

CSS 3

André Restivo

Index

Introduction

Linking

Resources

Selectors

Color

Dimensions

Units

Fonts

Text

Box Model

Border

Background

Lists

Tables

Transforms

Transitions

Positioning

Flexbox

Grid

Precedence

Responsive Design

Vars

Vendor Prefixes

Validation

Introduction

What are they?

- Cascading Style Sheets
- A style sheet language used for describing the look and formatting of a document written in a markup language (like HTML).
- Based on two concepts: **selectors** and **properties**.

History

- 1996 CSS 1 Limited and poorly supported by browsers
- 1998 CSS 2
- 1999 CSS 1 Supported by browsers
- 2003 CSS 2 Decently supported by browsers
- 2003 CSS Zen Garden (<http://www.csszengarden.com/>)
- 2011 CSS 2.1
- 2011–2012 CSS 3

Selectors

Allow us to select the HTML elements to which we want to apply some styles.

Properties

Define what aspect of the selected element will be changed or styled.

```
p  
color
```

Together, selectors and properties define CSS rules.

Linking to HTML

We can apply CSS styles to HTML documents in three different ways.

Inline

Directly in the HTML element

```
p
```

```
p
```

Internal Style Sheet

Using a stylesheet inside the HTML document

```
head
  style
p
  color

  style
head
body
  p
    body
```

External Style Sheet

In a separate stylesheet

```
head  
link  
head  
body  
  p  
body
```

style.css

```
p  
color
```

The preferred way. Allows for style **separation** and **reuse**.

Resources

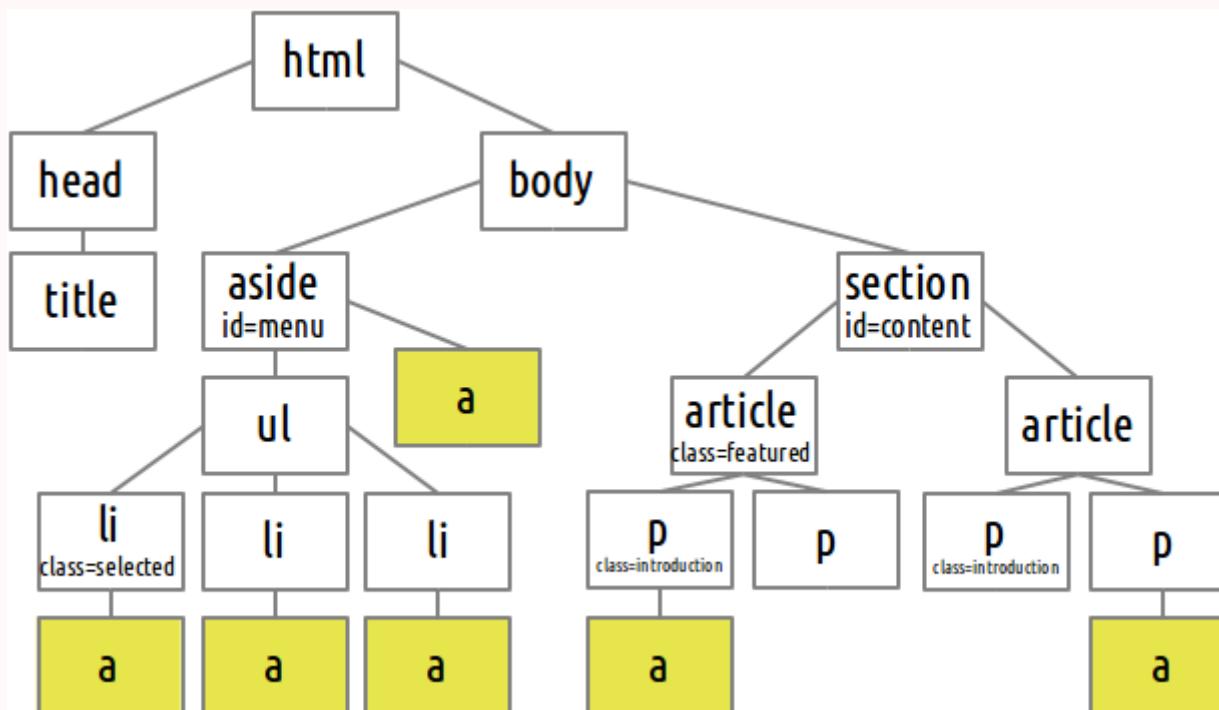
- References:
 - <https://developer.mozilla.org/en/docs/Web/CSS/Reference>
 - <http://www.w3.org/Style/CSS/specs.en.html>
- Tutorials:
 - <https://css-tricks.com/almanac/>
 - <http://www.htmldog.com/guides/css/>

Selectors

Element Selectors

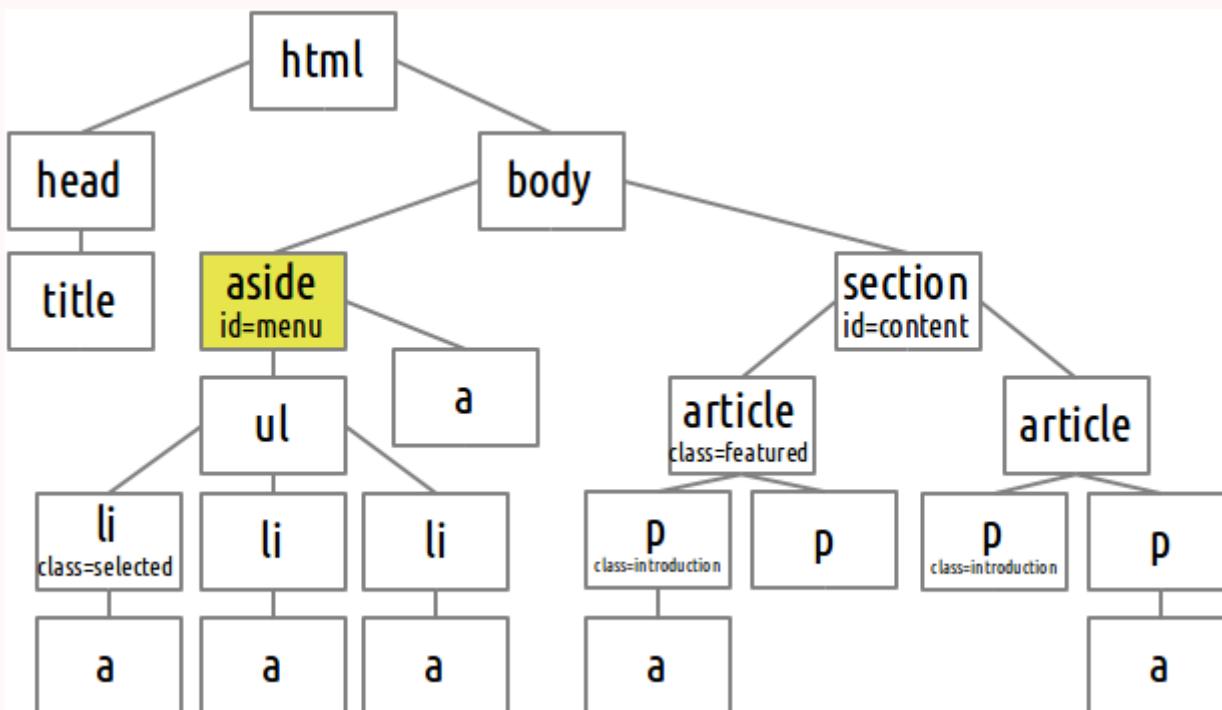
Select elements by their tag name

a



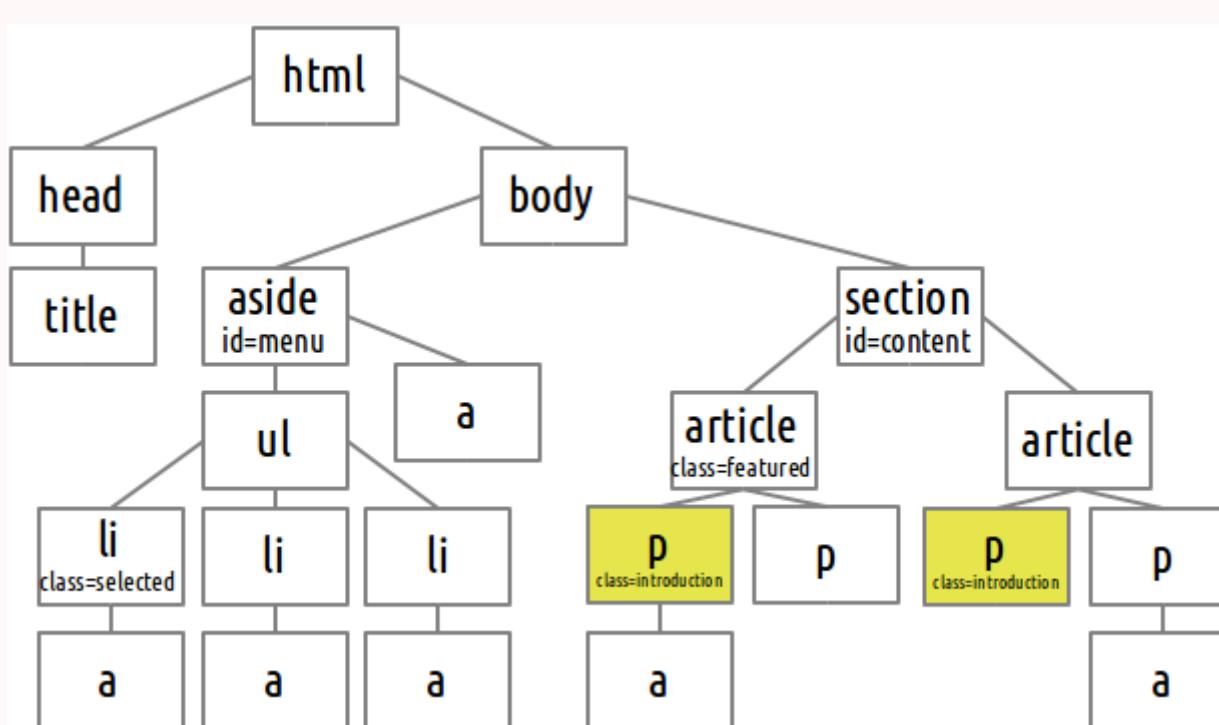
Id Selector

Selects element by their id (#)



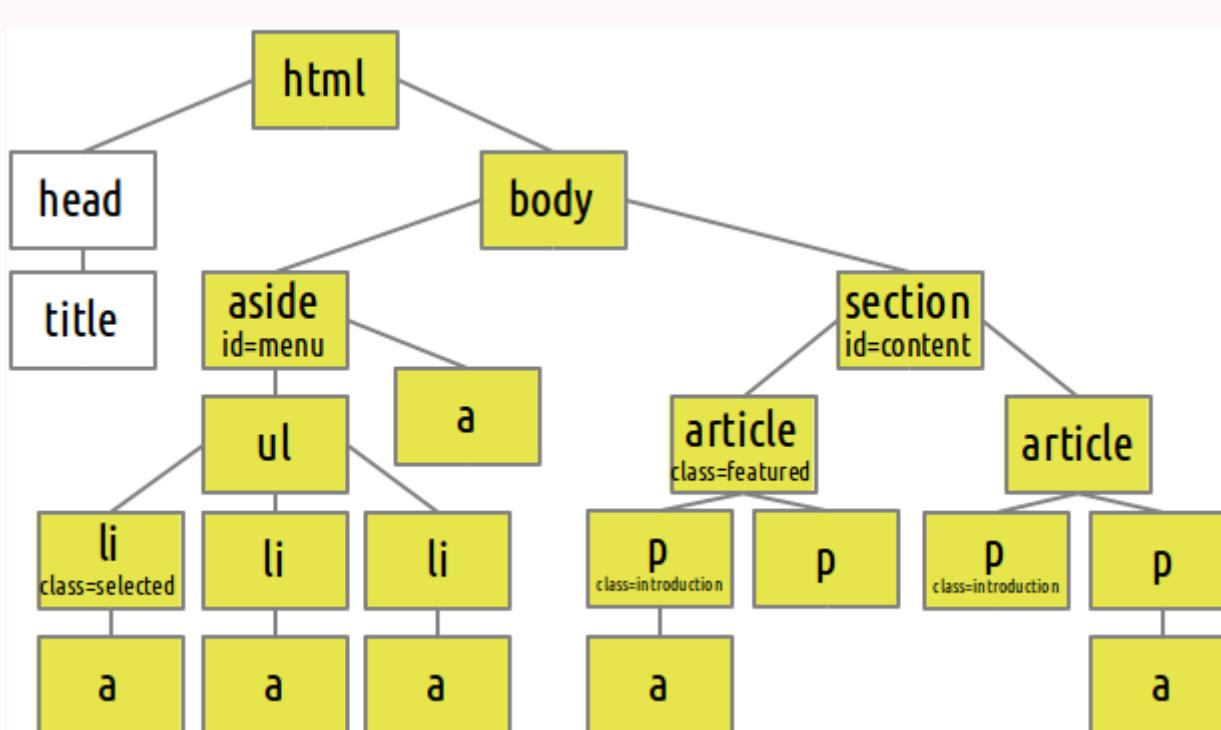
Class Selector

Selects element by their class (.)



