Cascading Style Sheets - CSS - Reference

By John Bentley

Overview

The latest Official Definition of Cascading Style Sheets (CSS):

(W3C The Latest, "CSS Snapshot - The Latest", https://www.w3.org/TR/CSS/#css) https://www.w3.org/TR/CSS/#css

See also:

(Bentley 2018, "CSS 2.1 Core - Quick Reference",
\\Atlas\Users\John\Documents\Sda\Info\Web\KB\Css\QuickReference\CSS2.1Core-QuickReference.docx)
\\Atlas\Users\John\Documents\Sda\Info\Web\KB\Css\QuickReference\CSS2.1Core-QuickReference.docx

Selectors

Overview

Pattern	Represents	Description	Level
Universal selector			
*	any element	Universal selector	2
Common			
E	an element of type E	Type selector	1
.warning	any whose class is "warning"		
E.warning	an E element whose class is "warning" (the document language specifies how class is determined).	Class selectors	1
#myID	any element with ID equal to "myid".		
E#myid	an E element with ID equal to "myid".	ID selectors	1
Combinators			
E F	an F element descendant of an E element	Descendant combinator	1
E > F	an F element child of an E element	Child combinator	2
E ~ F	an F element preceded by an E element (sibling)	Subsequent-sibling combinator	3
E + F	an F element immediately preceded by an E element (sibling)	Next-sibling combinator	2
Attribute Selectors			
E[foo]	an E element with a "foo" attribute	Attribute selectors	2
E[foo="bar"]	an E element whose "foo" attribute value is exactly equal to "bar"	Attribute selectors	2
E[foo~="bar"]	an E element whose "foo" attribute value is a list of whitespace-separated values, one of which is exactly equal to "bar"	Attribute selectors	2
E[foo^="bar"]	an E element whose "foo" attribute value begins exactly with the string "bar"	Attribute selectors	3
E[foo$="bar"]	an E element whose "foo" attribute value ends exactly with the string "bar"	Attribute selectors	3
E[foo*="bar"]	an E element whose "foo" attribute value contains the substring "bar"	Attribute selectors	3
E[foo|="en"]	an E element whose "foo" attribute has a hyphen-separated list of values beginning (from the left) with "en"	Attribute selectors	2
Language			

E:lang(fr)	an element of type E in language "fr" (the document language specifies how language is determined)	The :lang() pseudo-class	2
Structural pseudo-classes			
E:root	an E element, root of the document	Structural pseudo-classes	3
E:nth-child(n)	an E element, the n-th child of its parent	Structural pseudo-classes	3
E:nth-last-child(n)	an E element, the n-th child of its parent, counting from the last one	Structural pseudo-classes	3
E:nth-of-type(n)	an E element, the n-th sibling of its type	Structural pseudo-classes	3
E:nth-last-of-type(n)	an E element, the n-th sibling of its type, counting from the last one	Structural pseudo-classes	3
E:first-child	an E element, first child of its parent	Structural pseudo-classes	2
E:last-child	an E element, last child of its parent	Structural pseudo-classes	3
E:first-of-type	an E element, first sibling of its type	Structural pseudo-classes	3
E:last-of-type	an E element, last sibling of its type	Structural pseudo-classes	3
E:only-child	an E element, only child of its parent	Structural pseudo-classes	3
E:only-of-type	an E element, only sibling of its type	Structural pseudo-classes	3
E:empty	an E element that has no children (including text nodes)	Structural pseudo-classes	3
Links			
E:link E:visited	an E element being the source anchor of a hyperlink of which the target is not yet visited (:link) or already visited (:visited)	The link pseudo-classes	1
E:target	an E element being the target of the referring URI	The target pseudo-class	3
User action pseudo-classes			
E:active E:hover E:focus	an E element during certain user actions	The user action pseudo-classes	1 and 2
UI element states pseudo-classes			
E:enabled E:disabled	a user interface element E which is enabled or disabled	The UI element states pseudo-classes	3

E:checked	a user interface element E which is checked (for instance a radio-button or checkbox)	The UI element states pseudo-classes	3
The first x pseudo-elements			
E::first-line	the first formatted line of an E element	The ::first-line pseudo-element	1
E::first-letter	the first formatted letter of an E element	The ::first-letter pseudo-element	1
Generated Content pseudo-elements			
E::before	generated content before an E element	The ::before pseudo-element	2
E::after	generated content after an E element	The ::after pseudo-element	2
Negation			
E:not(s)	an E element that does not match simple selector s	Negation pseudo-class	3

(W3C The Latest, "Selectors Level 3", https://www.w3.org/TR/selectors-3/), "Selectors", https://www.w3.org/TR/selectors-3/#selectors

Match multiple classes.

```
// HTML
<div class="module accordion expand"></div>

// CSS. Will only match if element has both
.accordion.expand { }
```

(Coyier 2011, "Multiple Attribute Values", https://css-tricks.com/multiple-attribute-values/) https://css-tricks.com/multiple-attribute-values/

Match multiple values of the same attribute in general by concatenating the whitespace seperated slector.

```
// HTML
<div data-type="module accordion expand"></div>

// CSS
[data-type~="accordion"][data-type~="expand"] {

}
```

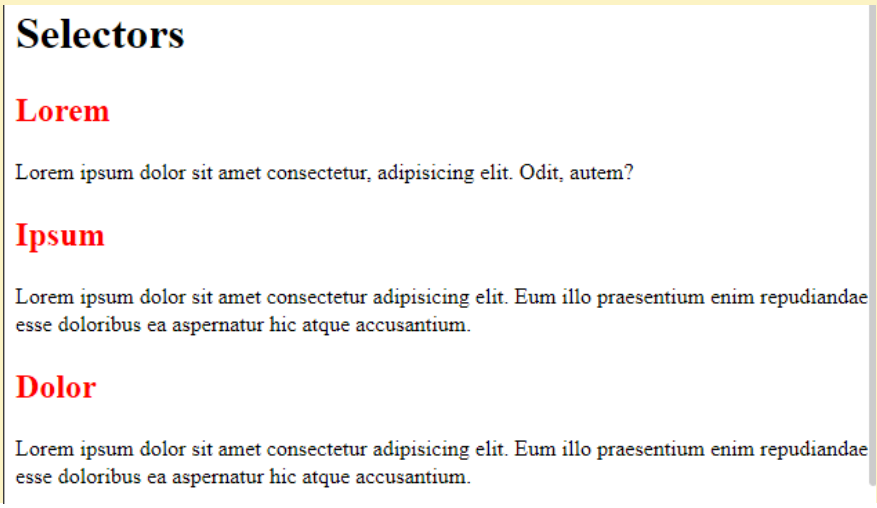
(Coyier 2011, "Multiple Attribute Values", https://css-tricks.com/multiple-attribute-values/) https://css-tricks.com/multiple-attribute-values/

Details

Sibling combinators

Preceded by sibling combinator.

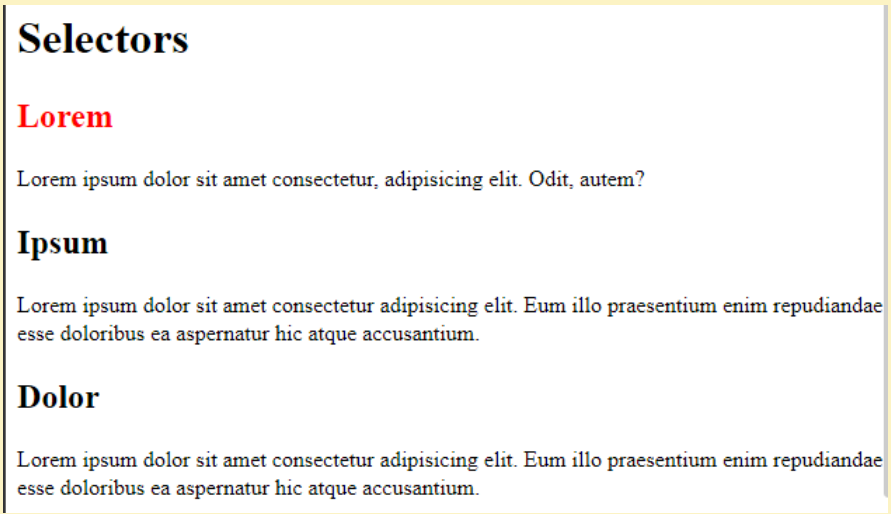
```
h1 ~ h2 {
  color: red;
}
```



C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\selectors.xhtml

Immediately preceded by sibling combinator.

```
h1 + h2 {
  color: red;
}
```



C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\selectors.xhtml

Structural pseudo-classes, nth notation.

Formally, `:nth-child(an+b)` notation: "represents an element that has `an+b-1` siblings before it in the document tree, for any positive integer or zero value of n". There are also the following features:

- Siblings are not required to have a parent.
- a and b must be integers (positive, zero, or negative).
- The sibling index starts at 1.

(W3C The Latest, "Selectors Level 3", https://www.w3.org/TR/selectors-3/), " :nth-child() pseudo-class", https://www.w3.org/TR/selectors-3/#nth-child-pseudo

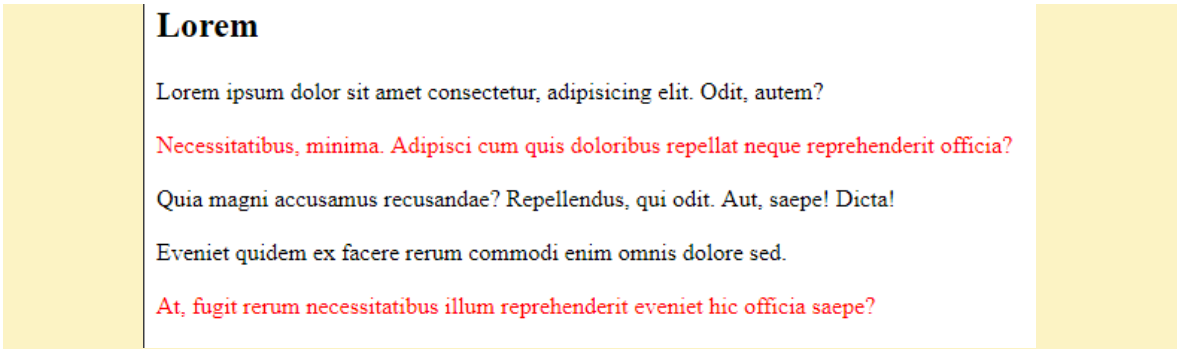
In the following examples, the given XHTML will be:

```
...

<main>
  <h1>Selectors</h1>

<h2 id="lorem">Lorem</h2>

<section>
  <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Odit, autem?</p>

  <p>Necessitatibus, minima. Adipisci cum quis doloribus repellat neque reprehenderit
officia?</p>

  <p>Quia magni accusamus recusandae? Repellendus, qui odit. Aut, saepe! Dicta!</p>

  <p>Eveniet quidem ex facere rerum commodi enim omnis dolore sed.</p>

  <p>At, fugit rerum necessitatibus illum reprehenderit eveniet hic officia saepe?</p>
</section>

<h2 id="ispum">Ipsum</h2>

<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Eum illo praesentium enim
repudiandae esse doloribus ea aspernatur hic atque accusantium.</p>

<p>Vel modi a tenetur eius! Quis tempore repudiandae, sequi iure quaerat sapiente ut
necessitatibus nesciunt. A nostrum expedita amet pariatur!</p>

...
```

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\selectors.xhtml

> [This] effectively divides the element's children into groups of *a* elements (the last group taking the remainder), and selecting the *b*th element of each group.

```
// The CSS
p:nth-child(3n+2) {
  color: red;
}

// Outputs
```



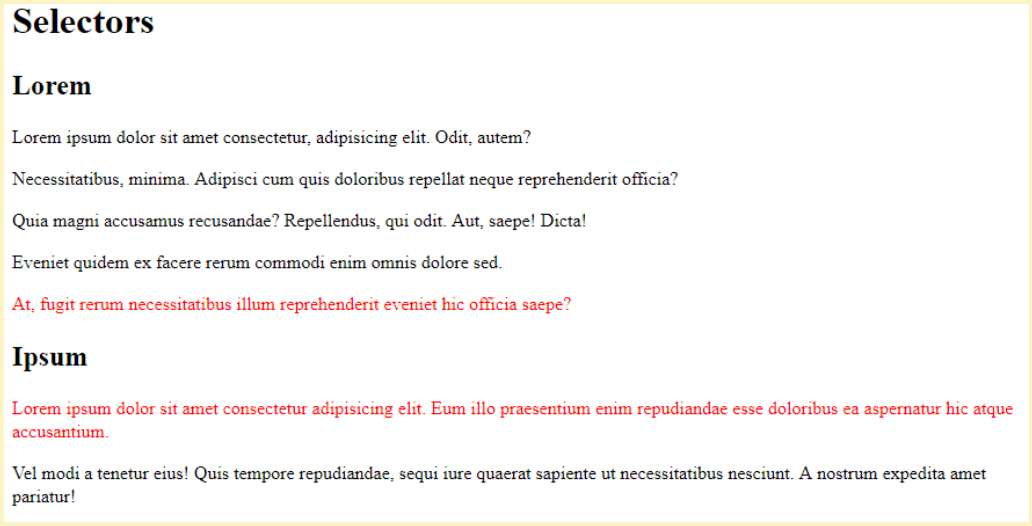
(W3C The Latest, "Selectors Level 3", https://www.w3.org/TR/selectors-3/), " :nth-child() pseudo-class", https://www.w3.org/TR/selectors-3/#nth-child-pseudo
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\selectors.xhtml

> When a=0, the *a*n part need not be included ... In this case the syntax simplifies to `:nth-child(b)`.

In effect `:nth-child(b)` will select the *b*th sibling and not repeat.

Note when a type is specified, as in `p:nth-child(b)`, this selects the *b*th sibling regardless of type, if that sibling is of type p.

```
p:nth-child(5) {
  color: red;
}

// Outputs
```

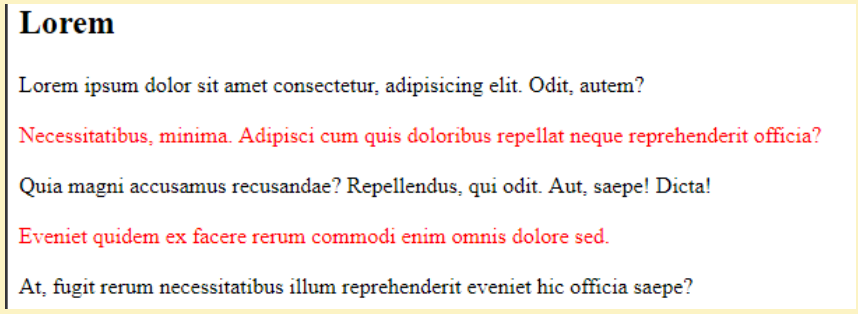



```
// Here the paragraph following the Ipsum h1 is the 5th sibling counting as follows:
// 1. h1
// 2. h2
// 3. section
// 4. h2
// 5. p (our selected sibling).
```

(W3C The Latest, "Selectors Level 3", https://www.w3.org/TR/selectors-3/), " :nth-child() pseudo-class", https://www.w3.org/TR/selectors-3/#nth-child-pseudo
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\selectors.xhtml

> If *b*=0, then every *a*th element is picked. In such a case, the +*b* (or -*b*) part may be omitted

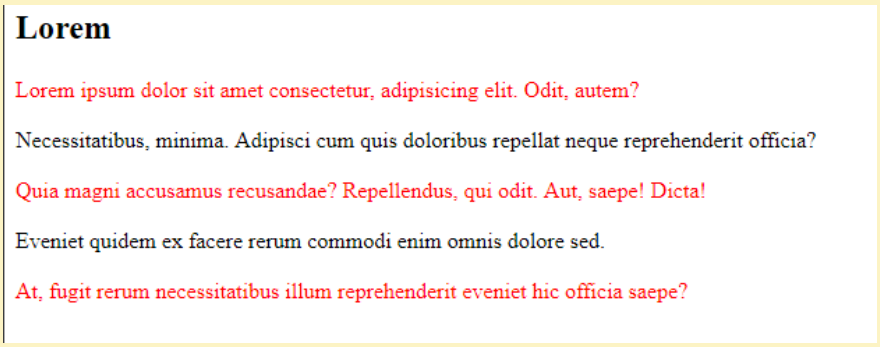
```
p:nth-child(2n) {
  color: red;
}

// Every second paragraph ...
```



(W3C The Latest, "Selectors Level 3", https://www.w3.org/TR/selectors-3/), " :nth-child() pseudo-class", https://www.w3.org/TR/selectors-3/#nth-child-pseudo
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\selectors.xhtml

The `:nth-child()` structural pseudo-class can also take `'odd'` and `'even'`.

```
p:nth-child(odd) {
  color: red;
}

// Every odd paragraph ...
```



(W3C The Latest, "Selectors Level 3", https://www.w3.org/TR/selectors-3/), " :nth-child() pseudo-class", https://www.w3.org/TR/selectors-3/#nth-child-pseudo
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\selectors.xhtml

Links

The link pseudo-class.

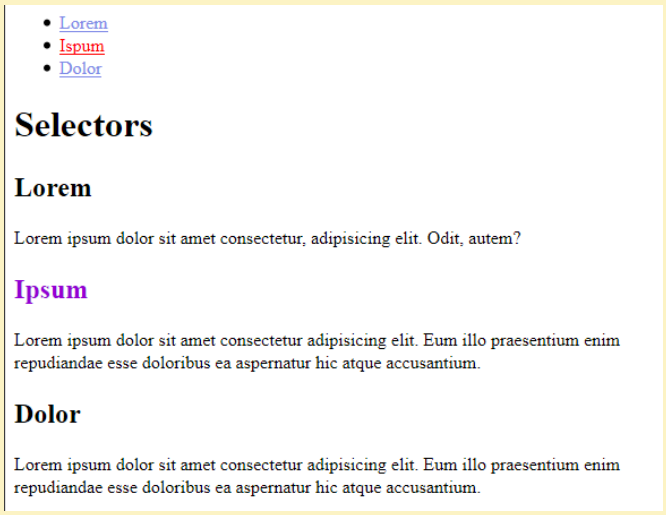
```
/* Unvisited */
:link {
  color: rgb(127, 138, 231);
}
:visited {
  color: red;
}

/* links with an external class */
.external:link{
  color: green;
```

```
}

/* E.g. picks up */
<a href="#somewhere" class="external">Somewhere</a>
```

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\selectors.xhtml

The target pseudo-class works only after a user has used a url to goto the target. A fairly useless thing.

```
// CSS
*:target{
  color: darkviolet;
}

// XHTML
<nav>
  <ul>
    <li>
      <a href="#lorem">Lorem</a>
    </li>
    <li>
      <a href="#ispum">Ispum</a>
    </li>
    <li>
      <a href="#dolor">Dolor</a>
    </li>
  </ul>
</nav>

<main>
  <h1>Selectors</h1>

  <h2 id="lorem">Lorem</h2>

  <section>

    <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Odit, autem?</p>

  </section>

  <h2 id="ispum">Ipsum</h2>

  <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Eum illo praesentium enim
repudiandae esse doloribus ea aspernatur hic atque accusantium.</p>

  <h2 id="dolor">Dolor</h2>

  <section>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Eum illo praesentium enim
repudiandae esse doloribus ea aspernatur hic atque accusantium.</p>
  </section>

// User action: clicks on "Ispum" link in nav

// Output
```



C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\selectors.xhtml

Links, Generated Content

If you want to style potential targets, then style elements with an id attribute.

```
[id] {
  font-style: italic;
}
[id]::before {
  content: "§ ";
}
```

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\selectors.xhtml
(W3C The latest, "Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification", https://www.w3.org/TR/CSS22/), "Generated content, automatic numbering, and lists", https://www.w3.org/TR/CSS22/generate.html
(W3C The Latest, "Selectors Level 3", https://www.w3.org/TR/selectors-3/), "The ::before and ::after pseudo-elements", https://www.w3.org/TR/selectors-3/#gen-content

User action pseudo-classes

The action pseudo-classes, :visited, :hover, and :active can be used on interactive elements, like links or input boxes. When used with links they are typically used in conjunction with the link pseudo-classes :link and :visited.

```
a:link {
  color: rgb(127, 138, 231);
}
a:visited {
  color: red;
}
a:focus {
  outline: 2px solid orange;
}
a:hover {
  outline: 2px solid blue;
  cursor: pointer; /* E.g. hand */
}
/* E.g. when clicking on a link */
a:active {
```

```
  outline: 2px solid green;
}
```

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\selectors.xhtml
(W3C The Latest, "Selectors Level 3", https://www.w3.org/TR/selectors-3/), " The user action pseudo-classes :hover, :active, and :focus", https://www.w3.org/TR/selectors-3/#the-user-action-pseudo-classes-hover-act

Styling link states should be done the right order: a:link; a:visited; a:focus; a:hover; a:active. The mnemonic: LoVe beFore HAte.

Still necessary as of 2018-05-09. Tested in Firefox.

UI element states pseudo-classes

Example for UI element states pseudo-classes.

