possibles, une parmi :
  visible | hidden | scroll | auto | paged-x | paged-y | paged-x-controls | paged-y-controls | fragments
  - fragments : non implémenté
  - paged-x | paged-y | paged-x-controls | paged-y-controls : pour les impressions (création de page)
  - visible : tout est visible
  - hidden : ce qui dépasse n'est pas visible
  - scroll : ajout de scrollbar (tout le temps, que cela dépasse ou non)
  - auto : ajout de scrollbar quand c'est nécessaire (dès que cela dépasse)

Faire une transition

- Lorsque le style css change (avec javascript ou des sélecteurs comme :hover), le navigateur peut calculer une animation fait d'étape intermédiaire
- transition
- Fait sans javascript, mais sur le même thread
- Permet de définit
  - Quelles propriétés (« animables ») sont concernés : transition-property
  - La durée de l'animation : transition-duration
  - Avec quelle vitesse/accélération : transition-timing-function
  - S'il y a un décalage temporelle (avant l'animation) : transition-delay

transition-property

- Indique ce qui sera pris en compte dans l'animation

- S'applique à tous les éléments (y compris :before et :after)
- Valeur par défaut : all (tout ce qui « animable » est animé)
- Pas d'héritage
- Cette propriété participe à définir les animations CSS

- Les valeurs possibles, une parmi :
  - all (tout ce qui « animable » est animé)
  - none (pas d'animation)
  - Une propriété CSS
    - Attention aux noms (et aux noms multiples quand c'est préfixé)
  - Une suite de propriétés css, séparées par des « , »
- La dernière déclaration transition-property avec la plus grande spécificité sera appliquée… donc faire attention à tout mettre sur "une ligne/déclaration"

- http://dabblet.com/gist/b8a1b5dce7e39cd678f2

transition-duration

- Indique la durée de chaque animation

- S'applique à tous les éléments (y compris :before et :after)
- Valeur par défaut : 0s (pas de transition)
- Pas d'héritage
- Cette propriété participe à définir les animations CSS

- Les valeurs possibles, une parmi :
  - Une durée (en s ou en ms)
  - Une suite de durées, séparées par des « , »
    - Chaque durée s'appliquera alors à la propriété correspondante dans transition-property
    - S'il n'y a pas assez de durée, on recommence avec la première durée et ainsi de suite
- http://dabblet.com/gist/ea3a60a66698fae40e35

transition-timing-function

- Indique la vitesse et l'accélération de l'animation

- S'applique à tous les éléments (y compris :before et :after)
- Valeur par défaut : ease
- Pas d'héritage
- Cette propriété participe à définir les animations CSS

- Les valeurs possibles, une parmi :
  - Une fonction
    ease | linear | ease-in | ease-out | ease-in-out | step-start | step-end | steps(<integer>[, [start | end]]?) | cubic-bezier(<number>, <number>, <number>, <number>)
  - Une suite de fonctions, séparées par des « , »
    - Chaque fonction s'appliquera alors à la propriété correspondante dans transition-property
  - c.f. http://www.w3.org/TR/css3-transitions/#transition-timing-function-property

transition-delay

- Indique le temps avec de commencer (ou de cesser) l'animation

- S'applique à tous les éléments (y compris :before et :after)
- Valeur par défaut : 0s (pas de délai)
- Pas d'héritage
- Cette propriété participe à définir les animations CSS

- Les valeurs possibles, une parmi :
  - Une durée (en s ou en ms)
  - Une suite de durées, séparées par des « , »
    - Chaque délai s'appliquera alors à la propriété correspondante dans transition-property
    - S'il n'y a pas assez de délai, on recommence avec le premier délai et ainsi de suite

transition

- Une notation raccourcie
- Valeur possible :
  - none (pas de transition)
  - <une propriété> || <une durée d'animation> || <une fonction> || <un délai>
    - Ordre important, mais elles sont facultatives
    - S'il n'y a qu'un temps => une durée d'animation (et donc une animation sur tout)
  - Une suite de valeurs comme celle-ci-dessus, séparées par des « , »

Gérer la transparence d'un élément : opacity

- Indique le degré d'opacité d'un élément

- S'applique à tous les éléments
- Valeur par défaut : 1 (complétement opaque)
- Chaque élément n'hérite pas de l'opacité de l'élément parent
- Cette propriété est prise en compte dans les animations CSS