Select All

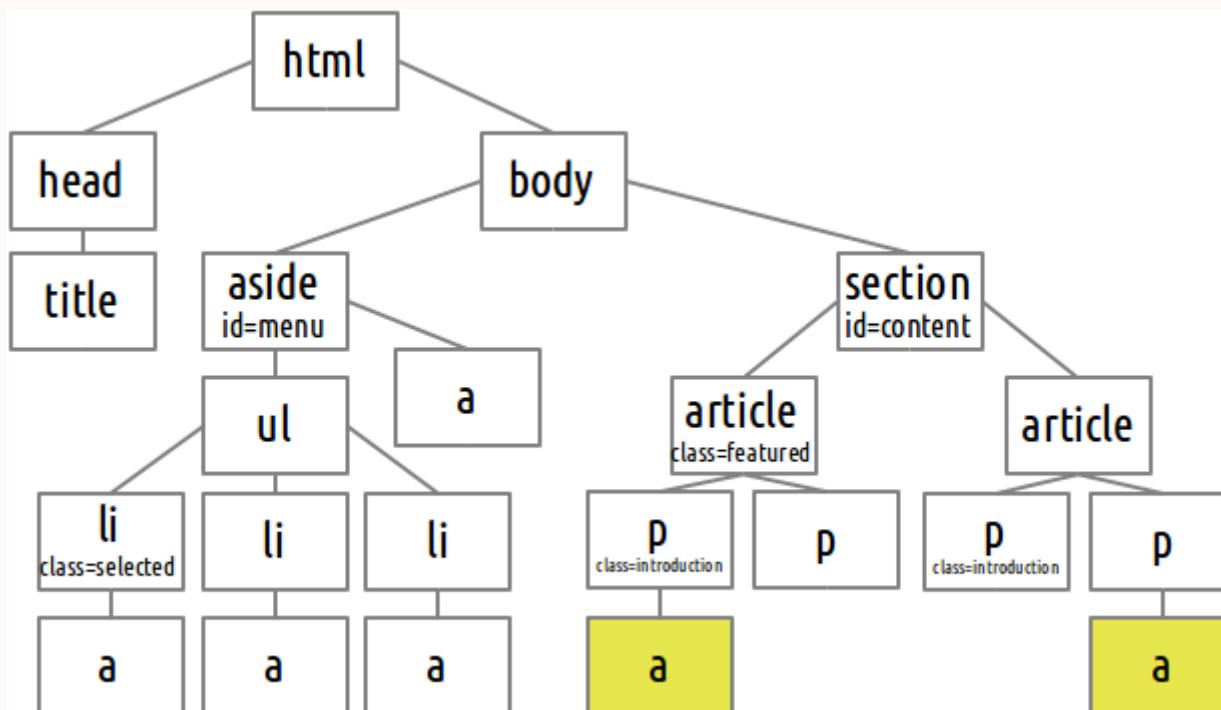
Selects all elements (*)



Descendant Selector

Selects all descendants (space)

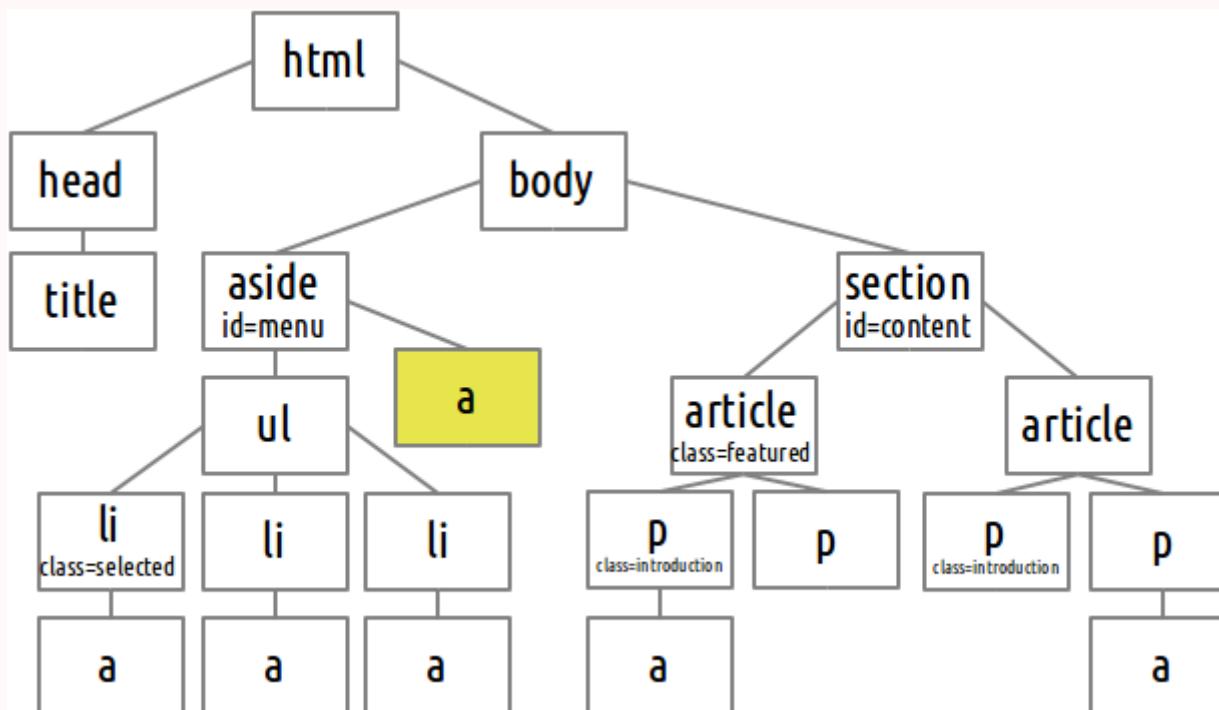
article a



Child Selector

Selects all children (>)

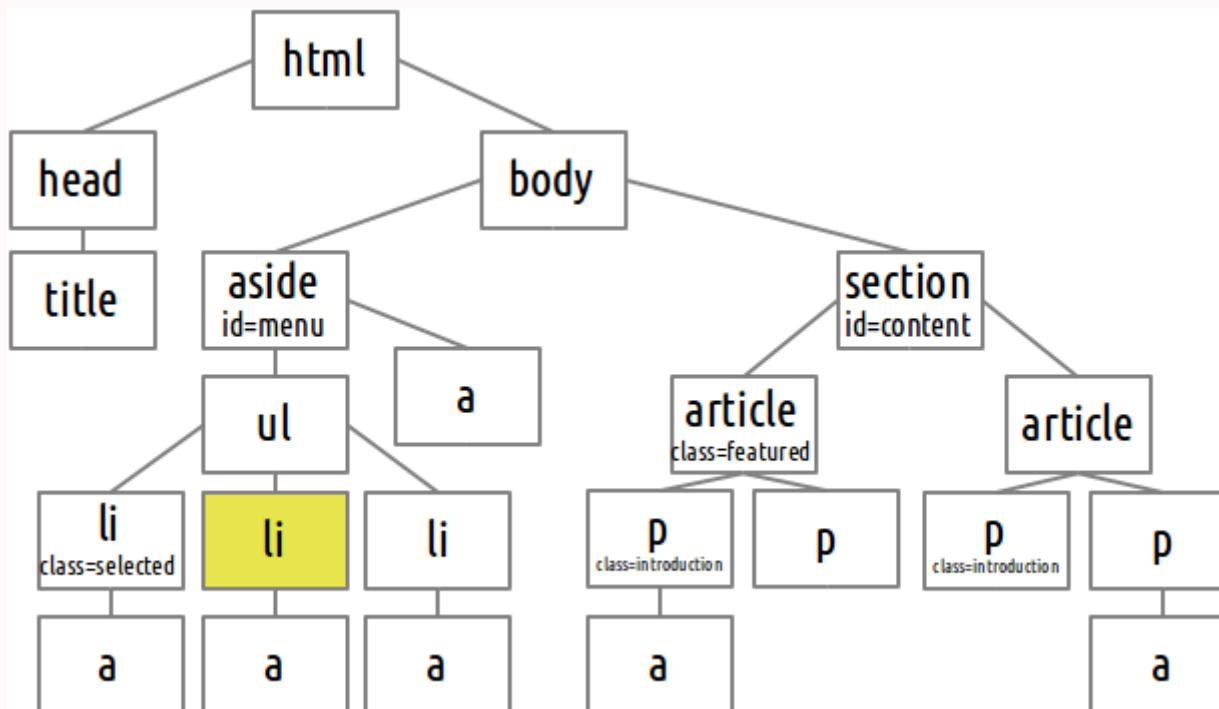
```
aside a
```



Immediately After Selector

Selects next sibling (+)

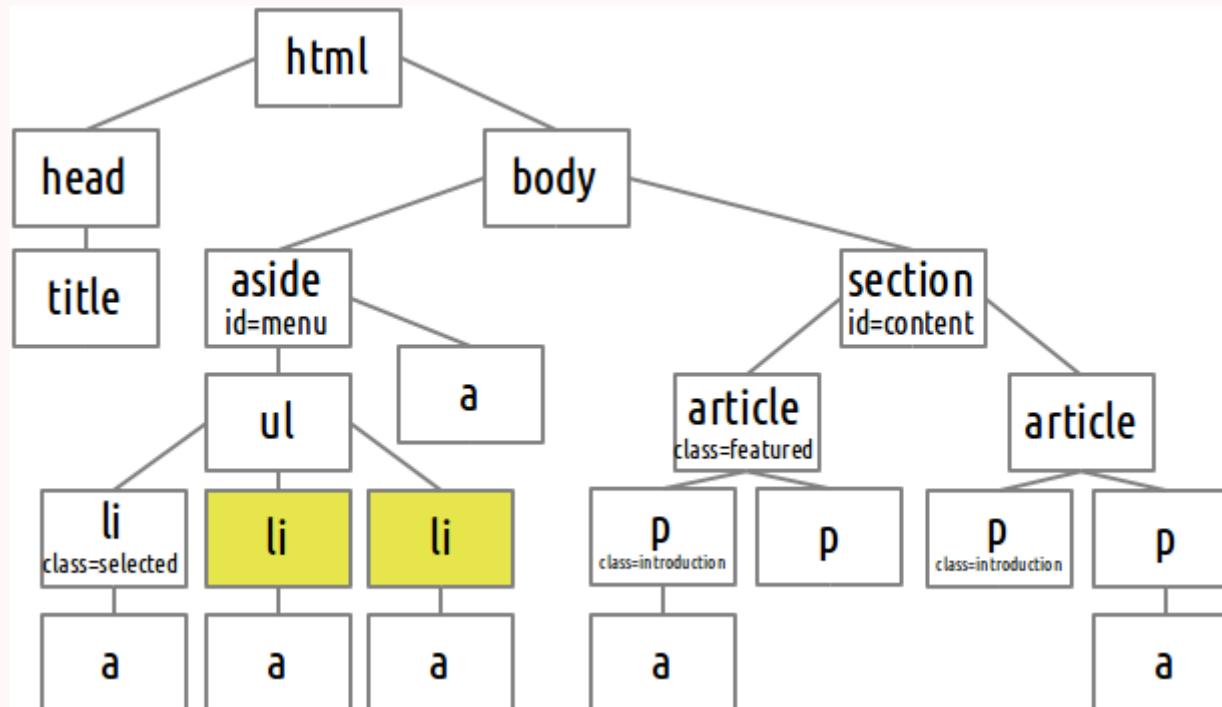
li



After Selector

Selects next siblings (~)