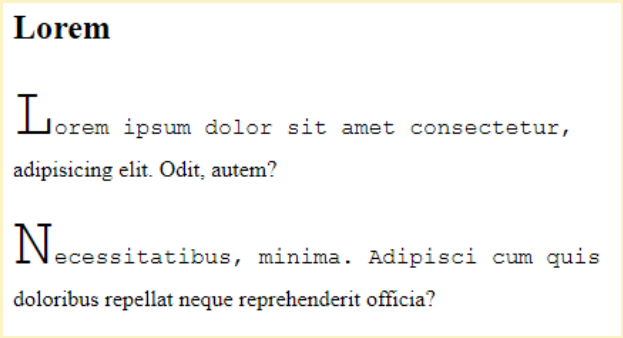
```
[type="checkbox"]:disabled {
  outline: 2px solid gray;
}
[type="checkbox"]:enabled {
  outline: 2px solid greenyellow;
}
[type="checkbox"]:checked {
  outline: 3px solid salmon;
}
```

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\selectors.xhtml
(W3C The Latest, "Selectors Level 3", https://www.w3.org/TR/selectors-3/), "The UI element states pseudo-classes", https://www.w3.org/TR/selectors-3/#UIstates

The first x pseudo-elements.

The first x pseudo-elements example.

```
p::first-letter {
  font-size: 3em;
}
p::first-line {
  font-family: 'Courier New', Courier, monospace;
}
```

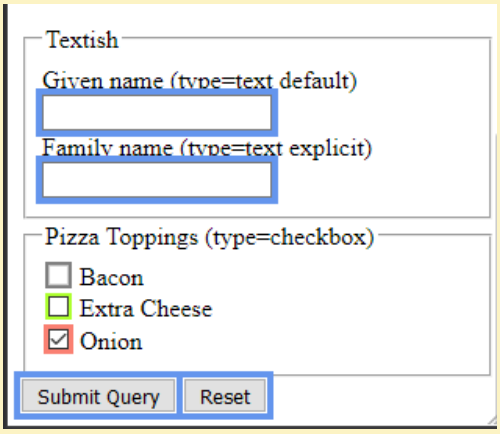


C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\selectors.xhtml
(W3C The Latest, "Selectors Level 3", https://www.w3.org/TR/selectors-3/), "Pseudo-elements", https://www.w3.org/TR/selectors-3/#pseudo-elements

Negation

Negation example.

```
/* Select all input elements that are not checkboxes */
input:not([type="checkbox"]) {
  outline: 4px solid cornflowerblue;
}
```




```
article > section:not(:last-of-type)

nav > ul > li:not(:target):not(:hover) > ul {
  visibility: collapse;
}s
```

(W3C The Latest, "Selectors Level 3", https://www.w3.org/TR/selectors-3/), "The negation pseudo-class", https://www.w3.org/TR/selectors-3/#negation
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\selectors.xhtml

Obtaining element property values.

Overall

Obtain an element's property value for each media type as follows:

- If there are one or more declarations for the element, use the value from the declaration that wins the cascade; else
- If a value can be inherited use it; else
- Use the default value (the initial setting).

(W3C The Latest, "CSS Cascading and Inheritance Level 4", https://www.w3.org/TR/css-cascade-4/), "Value Processing", https://www.w3.org/TR/css-cascade-4/#value-stages "Specified Values", https://www.w3.org/TR/css-cascade-4/#specified (W3C The latest, "Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification", https://www.w3.org/TR/CSS22/), "Assigning property values, Cascading, and Inheritance", https://www.w3.org/TR/CSS22/cascade.html

The Cascade

If obtaining a value via the cascade then do so by sorting the list of applicable declarations as follows:

- Importance (normal or important); origin (user agent, user, author); and whether it is a animation or transition declaraction. Later declaractions when over earlier:

Importance	**Origin**
Normal	User agent User Author
	Animation
Important	User agent User Author
	Transition

(W3C The Latest, "CSS Cascading and Inheritance Level 4", https://www.w3.org/TR/css-cascade-4/), "Origin and Importance", https://www.w3.org/TR/css-cascade-4/#cascade-origin

- [Obsolete] Scope (scope attribute on style element).

  w3c/csswg-drafts > [css-scoping] scoped attribute on style element removed from HTML #137, https://github.com/w3c/csswg-drafts/issues/137

- Specificity.

  Count:

- ID's (e.g. `#myThing`) (=A).
- Classes (e.g. `.coolClass`), attribute (e.g. `E[foo="bar"]`), and pseudo-classes (e.g. `E:nth-child(n)`) (=B).
- Types (aka "tags", e.g. `div`) and pseudo-elements (e.g. `E::before`) (=C).
- Don't count the universal selector (*).

Some pseudo-classes have special rules by setting up "evaluation contexts" for counting.

Pseudo-class	Count rule	Example
:is(), :not(), or :has()	Most specific complex selector in list of arguments	:is(em, #foo) has a specificity of (1,0,0)—like an ID selector (#foo)—when matched against any of , <p id=foo>, or . :not(em, strong#foo) has a specificity of (1,0,1)—like a tag selector (strong) combined with an ID selector (#foo)—when matched against any element.
:nth-child() or :nth-last-child()	Most specific complex selector in list of arguments (if any) plus the pseudo-class itself	:nth-child(even of li, .item) has a specificity of (0,2,0)—like a class selector (.item) plus a pseudo-class—when matched against any of , <ul class=item>, or <li class=item id=foo>.
:where()	0	.qux:where(em, #foo#bar#baz) has a specificity of (0,1,0): only the .qux outside the :where() contributes to selector specificity.

> Specificities are compared by comparing the three components in order: the specificity with a larger A value is more specific; if the two A values are tied, then the specificity with a larger B value is more specific; if the two B values are also tied, then the specificity with a larger C value is more specific; if all the values are tied, the two specificities are equal.

```
*               /* a=0 b=0 c=0 */
LI              /* a=0 b=0 c=1 */
UL LI           /* a=0 b=0 c=2 */
UL OL+LI        /* a=0 b=0 c=3 */
H1 + *[REL=up]  /* a=0 b=1 c=1 */
UL OL LI.red    /* a=0 b=1 c=3 */
LI.red.level    /* a=0 b=2 c=1 */
#x34y           /* a=1 b=0 c=0 */
#s12:not(FOO)   /* a=1 b=0 c=1 */
.foo :is(.bar, #baz)
                /* a=1 b=1 c=0 */
```

(W3C The latest, "Selectors Level 4", https://www.w3.org/TR/selectors-4/,, "Calculating a selector's specificity", https://www.w3.org/TR/selectors/#specificity-rules

Declarations in a style attribute (of an HTML element) have a specificity higher than declarations in a style rule (i.e. from a style sheet or style block).

(W3C The Latest, "CSS Cascading and Inheritance Level 4", https://www.w3.org/TR/css-cascade-4/), https://www.w3.org/TR/css-cascade-4/#cascade-specificity

- Order of appearance. If declarations have the same specificity then we rely on the order of appearance, the lower rule and declaration wins. Style attributes are placed after any style sheets.

Inheritance

If there is no value from a cascade and the property is defined as inherited then get an "inherited value":

- For the root element the inherited value is the initial value of the property; otherwise ...
- The computed value of the element's parent.

On whether a property is inheritede note, for example, that `color` is inherited (https://www.w3.org/TR/CSS22/colors.html#propdef-color) but `background-color` is not (https://www.w3.org/TR/CSS22/colors.html#propdef-background-color) .

(W3C The Latest, "CSS Cascading and Inheritance Level 4", https://www.w3.org/TR/css-cascade-4/), "7.2. Inheritance", https://www.w3.org/TR/css-cascade-4/#inheriting
(W3C The latest, "Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification", https://www.w3.org/TR/CSS22/), "Assigning property values, Cascading, and Inheritance", https://www.w3.org/TR/CSS22/cascade.html

Default

If there is no value from a cascade nor inheritance then obtain a value by defaul. That is, get the property's "initial value".

(W3C The Latest, "CSS Cascading and Inheritance Level 4", https://www.w3.org/TR/css-cascade-4/), Defaulting", https://www.w3.org/TR/css-cascade-4/#defaulting

Display

Overview

The spec.

(W3C The Latest, "CSS Display Module Level 3", https://www.w3.org/TR/css-display-3/) https://www.w3.org/TR/css-display-3/

Display property values:

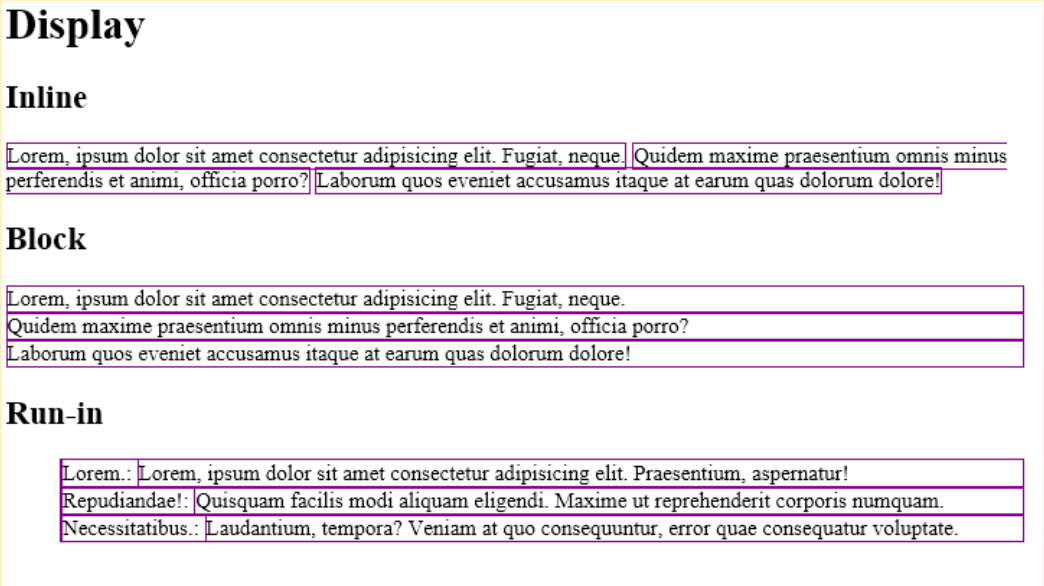
```
[<display-outside> || <display-inside> ] | <display-listitem> | <display-internal> |
<display-box> | <display-legacy>

<display-outside>  = block | inline | run-in
<display-inside>   = flow | flow-root | table | flex | grid | ruby
<display-listitem> = <display-outside>? && [ flow | flow-root ]? && list-item
<display-internal> = table-row-group | table-header-group |
                     table-footer-group | table-row | table-cell |
                     table-column-group | table-column | table-caption |
                     ruby-base | ruby-text | ruby-base-container |
                     ruby-text-container
<display-box>      = contents | none
<display-legacy>   = inline-block | inline-table | inline-flex | inline-grid
```

(W3C The Latest, "CSS Display Module Level 3", https://www.w3.org/TR/css-display-3/), "Box Layout Modes: the display property", https://www.w3.org/TR/css-display-3/#the-display-properties

Display-outside values.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>Display</title>
  <style>
    /* <![CDATA[ */
    div > * {
      border: 1px solid darkmagenta;
    }
    .box {
      margin-bottom: 1em;
    }

    #box01 > div {
      display: inline;
    }
    #box02 > div {
      display: block;
    }

    #box03 > dt {
      display: run-in;
    }
    dt::after {
      content: ": ";
    }
    /* ]]> */
  </style>

</head>
<body>
  <h1>Display</h1>
  <h2>Inline</h2>
  <div class="box" id="box01">
      <div>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Fugiat, neque.</div>
```

```
      <div>Quidem maxime praesentium omnis minus perferendis et animi, officia porro?</div>
      <div>Laborum quos eveniet accusamus itaque at earum quas dolorum dolore!</div>
  </div>

  <h2>Block</h2>
  <div class="box" id="box02">
    <div>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Fugiat, neque.</div>
    <div>Quidem maxime praesentium omnis minus perferendis et animi, officia porro?</div>
    <div>Laborum quos eveniet accusamus itaque at earum quas dolorum dolore!</div>  </div>

  <h2>Run-in</h2>
  <div class="box" id="box03">
    <dt>Lorem.</dt>
    <dd>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Praesentium,
aspernatur!</dd>
    <dt>Repudiandae!</dt>
    <dd>Quisquam facilis modi aliquam eligendi. Maxime ut reprehenderit corporis
numquam.</dd>
    <dt>Necessitatibus.</dt>
    <dd>Laudantium, tempora? Veniam at quo consequuntur, error quae consequatur
voluptate.</dd>
  </div>
</body>
</html>
```



C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\display.xhtml

> A run-in box is a box that merges into a block that comes after it, inserting itself at the beginning of that block's inline-level content.

(W3C The Latest, "CSS Display Module Level 3", https://www.w3.org/TR/css-display-3/), " Run-In Layout", https://www.w3.org/TR/css-display-3/#run-in-layout

Run-in is only supported in IE at 2019-01-01.

(MDN 2018, "Web Technology for Developers > CSS: Cascading Style Sheets > Display", https://developer.mozilla.org/en-US/docs/Web/CSS/display) https://developer.mozilla.org/en-US/docs/Web/CSS/display#Browser_compatibility

Legacy values specify display-outside and display-inside as one word.

```
// New School: display-outside display-inside.
.container {
  display: inline flex;
```

```
}

// Legacy (Old School)
.container {
  display: inline-flex;
}
```

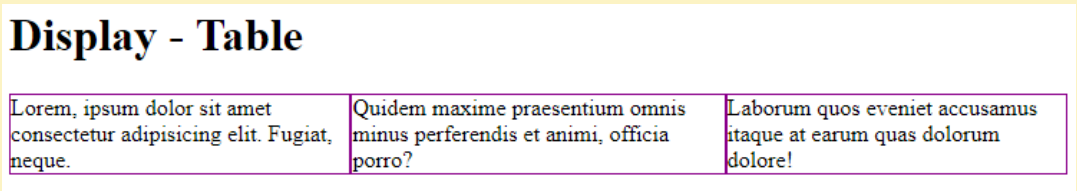
> The Level 3 specification details two values for the display property — enabling the specification of the outer and inner display type explicitly — but this is not yet well-supported by browsers.
>
> The display-legacy methods allow the same results with single keyword values, and should be favoured by developers until the two keyword values are better supported.

(MDN 2018, "Web Technology for Developers > CSS: Cascading Style Sheets > Display", https://developer.mozilla.org/en-US/docs/Web/CSS/display), "Syntax", https://developer.mozilla.org/en-US/docs/Web/CSS/display#Syntax

Display table

Display table example.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>Display - Table</title>
  <style>
    /* <![CDATA[ */
    div > * {
      border: 1px solid darkmagenta;
    }
    .box {
      margin-bottom: 1em;
    }

    #box03 {
      display: table;
    }
    #box03 > div {
      display: table-cell;
    }
    /* ]]> */
  </style>

</head>
<body>
  <h1>Display - Table</h1>

  <div class="box" id="box03">
    <div>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Fugiat, neque.</div>
    <div>Quidem maxime praesentium omnis minus perferendis et animi, officia porro?</div>
    <div>Laborum quos eveniet accusamus itaque at earum quas dolorum dolore!</div>
  </div>

</body>
</html>
```



C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\display-table.xhtml

Margin collapsing

Adjoining blocks with vertical margins sometimes "collapse", that is combine to form a single margin. There's various circumstances in which this does or does not occur.

> *(W3C The latest, "Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification", https://www.w3.org/TR/CSS22/), "8.3.1 Collapsing Margins", https://www.w3.org/TR/CSS22/box.html#collapsing-margins*

Horizontal margins never collapse.

> *(W3C The latest, "Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification", https://www.w3.org/TR/CSS22/), "8.3.1 Collapsing Margins", https://www.w3.org/TR/CSS22/box.html#collapsing-margins*

Media Queries

Media queries can be expressed in three locations:

- HTML: a link element (as a child of head).
- CSS: an import rule.
- CSS: a media rule.

```
<!-- HTML: a link element (as a child of head). -->
<head>
  <link rel="stylesheet" href="media-queries.css" media="screen and (min-width: 500px)" />
</head>

<!-- CSS: an import rule. -->
@import url(media-queries.css) screen and (min-width: 500px);

/* CSS: a media rule. */
@media screen and (min-width: 500px) {
  body {
    color: red;
  }
}
```

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\media-queries.xhtml
(W3C The Latest CSS Media Queries https://www.w3.org/TR/css3-mediaqueries/), "Media Queries",
https://www.w3.org/TR/css3-mediaqueries/#media0

Media query expressions:

- Composition: a media type ("the media type expression"), optionally followed by expressions that check for media features ("the media features expression").

```
@media screen and (min-width: 500px)

// Media type expression: "screen"
// Media feature expression: "(min-width: 500px)"
// Media query expression: "screen and (min-width: 500px)"
```

- The media query expression returns a boolean: it evaluates to either true or false.
- The "all" media type is the default and can be omitted.

```
// Equivalent
@media all and (orientation: portrait) {...}
@media (orientation: portrait) {...}
```

- Logical AND is represented by "and", logical OR is represented by a comma ",".

```
// Logical AND
@media screen and (min-width: 500px) {...}

// Logical OR
@media screen and (color), projection and (color) {...}
```

- Logical NOT is represented by "not". If at the beginning the media query expression, it negates the whole expression.

```
@media not screen and (min-width: 500px) {...}
```

(W3C The Latest, "CSS Media Queries", https://www.w3.org/TR/css3-mediaqueries/), "Media Queries",
https://www.w3.org/TR/css3-mediaqueries/#media0

Media feature expressions:

- Most media features accept optional '`min-`' and '`max-`' prefixes, meaning 'great than or equal to' and 'smaller than or equal to' respectively.

```
@media screen and (min-width: 500px) {...}
```

- Media featues can be without a value. Without a value `(feature)` will evaluate to true if `(feature:x)` equals a non zero value.

```
/* Return true if the media is a color device.
The color media feature references 'the number of bits per color component of the
output device' */
@media all and (color) { … }
```

(W3C The Latest, "CSS Media Queries", https://www.w3.org/TR/css3-mediaqueries/), "Media features", https://www.w3.org/TR/css3-mediaqueries/#media1

Media types:

- all
- screen
- print
- braille
- handheld
- projection
- tty
- tv
- embossed
- speech

(W3C The Latest, "CSS Media Queries", https://www.w3.org/TR/css3-mediaqueries/), "Background" https://www.w3.org/TR/css3-mediaqueries/#background

Media features:

Command	Description	Values	Accepts min/max
width	the width of the targeted display area of the output device	<length>	yes
height	the height of the targeted display area of the output device	<length>	yes
device-width	the width of the rendering surface of the output device	<length>	yes
device-height	the height of the rendering surface of the output device	<length>	tes
orientation	'portrait' when the value of the 'height' media feature is greater than or equal to the value of the 'width' media feature. Otherwise 'orientation' is 'landscape'.	portrait, landscape	no
aspect-ratio	the ratio of the value of the 'width' media feature to the value of the 'height' media feature.	<ratio>	yes
device-aspect-ratio	the ratio of the value of the 'device-width' media feature to the value of the 'device-height' media feature.	<ratio>	yes

color	the number of bits per color component of the output device. If the device is not a color device, the value is zero.	<integer>	yes
color-index	the number of entries in the color lookup table of the output device. If the device does not use a color lookup table, the value is zero.	<integer>	yes
monochrome	the number of bits per pixel in a monochrome frame buffer. If the device is not a monochrome device, the output device value will be 0.	<integer>	yes
resolution	the resolution of the output device, i.e. the density of the pixels.	<resolution> e.g. "300dpi" or "300dpcm"	yes
scan	scanning process of "tv" output devices	progressive, interlace	no
grid	query whether the output device is grid or bitmap. If the output device is grid-based (e.g., a "tty" terminal, or a phone display with only one fixed font), the value will be 1. Otherwise, the value will be 0.	0 or 1	no

(W3C The Latest, "CSS Media Queries", https://www.w3.org/TR/css3-mediaqueries/), "Media features", https://www.w3.org/TR/css3-mediaqueries/#media1

Apply screen sizing media queries from smallest width (or height) to largest.

Custom Properties and Variables

Basics

Official reference ...

> *(W3C 2015, "CSS Custom Properties for Cascading Variables Module Level 1", https://www.w3.org/TR/css-variables-1/)*

It is said "custom properties define variables". But I'll speak of "defining variables" and "using variables".