- Les valeurs possibles, une valeur qui peut être :
  - inherit (l'opacité du parent)
  - Un nombre en 0.0 et 1.0, e.g., 0.2412
    - 1 => complétement opaque
    - 0 => complétement transparent (non visible)
    - Entre : mixage de couleur avec ce qui est dessous

position (1ère rencontre)

- Indique comment calculer la position de l'élément dans la page / le document

- S'applique à tous les éléments
- Valeur par défaut : static (à l'endroit où il est inséré dans le document)
- Chaque élément n'hérite pas du positionnement de l'élément parent (mais cela peut avoir un effet)
- Cette propriété n'est pas prise en compte dans les animations CSS

- Valeurs possibles, 1 parmi :
  - static | relative | absolute | center | page | fixed
  - center | page ne sont pas implémentés
  - static : à l'endroit où il est inséré dans le document
  - relative : à l'endroit où il est inséré dans le document modulo un décalage à préciser avec top | right | bottom | left
  - absolute : une position fixe par rapport au document (si on scrolle, l'élément suit le mouvement), en précisant la position avec top | right | bottom | left
  - fixed: une position fixe par rapport à la fenêtre (si on scrolle, l'élément ne bouge pas, il reste où il est), en précisant la position avec top | right | bottom | left

top | right | bottom | left

- Indique la distance par rapport au bord indiqué
  - Si la position est relative, c'est par rapport à l'endroit où l'élément aurait du être
  - Si la position est absolute, c'est par rapport à boite du parent si le parent est en position absolute ou fixed, ou relative, sinon au document
  - Si la position est fixed, c'est par rapport à la fenêtre (viewport)

- S'applique à tous les éléments
- Valeur par défaut : auto (valeur calculer pour une position: static)
- Chaque élément n'hérite pas du positionnement de l'élément parent
- Cette propriété est prise en compte dans les animations CSS

- Valeurs possibles, 1 parmi :
  - auto : la valeur calculée normalement prévue
  - Une longueur (qui peut être négative, mais l'élément peut ne plus être visible)
  - Un pourcentage par rapport:
    - À sa propre largeur et hauteur totale si c'est relative
    - À la largeur et hauteur totale du parent si c'est absolute et que le parent est en position absolute ou fixed ou relative, sinon du document
    - À la propre largeur et hauteur totale du viewport si c'est fixed
- Utilisée deux propriétés opposées définit une largeur ou une hauteur (mais elle ne supplante pas width et height)

Gestion de la superposition : z-index

- Indique à quelle « hauteur » se trouve l'élément

- S'applique à tous les éléments repositionnés
- Valeur par défaut : auto (0, comme les « autres »)
- Chaque élément n'hérite pas du positionnement de l'élément parent
- Cette propriété est prise en compte dans les animations CSS (valeur entière)

- Valeurs possibles, 1 parmi :
  - auto (= 0)
  - Un entier
    - positif ou négatif
    - Plus la valeur est grande, plus l'élément est haut, plus il est au dessus des autres et se verra
    - Plus la valeur est petite, moins ou le voit
    - Les éléments « static », affichés à leur place vis-à-vis du document html sont au niveau 0

float

- http://www.w3.org/TR/CSS2/visuren.html#floats
- Permet de positionner l'élément sur le côté (comme dans un article de journal)
- Attention à l'ordre du html qui n'est pas sans conséquence
- http://dabblet.com/gist/72c7f579778e51dc2c96
  - Voir l'effet en cas de largeur insuffisante…
  - Voir l'effet des marges…

float

- Indique la boite est flottant ou pas

- S'applique à tous les éléments (mais voir les relations avec display et position : http://www.w3.org/TR/CSS2/visuren.html#dis-pos-flo)
- Valeur par défaut : none
- Pas d'héritage
- Cette propriété n'est pas prise en compte dans les animations CSS

- Valeurs possibles, 1 parmi :
  left | right | none | inherit
  - left : pour mettre sur la gauche
  - right : pour mettre sur la droite
  - none : ce n'est pas flottant
  - De nouvelles valeurs à venir (top bottom et snap), notamment avec le multi-columns, c.f., http://alistapart.com/blog/post/ten-css-one-liners-to-replace-native-apps/

clear

- Permet de dire qu'on ne veut rien de flottant (ou au contraire) sur tel ou tel côté

- S'applique à tous les éléments de niveau block
- Pas d'héritage
- Cette propriété n'est pas prise en compte dans les animations CSS