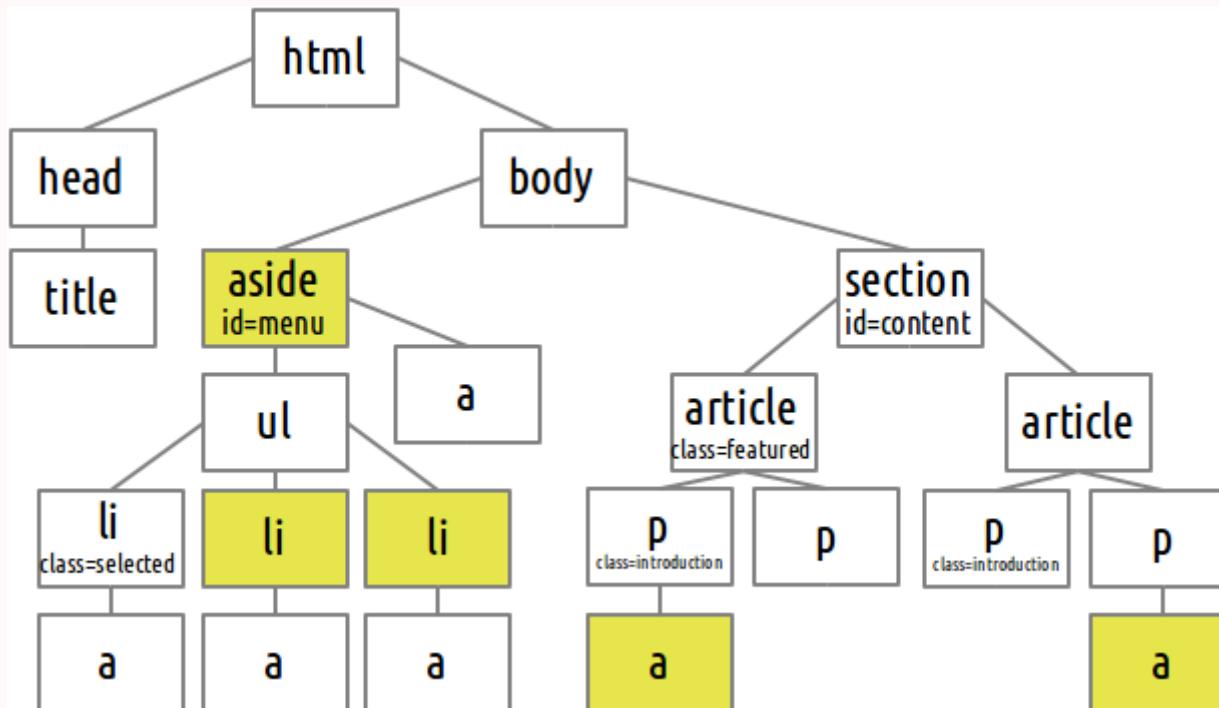
li



Multiple Selectors

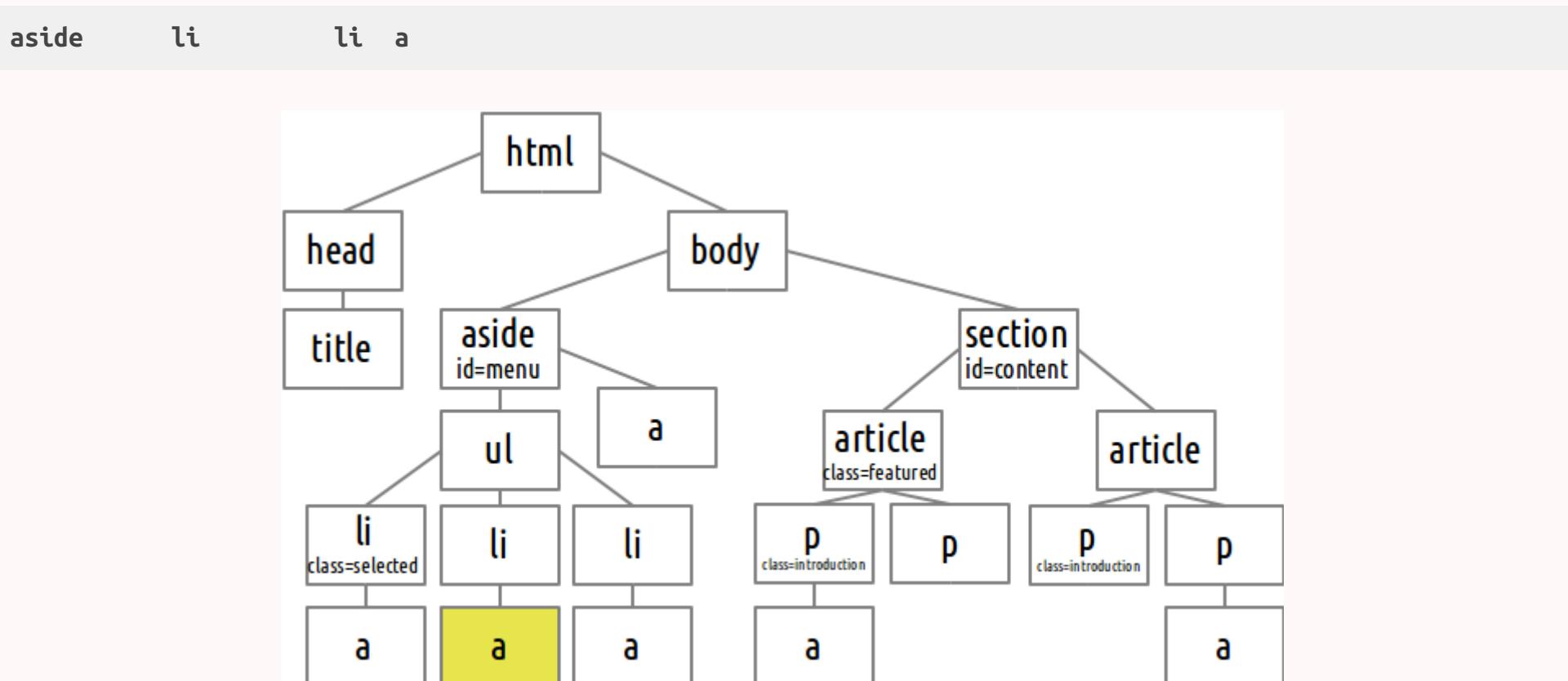
Multiple selectors (,)

li p a



Combinations

All these concepts can be combined to form powerful selectors



Pseudo Classes and Elements

- A pseudo-class is a way of selecting existing HTML elements, based on some specific characteristic (e.g. a visited link)
- Pseudo-elements allow logical elements to be defined which are not actually in the document element tree (e.g. The first letter of a paragraph)

More on [pseudo-classes](#) and [pseudo-elements](#)

Anchor Pseudo-classes

Selects anchors (links) based on their state:

a

- **link**: Link was not visited
- **visited**: Link was visited previously
- **active**: Link is active
- **hover**: Mouse is over the link (works on other elements)

Form Pseudo-classes

Selects form controls that have input focus:

```
input
```

```
input  
input
```

```
input  
input
```

```
input  
input
```

```
radio
```

Target Pseudo-class

The **target** pseudo-class represents the unique element, if any, with an id matching the fragment identifier of the URI of the document.

```
a           a  
div        div
```

```
div  
border
```

First and Last Pseudo-classes

Selects elements based on their position in the tree:

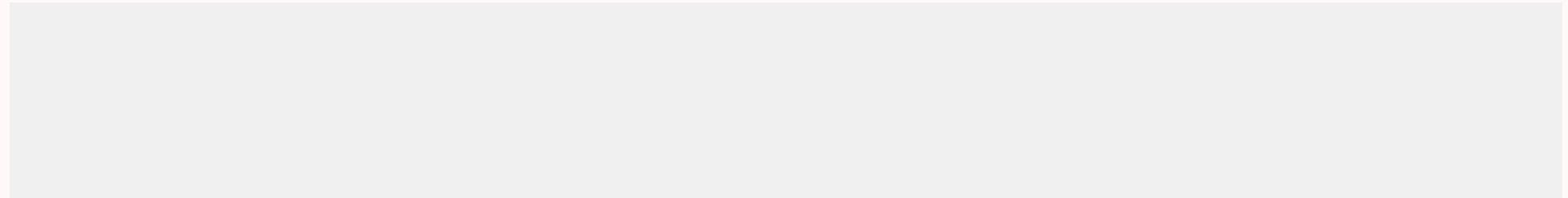
p

- **first-child**: Selects elements that are the first child of their parents
- **last-child**: Selects elements that are the last child of their parents
- **first-of-type**: Selects elements that are the first child of their type in their parents children's list
- **last-of-type**: Selects elements that are the last child of their type in their parents children's list

Nth Child Pseudo-classes

The **nth-child(an+b)** selector, selects elements that are the **bth** child of an element after all its children have been split into groups of **a** elements each.

In other words, this class matches all children whose index fall in the set $\{ an + b; n = 0, 1, 2, \dots \}$.



The **nth-of-type(an+b)** selector does the same thing but counts only siblings with the same name.

First and Last Pseudo-elements

Selects parts of elements based on their position in the tree:

p

- **first-line**: Selects the first line of the selector
- **first-letter**: Selects the first character of the selector

Before and After Pseudo-elements

Before and after pseudo-elements can be combined with the **content** property to generate content in an element.

The **content** property can have the following values:

- **none** The default value, adds nothing. Cannot be combined with other values: *none*
- **a string** Using single quotes. Adds the text to the element: '*Chapter*'
- **an url** An external resource (such as an image): *url('dog.png')*
- **counter** Variables maintained by CSS whose values may be incremented by CSS rules to track how many times they're used: *counter(section)* [Learn more](#).
- **open-quote** and **close-quote** Open and close quotes: *open-quote*

```
blockquote      content
blockquote      content
```

Attribute Selectors

Select elements based on their attribute existence and values:

form

- [attribute] exists
- [attribute=value] equals
- [attribute~=value] containing value (word)
- [attribute|=value] starting with value (word)
- [attribute^=value] starting with value
- [attribute\$=value] ending with value
- [attribute*=value] containing value

Color

Text Color

Setting the text color of any element.

```
p  
color
```

Background Color

Setting the background color of any element.

```
p  
background-color
```

Color by Name

Colors can be referenced using one of these pre-defined names:

```
aqua black blue fuchsia gray green  
lime maroon navy olive orange purple  
red silver teal white and yellow
```

```
p  
background-color
```

Color by Hexadecimal Value

A hexadecimal color is specified with: #**RRGGBB**, where the **RR** (red), **GG** (green) and **BB** (blue) hexadecimal integers specify the components of the color. All values must be between 00 and FF.

```
p  
background-color
```

#**RGB** is a shorthand for #**RRGGBB**

```
p  
background-color
```

Color by Decimal Value

An RGB color value also be specified using: `rgb(red, green, blue)`. Each parameter (`red`, `green` and `blue`) defines the intensity of the color and can be an integer between 0 and 255 or a percentage value (from 0% to 100%).

```
p  
background-color
```

Opacity

Specifies the transparency of an element. Values can go from 0.0 (completely transparent) to 1.0 (fully opaque).

```
p  
  opacity
```

Dimensions

Width and Height

Set the width and height of an element. Values can be a **length**, a **percentage** or **auto**.

```
div  
width  
height
```

Auto is the default value.

Minimum and Maximum

Set the minimum and maximum width and height of an element. Values can be a **length**, a **percentage** or **none**.

```
div  
  max-width  
  min-height
```

None is the default value.

Length Units

<https://developer.mozilla.org/en-US/docs/Web/CSS/length>

Absolute length

Absolute length units represents a physical measurement. They are useful when the physical properties of the output medium are known, such as for print layout.

mm cm in pt and pc

- **mm** One millimeter.
- **cm** One centimeter (10 millimeters).
- **in** One inch (2.54 centimeters).
- **pt** One point (1/72nd of an inch).
- **pc** One pica (12 points).

Font relative length

Font relative length units are relative to the size of a particular character or font attribute in the font currently in effect in the element (or parent element in some cases).

They are useful when the physical properties of the output medium are unknown, such as for screen layout.

- **rem** Represents the size of the root element font. If used in the root element, represents the initial (default) value of the browser (typically 16px).
- **em** When used with *font-size*, represents the size of the parent element font. For lengths, represents the size of the current element font.

Example (rem)

```
div
  p      p
  div
    p      p
    div
  div
```

```
div
  font-size
```

Some text

Some more text

Example (em)

```
div
  p          p
  div
    p          p
    div
  div
```

```
div
  font-size
```

Some text

Some more text

Pixel

- On low dpi screens, the **pixel (px)** represents one device pixel (**dot**).
- On higher dpi devices, a pixel represents an integer number of device pixels so that $1\text{in} \approx 96\text{px}$.

Percentage

The *percentage* CSS data type represents a percentage value. A percentage consists of a *number* followed by the percentage sign %. There is no space between the symbol and the number.

```
div  
width
```

Many CSS properties (width, margin, padding, font-size, ...) can take *percentage* values to define a size as relative to its parent object.

Fonts

Font Family

In CSS, there are two types of font family names:

- **generic family** - a group of font families with a similar look.
- **font family** - a specific font family (e.g. Times New Roman).