> *(W3C 2015, "CSS Custom Properties for Cascading Variables Module Level 1", https://www.w3.org/TR/css-variables-1/) , "2. Defining Custom Properties: the --* family of properties", https://www.w3.org/TR/css-variables-1/#defining-variables*

Define variables by using two hypens (--) in front of a name, in a :root rule.

```
:root {
  --main-indent: 12%;
  --main-color: blue;
  --accent-color: pink;
}
```

> *(W3C 2015, "CSS Custom Properties for Cascading Variables Module Level 1", https://www.w3.org/TR/css-variables-1/) , "2. Defining Custom Properties: the --* family of properties", https://www.w3.org/TR/css-variables-1/#defining-variables*
> *C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\custom-properties-and-variables.xhtml*

Use variables by surrounding the variable name with `var()`.

```
h1 {
  color: var(--main-color);
  margin-left: var(--main-indent);
  border-bottom: 1px solid var(--accent-color);
}

p {
  margin-left: var(--main-indent);
}
```

> *(W3C 2015, "CSS Custom Properties for Cascading Variables Module Level 1", https://www.w3.org/TR/css-variables-1/) , "2. Defining Custom Properties: the --* family of properties", https://www.w3.org/TR/css-variables-1/#defining-variables*
> *C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\custom-properties-and-variables.xhtml*

Advanced usage

You can use variables for internationalization.

```
:root,
:root:lang(en) {--external-link: "external link";}
:root:lang(de) {--external-link: "externer Link";}

a[href^="http"]::after {content: " (" var(--external-link) ")"}
```

(W3C 2015, "CSS Custom Properties for Cascading Variables Module Level 1", https://www.w3.org/TR/css-variables-1/), "2.1. Custom Property Value Syntax", https://www.w3.org/TR/css-variables-1/#syntax

To use variables in calculations, use `calc()`.

```
/* Do this */
#second-paragraph {
  --gap: 100;
  margin-top: calc(var(--gap) * 1px);
}

/* Not this */
#second-paragraph {
  --gap: 100;
  margin-top: var(--gap)px;
}
```

(W3C 2015, "CSS Custom Properties for Cascading Variables Module Level 1", https://www.w3.org/TR/css-variables-1/), "3. Using Cascading Variables: the var() notation", https://www.w3.org/TR/css-variables-1/#using-variables
(W3C 2016, "CSS Values and Units Module Level 3", https://www.w3.org/TR/css-values-3/), " 8. Functional Notations", https://www.w3.org/TR/css-values-3/#functional-notations
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\custom-properties-and-variables.xhtml

Fallbacks

For browsers that don't support custom variables (like IE 11 https://caniuse.com/#feat=css-variables) you can use fallbacks. That is, provide the specific value first then the variable.

```
a {
  color: #7F583F;
  color: var(--primary);
}
```

https://vgpena.github.io/winning-with-css-variables/

Downloadable font resources (Web Fonts)

Windows usage

To install third party fonts for usage on windows (e.g. to add fonts to MS Word):

- Go https://fonts.google.com/
- Browse and select your fonts.
- Click on the "X Families Selected" Box > download arrow (Top right corner)
- Windows Explorer. Unzip your downloaded fonts.
- Select all the font files > Right click > Install.
- Windows > Control Panel > Fonts. Observe your font family.
- MS Word > New > Blank Document > Home Menu > Font section. Click on font family combo box. Observe your downloaded font family (or families).

Web usage

Info

The official specs: (|CSS Fonts 3|, https://www.w3.org/TR/css-fonts-3/), https://www.w3.org/TR/css-fonts-3/#font-resources

Open source third party fonts are available from: https://fonts.google.com/

The two techniques

Download fonts using the `@font-face` rule, but attempt to source the font locally first.

- This requires the served fonts to be from the same domain as your other source (XHTML, CSS) files. That is, as web dev you install the fonts to your own server.
- We also try to use a locally installed copy (installed via the OS), should the user have such an installation, to avoid downloading.
- Use both the "full font name" and the "Postscript name" for cross platform compatibility.
- The name should target the particular font within the family, not the font family (despite the name)

```
<style>
  /* <![CDATA[ */
  /**
    CSS technique for same-domain access, with local try first
  **/
  @font-face {
    font-family: Lobster Regular;
    src:
          local(Loster Regular), /* Full font name, common on Windows */
          local(Lobster-Regular), /* Postscript name, common on OSX */
          url('fonts/lobster/Lobster-Regular.ttf');
```

```
    }
    /** Use a downloaded or native font **/
    p {
      font-family: Lobster Regular, cursive;
    }
    /* ]]> */
  </style>

</head>
<body>
  <h1>Fonts - Downloadable</h1>

  <p>How does <strong>strong</strong> and <em>emphasis</em> look?</p>
```

Using the `@font-face` rule is the technique pushed in the official specs. And it give you more control compared to an import technique: with @font-face you can specify an attempt to use locally installed fonts.

(W3C 2013, CSS Fonts 3, https://www.w3.org/TR/css-fonts-3/), https://www.w3.org/TR/css-fonts-3/#font-resources
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\fonts-downloadable-font-face-rule.xhtml

If you want to violate the no-cross-domain principle (a security vulnerability?) download fonts using the @import rule. But use the family name (e.g. Lobster), not the particular font name (e.g. Lobster Regular).

```
  <style>
    /* <![CDATA[ */
    @import url('https://fonts.googleapis.com/css?family=Lobster');

    p { font-family: Lobster, cursive; }
    /* ]]> */
  </style>

</head>
<body>
  <h1>Fonts - Downloadable</h1>

  <p>Lorem ipsum dolor sit.</p>
```

You could also achieve the same effect with a "stylesheet" link (in (X)HTML) ...

```
    <link href="https://fonts.googleapis.com/css?family=Lobster"
          rel="stylesheet" />

    <style>
      /* <![CDATA[ */

      p { font-family: Lobster, cursive; }
      /* ]]> */
    </style>

  </head>
  <body>
    <h1>Fonts - Downloadable</h1>

    <p>Lorem ipsum dolor sit amet.</p>
```

... but this is discouraged. The CSS @import technique is recommended as: It is not an (X)HTML solution, which would conflate semantics with presentation: we want as much presentation as possible to be done from CSS;

(W3C 2013, CSS Fonts 3, https://www.w3.org/TR/css-fonts-3/), https://www.w3.org/TR/css-fonts-3/#font-resources
(Google 2017, "Get Started with the Google Fonts API | Google Fonts", https://developers.google.com/fonts/docs/getting_started)
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\fonts-downloadable-Import-rule.xhtml

Minutiae

> When authors would prefer to use a locally available copy of a given font and download it if it's not, `local()` can be used. ....

```
@font-face {
  font-family: Lobster Regular;
  src:
        local(Loster Regular),
        local(Loster-Regular),
        url('fonts/lobster/Lobster-Regular.ttf');
}

/** Use a downloaded or native font **/
p {
  font-family: Lobster, cursive;
}
```

You can observe the network benefit with {Firefox} > Tools > Web Developer > Toogle Tools [F12] > |Network|: Ctrl+F5 (Hard Reload). With ...

```
@font-face {
  font-family: Lobster Regular;
  src:
        url('fonts/lobster/Lobster-Regular.ttf');
}
```

... in my test files I can observe two network requests (one of which fetches the font), for a total load time of 211ms. But with ...

```
@font-face {
  font-family: Lobster Regular;
  src:
        local(Loster Regular),
        local(Lobster-Regular),
        url('fonts/lobster/Lobster-Regular.ttf');
}
```

... there is only one network request (the xhtml file) and a total load time of 127ms.

(W3C 2013, CSS Fonts 3, https://www.w3.org/TR/css-fonts-3/), https://www.w3.org/TR/css-fonts-3/#src-desc
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\fonts-downloadable-options.xhtml

When using `local()`, use it twice to include the "full font name" and "Postsript name", for cross platform purposes.

For OpenType and TrueType fonts, this string is used to match only the Postscript name or the full font name in the name table of locally available fonts. Which type of name is used varies by platform and font, so authors should include both of these names to assure proper matching across platforms

```
/**
  CSS technique for same-domain access, with local try first
**/
@font-face {
  font-family: Lobster Regular;
  src:
        local(Loster Regular), /* Full font name, common on Windows */
        local(Lobster-Regular), /* Postscript name, common on OSX */
        url('fonts/lobster/Lobster-Regular.ttf');
}

/** Use a downloaded or native font **/
p {
  font-family: Lobster, cursive;
}
```

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\fonts-downloadable-options.xhtml (|CSS Fonts 3|, https://www.w3.org/TR/css-fonts-3/) https://www.w3.org/TR/css-fonts-3/#src-desc

[A] `@font-face` rule specifies the characteristics of a single font within a family, the unique name used with local() specifies a single font, not an entire font family [Despite the name "font-family"].

```
/**
  Do this
**/
@font-face {
  font-family: Lobster Regular;
  src:
        local(Loster Regular), /* Full font name, common on Windows */
        local(Lobster-Regular), /* Postscript name, common on OSX */
        url('fonts/lobster/Lobster-Regular.ttf');
}

p {
  font-family: Lobster Regular, cursive;
}

/**
  Not this
**/
@font-face {
  font-family: Lobster;
  src:
        local(Loster Regular), /* Full font name, common on Windows */
        local(Lobster-Regular), /* Postscript name, common on OSX */
        url('fonts/lobster/Lobster-Regular.ttf');
}

p {
  font-family: Lobster, cursive;
}
```

In Windows determining the particular font name, rather than the font family to which it belongs, is done by:

- Windows Search > "Fonts".
- Where a font has a singled paged icon, observe that name. Where a font has a multiple paged icon double click through to see the particular fonts included within that family.

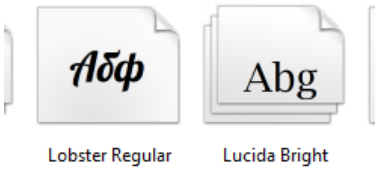


Figure 1 Particular Font Versus Font Family

Figure 2 Having clicked on a Font Familty to see particular fonts

In the google case, when using the `@import` rule, you can download several fonts at once:

```
// CSS import
@import url('https://fonts.googleapis.com/css?family=Lato|Roboto');
```

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\fonts-downloadable-options.xhtml

Use the downloaded font as for native fonts.

```
p { font-family: Lobster Regular, cursive; }
```

(W3C 2013, CSS Fonts 3, https://www.w3.org/TR/css-fonts-3/), https://www.w3.org/TR/css-fonts-3/#font-resources
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\fonts-downloadable-options.xhtml

Quoting font-family names, either in the @font-face rule or for use on an element (h1, p, etc.), is optional.

```
/* Quoting: Either of the following forms are valid */
p { font-family: Trebuchet MS, sans-serif; }
p { font-family: "Trebuchet MS", sans-serif; }
p { font-family: 'Trebuchet MS', sans-serif; }
```

"Font family names other than generic families must either be given quoted as strings, or unquoted as a sequence of one or more identifiers." (|CSS Fonts 3|, https://www.w3.org/TR/css-fonts-3/) https://www.w3.org/TR/css-fonts-3/#font-family-prop

" The name can optionally be enclosed in quotes. If unquoted, the unquoted font family name processing conventions apply; the name must be a sequence of identifiers separated by whitespace which is converted to a string by joining the identifiers together separated by a single space." (|CSS Fonts 3|, https://www.w3.org/TR/css-fonts-3/) https://www.w3.org/TR/css-fonts-3/#font-family-desc

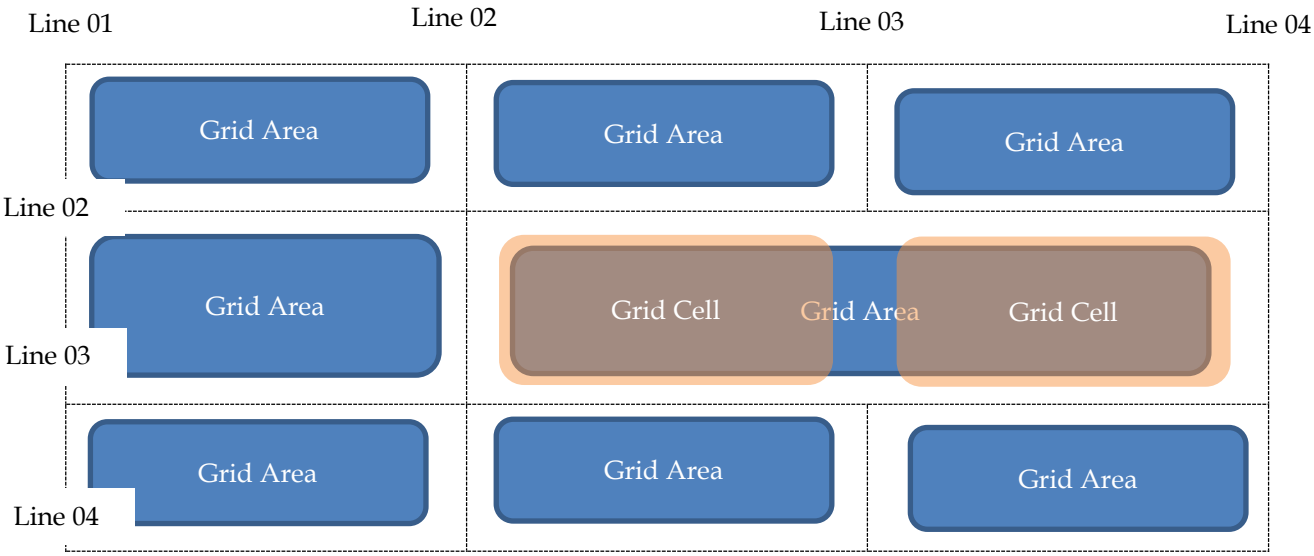
Grid Layout Module

Overview

Parts and terminology

Grid parts and terminology:

- Grid container. The whole thing which surrounds the whole.
- Grid lines. Vertical and horizontal. "A grid line exists on either side of a column or row."
- Grid tracks. "Grid track is a generic term for a grid column or grid row—in other words, it is the space between two adjacent grid lines ... Adjacent grid tracks can be separated by gutters".
- Grid cell. "A grid cell is the intersection of a grid row and a grid column."
- Grid areas. "the logical space used to layout one or more grid items. A grid area consists of one or more adjacent grid cells".
- Grid items.
- Rows and columns.



(W3C 2017, "CSS Grid Layout Module Level 1", https://www.w3.org/TR/css-grid-1/),
"Grid Lines", https://www.w3.org/TR/css-grid-1/#grid-line-concept
"Grid Tracks and Cells", https://www.w3.org/TR/css-grid-1/#grid-track-concept

Coding procedure

The grid layout coding procedure:

1. In HTML have in mind (and possibly provide id's for) a grid container and a set of grid items.

```
<div id="gridContainer">
  <div id="title">Game Title</div>
  <div id="score">Score</div>
  <div id="stats">Stats</div>
  <div id="board">Board</div>
  <div id="controls">Controls</div>
</div>
```

2. In CSS declare a container has having grid display.

```
#gridContainer {
  display: grid;
  ...
```

3. Define the grid areas and their sizing.

```
#gridContainer {
  ...

  grid-template-columns: auto 1fr;
  grid-template-rows:
    auto
    1fr
    auto;
}
```

4. Use break-word (not break-all) to ensure shrunk areas don't overflow.

```
#gridContainer  {
  ...
  word-break: break-word;
  ...
}
```

5. Place grid items.

```
#title    { grid-column: 1; grid-row: 1; }
#score    { grid-column: 1; grid-row: 3; }
#stats    { grid-column: 1; grid-row: 2; align-self: start; }
#board    { grid-column: 2; grid-row: 1 / span 2; }
#controls { grid-column: 2; grid-row: 3; justify-self: center; }
```

6. Add aesthetic sugar and further customisation.

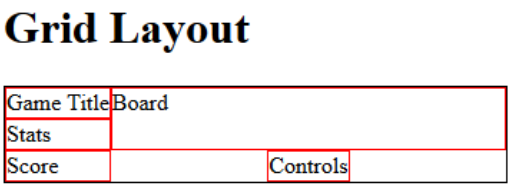
```
#gridContainer  {
  border: 1px solid black;
}
#gridContainer div {
  border: 1px solid red;
}
```

(W3C 2017, "CSS Grid Layout Module Level 1", https://www.w3.org/TR/css-grid-1/), "Adapting Layouts to Available Space", https://www.w3.org/TR/css-grid-1/#adapting-to-available-space
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\grid-layout-basic-example.xhtml

Complete example.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>Grid Layout</title>
```

```
  <style>
    /* <![CDATA[ */
    #gridContainer {
      /* 02. Declare a container has having grid display */
      display: grid;

      /* 03. Define the grid areas and their sizing */
      grid-template-columns: auto 1fr;
      grid-template-rows:
        auto
        1fr
        auto;
    }

    /* 04. Specify the position of grid items */
    #title    { grid-column: 1; grid-row: 1; }
    #score    { grid-column: 1; grid-row: 3; }
    #stats    { grid-column: 1; grid-row: 2; align-self: start; }
    #board    { grid-column: 2; grid-row: 1 / span 2; }
    #controls { grid-column: 2; grid-row: 3; justify-self: center; }


    /* 05. Add aesthetic sugur and further customisation */
    #gridContainer  {
      border: 1px solid black;
    }
    #gridContainer div {
      border: 1px solid red;
    }
    /* ]]> */
  </style>

</head>
<body>
  <h1>Grid Layout</h1>

  <!-- 01. In HTML designate a grid container and set of grid items  -->
  <div id="gridContainer">
    <div id="title">Game Title</div>
    <div id="score">Score</div>
    <div id="stats">Stats</div>
    <div id="board">Board</div>
    <div id="controls">Controls</div>
  </div>
</body>
</html>
```



C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\grid-layout-basic-example.xhtml

Coding Tools

You can use firefox developer tools to analyze grids:

- Firefox > {F12} Developer Tools > |Layout| > Overlay Grid: Ticked.
- Click "Grid Display Settings" as desired.

Details

Define the grid areas

Overview

Grid areas and their sizing are defined explicitly or implicitly.

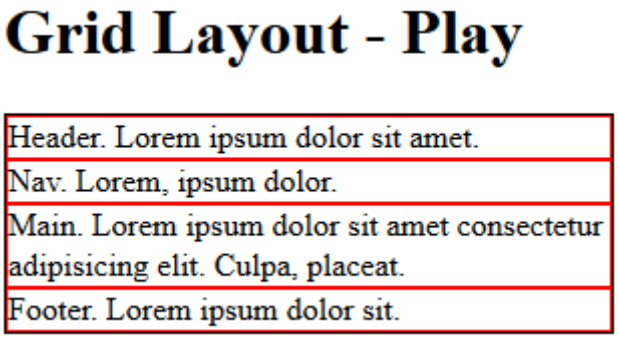
Grid areas can be defined explicitly with:

- `grid-template-rows` and `grid-template-columns`; and/or
- `grid-template-areas`;

(W3C 2017, "CSS Grid Layout Module Level 1", https://www.w3.org/TR/css-grid-1/), "Explicit Track Sizing: the grid-template-rows and grid-template-columns properties", https://www.w3.org/TR/css-grid-1/#track-sizing

As soon as we define an element as a grid container, all the direct children become grid items of that grid container.

```
#gridContainer {
  /* 02. Declare a container has having grid display */
  display: grid;
}

<div id="gridContainer">
  <header>Header. Lorem ipsum dolor sit amet.</header>
  <nav>Nav. Lorem, ipsum dolor.</nav>
  <main>Main. Lorem ipsum dolor sit amet consectetur adipisicing elit. Culpa,
placeat.</main>
  <footer>Footer. Lorem ipsum dolor sit.</footer>
</div>
```



(Mozilla n.d., "Basic Concepts of Grid Layout", Accessed 2018-04-15. https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout/Basic_Concepts_of_Grid_Layout), "Basic concepts of grid layout", https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout/Basic_Concepts_of_Grid_Layout
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\grid-layout-play.xhtml

Define grid area explicitly

With grid-template-rows and and grid-template-columns

Grid areas can be defined explicitly with `grid-template-rows` and `grid-template-columns`. These can include a track listing. A track listing can include: line names; track sizing; and track repeat commands.

```
/* Line names (in '[]') and track sizing */
grid-template-columns: [first nav-start] 150px [main-start] 1fr [last];

/* A repeat command */
grid-template-columns: repeat(auto-fill, minmax(25ch, 1fr));
```

(W3C 2017, "CSS Grid Layout Module Level 1", https://www.w3.org/TR/css-grid-1/), "Explicit Track Sizing: the grid-template-rows and grid-template-columns properties", https://www.w3.org/TR/css-grid-1/#track-sizing

With grid-template-areas

Grid areas can be defined explicitly using named grid areas. Do this with `grid-template-areas`. The value is <none> or <string>+ (one or more) with the following rules:

- Whitespace separated tokens represent cells in in a row.

```
/* Two grid areas in one row */
grid-template-areas: "nav main";
```

- Each string, surrounded by quotes (""), represents a row

```
/* Two rows */
grid-template-areas: "nav main"
                     "foot contactInfo";
```

- To have a named area span cells, repeat it's name.

```
/* The named area "head" spans two cells in the top row */
grid-template-areas: "head head"
                     "nav  main";
```

- One or more full stops (.) represents a null cell: an unnamed area.

```
/* Column 2, row 3 has an unnamed area */
grid-template-areas: "head head"
                     "nav  main"
                     "foot ...."
```

(W3C 2017, "CSS Grid Layout Module Level 1", https://www.w3.org/TR/css-grid-1/), "Named Areas: the grid-template-areas property", https://www.w3.org/TR/css-grid-1/#grid-template-areas-property

Named grid areas example, without sizing. Name the grid areas with `grid-template-areas`. Then (you can) explicIty assign grid items the defined areas.