- Valeurs possibles, une parmi :
  none | left | right | both | inherit
  - none : on accepte encore des éléments flottant sur les côtés
  - both : on n'en accepte plus, sur aucun des deux côtés (place en dessous des éventuels éléments flottant qui le précèdent)
  - left : on n'en accepte plus sur le côté gauche (place en dessous des éventuels éléments flottant sur la gauche qui le précèdent)
  - right: on n'en accepte plus sur le côté droit (place en dessous des éventuels éléments flottant sur la droite qui le précèdent)

Changer la façon de s'afficher : display

- Permet de dire si l'élément va s'afficher comme un morceau de phrase (inline), un paragraphe (block), un tableau, etc.
- Se décompose en théorie de 3 propriétés
  - display-inside : définit le comportement de l'affichage, à l'intérieur de l'élément
  - display-outside : permet de placer l'élément dans le reste de la page
  - pour dire si c'est une liste : display-list
  - (implémentation à vérifier, a priori pas encore supporté)

display

- Indique le type d'affiche de l'élément

- S'applique à tous les éléments
- Valeur par défaut : inline
- Pas d'héritage
- Cette propriété n'est pas prise en compte dans les animations CSS

Display : Valeurs possibles

- inline — comme un morceau de phrase
- block — comme un “block” (p, h1, etc.)
- inline-block — à l’intérieur comme un block, à l’extérieur traité comme un inline
- flex — pour une mise en page avec des boites facilements déplaçables
- grid — pour organizer comme une grille (un tableau)
- inline-flex — comme s’il était dans un parent lui même en flex
- inline-table — un morceau de phrase dans un tableau
- list-item — comme un item de liste
- table — comme <table> element
  - table-caption — comme <caption> element
  - table-column-group — comme <colgroup> element
  - table-header-group — comme <thead> element
  - table-footer-group — comme <tfoot> element
  - table-row-group — comme <tbody> element
  - table-cell — comme <td> element
  - table-column — comme <col> element
  - table-row — comme <tr> element
- none — l’élément n’apparait pas, ne prend pas de place, comme s’il n’existait pas
- Etc.

vertical-align

- Indique l'alignement vertical, notation raccourcie pour alignment-baseline , alignment-adjust , baseline-shift et dominant-baseline
- S'applique aux éléments inline ou aux cellules des table (td, th)
- Valeur par défaut : voir les propriétés
- Chaque élément n'hérite pas de la largeur de l'élément parent (pas systématiquement)
- Cette propriété est prise en compte dans les animations CSS (sauf si la valeur d'arrivée ou de départ est auto)
- c.f. http://www.w3.org/TR/css3-linebox/#vertical-align-prop
- Les valeurs possibles, une valeur qui peut être :
  auto | use-script | baseline | sub | super | top | text-top | central | middle | bottom | text-bottom | <percentage> | <length>
  - top : aligne la limite supérieur de la ligne d'écriture de l'élément avec celle de son parent (de la ligne du haut)
  - bottom : aligne la limite bas de la ligne d'écriture de l'élément avec celle de son parent (de la ligne du bas)
  - middle : aligne le milieu de la ligne d'écriture de l'élément avec celle de son parent
  - baseline : aligne les « alpabetic » (de la ligne du bas)
  - Etc.

SNT

Flex box

Gérer facilement des alignements…

Philippe Renevier-Gonin

Web : css

Des alignements… souples

- Permets de faire des « flow » de données, en ligne ou en colonne
- Supporte des tailles différentes
  - Contrôle sur la réorganisation par rapport à l'idéal
- Se décompose en différentes propriétés
  1. **display : flex ;** (pour le contenant)
     - /* display : inline-flex ; (pour faire comme si c'était le contenu d'un contenant « flex ») : statut automatiquement gagné si le parent est display: flex */
  2. Des propriétés spéciales
     - flex-direction + flex-wrap = flex-flow
       - order
     - Flexibilité : flex-grow + flex-shrink + flex-basis = flex
     - Alignement : justify-content / align-items / align-self / align-content

flex-direction

- Permet de dire l'orientation (horizontal ou vertical)

- S'applique aux flex containers (display: flex)
- Valeur par défaut : row
- Il n'y a pas héritage
- Cette propriété n'est pas prise en compte dans les animations CSS

- Valeurs possibles, 1 parmi :
  row | row-reverse | column | column-reverse
- http://dabblet.com/gist/fc3daa5052f4514a44e3

flex-wrap

- Permet de savoir s'il peut y avoir plusieurs lignes (ou colonnes)

- S'applique aux flex containers (display: flex)
- Valeur par défaut : nowrap
- Il n'y a pas héritage
- Cette propriété n'est pas prise en compte dans les animations CSS