Specific Font Family

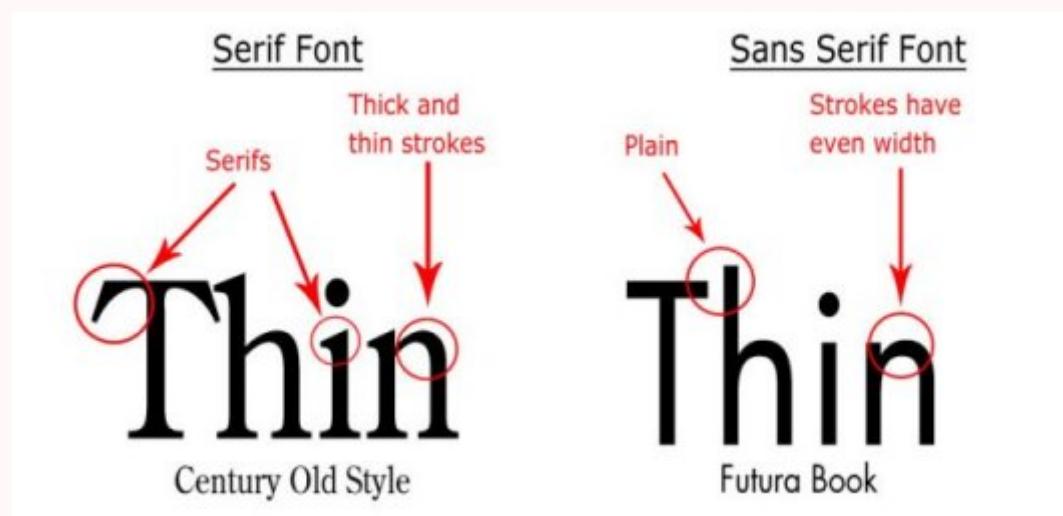
You can define a specific font family to be used. Be careful as the it might not exist in the target computer.

```
p  
font-family
```

Generic Font Family

Or a generic family like: **serif**, **sans-serif** and **monospace**.

```
p  
font-family
```



Web Safe Fonts

- To ensure that websites look the same across different platforms we should use *web safe* fonts like: Arial, Helvetica, Times New Roman, Times, Courier New or Courier.
- You can specify several fonts. The browser will try to use the first and continue down the list if it doesn't exist.
- Start with the font you want and gradually fall back to platform defaults and finally generic defaults:

```
p  
font-family
```

Remote Fonts

- The `@font-face` rule specifies a custom font with which to display text.
- The font can be loaded from a remote server making it possible to use all kind of fonts.

```
font-face  
font-family  
src
```

- An easier way to provide remote fonts is to use Google's fonts service:
<https://fonts.googleapis.com/>

Font Weight

You can specify the weight of the font using the `font-weight` property. Values can be **normal**, **bold**, **bolder**, **lighter** or values from **100** to **900**.

```
p  
  font-weight
```

Font Style

The `font-style` property allows you to specify if the font style should be italic. Values can be **normal**, **italic**, or **oblique**.

```
span  
font-style
```

Font Size

To define the font size you use the **font-size** property.

```
p  
  font-size
```

Use **rem** or **em**.

Text

Decoration

The **text-decoration** property is mostly used to remove underlines from links. But it has other possible values: **none**, **underline**, **overline** and **~~line-through~~**.

```
a  
text-decoration
```

Alignment

Text can be aligned **left**, **right**, **center** or justified (**justify**) using the **text-align** property. This property should be used for aligning text only.

text-align

left

Lorem ipsum, dolor sit amet, consectetur adipiscing elit.

center

Lorem ipsum, dolor sit amet, consectetur adipiscing elit.

justify

Lorem ipsum, dolor sit amet, consectetur adipiscing elit.

right

Lorem ipsum, dolor sit amet, consectetur adipiscing elit.

Transformation

The **text-transform** property can be used to make the text uppercase, lowercase or capitalized (capitalize first letter of each word).

```
h1  
text-transform
```

Indentation

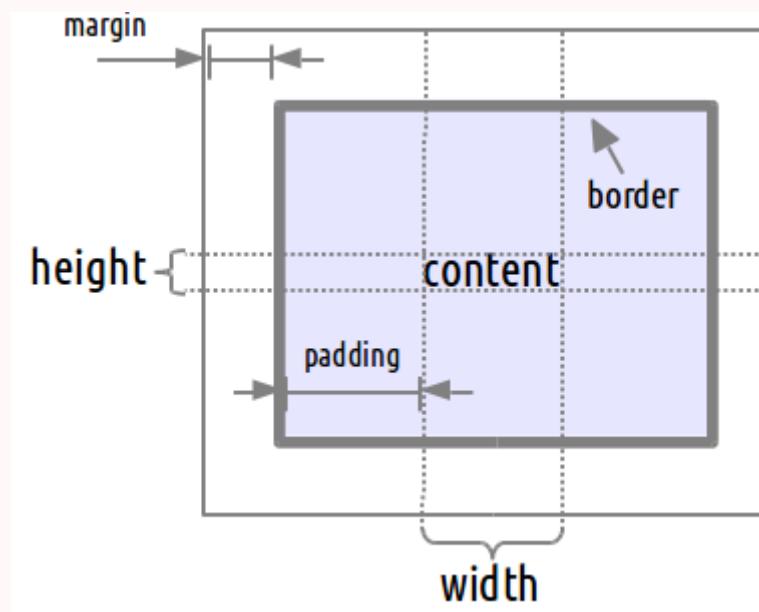
The first line of each paragraph can be indented using the **text-indent** property. This property takes a length as its value.

```
p  
text-indent
```

Box Model

Box Model

Elements all live inside a box. They can have a **border**, some space between themselves and that border (**padding**) and some space between themselves and the next element (**margin**).



Display

There are 41 different possible values for the `display` property. For now, we will concentrate on only four of them: **none**, **inline**, **block** and **inline-block**.

<https://developer.mozilla.org/en-US/docs/Web/CSS/display>

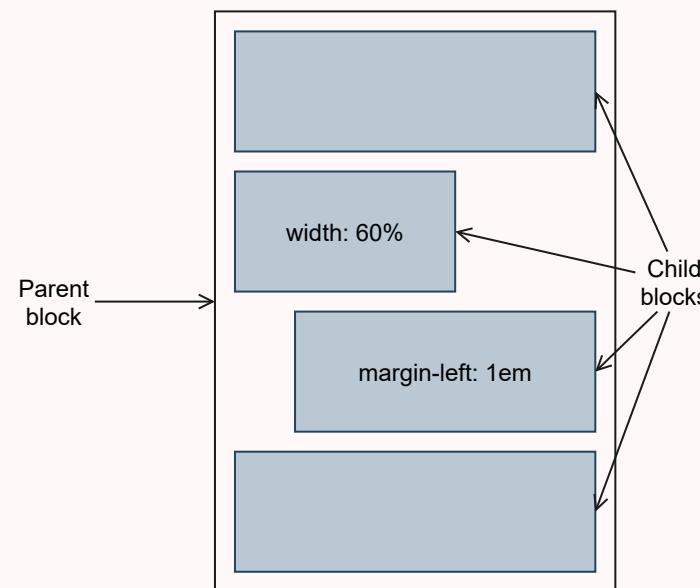
Block Elements

- Block elements are laid out one after the other, **vertically**.
- Respect **any** margins and padding.
- If no **width** is set, they will **expand** naturally to **fill** their parent container.
- If no **height** is set, they will **expand** naturally to **fit** their child elements and content.

Examples: p, div, h1-h6, section, article, header, footer, ...

```
img  
display
```

Blocks



Inline Elements

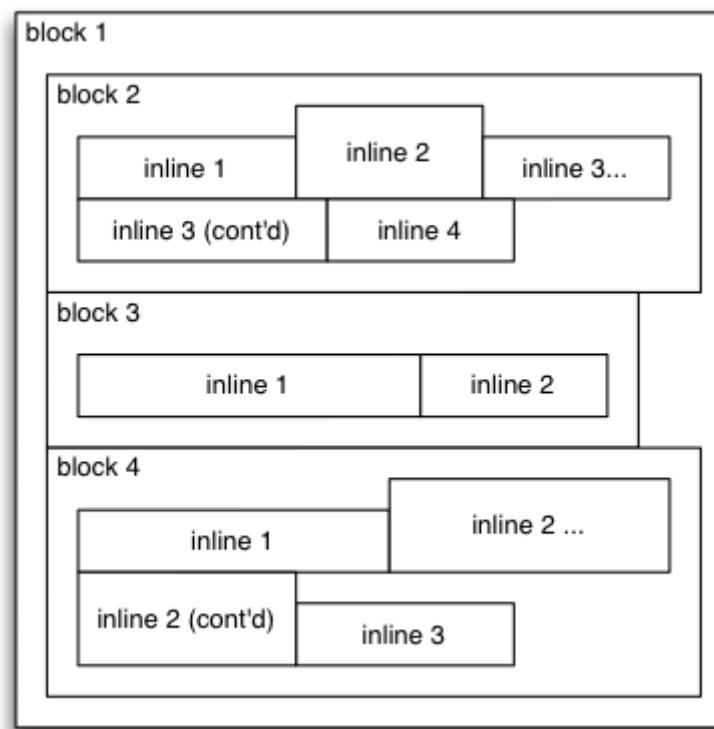
- Respect **left** and **right** margins and padding, but **not** top & bottom.
- **Cannot** have a width and height set.
- Allow other elements to sit to their left and right.
- Can break from one line to the next if there is no more space.

Examples: img, span, strong, a, em, ...

```
span  
display  
margin  
padding  
background-color
```

 Lorem ipsum dolor sit amet, consectetur
dipiscing elit. Aliquam gravida, massa ut vulputate iaculis,
libero nulla iaculis metus, in volutpat nisl neque eget libero.
Ut condimentum felis at mi accumsan ultrices. Cras eget
condimentum purus. Etiam tempor ipsum vitae suscipit
interdum.

Blocks and Inlines



Inline-Block Elements

- Inline elements that **behave** as block elements.
- Block elements that **stack** horizontally.

```
span  
display  
margin  
padding  
background-color
```

 Lorem ipsum dolor sit amet, consectetur

 dipiscing elit. Aliquam gravida, massa ut vulputate iaculis,
 libero nulla iaculis metus, in volutpat nisl neque eget libero.
 Ut condimentum felis at mi accumsan ultrices. Cras eget
 condimentum purus. Etiam tempor ipsum vitae suscipit
 interdum.

Display None

- Setting the **display** property to none, **removes** the element from the page completely.
- Different from making it invisible (with the *visibility* attribute).

```
display
```

Margin and Padding

- To change the margin and padding of an element we use the following properties: **margin-top**, **margin-right**, **margin-bottom**, **margin-left**, **padding-top**, **padding-right**, **padding-bottom** and **padding-left**.
- They all take a length as their value.

```
h1  
margin-top
```

- **Auto** A suitable margin is calculated by the browser (useful for centering).
- **Percentage** Calculated with respect to the width of the generated containing block (even for top/bottom).

Shorthands

To make it easier to define the margin and padding properties, shorthands can be used:

- Using **two values**, the **top/bottom** (vertical) and **left/right** (horizontal) margins are defined simultaneously.
- Using **three values**, the **top**, **horizontal** and **bottom** margins are defined.
- Using **four values**, the **top**, **right**, **bottom** and **left** values are defined (in that order i.e. **clockwise**).
- Using **one value**, **all values** are defined the same.

Shorthand Examples

```
h1  
margin
```

```
margin
```

```
padding
```

Common shorthand used to center the *body* element:

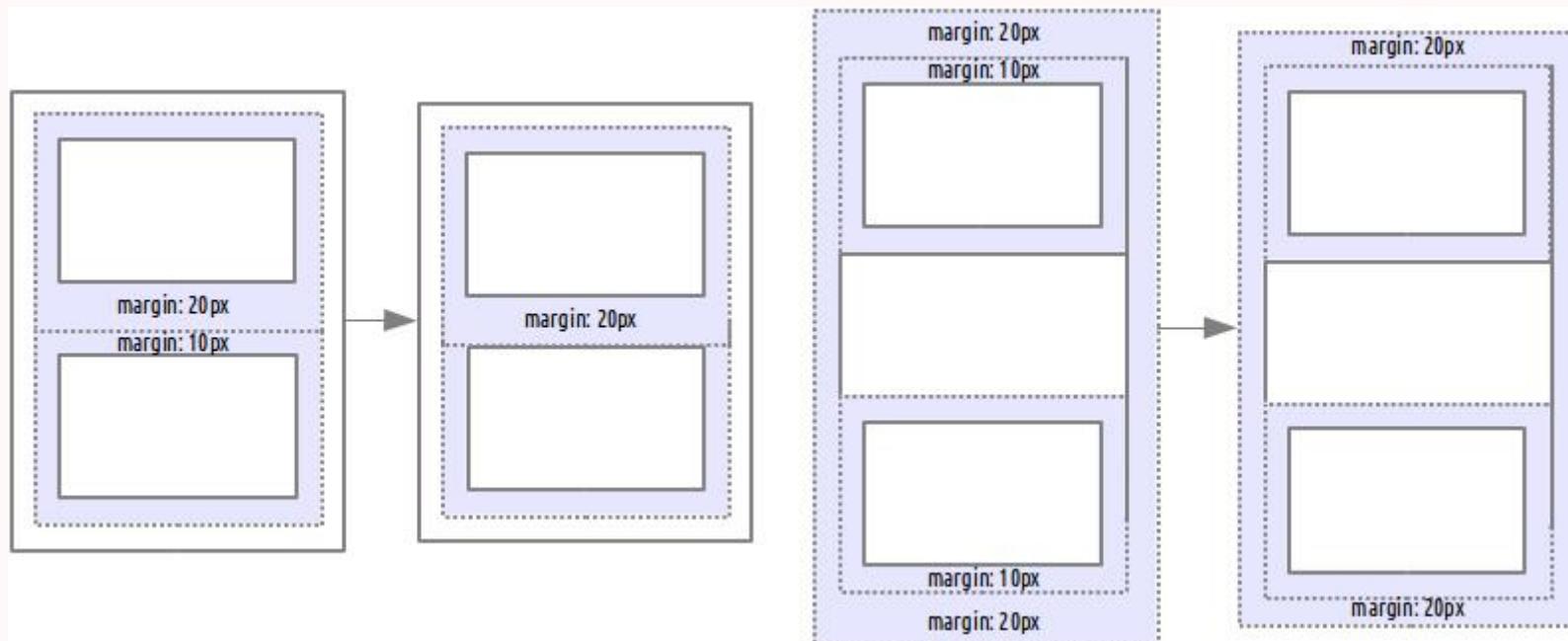
```
body  
margin
```

Margin Collapse

Adjacent margins collapse in three different cases

- The margins of **adjacent siblings** are collapsed.
- If there is no border, padding, inline content, or clearance to separate the margin-top of a block with the margin-top of its **first child** block, or no border, padding, inline content, height, min-height, or max-height to separate the margin-bottom of a block with the margin-bottom of its **last child**, then those margins collapse.
- If there is no border, padding, inline content, height, or min-height to separate a block's margin-top from its margin-bottom, then its top and bottom margins collapse.