```
#gridContainer {
  /* 02. Declare a container has having grid display */
  display: grid;

  /* 03. Define the grid areas */
  grid-template-areas:
    "head head"
    "nav  main"
    "foot ...."
}
```

```
/* 04. Specify the position of grid items */
header  { grid-area: head; }
nav     { grid-area: nav; }
main    { grid-area: main; }
footer  { grid-area: foot; }
```

Grid Layout - Grid Layout - Named Grid Areas

Header. Lorem ipsum dolor sit amet.	
Nav. Lorem, ipsum dolor.	Main. Lorem ipsum dolor sit amet consectetur adipisicing elit. Culpa, placeat.
Footer. Lorem ipsum dolor sit.	

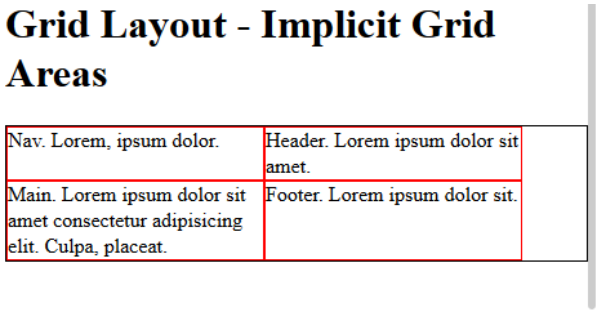
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\grid-layout-play.xhtml (W3C 2017, "CSS Grid Layout Module Level 1", https://www.w3.org/TR/css-grid-1/), "Named Areas: the grid-template-areas property", https://www.w3.org/TR/css-grid-1/#grid-template-areas-property

Define grid area implicitly and position grid items automatically

Grid areas can be defined implicitly, and grid items can be positioned automatically, by:

- Using `grid-auto-columns` and `grid-auto-rows` to define the size of columns and rows that are implicitly generated to accommodated new grid items.
- Add the first few grid items explicitly to give starting shape to the grid.
- Then remaining grid items will be placed in order, with new grid areas being implicitly generated as necessary.

```
<style>
  /* <![CDATA[ */

  #gridContainer {
    /* 02. Declare a container has having grid display */
    display: grid;
    /* 03. Define the grid areas and their sizing: implicitly */
    grid-auto-columns: 200px;
    grid-auto-rows: auto;
  }

  /* 04. Specify the position of grid items */
  nav     { grid-column: 1; grid-row: 1; }
  header  { grid-column: 2; grid-row: 1; }

  /* ]]> */
</style>

..
<div id="gridContainer">
  <header>Header. Lorem ipsum dolor sit amet.</header>
  <nav>Nav. Lorem, ipsum dolor.</nav>
  <main>Main. Lorem ipsum dolor sit amet consectetur adipisicing elit. Culpa,
placeat.</main>
  <footer>Footer. Lorem ipsum dolor sit.</footer>
</div>
```



C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\grid-layout-implicit-grid-areas.xhtml (W3C 2017, "CSS Grid Layout Module Level 1", https://www.w3.org/TR/css-grid-1/), " Implicit Track Sizing: the grid-auto-rows and grid-auto-columns properties", https://www.w3.org/TR/css-grid-1/#auto-tracks

Sizing grid areas

Track sizing functions

Track sizing functions are expressed either as:

- A fixed length (e.g. `100px`);
- A percentage of the grid container's size (e.g. `10%`);
- A relation to the content's occupying the column or row:
    a. `min-content`. In practice the size of the largest word of the content;
    b. `max-content`. In practice the size of the largest line of conent;
    c. `auto`;
- As a maxium = max-content;
- As a minimum = the largest minimum of the grid item (occupying the grid area) specified by `min-width` or `min-height`;
- A fraction of free space in the grid (e.g. `1fr`). The 'fr' unit goes by various names: flexible length; flex; or, as I'll call it "fractional length". It represents a ratio of the space left in the grid track.
- Some combination of the above using the range `minmax()` function.
- `fit-content(percentage)`. Calculated like `max(auto, max-content)` except the maxium is limited to max-content if the percentage is greater than max-content. That is, mostly: limit to the smaller width between `percentage` and `max-content`.

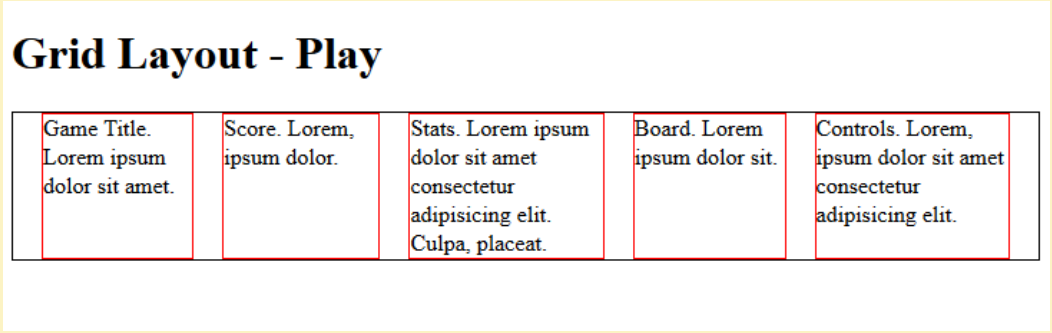
```
grid-template-columns: 100px 10% max-content 1fr minmax(min-content, 1fr) fit-content(60%);
grid-template-columns: fit-content(60%) auto;
```

(W3C 2017, "CSS Grid Layout Module Level 1", https://www.w3.org/TR/css-grid-1/), " Explicit Track Sizing: the grid-template-rows and grid-template-columns properties", " <track-list> | <auto-track-list>", https://www.w3.org/TR/css-grid-1/#track-listing

The repeat() function

The `repeat()` function allows you to repeat a fragment of a track list (e.g. line names and track sizing functions).

```
/* Given */
#title    { grid-column: 2; grid-row: 1; }
#score    { grid-column: 4; grid-row: 1; }
#stats    { grid-column: 6; grid-row: 1; }
#board    { grid-column: 8; grid-row: 1; }
#controls { grid-column: 10; grid-row: 1; }


#gridContainer {
  /* 02. Declare a container has having grid display */
  display: grid;

  /* The following are equivalent */
  grid-template-columns:
                          20px auto
                          20px auto
                          20px auto
                          20px auto
                          20px auto 20px;
  grid-template-columns: repeat(5, 20px auto) 20px;
                                                }
/* Creates 'gutter' columns in-between and on the ends */
```



C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\grid-layout-play.xhtml
(W3C 2017, "CSS Grid Layout Module Level 1", https://www.w3.org/TR/css-grid-1/), Repeating Rows and Columns: the repeat() notation, https://www.w3.org/TR/css-grid-1/#repeat-notation

Size named grid areas

Named grid areas, with sizing. Name the grid areas with `grid-template-areas`. Size using `grid-template-rows` and `grid-template-columns`.

```
#gridContainer {
  /* 02. Declare a container has having grid display */
  display: grid;

  /* 03. Define the grid areas and their sizing */
  grid-template-areas:
    "head head"
    "nav  main"
    "foot ....";
  grid-template-columns: 1fr 2fr;
}

/* 04. Specify the position of grid items */
header  { grid-area: head; }
nav     { grid-area: nav; }
main    { grid-area: main; }
footer  { grid-area: foot; }

<!-- 01. In HTML designate a grid container and set of grid items  -->
...
<div id="gridContainer">
  <header>Header. Lorem ipsum dolor sit amet.</header>
  <nav>Nav. Lorem, ipsum dolor.</nav>
```

```
  <main>Main. Lorem ipsum dolor sit amet consectetur adipisicing elit. Culpa,
placeat.</main>
  <footer>Footer. Lorem ipsum dolor sit.</footer>
</div>
```

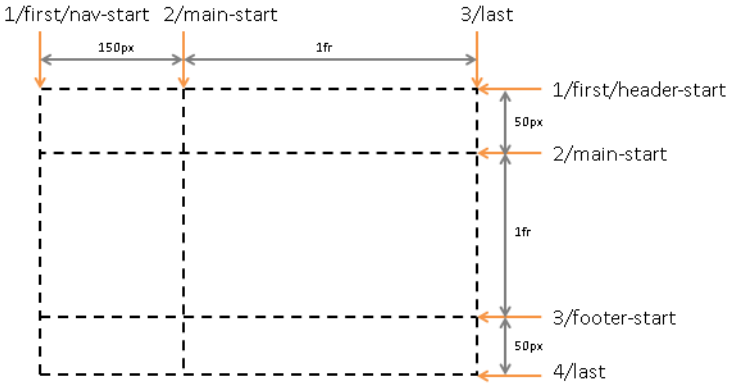
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\grid-layout-named-grid-areas.xhtml

Grid lines

Grid lines can be referenced either:

- By their numerical index, starting at 1 for the leftmost column or top row - and starting at -1 for the rightmost column or bottom row; or
- Their name, if defined.

```
#grid {
  display: grid;
  grid-template-columns: [first nav-start] 150px [main-start] 1fr [last];
  grid-template-rows: [first header-start] 50px [main-start] 1fr [footer-start] 50px
[last];
}
```



(W3C 2017, "CSS Grid Layout Module Level 1", https://www.w3.org/TR/css-grid-1/),
"Named Grid Lines: the [<custom-ident>] syntax", https://www.w3.org/TR/css-grid-1/#named-lines*
"The Explicit Grid", https://www.w3.org/TR/css-grid-1/#explicit-grids

Implicitly named lines are created when `grid-template-areas` is used. Four implicit line names are created for each named area. E.g. for area "foo":

- "foo-start" for row start and column start; and
- "foo-end" for row end and column end.

Place Grid Items

Grid items are placed in grid areas. Within the grid area grid items are aligned (e.g. with `justify-self` and `align-self`). A placement entails:

- Position, an assignment to a location (e.g. a grid area); and
- Span, how many grid tracks the grid items takes up in each axis.

(W3C 2017, "CSS Grid Layout Module Level 1", https://www.w3.org/TR/css-grid-1/), "Placing Grid Items", https://www.w3.org/TR/css-grid-1/#placement

Placement is achieved using three sets of properties:

- `grid-area`;
- `grid-column` and `grid-row`; and
- `grid-column-start`, `grid-column-end`, `grid-row-start`, and `grid-row-end`.

(W3C 2017, "CSS Grid Layout Module Level 1", https://www.w3.org/TR/css-grid-1/), "Placing Grid Items", https://www.w3.org/TR/css-grid-1/#placement

The patterns of grid item placement include ...

- Place in a named grid-area.

```
nav { grid-area: head; }
```

- Place at the edge of a named-grid area.

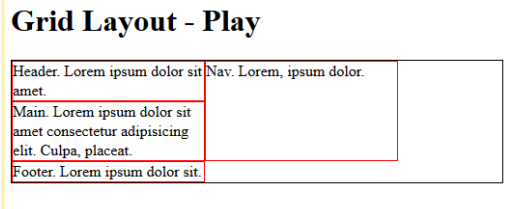
```
nav { grid-row-start: head; }
```

- Place using grid axis co-odrinates.

```
nav {
  grid-row: 2;
  grid-column: 2;
}

/* Shorthand */
nav {
  grid-area: 2 / 2; /* row / column */
}
```

- Place using the grid axis co-ordinates, with a span.

```
nav {
  grid-row: 1 / span 2;
  grid-column: 2;
}
```



- Place using grid lines: span between lines.

```
nav {
  grid-column: 1 / 3;
}
```

```
/* Named lines */
nav {
  grid-column: race-start / race-end;
}
```

- Auto placement.

```
/* The default */
nav {
  grid-area: auto;
}

/* Auto-placed item, covering two rows and two columns. */
nav {
  grid-area: span 2 / span 2;
}
```

(W3C 2017, "CSS Grid Layout Module Level 1", https://www.w3.org/TR/css-grid-1/), "Common Patterns for Grid Placement", https://www.w3.org/TR/css-grid-1/#common-uses

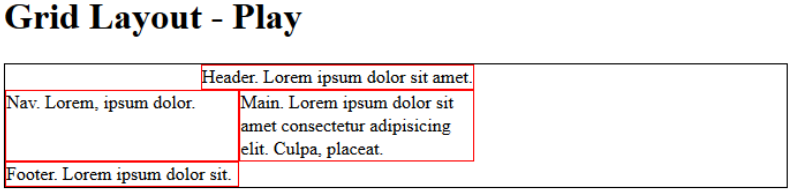
Alignment

Of grid items within their grid area

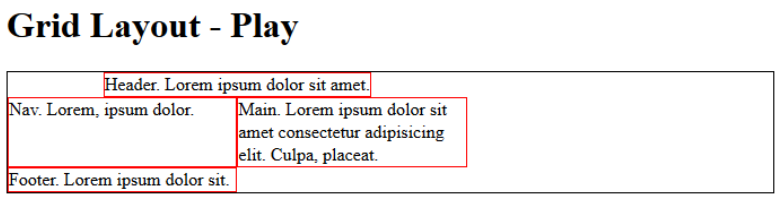
A grid item is either one of two states with respect to their grid area:

- Stretched to fill. The default.
- Sized to fit their contents. Effected if the grid item properties:
  - `justify-self` or `align-self` have a value other than `'stretch'`; or
  - `margins` are `'auto'`.

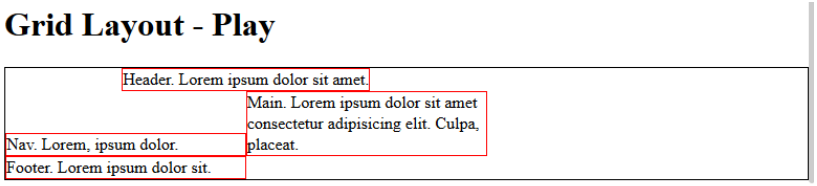
```
#gridContainer {
  /* 02. Declare a container has having grid display */
  display: grid;
  /* 03. Define the grid areas and their sizing: implicitly */
  grid-template-areas:
  "head head"
  "nav  main"
  "foot  ...";
  grid-template-columns:  30% 30%;
}

/* 04. Specify the position of grid items */
header { grid-area: head; justify-self: end; }
nav {grid-area: nav;  }
main {grid-area: main; }
footer { grid-area: foot; }
```




```
header { grid-area: head; margin: 0 auto; }
```



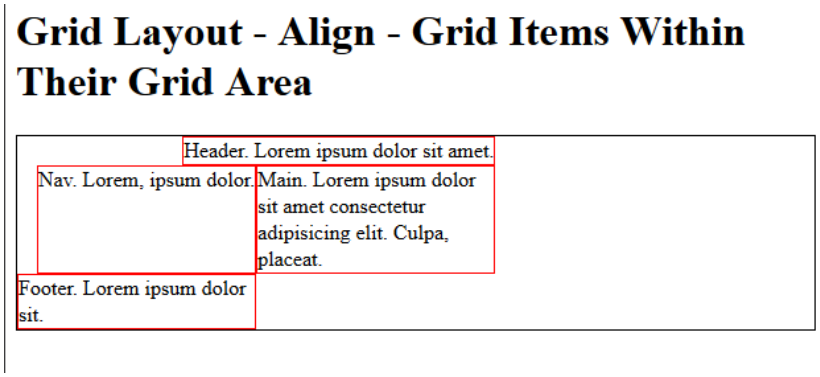

```
  nav { align-self: end; }
```



(W3C 2017, "CSS Grid Layout Module Level 1", https://www.w3.org/TR/css-grid-1/), "Alignment and Spacing", https://www.w3.org/TR/css-grid-1/#alignment

The above effects can also be achieved by using, on the grid container, `justify-items` and `align-items`.

```
#gridContainer {
  display: grid;
  grid-template-areas:
    "head head"
    "nav  main"
    "foot  ...";
  grid-template-columns:  30% 30%;
  justify-items: end;
}

header  { grid-area: head; }
```



C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\grid-layout-align-gridItemsWithinTheirGridArea.xhtml
(W3C 2017, "CSS Grid Layout Module Level 1", https://www.w3.org/TR/css-grid-1/), "Row-axis Alignment: the justify-self and justify-items properties", https://www.w3.org/TR/css-grid-1/#row-align
(W3C 2017, "CSS Grid Layout Module Level 1", https://www.w3.org/TR/css-grid-1/), " Column-axis Alignment: the align-self and align-items properties", https://www.w3.org/TR/css-grid-1/#row-align

Of the grid within the grid container

When the grid's outer edges do not line up with the grid container's then the grid can be aligned in the vertical and horizontal axis with respect to the grid. To do this:

- In the horizontal axis, use `justify-content` on the grid container;
- In the vertical axis, use `align-content` on the grid container;

```
#gridContainer {
  display: grid;
  grid-template-areas:
  "head nav"
  "main  foot";

  /* 01. Give the grid container some height so the grid edges in the vertical axis
  don't line up with the grid container */
  min-height: 200px;

  /* 02. Size the grid areas so they are smaller than the grid container dimensions */
  grid-template-columns:  auto 30%;
  grid-template-rows: 50px 50px;

  /* 03. Align the grid within the grid container */
  justify-content: start;
  align-content: center;
}
```

Grid Layout - Align - Grid Within Container

Header. Lorem ipsum dolor sit amet.	Nav. Lorem, ipsum dolor.
Main. Lorem ipsum dolor sit amet consectetur adipisicing elit. Culpa, placeat.	Footer. Lorem ipsum dolor sit.

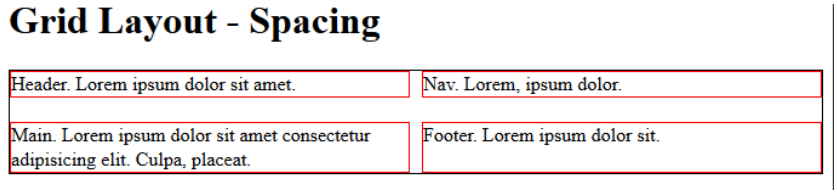
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\grid-layout-align-gridWithinContainer.xhtml

Spacing

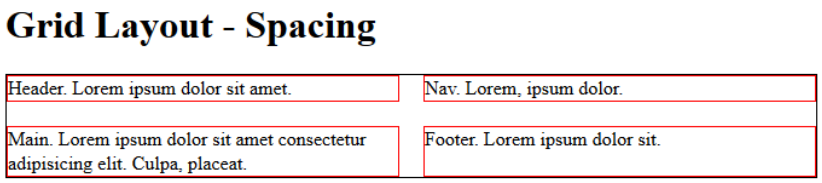
Create spacing between columns and rows, also known as "gutters", with:

- row-gap;
- column-gap; or
- grid-gap

```
#gridContainer {
  display: grid;
  grid-template-columns: 1fr 1fr;
  row-gap: 20px;
  column-gap: 10px;
}

/* Equivalent */
#gridContainer {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-gap: 20px 10px;
}
```



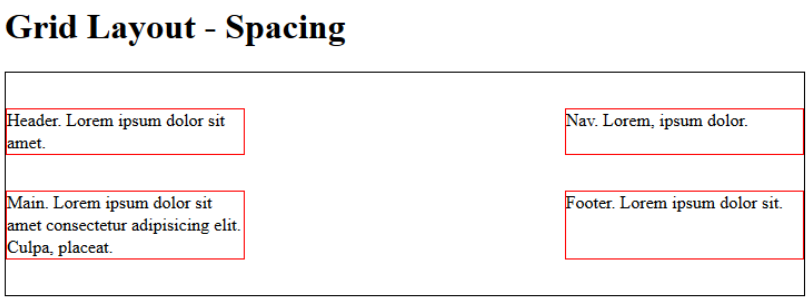

```
#gridContainer {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-gap: 20px; /* Both rows and columns */
}
```



(W3C 2017, "CSS Grid Layout Module Level 1", https://www.w3.org/TR/css-grid-1/), "Gutters: the row-gap, column-gap, and gap properties", https://www.w3.org/TR/css-grid-1/#gutters
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\grid-layout-spacing.xhtml

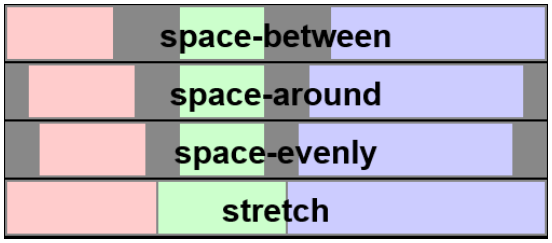
In addition, or alternatively, gutters can be created by using "distributed alignment" values on justify-content and/or align-content. The grid items must have edges that do not line up with the grid container, for the full effect. The distributed alignment values are: stretch (the default), space-between, space-around, and space-evenly.