- Valeurs possibles, 1 parmi :
  nowrap | wrap | wrap-reverse
  - nowrap : reste sur une ligne (ou colonne)
  - wrap : ajout de ligne (ou colonne) possible, dans le sens de lecture
  - wrap-reverse : ajout de ligne (ou colonne) possible, dans le sens opposé à celui de lecture

- Même exemple (il faut redimensionner votre fenêtre) : http://dabblet.com/gist/fc3daa5052f4514a44e3

flex-flow

- Notation raccourcie pour flex-direction + flex-wrap

- S’applique aux flex containers (display: flex)
- Valeur par défaut : row nowrap
- Il n’y a pas héritage
- Cette propriété n’est pas prise en compte dans les animations CSS

order

- Permet de contrôler l'ordre de positionnement des boites

- S'applique aux éléments contenus dans un flex container (display : inline-flex ou positionnement absolu dans le container)
- Valeur par défaut : 0
- Il n'y a pas héritage
- Cette propriété n'est pas prise en compte dans les animations CSS

- Valeurs possibles : 1 entier
  - Cette valeur permet de regrouper les éléments avec la même valeur
  - Les éléments sont ensuite affichés par groupe, en commençant par celui avec le n° le plus petit
  - Interaction avec l'affichage inverse : l'ordre d'affichage est conservé (le premier groupe sera le dernier…)
- http://dabblet.com/gist/011f764c855144ff5bf1

Flexibilité : il y a trop d'espace, pas assez, la taille de référence

- Il peut arriver qu'il y ait trop d'espace, comme dans l'exemple précédent… on peut le répartir (et déformer les élémets)
  ➔ flex-grow
- Idem s'il n'y a pas assez de place (et rétrécir)
  ➔ flex-shrink
- À partir d'une taille de référence
  ➔flex-basis

flex-grow

- Permet de déterminer comment agrandir les éléments en cas d'espace restant

- S'applique aux éléments contenus dans un flex container (display : inline-flex)
- Valeur par défaut : 0
- Il n'y a pas héritage
- Cette propriété est prise en compte dans les animations CSS, sauf pour le passage vers ou depuis 0

- Valeurs possibles : 1 nombre (positif)
  - En cas d'espace supplémentaire, celui-ci est réparti entre les éléments de manière pondérée.
    Si la somme des flex-grow vaut 10 et que celui d'un élément vaut 5 il recevra la moitié de l'espace supplémentaire
    Si un autre élément a un flex-grow de 0, il ne recevra rien
- http://dabblet.com/gist/21f54aa77edc8dd04885

flex-shrink

- Permet de déterminer comment réduire les éléments en cas d'espace manquant

- S'applique aux éléments contenus dans un flex container (display : inline-flex)
- Valeur par défaut : 1
- Il n'y a pas héritage
- Cette propriété est prise en compte dans les animations CSS, sauf pour le passage vers ou depuis 0

- Valeurs possibles : 1 nombre (positif)
  - En cas d'espace insuffisant, celui-ci est trouvé parmi les éléments de manière pondérée.
    Si la somme des flex-shrink vaut 10 et que celui d'un élément vaut 5 il donnera la moitié de l'espace manquant
    Si un autre élément a un flex-shrink de 0, il ne sera pas retaillé
- http://dabblet.com/gist/93463d0d73aae793edf6

flex-basis

- Permet de déterminer la taille de référence des éléments, avant tout redimensionnement

- S'applique aux éléments contenus dans un flex container (display : inline-flex)
- Valeur par défaut : auto
- Il n'y a pas héritage
- Cette propriété est prise en compte dans les animations CSS

- Valeurs possibles : auto ou une largeur (positive)
  - Les pourcentages sont par rapport à la dimension du contenu du parent concernée par le flex layout (vertical <->height ; horizontal <->width)
  - Auto : se base sur les dimensions « usuelles » de l'élément
  - 0 : l'élément n'a pas de dimension (mais peu grandir avec flex-grow)
- http://dabblet.com/gist/5c6f1b951653fc895aa8

flex

- Propriété raccourcie pour ‘flex-grow’, ‘flex-shrink’ et ‘flex-basis’

- S’applique aux éléments contenus dans un flex container (display : inline-flex)
- Valeur par défaut : 0 1 auto (c.f. les propriétés)
- Il n’y a pas héritage
- Cette propriété est prise en compte dans les animations CSS (c.f. les propriétés)