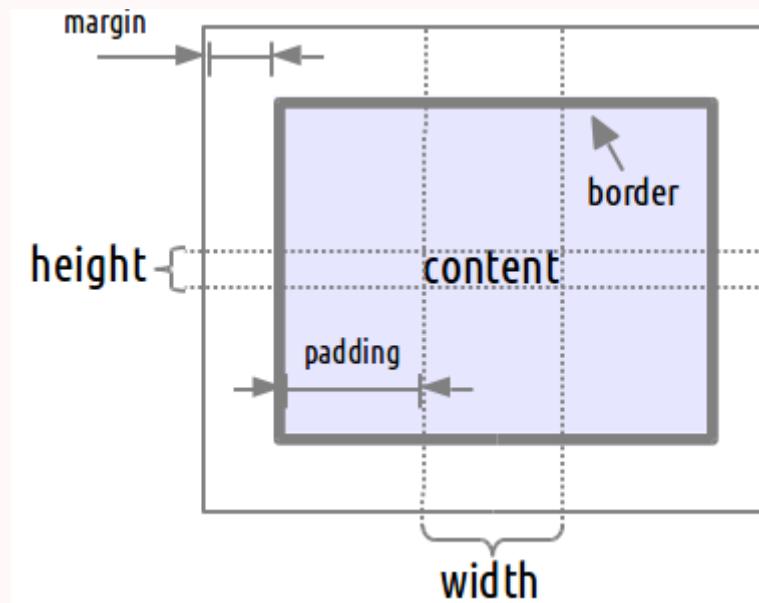
Margin Collapse Examples



Border

Element Border

An element border is a line that separates the padding from the margin.

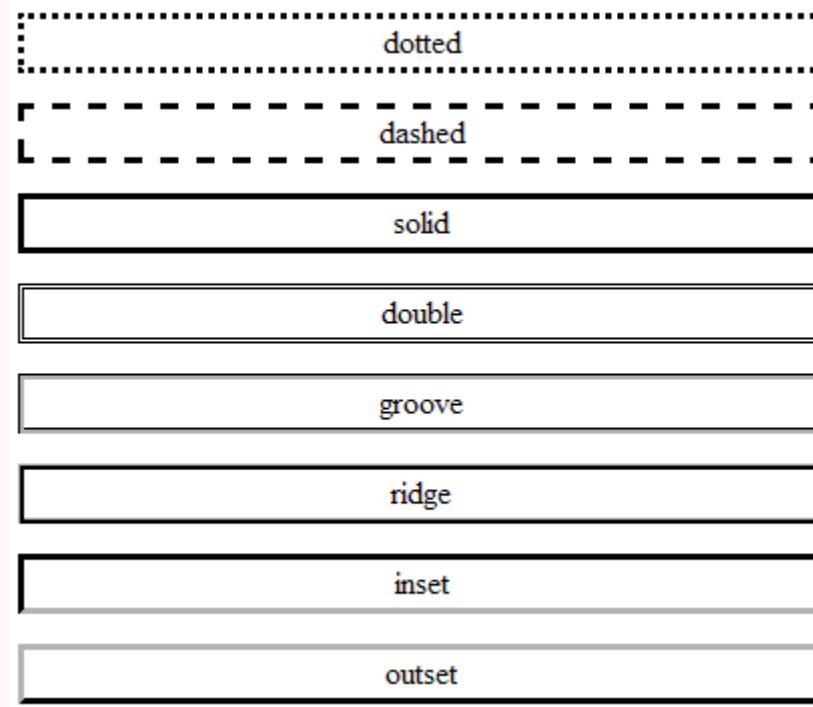


Border Style

- The **border-style** property specifies what kind of border to display. The following values are possible: **none**, **dotted**, **dashed**, **solid**, **double**, **groove**, **ridge**, **inset** and **outset**.
- We can set different border styles for each side using the properties: **border-top-style**, **border-right-style**, **border-bottom-style** and **border-left-style**.

```
border-style
```

Border Style Examples



Border Width

- The **border-width** property is used to specify the width of the border. Its value can be a length or a predefined value: **thin**, **medium**, or **thick**.
- We can set different border widths for each side using the properties: **border-top-width**, **border-right-width**, **border-bottom-width** and **border-left-width**.

```
border-left-width  
border-right-width
```

Border Color

- The **border-color** property is used to specify the color of the border.
- We can set different border colors for each side using the properties: **border-top-color**, **border-right-color**, **border-bottom-color** and **border-left-color**.

```
border-color
```

Shorthands

- As with the padding and margin properties we can use more than one value in the style, color and width properties to change the border of several sides at the same time.
- Using **two values**, the top/bottom and left/right border properties are defined simultaneously.
- Using **four values**, the **top**, **right**, **bottom** and **left** values are defined (in that order i.e. **clockwise**).
- Using **one value**, **all values** are defined the same.

`border-width`

Shorthands

- The **border** property allows us to define all border properties in one declaration.
- The properties that can be set, are (in order): **border-width**, **border-style**, and **border-color**.
- It does not matter if one of the values above is missing.

```
border
```

Border Radius

- The **border-radius** property is used to define how rounded border corners are.
- The curve of each corner is defined using one or two radii, defining its shape: circle or ellipse.
- We can set different border radius for each corner using the properties: **border-top-left-radius**, **border-top-right-radius**, **border-bottom-right-radius** and **border-bottom-left-radius**.
- Values can be a length or a percentage.
- If two radii are used, they are separated by a /.

Shorthands

- As with other properties we can use more than one value in the radius property to change the border radius of several corners at the same time.

The possible combinations are as follows:

- One value: single radius for the whole element
- Two values: **top-left-and-bottom-right** and **top-right-and-bottom-left**
- Three values: **top-left**, **top-right-and-bottom-left** and **bottom-right**
- Four values: **top-left**, **top-right**, **bottom-right**, **bottom-left**

Examples

```
div      div div      div div      div  
div      div div      div div      div
```

```
div  
background-color  
width      height  
margin      float  
  
border-radius      background-color  
border-radius      background-color  
border-radius      background-color  
border-radius      background-color  
border-radius      background-color  
border-radius      background-color
```



Background

Image

- Besides having a background color, elements can also have an image as background using the **background-image** property.
- This property accepts an URL as its value.

```
div  
background-image
```

Position

- The position of the background image can be set using the **background-position** property. This property receives two values.
- The first one can be **left**, **right**, **center** or a **length**.
- The second one can be **top**, **bottom**, **center** or a **length**.

```
div  
background-image  
background-position
```

Attachment

- Using the **background-attachment** property, we can specify if the background should or not scroll with the page or element.
- Possible values are **fixed** (in relation to the viewport), **scroll** (in relation to the element) and **local** (in relation to the content).
- Scroll is the default value.

```
div  
background-image  
background-position  
background-attachment
```

<https://css-tricks.com/almanac/properties/b/background-attachment/>

Repeat

We can also define if the background repeats along one or both axis with the **background-repeat** property. Possible values are **no-repeat**, **repeat-x**, **repeat-y** and **repeat**.

```
div  
background-image  
background-position  
background-attachment  
background-repeat
```

Clipping

- By default, background properties, like `background-color`, apply to the space occupied by the element, its padding and border.
- This can be changed using the `background-clip` property.
- The possible values are: `border-box` (default), `padding-box` (only content and border) and `content-box` (only content).

<https://css-tricks.com/almanac/properties/b/background-clip/>

Shorthands

- The **background** shorthand property sets all the background properties (including color) in one declaration.
- The properties that can be set, are: **background-color**, **background-position**, **background-size**, **background-repeat**, **background-origin**, **background-clip**, **background-attachment**, and **background-image**.
- It does not matter if one of the values above are missing.

```
div  
background
```

Lists

Markers

- Each item, in ordered and unordered lists, have left marks defining its position.
- We can change the markers of both types of lists using the **list-style-type** property.
- Some possible values for unordered lists are: **none**, **disc** (default), **circle** and **square**.
- For ordered lists we can use: **none**, **decimal** (default), **lower-alpha**, **lower-greek**, **lower-roman**, **upper-alpha** and **upper-roman**.

```
ul list-style-type  
ol list-style-type
```

Images as Markers

It is also possible to use an arbitrary image as the list marker:

```
div      ul  
list-style-image
```

Tables

Borders

To draw border around table elements we can use the **border** property that we've seen before:

```
table th td  
border
```

Collapse Borders

- Both tables and cells have borders.
- Specifying borders for both will result in a double border effect.
- To collapse borders from these two elements into one single border we can use the **border-collapse** property:

```
table  
border-collapse
```

Transforms

Transform

- The **transform** property modifies the coordinate space of the CSS visual formatting model. A space separated list of transforms, which are applied one after the others.
- The **transform-origin** property specifies the position of the transform origin. By default it is at the center of the element. It takes two values (x-offset and y-offset) that can be a length, a percentage or one of left, center, right, top and bottom.

Examples

```
div      div  div      div  div      div  
div      div  div      div  div      div
```

```
div  
margin  
float  
width      height  
  
transform      background-color  
transform      background-color  
transform      background-color  
transform      background-color  
transform      background-color  
transform      background-color  
transform      background-color
```

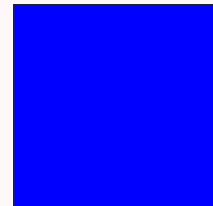


Transitions

Transitions

- Provide a way to control **animation speed** when changing CSS properties
- Instead of having property changes take effect immediately, you can cause the changes in a property to take place over a period of time.
- CSS transitions let you decide:
 - which properties to animate (**list**)
 - when the animation will start (**delay**)
 - how long the transition will last (**duration**)
 - how the transition will run (**timing function**)

Example



```
border-style  
border-width  
width  
height  
background-color  
transition
```

```
background-color  
width  
height  
transform
```

Positioning

The Flow

- By default, elements follow something called **the flow** of the document.
- In order to make page drawing easier for browsers, elements are always placed from **top to bottom** and **left to right**. Unless they are removed from the flow.

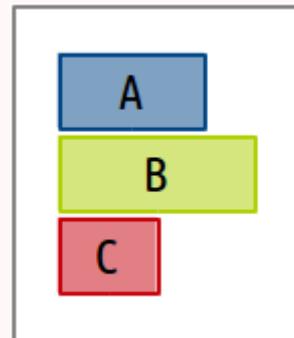
Position

The **position** property allows the developer to alter the way an element is positioned. There are 4 possible values:

- static
- relative
- fixed
- absolute

Position Static

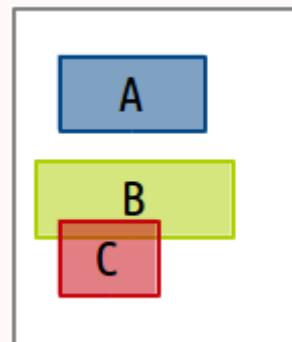
- The default value.
- The element keeps its place **in the document flow**.



position

Position Relative

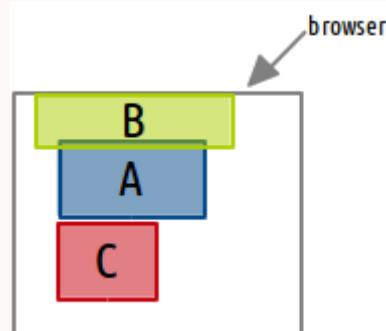
- The element keeps its position **in the flow**.
- But can be moved relatively to its static position using the properties: **top**, **right**, **bottom** and **left**.



```
position  
left  
top
```

Position Fixed

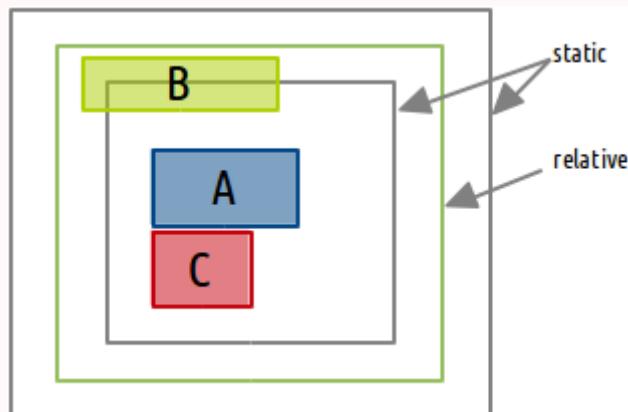
- The element is no longer a part of the flow.
- Can be positioned relatively to the browser window.
- Scrolling doesn't change the element's position.



```
position  
left  
top  
height
```