```
#gridContainer {
  display: grid;

  /* Ensure the total of grid items are less than the grid container dimensions
     in the vertical and horizontal axis */
  grid-template-columns: 30% 30%;
  min-height: 200px;

  align-content: space-evenly;
  justify-content: space-between;
}
```



(W3C The Latest, "CSS Box Alignment Module Level 3", https://www.w3.org/TR/css-align-3/#valdef-align-content-space-around), " Distributed Alignment: the stretch, space-between, space-around, and space-evenly keywords", https://www.w3.org/TR/css-align-3/#distribution-values
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\grid-layout-spacing.xhtml

The effects of the distributed alignment values.



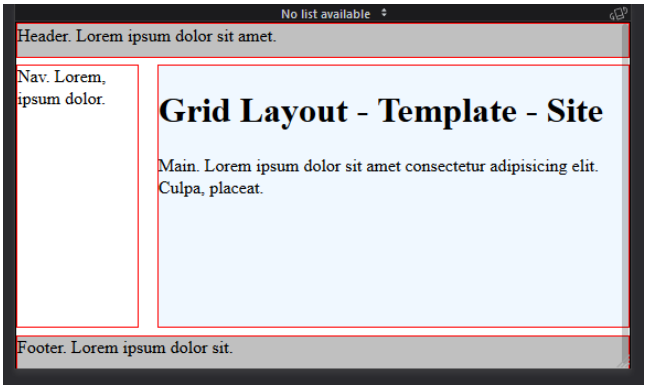
(W3C The Latest, "CSS Box Alignment Module Level 3", https://www.w3.org/TR/css-align-3/#valdef-align-content-space-around), " Distributed Alignment: the stretch, space-between, space-around, and space-evenly keywords", https://www.w3.org/TR/css-align-3/#distribution-values

Template Examples

For a site (using a grid)

Template for a site.

```
:root {
  --header-footer-height: 10vh;
}
body {
  padding: 0;
  margin: 0;
}
#gridContainer {
  /* 02. Declare a container has having grid display */
  display: grid;
  /* 03. Define grid areas */
  grid-template-areas:
  "head head"
  "nav  main"
  "foot foot";
  min-height: 100vh;
  grid-template-columns:  20% auto;
  grid-template-rows:
      var(--header-footer-height)
      auto
      var(--header-footer-height);
  grid-gap: 3%;
}

/* 04. Specify the position of grid items */
header  { grid-area: head; background-color: silver;  } */
nav     { grid-area: nav;  background-color: blanchedalmond }
main    { grid-area: main; background-color: aliceblue; }
footer  { grid-area: foot; background-color: silver;  }

/* 05. Add aesthetic sugur and further customisation */
#gridContainer {
  border: 1px solid black;
}
#gridContainer > * {
  border: 1px solid red;
}
```



C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\grid-layout-template-site.xhtml (W3C 2017, "CSS Grid Layout Module Level 1", https://www.w3.org/TR/css-grid-1/), "Reordering and Accessibility", https://www.w3.org/TR/css-grid-1/#order-accessibility

For a form

XHTML

Template for a form, XHTML:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>
    Forms - all controls, div enclosing.
  </title>
  <!-- <link rel="stylesheet" href="style/forms-div-enclosing-traditional.css" /> -->
  <link rel="stylesheet" href="style/forms-div-enclosing-grid.css" />
</head>
<body>
  <h1>Forms - all controls,  div enclosing.</h1>

  <form method="post" enctype="application/x-www-form-urlencoded"
action="http://localhost:8080/web/Examples/Html5Examples/forms-process.php">

    <section>
      <h2>Input element of various types</h2>

      <fieldset>
        <legend>Textish</legend>
        <input type="hidden" name="SecretValue" value="Kenny Everett" />
        <div>
          <label>Given name (type=text default)</label>
          <input name="FirstName" />
        </div>
        <div>
          <label>Family name (type=text explicit)</label> <input type="text"
name="FamilyName" />
        </div>
        <div>
          <label>Search (type=search)</label>
          <input type="search" name="Search" />
        </div>
      </fieldset>

     ...
      <fieldset>
        <legend>Pizza Toppings (type=checkbox)</legend>
        <div>
          <input type="checkbox" name="Toppings[]" value="Bacon" />
          <label>Bacon</label>
        </div>
```

```
        <div>
          <input type="checkbox" name="Toppings[]" value="Extra cheese" />
          <label>Extra Cheese</label>
        </div>
        <div>
          <input type="checkbox" name="Toppings[]" value="onion" />
          <label>Onion</label>
        </div>
      </fieldset>

...

        <p class="buttonGroup">
          <input type="submit" />
          <input type="reset" />
          <input type="image" name="ImageButton" src="images/beautiful-scene.jpg"
alt="Beautiful scene of mountains" />
        </p>

      </fieldset>
    </section>

    <section>
      <h2>Non-input elements</h2>

      ...

      <div>
        <label>Delivery instructions (textarea)</label>
        <textarea name="deliveryInstructions"></textarea>
      </div>

      <p class="buttonGroup">
        <button type="submit">Submit order</button>
        <button type="reset">Reset</button>
      </p>
    </section>
  </form>
</body>
</html>
```

C:\Users\John\Documents\Sda\Code\web\Examples\Html5Examples\forms-all-controls-div-enclosing.xhtml

CSS, using a grid

Template for a form, the CSS, using a grid.

```
:root{
  --column01-size: 30%;
  --buttonGroup-internalMarginLeft: 0.4em;
}
form {
  margin-bottom: 5em;
}
section {
  margin: 0 10%;
}

/* Grid */
div {
  display: grid;
  grid-template-areas: "column01 column02";
  grid-template-columns: var(--column01-size) 60%;
  grid-auto-rows: auto;
  align-items: center;
}

label {
  grid-area: column01;
  justify-self: end;
}
label:after{
  content: ":";
```

```
}

input, select, optgroup, textarea, output, progress, meter {
  grid-area: column02;
  margin: 0.5em;
}

input[type="checkbox"], input[type="radio"]  {
  grid-area: column02;
  justify-self: start;
}

/* buttonGroup */
fieldset .buttonGroup{
  margin-left: calc(var(--column01-size) +  calc(var(--buttonGroup-internalMarginLeft)));
}

.buttonGroup{
  margin-left: calc(var(--column01-size) + 0.5em);
}

.buttonGroup input[type=submit], .buttonGroup input[type=reset] {
  margin: 1em 1em 1em 0;
  display: inline-block;
  width: auto;
}

.buttonGroup input[type=image] {
  display: block;
  width: 200px;
  margin-left: 0;
}

@media (max-width: 600px) {
  div {
    display: block;
  }

  fieldset .buttonGroup, .buttonGroup {
    margin-left: 0;
  }

  h2 {
    color: red;
  }

  div label {
    text-align: left;
    margin-bottom: 0.5em;
    width: auto;
  }

  input {
    display: block;
  }

  input, textarea, select {
    width: 95%;
  }

  input[type=checkbox], input[type=radio] {
    display: inline-block;
    width: auto;
  }
  input[type=checkbox] ~ label::after, input[type=radio] ~ label::after {
    content: "";
  }
}
```

C:\Users\John\Documents\Sda\Code\web\Examples\Html5Examples\style\forms-div-enclosing-grid.css

Css, traditional

Template for a form, CSS, traditional

```
:root {
  --label-text-width: 250px;
  --first-in-line-labelless-control-indent: calc(var(--label-text-width) + 19px);
}

body {
  margin: 0;
  padding: 0 1em;
}

form {
  border: 1px solid black;
  margin: 0;
  padding: 0 1em;
}

fieldset {
  margin-top: 0.5em;
}

h2 {
  border-bottom: 1px solid;
  margin-top: 2em;
}

div {
  display: block;
  margin: 1em 0;
}

label {
  display: inline-block;
  text-align: right;
  margin-right: 1em;
}

div label {
  width: var(--label-text-width);
}

div label:first-child::after {
  content: ":";
}

div label:not(:first-child) {
  width: 1em;
}

input, textarea, select {
  margin: 0 1em 0 0;
  padding: 2px;
  width: 45%;
}

input[type=checkbox], input[type=radio] {
  width: auto;
  margin-left: 5%;
}

textarea {
  vertical-align: top;
}

[type=submit], [type=reset] {
  margin: 1em 1em 1em 0;
  display: inline-block;
  width: auto;
}

[type=submit], [type=image] {
  margin-left: var(--first-in-line-labelless-control-indent);
}

[type=image] {
```

```
  display: block;
  width: 200px;
}

@media (max-width: 600px) {
  :root {
    --label-text-width: 0px;
    --first-in-line-labelless-control-indent: 0px;
  }

  h2 {
    color: red;
  }

  div label {
    text-align: left;
    margin-bottom: 0.5em;
    width: auto;
  }

  input {
    display: block;
  }

  input, textarea, select {
    width: 95%;
  }

  input[type=checkbox], input[type=radio] {
    display: inline-block;
  }
}
```

C:\Users\John\Documents\Sda\Code\web\Examples\Html5Examples\style\forms-div-enclosing-traditional.css

Grid layout a table

Grid layout a table. Don't use flexbox to layout a table. When using grid layout on tables media queries you can change a row to become a column, thereby making a table "responsive".

You can layout a table with flexbox. See https://atlas/web/Examples/CssExamples/css2017/flexbox-table-production.xhtml. But it is generally harder to line up the columns exactly when shrunk. E.g. You'll get the following effect ...

Location	Type	Name
Version	**Home page**	**Licence**
Front end	Markup standard	HTML Living Standard
"Living"	https://html.spec.whatwg.org/multipage/	Not applicable (I don't distribute the document), but https://html.spec.whatwg.org/#acknowledgments, https://creativecommons.org/licenses/by/4.0/

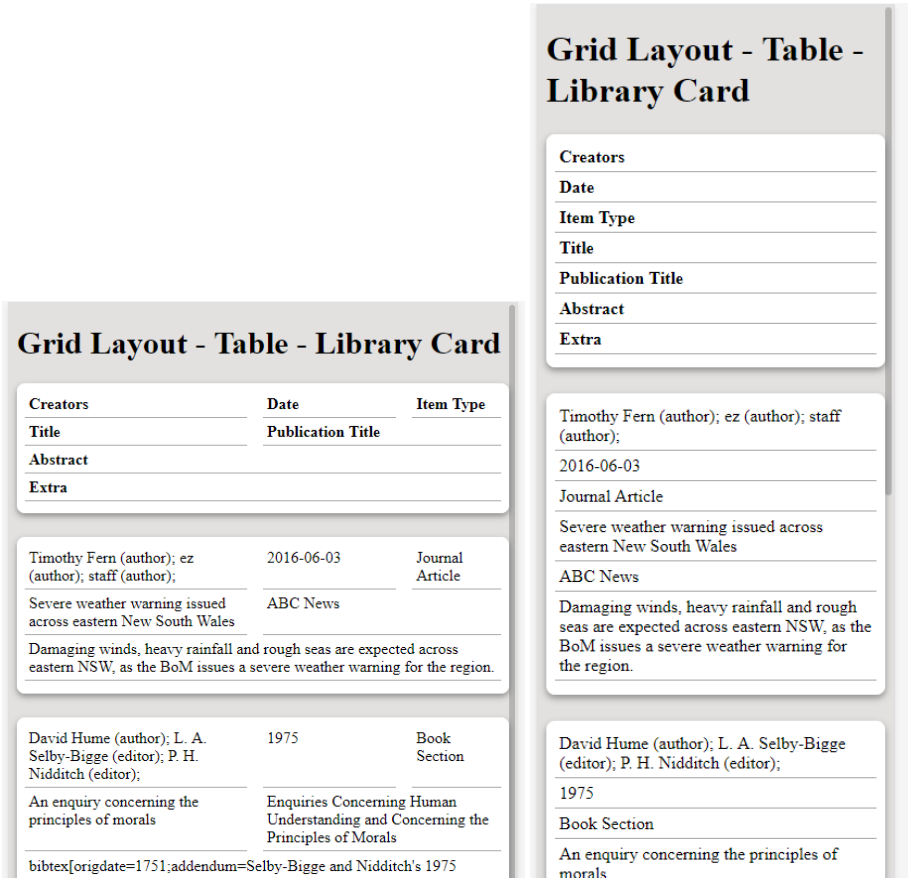
```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />

  <title>Grid Layout - Table - Library Card</title>

  <style>
    /* <![CDATA[ */

    table {
      width: 100%;
    }

    table tr {
      /* 02. Declare a container has having grid display */
      display: grid;

      /* 03. Define the grid areas and their sizing */
      /* Done in media queries bellow */

      /* 04. Use break-word (not break-all) to ensure shrunk areas don't overflow */
      word-break: break-word;

      /* Aesthetic sugar */
      row-gap: 1px;
      column-gap: 1em;
      padding-bottom: 12px;
    }

    /* 05. Specify the position of grid items */
    /* .creators { grid-area: creators;}
    .date { grid-area: date;}
    .itemType { grid-area: itemType;}
    .title {grid-area: title;} */
    .publicationTitle {grid-area: span 1 / span 2; }
    .abstractNote { grid-area: span 1 / span 3;}
    .extra { grid-area: span 1 / span 3;}

    /* 06. Add aesthetic sugur and further customisation */
    :root {
      --border-width: 1px;
    }
    body {
      background-color: #e2e1e0;
    }
    table {
      margin-top: 1em;
      border-collapse: collapse;
      max-width: 768px;
      margin-left: auto;
      margin-right: auto;
    }
    tr {
      background-color: white;
      border-radius: 8px;
      /* Material Design Box Shadows, A Pen By Samuel Thornton, Card 2,
https://codepen.io/sdthornton/pen/wBZdXq */
      box-shadow: 0 3px 6px rgba(0,0,0,0.16), 0 3px 6px rgba(0,0,0,0.23);
      padding: 0.5em;
      margin-bottom: 1.5em;
      text-align: left;
    }
    th, td {
      border-color: darkgray;
      /* border-color: transparent; */
      border-style: solid;
      border-left-width: 0;
      border-bottom-width: var(--border-width);
      border-right-width: 0;
      border-top-width: 0;
      padding: 0.25em;
    }
    td:empty {
      padding: 0;
      border: none;
    }

    /* 03. Define the grid areas and their sizing */
    @media screen and (min-width: 0px) {
```

```
      table tr {
        grid-template-columns: 1fr;
      }
      .publicationTitle {grid-area: span 1 / span 1; }
      .abstractNote { grid-area: span 1 / span 1;}
      .extra { grid-area: span 1 / span 1;}

    }
    @media screen and (min-width: 475px) {
      table tr {
        grid-template-columns: 5fr 3fr 2fr;
      }
      .publicationTitle {grid-area: span 1 / span 2; }
      .abstractNote { grid-area: span 1 / span 3;}
      .extra { grid-area: span 1 / span 3;}
    }

    /* ]]> */
  </style>

</head>
<body>
  <h1>Grid Layout - Table - Library Card</h1>

  <!-- 01. In HTML designate a grid container and set of grid items  -->
  <table>
    <thead>
      <tr>
        <th class="creators">Creators</th>
        <th class="date">Date</th>
        <th class="itemType">Item Type</th>
        <th class="title">Title</th>
        <th class="publicationTitle">Publication Title</th>
        <th class="abstractNote">Abstract</th>
        <th class="extra">Extra</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td class="creators">Timothy Fern (author); ez (author); staff (author); </td>
        <td class="date">2016-06-03</td>
        <td class="itemType">Journal Article</td>
        <td class="title">Severe weather warning issued across eastern New South Wales</td>
        <td class="publicationTitle">ABC News</td>
        <td class="abstractNote">Damaging winds, heavy rainfall and rough seas are expected
across eastern NSW, as the BoM issues a severe weather warning for the region.</td>
        <td class="extra"></td>
      </tr>
      <tr>
        <td class="creators">David Hume (author); L. A. Selby-Bigge (editor); P. H.
Nidditch (editor); </td>
        <td class="date">1975</td>
        <td class="itemType">Book Section</td>
        <td class="title">An enquiry concerning the principles of morals</td>
        <td class="publicationTitle">Enquiries Concerning Human Understanding and
Concerning the Principles of Morals</td>
        <td class="abstractNote"></td>
        <td class="extra">bibtex[origdate=1751;addendum=Selby-Bigge and Nidditch's 1975
edition is based off a collection of Hume's essays posthumously published in 1777. However,
Hume's "An Enquiry Concerning the Principles of Morals" was first published in 1751, and
that's the date we use here.]
  bibtex: hume_1751_enquiry</td>
      </tr>
      <tr>
        <td class="creators">David J. Chalmers (author)</td>
        <td class="date">1995</td>
        <td class="itemType">Journal Article</td>
        <td class="title">Facing up to the problem of consciousness</td>
        <td class="publicationTitle">Journal of Consciousness Studies</td>
        <td class="abstractNote"></td>
        <td class="extra"></td>
      </tr>
      <tr>
        <td class="creators">G. C. Abiogu (author); I. N. Mbaji (author); A. O. Adeogun
(author); </td>
        <td class="date">2015</td>
        <td class="itemType">Journal Article</td>
```

```
        <td class="title">Music Education and Youth Empowerment: A Conceptual
Clarification</td>
        <td class="publicationTitle"></td>
        <td class="abstractNote"></td>
        <td class="extra">bibtex: abiogu_2015_music
  bibtex[journaltitle=Open Journal of Philosophy]</td>
      </tr>
    </tbody>
  </table>
</body>
</html>
```

The resulting table for a wider and narrower screen respectively ....



Grid Layout - Table - Library Card

Creators	Date	Item Type
Title	Publication Title	
Abstract		
Extra		
Timothy Fern (author); ez (author); staff (author);	2016-06-03	Journal Article
Severe weather warning issued across eastern New South Wales	ABC News	
Damaging winds, heavy rainfall and rough seas are expected across eastern NSW, as the BoM issues a severe weather warning for the region.		
David Hume (author); L. A. Selby-Bigge (editor); P. H. Nidditch (editor);	1975	Book Section
An enquiry concerning the principles of morals	Enquiries Concerning Human Understanding and Concerning the Principles of Morals	
bibtex[origdate=1751;addendum=Selby-Bigge and Nidditch's 1975		

Grid Layout - Table - Library Card

Creators
Date
Item Type
Title
Publication Title
Abstract
Extra
Timothy Fern (author); ez (author); staff (author);
2016-06-03
Journal Article
Severe weather warning issued across eastern New South Wales
ABC News
Damaging winds, heavy rainfall and rough seas are expected across eastern NSW, as the BoM issues a severe weather warning for the region.
David Hume (author); L. A. Selby-Bigge (editor); P. H. Nidditch (editor);
1975
Book Section
An enquiry concerning the principles of morals

https://atlas/web/Examples/CssExamples/css2017/grid-layout-table--library-card.xhtml
https://atlas/web/Examples/CssExamples/css2017/grid-layout-table--tech-stack.xhtml

Flexbox

Overview

The spec:

> *(W3C The Latest, "CSS Flexible Box Layout Module Level 1", https://www.w3.org/TR/css-flexbox-1/)* https://www.w3.org/TR/css-flexbox-1/

A very handy summary:

> *(Coyier, Chris 2018, "A Complete Guide to Flexbox", https://css-tricks.com/snippets/css/a-guide-to-flexbox/)* https://css-tricks.com/snippets/css/a-guide-to-flexbox/

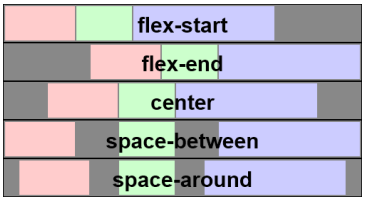
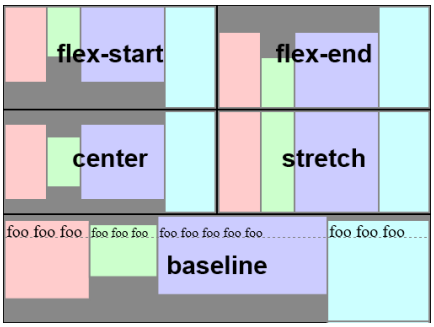
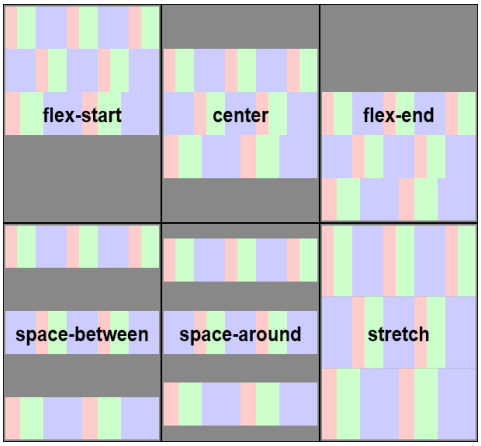
Flexbox operates on a "flex container" and child "flex items". To use flexbox set css declarations on the container and child items.