- Valeurs possibles :
  - none (⇔ 0 0 auto)
  - <‘flex-grow’> <‘flex-shrink’>? || <‘flex-basis’>
    - flex-grow > si non précisé, = 1
    - flex-shrink > si non précisé, = 1
    - flex-basis > si non précisé, = 0%

Alignement dans le flex box layout

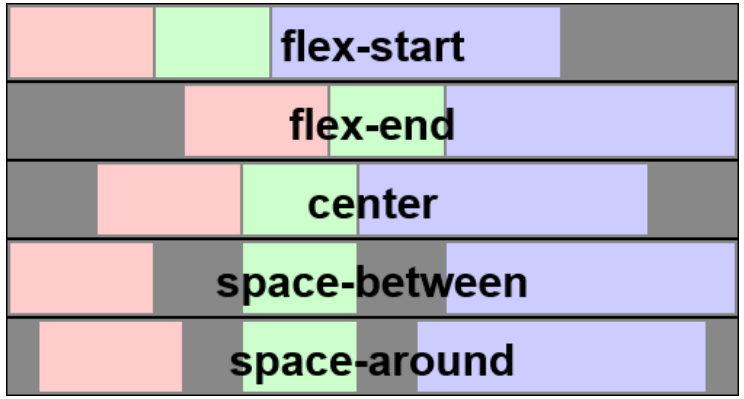
- margin: auto & flex
  - http://dabblet.com/gist/33edfc345b633434d161
  - Permet de distribuer l'espace restant
- Alignement sur l'axe du flex box :
  justify-content
- Alignement sur l'autre axe :
  align-items et
  align-self

justify-content

- Alignement selon l'orientation

- S'applique aux containers « display: flex »
- Valeur par défaut : flex-start
- Il n'y a pas héritage
- Cette propriété n'est pas prise en compte dans les animations CSS

- Valeurs possibles, une parmi :
  flex-start | flex-end | center
  space-between | space-around



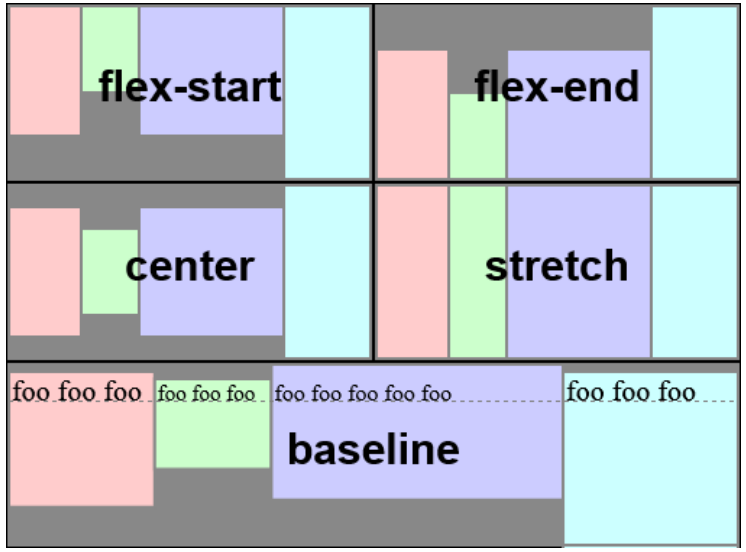
- http://dabblet.com/gist/2276cc05e685b027a729

align-items

- Alignement selon l'autre orientation

- S'applique aux containers « display: flex »
- Valeur par défaut : stretch
- Il n'y a pas héritage
- Cette propriété n'est pas prise en compte dans les animations CSS

- Valeurs possibles, une parmi : flex-start | flex-end | center | baseline | stretch



align-self

- Permet de modifier l'alignement d'un élément dans une flex box

- S'applique aux éléments contenu « display: inline-flex »
- Valeur par défaut : auto
- Il n'y a pas héritage
- Cette propriété n'est pas prise en compte dans les animations CSS

- Valeurs possibles :
  auto | flex-start | flex-end | center | baseline | stretch
  - idem que précédemment
  - auto pour garder la valeur spécifiée dans le container

align-content

- Pour aligner les éléments à l'intérieur du contenant, en cas d'espace supplémentaire

- S'applique aux containers « display: flex »
- Valeur par défaut : stretch
- Il n'y a pas héritage
- Cette propriété n'est pas prise en compte dans les animations CSS

- Valeurs possibles :
  flex-start | flex-end | center | space-between |
  space-around | stretch