Position Absolute

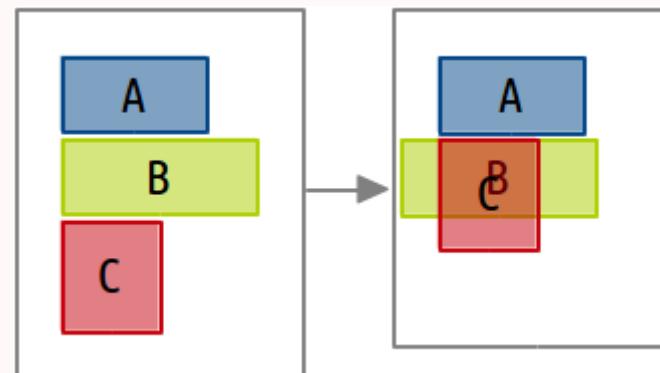
- No longer a part of the flow and scrolls with the page.
- Can be positioned relatively to its first non static parent.



```
position  
left  
top  
height
```

Float

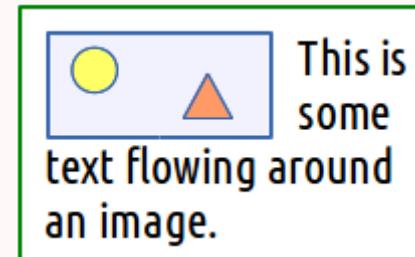
The **float** property removes an element from the document flow and shifts it to the **left** or to the **right** until it touches the edge of its containing box or another floated element.



float

Floats and Text

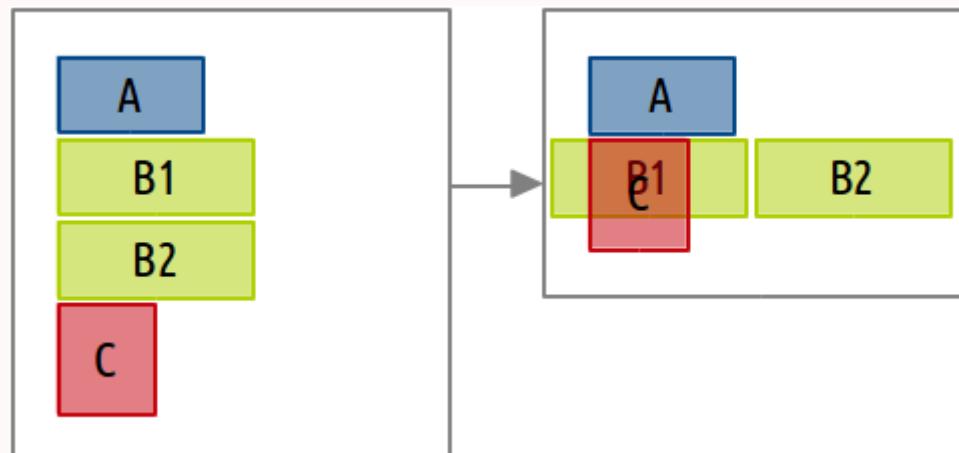
Text always flows around floated elements. This is useful to make text that flows around images.



```
img  
float
```

Multiple Floats

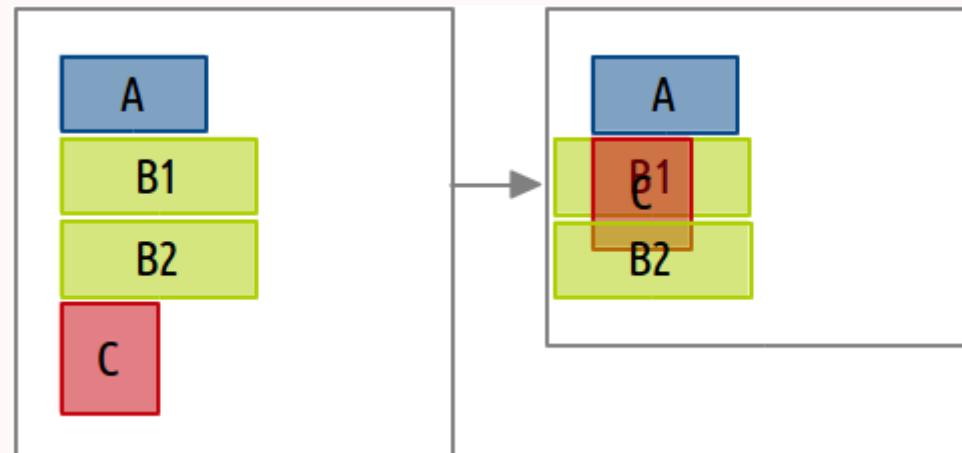
Floats go right or left until they find another float or the parent container.



`float`

Clear

- The **clear** property indicates if an element can be next to floating elements that precede it or must be moved down.
- Values can be **left**, **right** or **both**.



```
float  
clear
```

Ordering

- When elements are positioned outside the normal flow, they can overlap other elements. The **z-index** property specifies the stack order of an element.
- An element with greater stack order is always in front of an element with a lower stack order.

z-index

By default, the elements are stacked following the order they are declared in the HTML.

Overflow

- The **overflow** property specifies the behavior of an element when its contents don't fit its specified size.
- Possible values are:
 - **visible**: The overflow is not clipped. It renders outside the element's box. This is default.
 - **hidden**: The overflow is clipped, and the rest of the content will be invisible.
 - **scroll**: The overflow is clipped, but a scroll-bar is added to see the rest of the content.
 - **auto**: If overflow is clipped, a scroll-bar should be added to see the rest of the content.

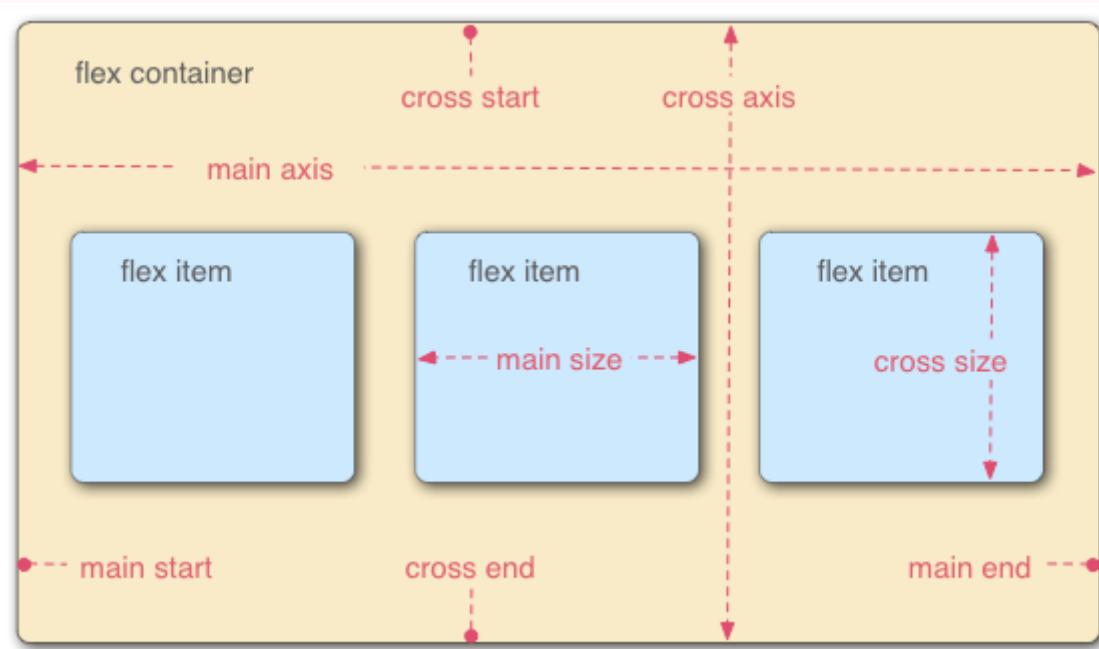
Flexbox

Flexbox

- A direction agnostic alternative to the box model layout model.
- Flexbox provides block level arrangement of **parent** and **child** elements that are **flexible** to adapt to display size.
- Flexbox items **cannot** be floated.
- The flex container's margins **do not collapse** with the margins of its contents.

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

Flexbox Vocabulary



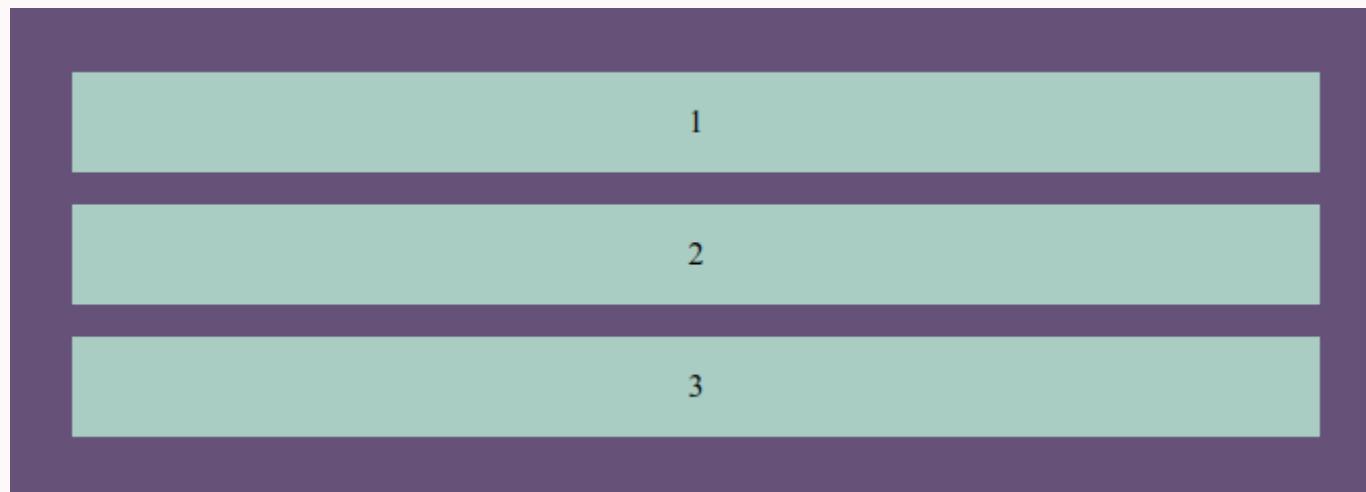
Running Example

```
div
  div          div
  div          div
  div          div
  div
```

```
background-color
padding
```

```
color
text-align
margin
padding
background-color
```

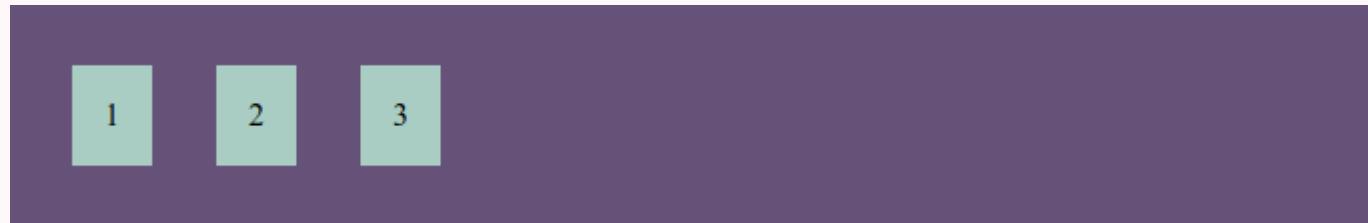
Running Example



Flex

Changing the *display* property of the container to *flex* transforms the contained items into flexboxes.

`display`

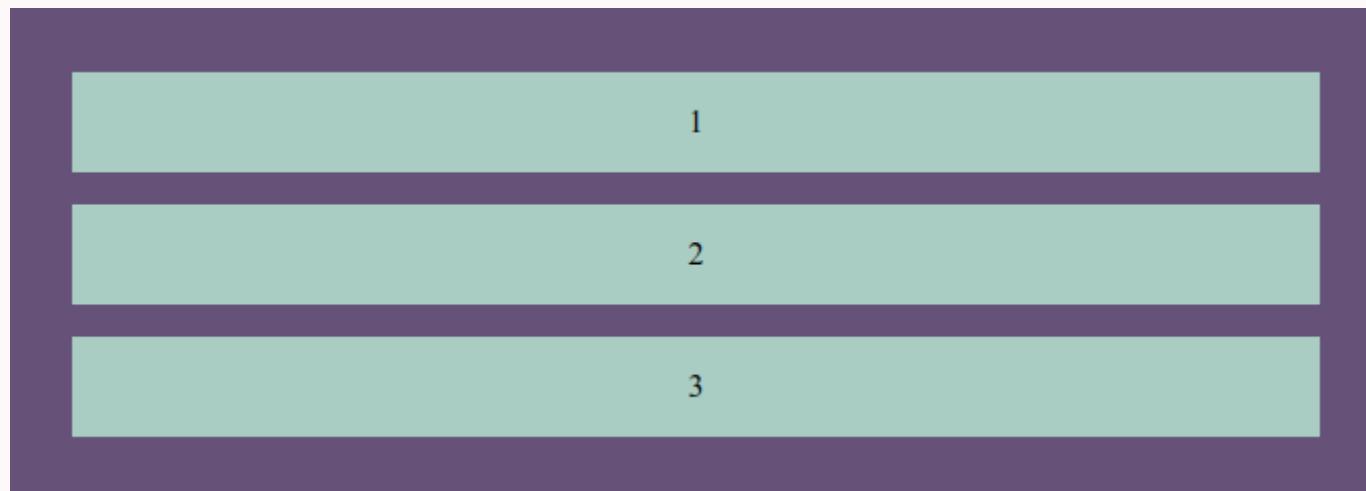


By default the main axis is horizontal from left to right.