Example.

```
.flexContainer {
  outline: 1px solid black;
  display: flex;
  flex-direction: row;
  flex-wrap: nowrap;
  max-width: 800px;
  margin-bottom: 1em;
  /* align-items: center; */
  /* justify-content: space-between; */
  /* align-items: flex-start; */
  /* height: 200px; */
}
/* In production you'd just target the relevant selector. Here I'm using a
    "flexItem" class to make the example clear */
.flexItem {
  outline: 1px solid blue;
}
/* The last item is pushes againt the other flex items */
.flexItem01 {
  flex: 1 1 0;
}
.flexItem02 {
  flex: 1 1 0;
}
.flexItem03 {
  flex: 1 1 800px;
}
```

> *C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\flexbox-basics.xhtml*

Container declaration summary

Property	Meaning	Values	Examples
display	Sets an element as a flexbox container	flex inline-flex inline block	display: flex;
flex-direction		row row-reverse column column-reverse	flex-direction: row;
flex-wrap	Whether to allow items to spill over onto another line if they can't be kept on the one line.	wrap wrap-reverse nowrap	flex-wrap: wrap;
flex-flow	Shorthand for flex-direction and flex-wrap		flex-flow: row wrap;
justify-content	Distributes any free space left over if the flex items are inflexible or have reached their maximum size.		justify-conent: space-around;
align-items	If the cross axis dimension of the container (e.g. the vertical dimension if the main axis is horizontal) exceeds that of one of the flex items then those items can be aligned in the cross axis within the container.	auto 	align-items: flex-start;
align-content	Aligns "flex container lines" on the cross axis. Only relevant when items have wrapped onto multiple lines.		align-content: center;

(W3C The Latest, "CSS Flexible Box Layout Module Level 1", https://www.w3.org/TR/css-flexbox-1/)

Item declaration summary

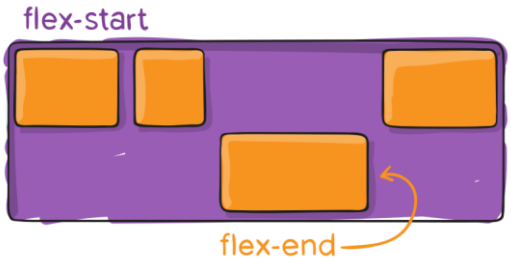
Property	Meaning	Values	Examples

`order`	Rearrange the order of flex items as desired.	`<integer>`
		`0 /* default */`
		`-1 /* lower than default*/`
			`order: 3;`

`flex-grow`		`<positive number>`	`flex-grow: 2;` `flex-grow: 0.2;`
`flex-shrink`		`<positive number>`	`flex-shrink: 2;` `flex-shrink: 0.2;`

`flex-basis`		`auto`
		`content`
		`<width> /* Can be 0 */`
			`flex-basis: auto;`
			`flex-basis: 0;`

`flex`	Shorthand for flex-grow, flex-shrink, and flex-basis. Authors are encouraged to use this shorthand as it sets other defaults intelligently.		`flex: 2;` `flex: 0 1 auto;`

`align-self`	Analogous to `align-items` on containers. Sets cross axis alignment on the item if the cross axis dimension of the container (e.g. the vertical dimension if the main axis is horizontal) exceeds that of one of the flex items	`auto`
		`flex-start`
		`flex-end`
		`center`
		`baseline`
		`stretch`
		 
			`align-self: flex-end;`

(W3C The Latest, "CSS Flexible Box Layout Module Level 1", https://www.w3.org/TR/css-flexbox-1/; Coyier, Chris 2018, "A Complete Guide to Flexbox", https://css-tricks.com/snippets/css/a-guide-to-flexbox/)

Overall procedure

To understand flexbox operations make a series of decisions in the following order ...

1. Decide if your child flex item(s) will get any fixed padding or margins. This doesn't effect flex settings directly, e.g. setting a fixed margin doesn't create free space against which a container's `justify-content` property can take effect.
2. Decide whether you want to create free space against which you might want to set flex effects (e.g. by setting `justify-content: space-between;`). The decision is:

   a. No. You don't want to create any free space.
   b. Yes. You can create free space in the following ways:
      i. Fix the child item(s) dimension(s) and don't set flex (let it default to the initial `flex: 0 1 auto;`). E.g. `width: 10em;`
      ii. Set a max-width or max-height on all of the child items, such that there is a possible container width or height that exceeds the total of all the child max-width's or max-heights. The child items can have varying max-* declaration's set.
      iii. Flex the child items with values between 0 and 1 ... "when the sum of the flex values on the line is less than 1, they will take up less than 100% of the free spac". E.g. `flex: 0.6;`

      (W3C The Latest, "CSS Flexible Box Layout Module Level 1", https://www.w3.org/TR/css-flexbox-1/)

3. Next decide if the container is flexible or has a fixed dimension.
   a. In the main axis; and
   b. In the cross axis. E.g. When `flex-direction: row;` is set on the container, and the container has it's height set to something that is larger than the height of one of it's children, only then do some other declarations become relevant (e.g. `align-items`).
4. Decide how you want your child items to flex ...

Understanding flex

Always use the flex shorthand rather then the longhand `flex-grow, flex-shrink, or flex-basis`.

> Authors are encouraged to control flexibility using the flex shorthand rather than with its longhand properties directly, as the shorthand correctly resets any unspecified components to accommodate common uses.

(W3C The Latest, "CSS Flexible Box Layout Module Level 1", https://www.w3.org/TR/css-flexbox-1/)
https://www.w3.org/TR/css-flexbox-1/#flex-components

Foundational concepts

See

(MDN 2018, "Controlling Ratios of Flex Items Along the Main Axis", https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Controlling_Ratios_of_Flex_Items_Along_the_Main_Ax)
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Controlling_Ratios_of_Flex_Items_Along_the_Main_Ax

To understanding flexing first consider what happens on a regular (non flexed) block when you set its width to (currently `*-content` has limited browser support https://caniuse.com/#feat=intrinsic-width) :

- A fixed width. `width: 100px;`

- `width: auto;` The block item expands to fill its container.

- `width: min-content;`

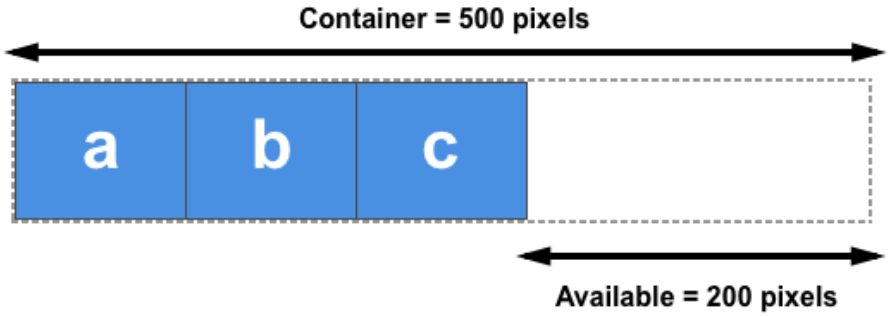
- `width: max-content;`

(W3C The Latest, "CSS Intrinsic & Extrinsic Sizing Module Level 3", https://www.w3.org/TR/css-sizing-3/)

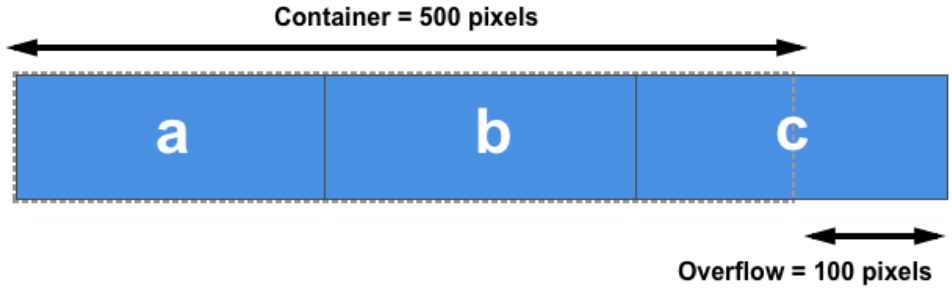
In (MDN 2018, "Controlling Ratios of Flex Items Along the Main Axis", https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Controlling_Ratios_of_Flex_Items_Along_the_Main_Ax) a flex item is described as having a "natural size", a size before in growth or shrinkage is applied.

A flex container will have positive or negative free space, along it's main axis:

- Positive free space if the total of natural size (e.g. width) of the flex items are **less** than the size (e.g. width) of the container; and



- Negative free space if the total of natural size (e.g. width) of the flex items are **greater** than the size (e.g. width) of the container;



(MDN 2018, "Controlling Ratios of Flex Items Along the Main Axis", https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Controlling_Ratios_of_Flex_Items_Along_the_Main_Ax)

Flex shorthand and flex-basis

Flexing the items (and/or container) entails establishing a "natural size" of the items then (optionally) growing or shrinking the items, thereby "distributing free space" to fill the container. After this there may, or may not, be space "left over" (john Bentley term).

Set the "natural size" of a flex item with flex-basis in the following ways:

- Fix the child item'(s) dimension(s) and don't allow any flex.

```
#item01 {
  width: 10em;

  // This is the default but better to be explicit
  // flex-grow set to 0.
  flex: 0 1 auto;
}
```

- Definite (but not fixed) size.

```
// Set flex-basis to auto and main width to a definite size.
#item01 {
  width: 100px;
  flex: 1 1 auto;
}

// Set flex-basis to a definite size.
#item01 {
  flex: 1 1 100px;
}
```

- Max-content.

```
// Flex-basis to auto and main width to auto.
#item01 {
  width: auto;
  flex: 1 1 auto;
}

// Note the initial value of width (if the prop. is omitted) is 'auto'
#item01 {
  flex: 1 1 auto;
}

// Flex-basis to content.
#item01 {
```

```
  flex: 1 1 content;
}
```

- None: use flex-grow entirely for distributing free space; and ignore flex-shrink values ... shrinking will be equal (see Flex shrink below).

```
// Flex-basis to 0.
#item01 {
  flex: 1 1 0;
}

// "When omitted from the flex shorthand, its specified value is 0."
#item01 {
  flex: 1 1;
}
```

(MDN 2018, "Controlling Ratios of Flex Items Along the Main Axis", https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Controlling_Ratios_of_Flex_Items_Along_the_Main_Ax)
(W3C The Latest, "CSS Flexible Box Layout Module Level 1", https://www.w3.org/TR/css-flexbox-1/)
https://www.w3.org/TR/css-flexbox-1/

`flex` shorthand defaults:

- The spec stipulates "initial" values of: `0 1 auto` (flex-grow; flex-shrink: flex-basis); but
- When flex-basis is omitted from the shorthand it gets a value of 0.

```
#flexItem01 {
  /* Initial: 0 1 auto */
}

#flexItem01 {
    flex: 2 4;  /* flex-basis is 0 when omitted */
}
```

(W3C The Latest, "CSS Flexible Box Layout Module Level 1", https://www.w3.org/TR/css-flexbox-1/) , "The flex Shorthand", https://www.w3.org/TR/css-flexbox-1/#flex-property

Flex grow

Flex grow will grow the item only if there is "positive free space" (the "natural size", controlled by flex-basis, of all the items is less than the "natural size" of the container).

If there is to be flex growth then:

- "Distribute free space" according to the ratio of the item's flex-grow factor compared to the total of the flex-grow values set on all the flex items;

```
.flexItem01 {
  flex: 1; /* defaults for flex-shrink: 1; flex-basis; 0; */
}
.flexItem02 {
  flex: 1; /* defaults for flex-shrink: 1; flex-basis; 0; */
}
.flexItem03 {
  flex: 4 /* defaults for flex-shrink: 1; flex-basis; 0; */
}
```

Lorem.	Architecto?	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id quaerat asperiores laudantium velit est itaque facilis tempora, unde architecto necessitatibus nostrum, aliquid autem.!
Lorem. Lorem, ipsum dolor.	Architecto? Lorem ipsum dolor sit amet consectetur adipisicing.	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id quaerat asperiores laudantium velit est itaque facilis tempora, unde architecto necessitatibus nostrum, aliquid autem.!

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\flexbox-basics.xhtml

- Before distributing free space start with the "natural size" of the item, as set by the flex-basis.

```
.flexItem01 {
  flex: 1 1 auto; /* auto here will, in effect mean max-content */
}
.flexItem02 {
  flex: 1 1 auto; /* auto here will, in effect mean max-content */
}
.flexItem03 {
  flex: 1 1 auto; /* auto here will, in effect mean max-content */
                                                    }
```

Lorem.	Architecto?	Repellat.
Lorem. Lorem, ipsum dolor.	Architecto? Lorem ipsum dolor sit amet consectetur adipisicing.	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id quaerat asperiores laudantium velit est itaque facilis tempora, unde architecto necessitatibus nostrum, aliquid autem.!

Since, in the example, all the flex items will be set to max-content the items will flex to fit their content, with free space left over as follows:

Then the free space is distributed equally (because the flex-grow factor is 1 for each item), while the larger item retains its larger width:

(MDN 2018, "Controlling Ratios of Flex Items Along the Main Axis", https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Controlling_Ratios_of_Flex_Items_Along_the_Main_Ax)
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\flexbox-basics.xhtml

- Alternatively you could force the items to flex in a way that doesn't take into account their content. Just set the flex-basis, in the flex shorthand, to zero (or omit flex-basis from the flex shorthand).

```
/* Flex-basis is 0;  use flex-grow entirely for distributing free space; and ignore
flex-shrink values ... shrinking will be equal (see Flex shrink below). */
.flexItem01 {
  flex: 1 1 0;
}
.flexItem02 {
```

```
  flex: 1 1 0;
}
.flexItem03 {
  flex: 1 1 0;
}
```

Lorem.	Architecto?	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit.
Lorem. Lorem, ipsum dolor.	Architecto? Lorem ipsum dolor sit amet consectetur adipisicing.	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id quaerat asperiores laudantium velit est itaque facilis tempora, unde architecto necessitatibus nostrum, aliquid autem.!

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\flexbox-basics.xhtml

- ... Or you could set some definite flex-basis values to alter the "natural size" of the items before growth is applied. Note a definite flex-basis value **doesn't** set the width (or height) of the item. Rather the higher the definite flex-basis value, the more room the child item takes up.

```
.flexContainer {
...
  max-width: 800px;
...
}
.flexItem01 {
  flex: 1 1 0;
}
.flexItem02 {
  flex: 1 1 0;
}
.flexItem03 {
  flex: 1 1 0;
}
```

Lorem.	Architecto?	Repe div.flexItem.flexItem03 266.667 × 90
Lorem. Lorem, ipsum dolor.	Architecto? Lorem ipsum dolor sit amet consectetur adipisicing.	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id quaerat asperiores laudantium velit est itaque facilis tempora, unde architecto necessitatibus nostrum, aliquid autem.!

```
.flexItem03 {
  flex: 1 1 1px;
}
```

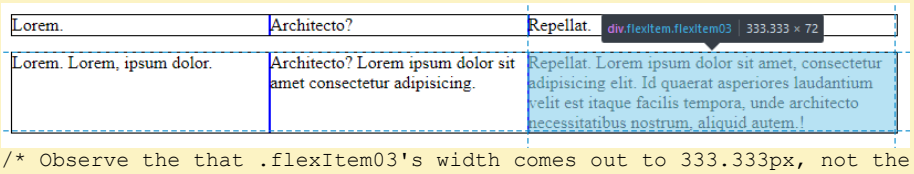
Lorem.	Architecto?	Repe div.flexItem.flexItem03 267.333 × 90
Lorem. Lorem, ipsum dolor.	Architecto? Lorem ipsum dolor sit amet consectetur adipisicing.	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id quaerat asperiores laudantium velit est itaque facilis tempora, unde architecto necessitatibus nostrum, aliquid autem.!

```
.flexItem03 {
  flex: 1 1 10px;
}
```

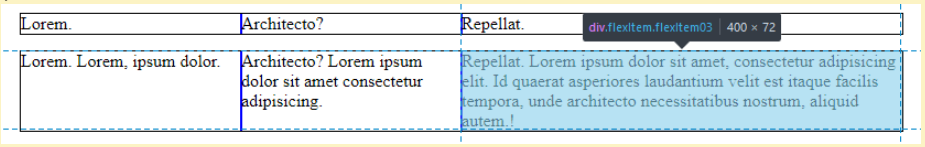
Lorem.	Architecto?	Repel div.flexItem.flexItem03 273.333 × 90
Lorem. Lorem, ipsum dolor.	Architecto? Lorem ipsum dolor sit amet consectetur adipisicing.	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id quaerat asperiores laudantium velit est itaque facilis tempora, unde architecto necessitatibus nostrum, aliquid autem.!

```
.flexItem03 {
  flex: 1 1 100px;
```

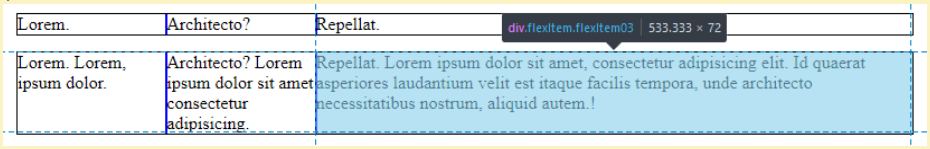
```
}
```



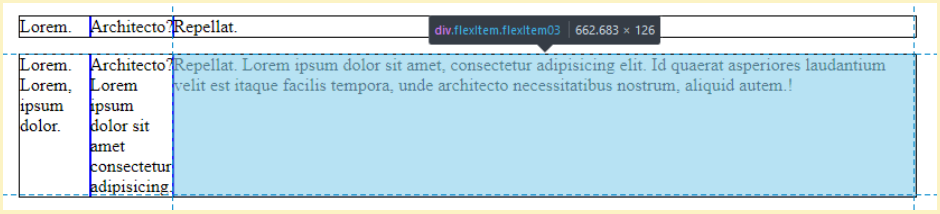

```
/* Observe the that .flexItem03's width comes out to 333.333px, not the 100px set as
the flex-basis */

.flexItem03 {
  flex: 1 1 200px;
}
```



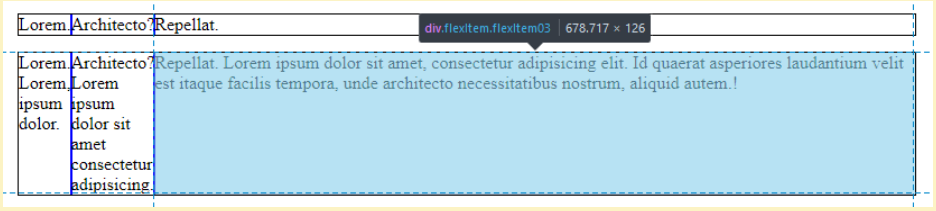

```
.flexItem03 {
  flex: 1 1 400px;
}
```




```
.flexItem03 {
  flex: 1 1 600px;
}
```




```
.flexItem03 {
  flex: 1 1 800px;
}
```



C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\flexbox-basics.xhtml

Flex shrink

> Where flex-grow deals with adding available space, flex-shrink manages taking away space to make boxes fit into their container without overflowing.