Flex Direction

We can change the direction of the main axis by changing the *flex-direction* property of the container to: **row**, **row-reverse**, **column** or **column-reverse**.

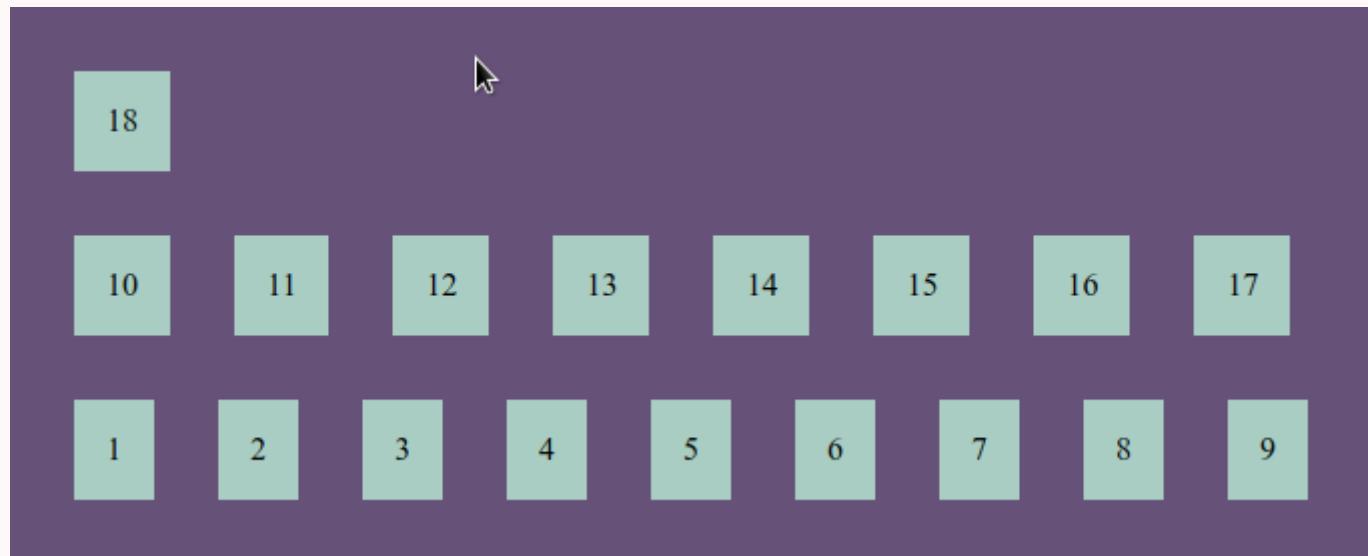
`flex-direction`



Flex Wrap

The *flex-wrap* property allows us to specify how items should wrap when changing lines: **nowrap**, **wrap**, **wrap-reverse**. The default is **nowrap**.

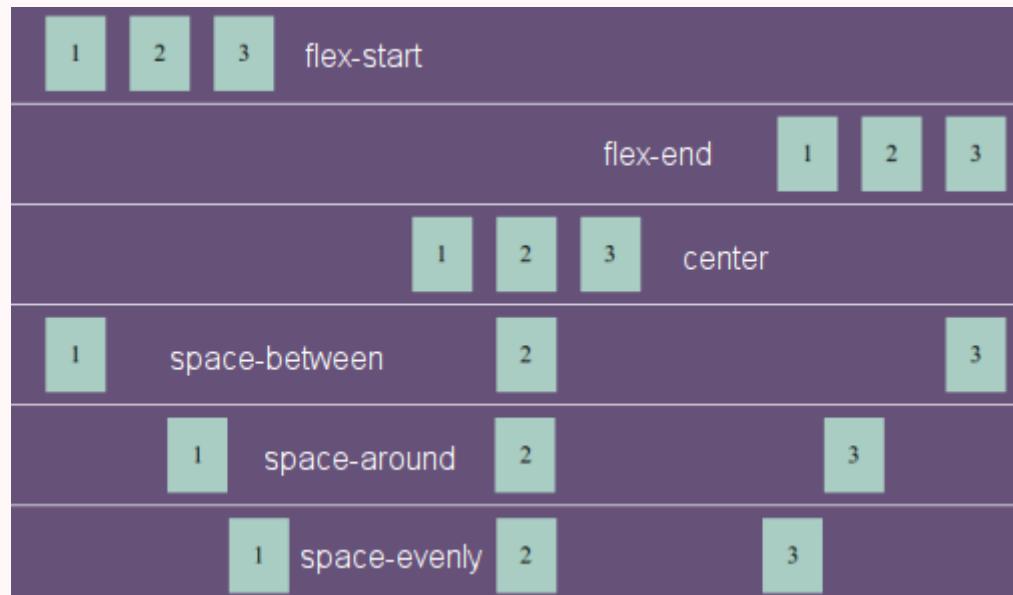
flex-wrap



Justify Content

The *justify-content* property defines the alignment along the main axis allowing the distribution of extra space: **flex-start**, **flex-end**, **center**, **space-around**, **space-between**, **space-evenly**. The default is **flex-start**.

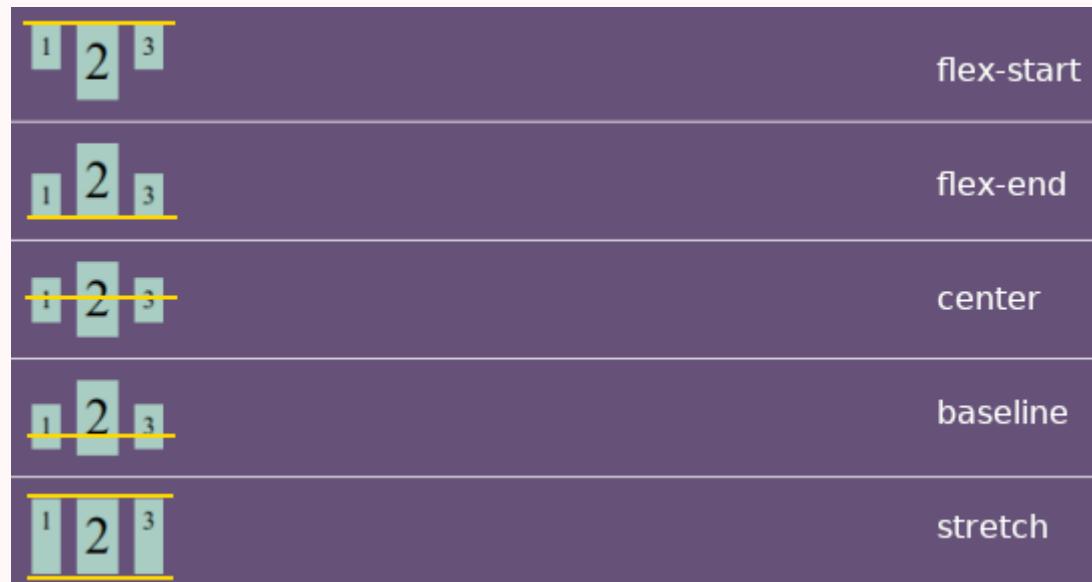
justify-content



Align Items

The `align-items` property defines the default behaviour for how flex items are laid out along the cross axis on the current line: **flex-start**, **flex-end**, **center**, **baseline**, **stretch**. The default is **stretch**.

`align-items`



Order

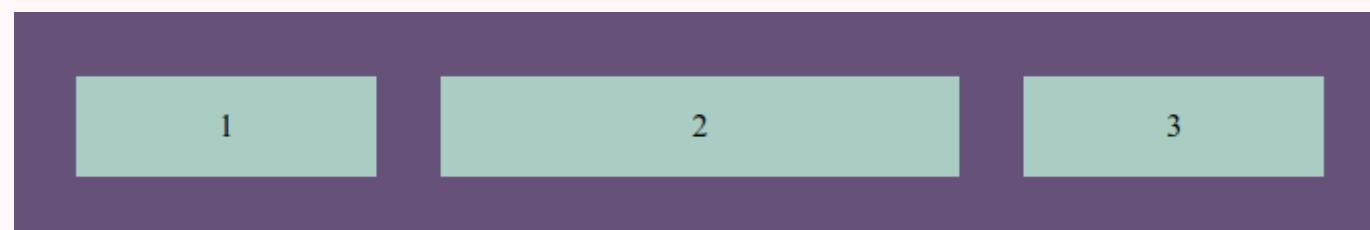
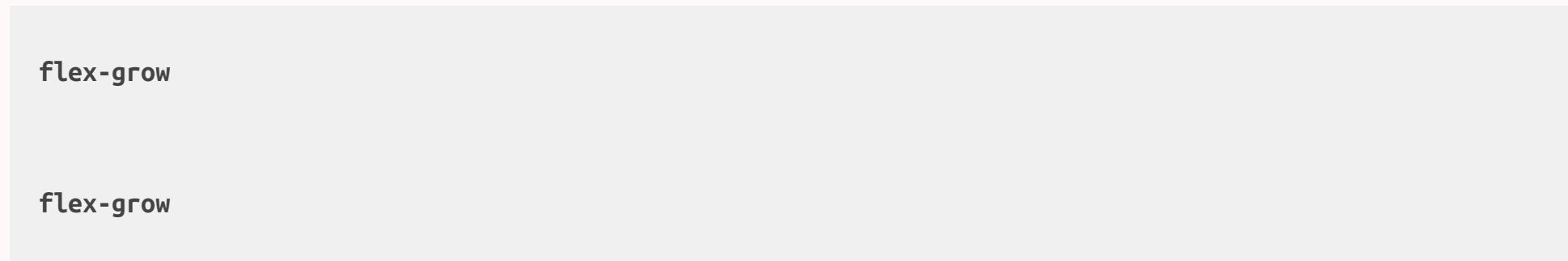
The **order** property alters the order in which a flex item is layed out in its container.

order



Grow and Shrink

The *flex-grow* and *flex-shrink* properties define the ability for a flex item to grow, if there is extra space, or shrink, if there isn't enough. They accept a unitless value that serves as a proportion. The default is **1** for both properties.



Align Self

Allows the alignment specified by align-items to be overridden for individual flex items. The default value is **auto** meaning that items follow the alignment specified by align-items.

```
align-items
```

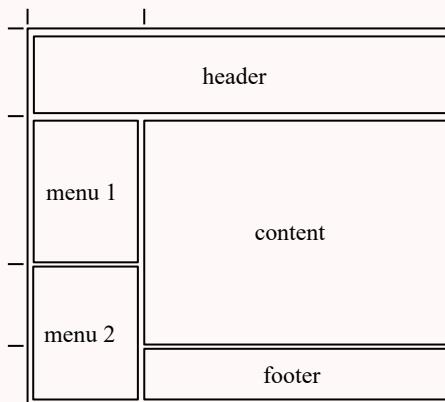
```
align-self
```

All items are aligned as **flex-start** except the second one that is **center**-aligned.

Grid

Grid

- A grid layout enables us to align elements into **columns** and **rows**.
- A grid container's child elements can position themselves so they **overlap** and **layer**.



<https://css-tricks.com/snippets/css/complete-guide-grid/>

Running Example

```
div
  div
  div
  div
  div
    div
  div
  div
```

```
background-color
padding
```

```
color
text-align
margin
padding
background-color
```

Running Example

Header
Menu 1
Menu 2
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ante mauris, sagittis ac consectetur a, sodales sed massa. Sed rutrum convallis commodo. Suspendisse consequat neque et erat condimentum vestibulum. Sed et tristique felis. Nunc sit amet convallis arcu, sed vulputate diam. Donec bibendum tellus ac nunc pretium, hendrerit congue leo tristique. Fusce malesuada lorem sem, a tincidunt augue mattis mattis. In viverra augue efficitur tincidunt imperdiet. Quisque neque tellus, tristique blandit nibh ut, aliquam mollis felis. Proin ornare ex lorem, sit amet bibendum tellus ultrices sit amet. Ut vitae urna nec massa condimentum auctor eget nec est. Quisque id dui tellus.</p>
Footer

Grid

Changing the *display* property of the container to *grid* transforms the container into a grid layout.

```
display
```

By default there is only one column.

Grid Templates

The `grid-template-columns` and `grid-template-rows` properties allow us to define the number and size of the columns and rows of our table.

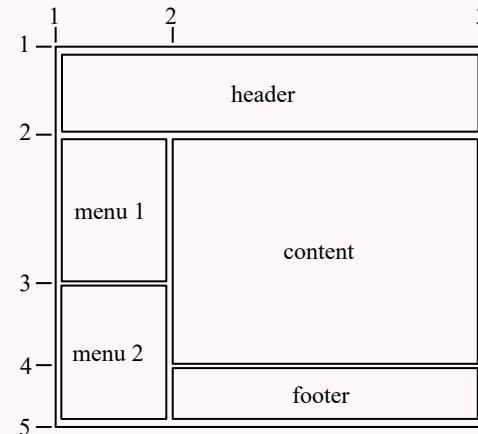
Sizes can be defined as **auto**, a **length**, a **percentage** or a **fraction** of the free space (using the `fr` unit).

```
grid-template-columns  
grid-template-rows
```

Header	Menu 1
Menu 2	<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam ante mauris, sagittis ac consectetur a, sodales sed massa. Sed rutrum convallis commodo. Suspendisse consequat neque et erat condimentum vestibulum. Sed et tristique felis. Nunc sit amet convallis arcu, sed vulputate diam. Donec bibendum tellus ac nunc pretium, hendrerit congue leo tristique. Fusce malesuada lorem sem, a tincidunt augue mattis mattis. In viverra augue efficitur tincidunt imperdiet. Quisque neque tellus, tristique blandit nibh ut, aliquam mollis felis. Proin ornare ex lorem, sit amet bibendum tellus ultrices sit amet. Ut vitae urna nec massa condimentum auctor eget nec est. Quisque id dui tellus.</p>
Footer	

Numerical Names

By default, grid lines are assigned numerical values.



Assigning Location

We can assign a **location** to an item within the grid by referring to specific grid lines using the *grid-column-start*, *grid-column-end*, *grid-row-start* and *grid-row-end* properties.

```
grid-column-start  
grid-column-end  
grid-row-start  
grid-row-end
```

Values can be the **numerical** default names of the grid lines or a name assigned by us.

The end values can also be the number of rows or columns to span. By default these values are a span of one.

```
grid-column-end  
grid-row-end
```

Location Shorthand

The *grid-column* and *grid-row* properties can be used as a shorthand for assigning the location of an item. Each one of them receives two values separated by a forward slash (start / end).

The *grid-area* property can be used as a shorthand for the four values at once: *row-start* / *column-start* / *row-end* / *column-end*.

```
grid-area
```

```
grid-column  
grid-row
```

```
grid-column  
grid-row
```

```
grid-column  
grid-row
```

```
grid-column  
grid-row
```

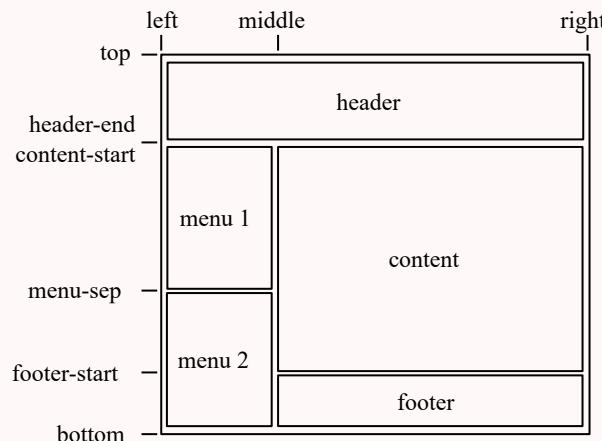
Location Result

Header	
Menu 1	
Menu 2	<p>Lore ipsum dolor sit amet, consectetur adipiscing elit. Etiam ante mauris, sagittis ac consectetur a, sodales sed massa. Sed rutrum convallis commodo. Suspendisse consequat neque et erat condimentum vestibulum. Sed et tristique felis. Nunc sit amet convallis arcu, sed vulputate diam. Donec bibendum tellus ac nunc pretium, hendrerit congue leo tristique. Fusce malesuada lorem sem, a tincidunt augue mattis mattis. In viverra augue efficitur tincidunt imperdiet. Quisque neque tellus, tristique blandit nibh ut, aliquam mollis felis. Proin ornare ex lorem, sit amet bibendum tellus ultrices sit amet. Ut vitae urna nec massa condimentum auctor eget nec est. Quisque id dui tellus.</p>
Footer	

Grid Line Names

When defining the grid template, we can assign names to the grid lines. A line can have more than one name.

```
grid-template-columns  
grid-template-rows  
  
grid-area
```



Grid Template Areas

By giving names to items using the *grid-area* property, we can define a grid template in a more visual fashion.

Any number of adjacent periods can be used to declare a single empty cell.

```
grid-template-columns  
grid-template-rows  
  
grid-template-areas  
  
grid-area  
grid-area  
grid-area  
grid-area  
grid-area
```

Precedence, Inheritance and Specificity

Example

The text becomes red but the links are still blue. Why?

```
div
  p This is some text with a link a
div
```

```
div
  color
```

Defaults

- Each browser has **its own** set of default values for the properties of each HTML element.
- These defaults are very similar between browsers but the little differences make cross-browser development harder.

Tip: There are several reset CSS available that redeclare each default value to have the same value in every browser.

Inherit

- A special value that can be used in almost every property.
- When a property is set to **inherit**, the value of that property is **inherited** from the element's **parent**.

```
div id menu
  h1 Menu  h1    -- inherits the blue color from the div --
  div
```

```
h1
  color
```

```
color
```

I Get it Now

- In most browsers the **anchor** color is defined as **blue**.
- On the other hand, the **paragraph** color is defined as **inherit**.

```
div
  p This is some text with a a link a p
div
```

```
a
color
```

```
p
color
```

```
div
color
```

Specificity

```
div id menu
  p What is my color  p
    div
```

```
      p
    color
```

```
div p
  color
```

Specificity

```
div id menu
  p What is my color  p
div
```

```
  p
color
```

```
div p
color
```

Green! Because the first rule is more specific than the second one.

Calculating Specificity

- The specificity of a rule is defined as 4 values (a, b, c, d).
- Each one of them is incremented when a certain type of selector is used:
 - **d**: Element, Pseudo Element
 - **c**: Class, Pseudo class, Attribute
 - **b**: Id
 - **a**: Inline Style

Specificity Examples

- p: 1 element – (0,0,0,1)
- div: 1 element – (0,0,0,1)
- #sidebar: 1 id – (0,1,0,0)
- div#sidebar: 1 element, 1 id – (0,1,0,1)
- div#sidebar p: 2 elements, 1 id – (0,1,0,2)
- div#sidebar p.bio: 2 elements, 1 class, 1 id – (0,1,1,2)

Specificity Calculator: <http://specificity.keegan.st>

Specificity Rules

- Rules with a bigger **a** value are **more specific**.
- If the **a** value is the same for both rules, the **b** value is used for comparison.
- If still needed, the **c** and **d** values are used.

Cascading

- The rule to be applied is selected using the following rules in order:
 - Origin (author, user, default)
 - Specificity (bigger is better)
 - Position (last is better)
- Origin Explanation:
 - **author:** The CSS rules defined by the page developer
 - **user:** User defined preferences
 - **default:** Browser defaults

CSS Vars

CSS Vars

- Entities that contain specific values to be reused throughout a document.
- Set using custom property notation:

```
body  
--main-bg-color  
--default-margin
```

- Accessed using the `var()` function:

```
body header  
margin
```

CSS Vars Inheritance

- CSS vars can be inherited.
- If no value is set for a var on a given element, the value of its parent is used.

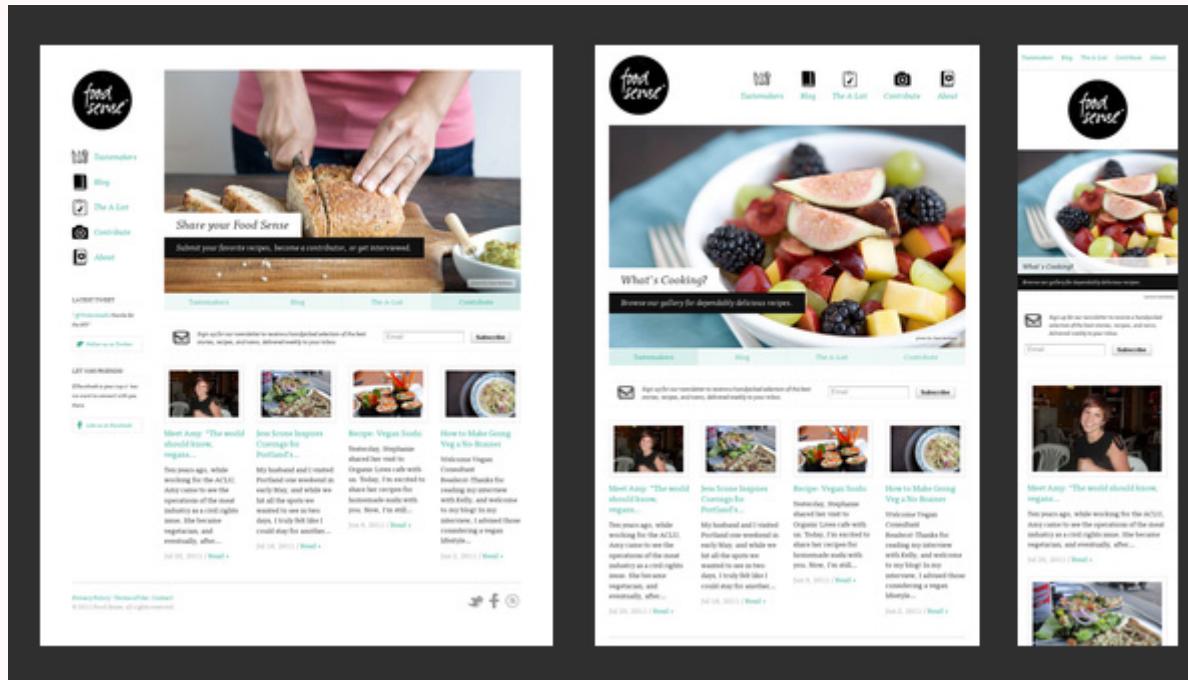
```
section
  header
    h1      h1
    h2      h2
  header
section
```

```
section --text-color
h1   --text-color
header color
```

Responsive Design

Responsive Design

Responsive web design is a way of making websites that works effectively on both desktop browsers and the myriad of mobile devices on the market.



<http://foodsense.is/> image taken from <http://designmodo.com/responsive-design-examples/>

Responsive vs Adaptive

Adaptive Design : Multiple **fixed width** layouts

Responsive Design : Multiple **fluid** grid layouts

Mixed Approach : Multiple fixed width layout for larger screens, multiple fluid layout for smaller screens.

Viewport

Pages optimized for a variety of devices must include a meta viewport element in the head of the document. A meta viewport tag gives the browser instructions on how to control the page's dimensions and scaling.

meta

- *width=device-width* matches the screen's width in device independent pixels.
- *initial-scale=1** establishes a 1:1 relationship between CSS pixels and device independent pixels.

Learn more: https://developer.mozilla.org/en/docs/Mozilla/Mobile/Viewport_meta_tag and a tale of two viewports [part 1](#) and [part 2](#)

Media Queries

A **media-query** is composed of a **media type** and/or a number of **media features**.

They can be used when linking to a CSS file from HTML or directly in the CSS code.

link

media

display

Media Types

The media type indicates the type of media the CSS is to be applied to.

- **all** - suitable for all devices.
- **print** - intended for paged material and for documents viewed on screen in print preview mode.
- **screen** - intended primarily for color computer screens.
- **speech** - intended for speech synthesizers (aural in CSS2).

link

Media Features

- **min-width** width over the value defined in the query.
- **max-width** width under the value defined in the query.
- **min-height** height over the value defined in the query.
- **max-height** height under the value defined in the query.
- **orientation=portrait** height is greater than or equal to the width.
- **orientation=landscape** width is greater than the height.

[link](#)

Parentheses are required around expressions; failing to use them is an error.

Logical Operators

- **and** used for combining multiple media features together
- **comma-separated** lists behave as the logical operator **or**
- **not** applies to the whole media query and returns true if the media query would otherwise return false

link

Learn more:

https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media_queries

Vendor Prefixes

Vendor Prefixes

While the specification of selectors, properties and values are still being finalized, it is normal for browsers to go through an **experimentation** period.

Browsers might also have **proprietary** extensions to the CSS standard.

In order to accommodate the release of vendor-specific extensions, the CSS specifications define a specific format that vendors should follow:

```
-webkit-border-radius  
-moz-border-radius  
border-radius
```

Prefixes: **-webkit-** (chrome, safari), **-moz-** (firefox), **-o-** (opera), **-ms-** (internet explorer), ...

Check browser support: <http://caniuse.com/>

Validation

<http://jigsaw.w3.org/css-validator/>

Extra stuff

- Frameworks: [Ink](#), [Bootstrap](#), [Flat UI](#), [Pure](#)
- Reset: [CSS Reset](#)
- Fonts: [Google Fonts](#)
- Advanced/Experimental: [Shadows](#), [Animations](#)
- Playgrounds: [JSFiddle](#), [CodePen](#)
- Pre-processors: [Less](#), [Sass](#)
- Information: [Google Web Essentials](#), [Mozilla Developer Network](#)
- Icons: [Font Awesome](#)