(MDN 2018, "Controlling Ratios of Flex Items Along the Main Axis", https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Controlling_Ratios_of_Flex_Items_Along_the_Main_Ax), "The flex-shrink property", https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Controlling_Ratios_of_Flex_Items_Along_the_Main_Ax#The_flex-shrink_property

When flex-shrink applies it works to make the item with the higher flex factor shrink faster.

```
/* Baseline:
  - the total width of items is equal to container width;
  - the flex-shrink ratios are equal */
.flexContainer {
  outline: 1px solid black;
  display: flex;
  flex-direction: row;
  flex-wrap: nowrap;
  max-width: 600px;
  margin-bottom: 1em;
}

.flexItem01 {
  flex: 1 1 200px;
}
.flexItem02 {
  flex: 1 1 200px;
}
.flexItem03 {
  flex: 1 1 200px;
}
```

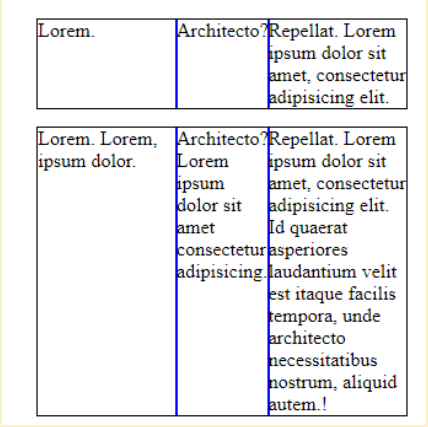
Lorem.	Architecto?	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit.
Lorem. Lorem, ipsum dolor.	Architecto? Lorem ipsum dolor sit amet consectetur adipisicing.	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id quaerat asperiores laudantium velit est itaque facilis tempora, unde architecto necessitatibus nostrum, aliquid autem.!

```
/* ... reduce the width of the container to effect overflow */
.flexContainer {
  ...
  max-width: 300px;
  ....
}

/* ... the items are shrunk and retain an equal distribution */
```

Lorem.	Architecto?	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit.
Lorem. Lorem, ipsum dolor.	Architecto? Lorem ipsum dolor sit amet consectetur adipisicing.	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id quaerat asperiores laudantium velit est itaque facilis tempora, unde architecto necessitatibus nostrum, aliquid autem.!

```
/* ... Now let's increase the flex-shrink factor on one of the items */
.flexItem01 {
  flex: 1 1 200px;
}
.flexItem02 {
  flex: 1 10 200px;
}
.flexItem03 {
  flex: 1 1 200px;
}
```

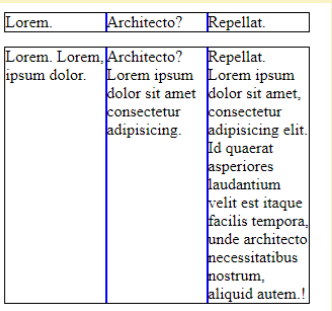
```
/* Compared wih the previous that item is now shrunk more (and the other items shrunk less)
... in this case up the limit of its min-content (this limit always applies) */
```



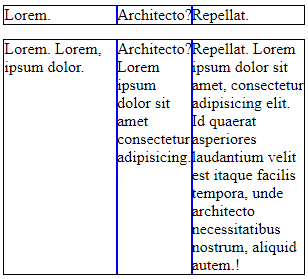
Flex-shrink only applies when all of the following conditions are met(???):

- The container's `flex-wrap` is set to `nowrap`. Otherwise the items will just wrap instead of shrink; and
- There is overflow: the total of the flex-basis width of the items is greater than the container's width (it is also a good idea to use `max-width` to preserve the ability of the container to shrink).
- The flex items have a flex-basis that is:
    a. Definite. E.g.:
        i. 200px set either directly (`flex: 1 1 200px`); or
        ii. With an auto value that relies on an item's width being definitely set (`width: 200px; flex: 1 1 auto`); or

```
/* Baseline: definite flex-basis set; and shrink factor is equal */
.flexContainer {
  ...
  max-width: 300px;
  ....
}
.flexItem01 {
  flex: 1 1 200px;
}
.flexItem02 {
  flex: 1 1 200px;
}
.flexItem03 {
  flex: 1 1 200px;
}
```




```
/* Change the flex shrink of the second item ... */
```



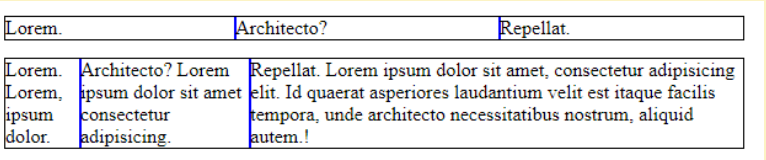

```
/* Where second item content was overflowing, it has shrunk faster than the
other items */
```

b. Max-content. E.g.:

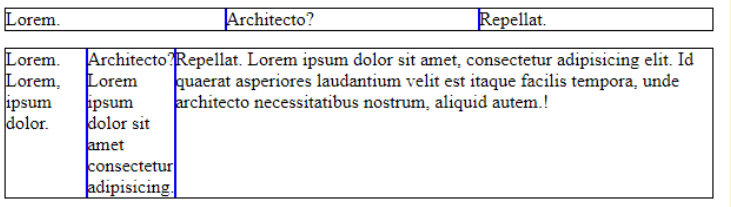
   i. By being set directly (`flex: 1 1 content`); or

   ii. By relying on auto values (`width: auto; flex: 1 1 auto`)

```
/* Baseline: max-content set and flex-shrink factor equal */
.flexContainer {
  ...
  max-width: 600px;
  ....
}
.flexItem01 {
  flex: 1 1 content;
}
.flexItem02 {
  flex: 1 1 content;
}
.flexItem03 {
  flex: 1 1 content;
}
```




```
/* Change the flex shrink of the second item ... */
.flexItem01 {
  flex: 1 1 content;
}
.flexItem02 {
  flex: 1 10 content;
}
.flexItem03 {
  flex: 1 1 content;
}
```




```
/* Where second item content was overflowing, it has shrunk faster than the
other items */
```

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\flexbox-basics.xhtml

When the flex-basis is set to 0 then flex shrink ratios have no effect.

```
/* Baseline: flex-basis set to 0; and flex-shrink ratios equal */
.flexContainer {
  ...
  max-width: 300px;
  ....
}.flexItem01 {
  flex: 1 1 0;
}
.flexItem02 {
  flex: 1 1 0;
}
.flexItem03 {
  flex: 1 1 0;
}
```

Lorem.	Architecto?	Repellat.
Lorem. Lorem, ipsum dolor.	Architecto? Lorem ipsum dolor sit amet consectetur adipisicing.	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id quaerat asperiores laudantium velit est itaque facilis tempora, unde architecto necessitatibus nostrum, aliquid autem.!

```
/* ... flex items have equal shrink */

/* Change the second item's flex-shrink factor ... */
}.flexItem01 {
  flex: 1 1 0;
}
.flexItem02 {
  flex: 1 10 0;
}
.flexItem03 {
  flex: 1 1 0;
}
```

Lorem.	Architecto?	Repellat.
Lorem. Lorem, ipsum dolor.	Architecto? Lorem ipsum dolor sit amet consectetur adipisicing.	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id quaerat asperiores laudantium velit est itaque facilis tempora, unde architecto necessitatibus nostrum, aliquid autem.!

```
/* ... no change from baseline */
```

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\flexbox-basics.xhtml

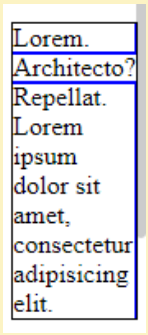
> The flex shrink factor is multiplied by the flex base size when distributing negative space.

(W3C The Latest, "CSS Flexible Box Layout Module Level 1", https://www.w3.org/TR/css-flexbox-1/), "The flex shorthand, <flex-shrink>" https://www.w3.org/TR/css-flexbox-1/#valdef-flex-flex-shrink

When settings a width on the container you generally want to use the `max-width` rather than `width` property.

You generally want to allow the container to shrink.

Flex shrink is constrained by the item's `width: min-content;` size. That is, an item can't shrink smaller than when there are no more opportunity to soft-wrap it's text.



C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\flexbox-basics.xhtml

Space left over - alignment

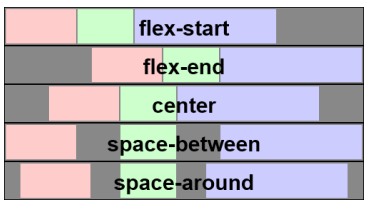
If the ratios of all the flex-grow items is less than 1 then this will ensure there is "space left over".

```
/* Flex-grow total is less than 1 therefore create space left over in the container. */
.flexItem01 {
  flex: 0.3 1 0;
}
.flexItem02 {
  flex: 0.3 1 0;
}
.flexItem03 {
  flex: 0.3 1 0;
}
```

Lorem.	Architecto?	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit.	
Lorem. Lorem, ipsum dolor.	Architecto? Lorem ipsum dolor sit amet consectetur adipisicing.	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id quaerat asperiores laudantium velit est itaque facilis tempora, unde architecto necessitatibus nostrum, aliquid autem.!	

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\flexbox-basics.xhtml

Only if there is space "left over" (John Bentley term) can you then decide how you want it further distributed, or "aligned", by setting `justify-content` (e.g. `justify-content: space-between`) on the container;



You can also set a child flex item's margin to auto (left, right, top, bottom, all of these, or some combination) to absorb free space, aligning the item against the container and possibly also pushing against other child flex items.

```
/* The last item is centered with the space left over. */
.flexItem01 {
  flex: 0.3 1 0;
}
.flexItem02 {
  flex: 0.3 1 0;
}
.flexItem03 {
  flex: 0.3 1 0;
  margin: 0 auto;
}
```

Lorem.	Architecto?		Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit.
Lorem. Lorem, ipsum dolor.	Architecto? Lorem ipsum dolor sit amet consectetur adipisicing.		Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id quaerat asperiores laudantium velit est itaque facilis tempora, unde architecto necessitatibus nostrum, aliquid autem.!

```
/* The last item is pushes againt the other flex items */
.flexItem01 {
  flex: 0.3 1 0;
}
.flexItem02 {
  flex: 0.3 1 0;
}
.flexItem03 {
  flex: 0.3 1 0;
  margin-left: auto;
}
```

Lorem.	Architecto?		Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit.
Lorem. Lorem, ipsum dolor.	Architecto? Lorem ipsum dolor sit amet consectetur adipisicing.		Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id quaerat asperiores laudantium velit est itaque facilis tempora, unde architecto necessitatibus nostrum, aliquid autem.!

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\flexbox-basics.xhtml

Constraining flex

Whether there is any space "left over" (John Bentley term) or not you can constrain an item's ability to flex by setting `max-*` and/or `min-*` constraints:

- Without `max-*` or `min-*` constraints:

```
.flexItem01 {
  flex: 1 1 0;
}
.flexItem02 {
  flex: 1 1 0;
}
```

```
.flexItem03 {
  flex: 4 1 0;
}
```

Lorem.	Architecto?	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit.
Lorem. Lorem, ipsum dolor.	Architecto? Lorem ipsum dolor sit amet consectetur adipisicing.	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id quaerat asperiores laudantium velit est itaque facilis tempora, unde architecto necessitatibus nostrum, aliquid autem.!

- For shrink you can set a `min-*` property to bound the shrink of items;

```
.flexItem01 {
  min-width: 300px;
  flex: 1 1 0;
}
.flexItem02 {
  flex: 1 1 0;
}
.flexItem03 {
  flex: 4 1 0;
}
```

Lorem.	Architecto?	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit.
Lorem. Lorem, ipsum dolor.	Architecto? Lorem ipsum dolor sit amet consectetur adipisicing.	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id quaerat asperiores laudantium velit est itaque facilis tempora, unde architecto necessitatibus nostrum, aliquid autem.!

- For growth you can set a `max-*` property to bound the growth of items;

```
.flexItem01 {
  flex: 1 1 0;
}
.flexItem02 {
  flex: 1 1 0;
}
.flexItem03 {
  max-width: 300px;
  flex: 4 1 0;
}
```

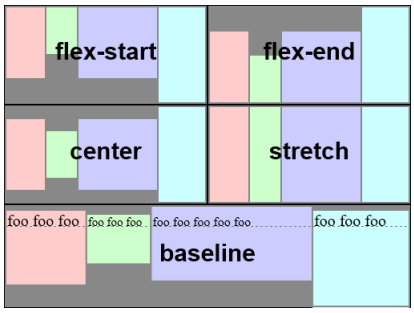
Lorem.	Architecto?	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit.
Lorem. Lorem, ipsum dolor.	Architecto? Lorem ipsum dolor sit amet consectetur adipisicing.	Repellat. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id quaerat asperiores laudantium velit est itaque facilis tempora, unde architecto necessitatibus nostrum, aliquid autem.!

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\flexbox-basics.xhtml

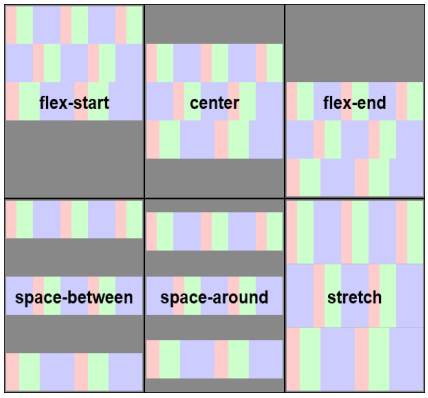
Generally avoid using `max-*` and/or `min-*` constraints as it places limits on the extent to which flex can work to preserve a responsive page (a page viewed on a range of screen sizes).

Cross dimensional alignment

For the cross dimension use `align-content` (e.g. `align-content: space-around;`)



For multilined items in the cross dimensions (when `flex-wrap: wrap;`) use align-content (e.g. `align-content: center;`)



Collapsed Flex item

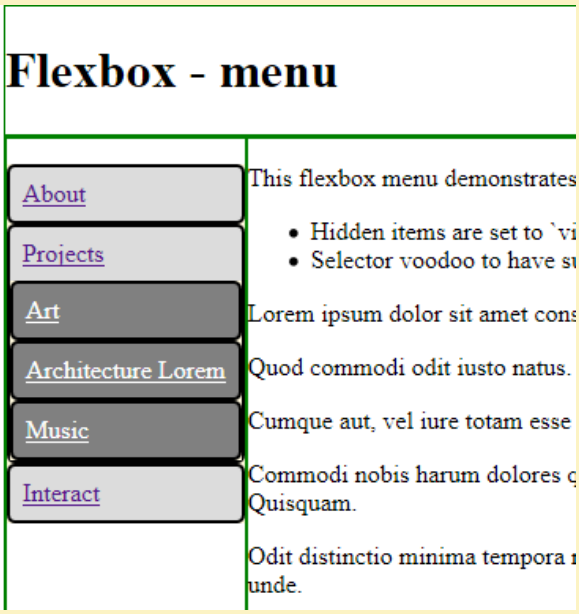
A collapsed flex item removes the item from rendering. But it leaves an invisible "strut" behind that keep's the flex item's cross-size (e.g. hortizontal width where the flex container has flex-direction set to column). To set use `visibility:collapse` on a flex item.

In the following example `visibility:collapse` ensures that the nav menu's width is sized to the largest li, whether visible or not.

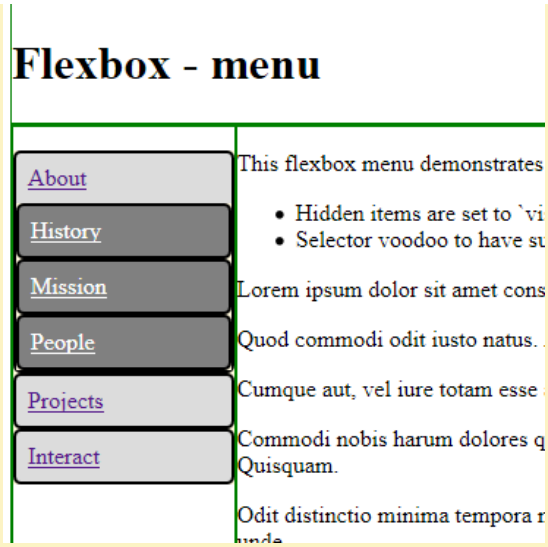
```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>Flexbox - menu</title>

  <style>
    /* <![CDATA[ */
    /* Debugging guides */
    header, main, nav, article {
      border: 1px solid green;
    }
    nav ul {
      background-color: cornsilk;
    }

    /* Flexbox for basic layout of page elements, with nav shrunk to fit content */
    main {
      display: flex;
      flex-wrap: wrap;
    }
    nav {
      flex: 1 0 min-content;
```

```
    }
    article {
      /* max-width: 700px; */
      flex: 6 0;
    }

    /* Flexbox for menu items */
    nav ul {
      display: flex;
      flex-direction: column;
      list-style: none;
      padding: 0;

    }

    /* Aesthetics */
    nav > ul > li {
      background: gainsboro;
      padding: 0;
      display: flex;
      flex-direction: column;
    }

    nav li {
      border: 2px solid black;
      border-radius: 5px;
    }
    nav li {
      background: grey;
      padding: 0.5em;
    }
    nav li li a {
      color: white;
    }
    nav > ul > li > a {
      margin: 0.5em;
    }

    /* Make a collapsed flex item (assumes ul is a flex item) */
    nav > ul > li:not(:target):not(:hover) > ul {
      visibility: collapse;
    }

    /* ]]> */
  </style>

</head>
<body>
  <header>
    <h1>Flexbox - menu</h1>
  </header>
  <main>

    <nav>
      <ul>
        <li id="nav-about">
          <a href="#nav-about">About</a>
          <ul>
            <li><a href="#">History</a>
            </li>
            <li><a href="#">Mission</a>
            </li>
            <li><a href="#">People</a>
            </li>
          </ul>
        </li>
        <li id="nav-projects">
          <a href="#nav-projects">Projects</a>
          <ul>
            <li><a href="#">Art</a>
            </li>
            <li><a href="#">Architecture Lorem</a>
            </li>
            <li><a href="#">Music</a>
            </li>
          </ul>
```

```
        </li>
        <li id="nav-interact">
          <a href="#nav-interact">Interact</a>
          <ul>
            <li><a href="#">Blog</a>
            </li>
            <li><a href="#">Forums</a>
            </li>
          </ul>
        </li>
      </ul>
    </nav>

    <article>

      <p>This flexbox menu demonstrates the following:</p>
      <ul>
        <li>Hidden items are set to `visibility:collapse` which preserves the cross-axis
width even though hidden.</li>
        <li>Selector voodoo to have submenus open and close in response to user
interaction, without needing javascript.</li>
      </ul>

      <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Temporibus praesentium
provident reprehenderit natus, iusto exercitationem itaque a totam commodi officiis amet
veritatis. Ipsam tenetur aut aperiam ut illo eum itaque!</p>

      ...

    </article>
  </main>
</body>
</html>

"Architecture Lorem" sets the nav menu width ...
```




```
... When the "Projects" submenu is collapsed then the nave menu overall retains it's width
(that is, even though "Architecture Lorem" is now hidden ...
```



(W3C The Latest, "CSS Flexible Box Layout Module Level 1", https://www.w3.org/TR/css-flexbox-1/), "Collapsed Items ", https://www.w3.org/TR/css-flexbox-1/ - abspos-items *https://www.w3.org/TR/css-flexbox-1/#visibility-collapse*

Flexbox techniques

Holy grail layout

See source for John Bentley's Holy Grail Layout. It has the following features:

- Have a header, footer, nav, article and aside block.
- These will have the same ordering in the DOM and the visual order.
- All middle columns will have the same height, regardless of which colum is the tallest.
- The footer should stick to the bottom of the viewport when the main content (i.e. in nav, article, and aside) is minimal.
- Have a "Skip to main content" link.
- It is a responsive layout. It is verified to work on mobile.

Production:
C:\Users\John\Documents\Sda\Code\web\Libraries\Html5Library\html5-polyglot-template-040-project\holyGrailLayout
C:\Users\John\Documents\Sda\Code\web\Libraries\Html5Library\html5-polyglot-template-040-project\index.xhtml

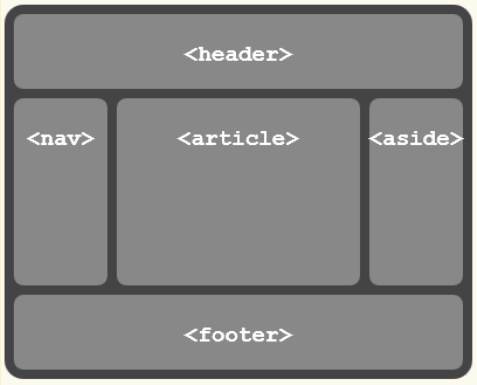
Simple/Demo: C:\Users\John\Documents\Sda\Code\web\Libraries\CssLibrary\flexbox-holygrail-layout-simple.xhtml

Don't use reordering in a Holy Grail Layout.

The W3C uses a "holy grail layout" to demonstrate reordering with flexbox:

```
// From
<!DOCTYPE html>
<header>...</header>
<main>
```

```
   <article>...</article>
   <nav>...</nav>
   <aside>...</aside>
</main>
<footer>...</footer>

// ... achieve
```




```
// ... with ...
main { display: flex; }
main > article { order: 2; min-width: 12em; flex:1; }
main > nav     { order: 1; width: 200px; }
main > aside   { order: 3; width: 200px; }
```

However, that would make the visual order different from the DOM order.

There is an Accessibility CSS technique "C27: Making the DOM order match the visual order", https://www.w3.org/WAI/WCAG21/Techniques/css/C27 from (W3C 2019, "Techniques for WCAG 2.1", https://www.w3.org/WAI/WCAG21/Techniques/):

> The objective of this technique is to ensure that the order of content in the source code is the same as the visual presentation of the content. ...

... and there's a fairly convincing explanation why contradicting the technique is bad for accessibility ...

> Each order may be meaningful in itself but may cause confusion for assistive technology users. This could be due to the user switching off the author-specified presentation, by accessing the content directly from the source code (such as with a screen reader), or by interacting with the content with a keyboard. If a blind user, who reads the page with a screen reader that follows the source order, is working with a sighted user who reads the page in visual order, they may be confused when they encounter information in different orders. A user with low vision who uses a screen magnifier in combination with a screen reader may be confused when the reading order appears to skip around on the screen. A keyboard user may have trouble predicting where focus will go next when the source order does not match the visual order.
>
> There may also be situations where the visually presented order is necessary to the overall understanding of the page, and if the source order is presented differently, it may be much more difficult to understand.
>
> When the source order matches the visual order, everyone will read the content and interact with it in the same (correct) order.

However, the success criteria for the technique is has the following to say:

With respect to "Understanding Success Criterion 1.3.2: Meaningful Sequence", https://www.w3.org/WAI/WCAG21/Understanding/meaningful-sequence

> Example 2: CSS is used to position a navigation bar, the main story on a page, and a side story. The visual presentation of the sections does not match the programmatically determined order, but the meaning of the page does not depend on the order of the sections.

... and with respect to " Understanding Success Criterion 2.4.3: Focus Order", https://www.w3.org/WAI/WCAG21/Understanding/focus-order

> An HTML Web page is created with the left hand navigation occurring in the HTML after the main body content, and styled with CSS to appear on the left hand side of the page. This is done to allow focus to move to the main body content first without requiring tabIndex attributes or JavaScript.

That is, against the technique "C27: Making the DOM order match the visual order" both relevant success criteria explicitly allow the visual order to be different form the DOM order with respect where the <nav> block is placed. In other words, placing the <nav> element after the main/article content in the DOM while reordering it in the visual order *does not* violate any accessibility "principle" or "guideline" (indeed doing so seems to be promoted to some degree).

However, I take this to be a bad thing. That is, that we should take the explanation in "C27: Making the DOM order match the visual order" to heart and never, for accessibility reasons, have <nav>, <main>, <article> etc. have a visual order that doesn't match the DOM order.

Moreover, if you have the nav come after the main content then a blind reader will not have convenient access to the navigation menu. To overcome this you'd have to introduce a "Skip to navigation" link. Therefore it is not as if, in the source, placing the nav block after the main content removes the need to provide a "Skip to" link.

My recommendation, then, is:

1. Maintain the DOM and Visual order.
2. Have the nav block before the main content (whether the main content is in a <main> or <article> block, or some such); and
3. Provide a " Skip to main content" link.

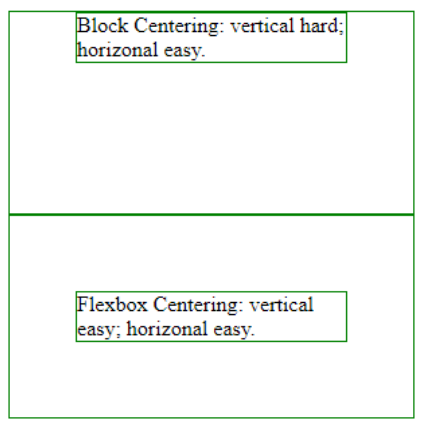
(W3C The Latest, "CSS Flexible Box Layout Module Level 1", https://www.w3.org/TR/css-flexbox-1/), "Reordering and Accessibility", https://www.w3.org/TR/css-flexbox-1/#order-accessibility

Centering

Flexbox can center in the vertical dimension (when flex-direction is row) while an ordinary block can not (so readily).

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>Flexbox - centering</title>

  <style>
    /* <![CDATA[ */
    /* General settings */
    div {
      border: 1px solid green;
    }
    div > div {
      width: 200px;
    }
    #blockContainer, #flexboxContainer {
      width: 300px;
      height: 150px;
    }

    /* Block centering */
    #blockContainer > div {
      /* Horizontal centering of a block */
      margin: auto;
    }
```

```
    /* Flexbox centering */
    #flexboxContainer {
      display: flex;
      flex-direction: row;
      flex-wrap: nowrap;

      /* Vertical centering (when flex-direction is row) */
      align-items: center;

      /* Horizontal centering (when flex-direction is row) */
      justify-content: center;
    }
    #flexboxContainer.div > div {
      flex: 1 1 0;
    }
    /* ]]> */
  </style>
</head>
<body>
  <h1>Flexbox - centering</h1>
  <div id="blockContainer">
    <div>Block Centering: vertical hard; horizonal easy.</div>
  </div>
  <div id="flexboxContainer">
    <div>Flexbox Centering: vertical easy; horizonal easy.</div>
  </div>
</body>
</html>
```



C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\flexbox-centering.xhtml
(W3C The Latest, "CSS Flexible Box Layout Module Level 1", https://www.w3.org/TR/css-flexbox-1/), " Absolutely-Positioned Flex Children", https://www.w3.org/TR/css-flexbox-1/#abspos-items

Right align a block within a container

Right align a block within a container.

```
<header>
  <h1>Tech Skills</h1>
  <p class='articleDates'>
    First Published: 2021-06-24 17:23. <br />
      Last Modified: 2021-06-24 17:24.
  </p>
</header>

/* Push the .articleDates block to the right of the article header */
article header {
  display: flex;
  flex-wrap: wrap;
  justify-content: flex-end;
}
```

```
article header h1 {
  flex: 1 1 100%;
}
article header .articleDates {
  flex: 0 1 max-content;
}
```

https://source.v2.softmake/tech-skills
C:\Users\John\Documents\Sda\Code\web\Apps\Softmake\V2\Source\200-themes\softmake-purple\themeSoftmakePurple.css

Flexbox a table

To layout tables use grid over flexbox. See Grid layout a table.

You can layout a table with flexbox. See https://atlas/web/Examples/CssExamples/css2017/flexbox-table-production.xhtml. But it is generally harder to line up the columns exactly when shrunk. E.g. You'll get the following effect ...

Location	Type	Name
Version	**Home page**	**Licence**
Front end	Markup standard	HTML Living Standard
"Living"	https://html.spec.whatwg.org/multipage/	Not applicable (I don't distribute the document), but https://html.spec.whatwg.org/#acknowledgments, https://creativecommons.org/licenses/by/4.0/

See https://atlas/web/Examples/CssExamples/css2017/flexbox-table-production.xhtml

CSS Transforms

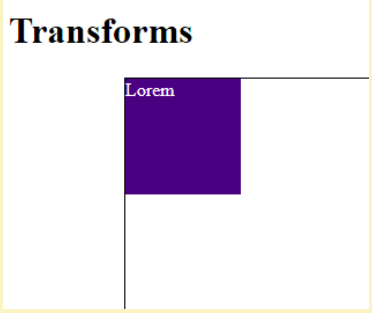
A css transform works on HTML and SVG elements.

> This specification is the convergence of the CSS 2D transforms and SVG transforms specifications.

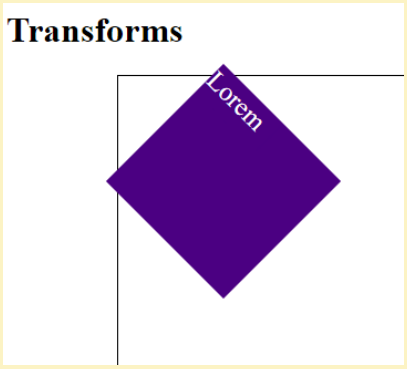
(W3C The latest, "CSS Transforms Module Level 1", https://www.w3.org/TR/css-transforms-1/)

A css transform can scale, translate (move), or rotate the target element.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>Transforms</title>

  <link rel="stylesheet" href="style/html5-basic.css" />
  <style>
    /* <![CDATA[ */
    /* A comment */
    main {
      border: 1px solid black;
      width: 400px;
      height: 400px;
      margin-left: 100px;
    }
    div {
      width: 100px;
      height: 100px;
      background-color: indigo;
      color: white;
    }

    div {
      transform: scale(1.5) translate(30px, 30px) rotate(45deg);
    }
    /* ]]> */
  </style>

</head>
<body>
  <h1>Transforms</h1>
  <main>
    <div>
        Lorem
    </div>
  </main>
</body>
</html>

// Before
```




```
// After
```



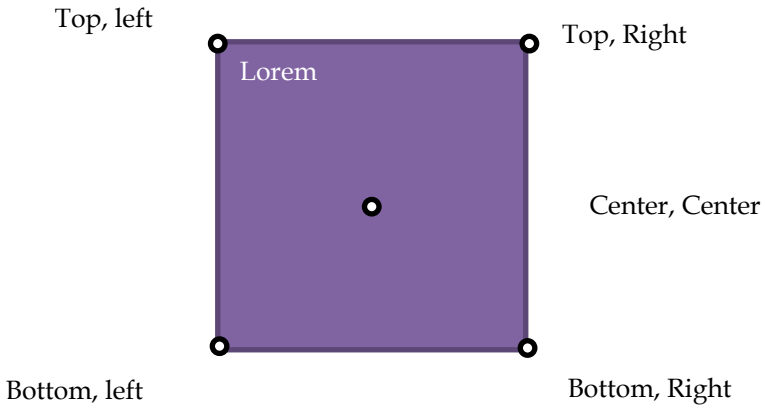
(W3C The latest, "CSS Transforms Module Level 1", https://www.w3.org/TR/css-transforms-1/), "The Transform Rendering Model", https://www.w3.org/TR/css-transforms-1/#transform-rendering
C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\transforms-html.xhtml

Transformations occur according to the "Transformation Matrix", an algorithm:

- Use a `'transform-origin'` property, if specified, to translate (move) the origin with the 'transform reference box', in html by default the element itself.
- Perform the transforms as specified by the `'transform'` property, from left to right.
- Undo the translation specified by `'transform-origin'`.

(W3C The latest, "CSS Transforms Module Level 1", https://www.w3.org/TR/css-transforms-1/), "The Transform Rendering Model", https://www.w3.org/TR/css-transforms-1/#transform-rendering

The origin is relative to the 'transform reference box', in html by default the element itself).



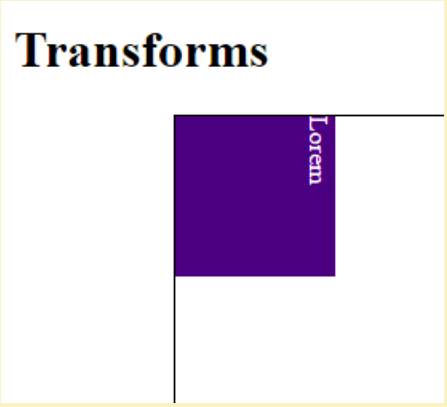
(W3C The latest, "CSS Transforms Module Level 1", https://www.w3.org/TR/css-transforms-1/), "The transform-origin Property", https://www.w3.org/TR/css-transforms-1/#transform-origin-property

The transform-origin is the point around which the transformation is applied.

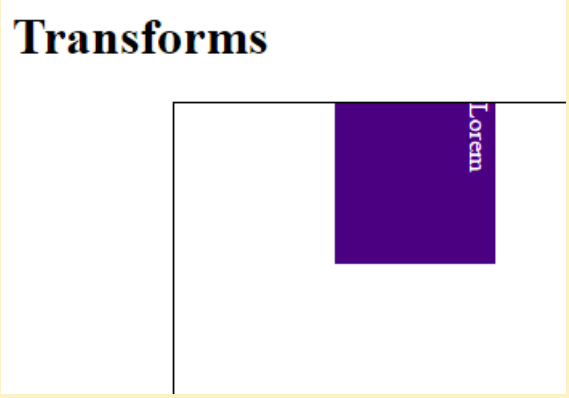
(Mozilla 2018, "MDN Web Docs: Transform-Origin", https://developer.mozilla.org/en-US/docs/Web/CSS/transform-origin), https://developer.mozilla.org/en-US/docs/Web/CSS/transform-origin

Example of the effects of `'transform-origin'`.

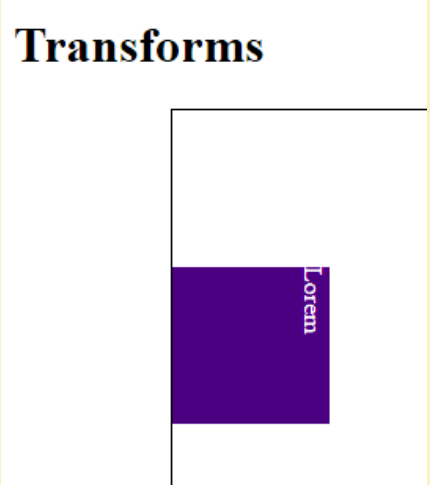
```
// Without 'transform-origin' set, defaults to 'center', 'center'
div {
  transform: rotate(90deg);
}
```




```
div {
  transform-origin: right bottom;
  transform: rotate(90deg);
}
```




```
div {
  transform-origin: left bottom;
  transform: rotate(90deg);
}
```



C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\transforms.xhtml

CSS Transitions

Basics

CSS transitions (W3C The latest, "CSS Transitions", https://www.w3.org/TR/css-transitions-1/). https://www.w3.org/TR/css-transitions-1/

CSS transitions enable CSS properties to change smoothly from one value to another over a given duration.

> *(W3C The latest, "CSS Transitions", https://www.w3.org/TR/css-transitions-1/), " Introduction", https://www.w3.org/TR/css-transitions-1/#introduction*

> Only animatable CSS properties can be transitioned.See https://www.w3.org/TR/css-transitions-1/#animatable-properties

> *(W3C The latest, "CSS Transitions", https://www.w3.org/TR/css-transitions-1/), "Transitions", https://www.w3.org/TR/css-transitions-1/#transitions*

CSS transition properties:

Property	Description
`transition-property`	Which properties will have a transition applied to it.
`transition-duration`	The length of time that a transition takes.
`transition-timing-function`	How the intermediate values used during a transition will be calculated. It allows for a transition to change speed over its duration. These effects are commonly called easing functions.
`transition-delay`	When the transition will start
`transition`	Shorthand.

> *(W3C The latest, "CSS Transitions", https://www.w3.org/TR/css-transitions-1/), "Transitions", https://www.w3.org/TR/css-transitions-1/#transitions*

Transistions start when the transition property is changed. The property can be changed through css or scripting. Either way the starting value of the property must be explicitly defined in css (we can't rely on initial values).

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>CSS Transitions</title>

  <style>
    /* <![CDATA[ */
    div {
      height: 2em;
      width: 50%;
      border: 1px solid black;
      background-color: black;
      color: yellow;
      position: relative;
    }
```

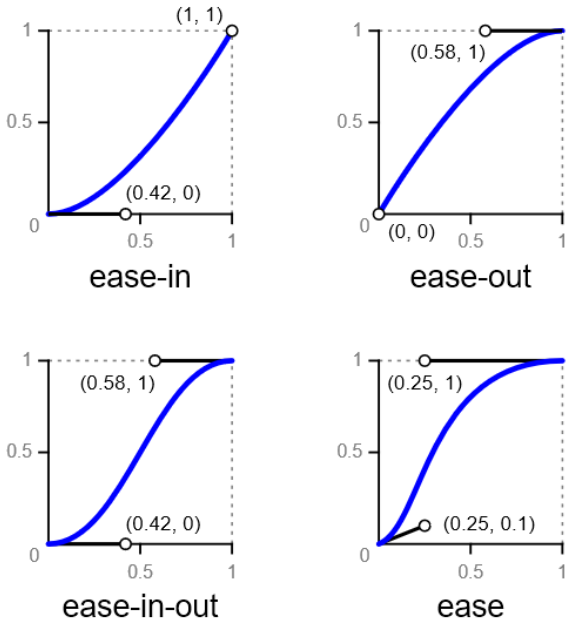
```
    div {
      transition-property: left;
      transition-duration: 2s;

      /* The start value must be explicitly defined */
      left: 0px;
    }

    /* Start the transition via css (and a hover event) */
    div:hover {
      left: 200px;
      /* Note that the transitions reverses back to the start value after hover */
    }


    /* ]]> */
  </style>

  <script>
    /* <![CDATA[ */

    function go() {
      var targetElement = document.getElementById("target");
      targetElement.style.left = "200px";
    }

    /* ]]> */
  </script>
</head>
<body>
  <h1>CSS Transitions</h1>
  <div id="target">Super div</div>
  <!-- Start the transition via scripting -->
  <button onclick="go();">Go</button>
</body>
</html>
```

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\css-transitions.xhtml
https://www.w3schools.com/css/css3_transitions.asp

Transition timing functions

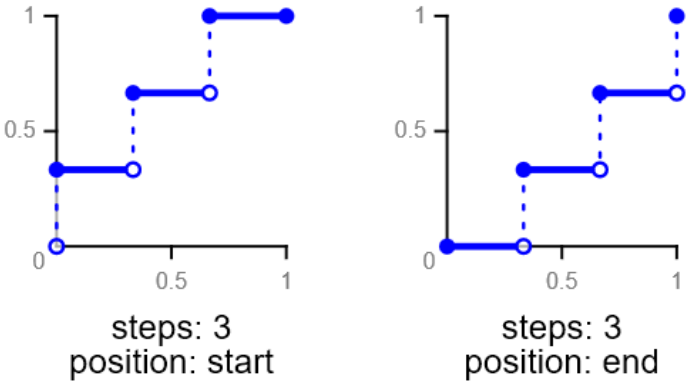
Transition timing functions include:

- Linear.
- Bezier timing functions:
    - ease (the default)
    - ease-in;
    - ease-out;
    - ease-in-out;
    - cubic-bezier(<number>, <number>, <number>, <number>): a custom Bezier function.



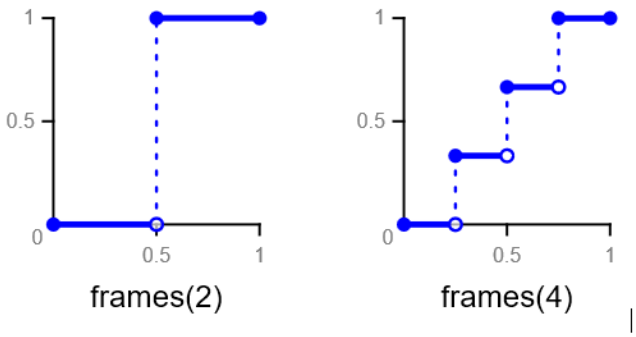
- Step timing functions.

```
steps(5, start);
steps(3, end);
```



- Frames timing function (not supported in firefox and chrome at 2018-09-06?).

```
frames(5);
```



(W3C n.d., "CSS Timing Functions Level 1", Accessed 2018-09-06. https://drafts.csswg.org/css-timing/), https://www.w3.org/TR/css-timing-1/

Timing function example.

```
div {
  transition-property: left;
  transition-duration: 2s;
  transition-timing-function: ease-in-out;

  /* The start value must be explicitly defined */
  left: 0px;
}
```

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\css-transitions.xhtml

Reversing transitions instanteously by setting the transitions duration to 0.

```
    div {
      height: 2em;
      width: 50%;
      margin-bottom: 1em;
      border: 1px solid black;
      background-color: black;
      color: yellow;
      position: relative;
    }

    div {
      transition-property: left;
      /* Set transition-duration with javascript */

      /* The start value must be explicitly defined */
      left: 0px;
    }

    #targetEase {
      transition-timing-function: ease;
    }

    #targetEaseIn {
      transition-timing-function: ease-in;
    }
...

    function go () {
      var elements = document.getElementsByTagName("div");
      for (var i = 0; i < elements.length; i++) {
        elements[i].style.transitionDuration = "1s";
        elements[i].style.left = "200px";
      }
    }

    function reset () {
      var elements = document.getElementsByTagName("div");
      for (var i = 0; i < elements.length; i++) {
        elements[i].style.transitionDuration = "0s";
```

```
        elements[i].style.left = "0px";
      }
    }
```

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\css-transitions-timingFunctionComparision.xhtml

Full example

Transition full example.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>CSS Transitions - Timing function comparison</title>

  <style>
    /* <![CDATA[ */
    div {
      height: 2em;
      width: 50%;
      margin-bottom: 1em;
      border: 1px solid black;
      background-color: black;
      color: yellow;
      position: relative;
    }

    div {
      transition-property: left;
      /* Set transition-duration with javascript */

      /* The start value must be explicitly defined */
      left: 0px;
    }

    #targetEase {
      transition-timing-function: ease;
    }

    #targetEaseIn {
      transition-timing-function: ease-in;
    }

    #targetEaseOut {
      transition-timing-function: ease-out;
    }

    #targetEaseInOut {
      transition-timing-function: ease-in-out;
    }

    #targetLinear {
      transition-timing-function: linear;
    }

    #targetStepStart {
      transition-timing-function: steps(5, start);
    }

    #targetStepEnd {
      transition-timing-function: steps(5, end);
    }

    #targetFrames {
      transition-timing-function: frames(5);
    }


    /* ]]> */
```

```
  </style>

  <script>
    /* <![CDATA[ */

    function go () {
      var elements = document.getElementsByTagName("div");
      for (var i = 0; i < elements.length; i++) {
        elements[i].style.transitionDuration = "1s";
        elements[i].style.left = "200px";
      }
    }

    function reset () {
      var elements = document.getElementsByTagName("div");
      for (var i = 0; i < elements.length; i++) {
        elements[i].style.transitionDuration = "0s";
        elements[i].style.left = "0px";
      }
    }

    /* ]]> */
  </script>
</head>
<body>
  <h1>CSS Transitions - Timing function comparison</h1>
  <div id="targetEase">Ease</div>
  <div id="targetEaseIn">Ease-in</div>
  <div id="targetEaseOut">Ease-out</div>
  <div id="targetEaseInOut">Ease-InOut</div>
  <div id="targetlinear">Linear</div>
  <div id="targetStepStart">Step Start</div>
  <div id="targetStepEnd">Step End</div>
  <div id="targetFrames">Frames (unsupported?)</div>

  <!-- Start the transition via scripting -->
  <button onclick="go();">Go</button>
  <button onclick="reset();" type="reset">Reset</button>
</body>
</html>
```

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\css-transitions-timingFunctionComparision.xhtml

CSS Animations

CSS animations (W3C The latest, "CSS Animations Level 1", https://www.w3.org/TR/css-animations-1/). https://www.w3.org/TR/css-animations-1/.

Values are specified using animation keyframes.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
  <meta charset="utf-8" />
  <title>CSS Animations</title>

  <style>
    /* <![CDATA[ */
    div {
      height: 2em;
      width: 50%;
      border: 1px solid black;
      background-color: black;
      color: yellow;
    }

    .animate {
      animation-name: slide-right;
      animation-duration: 2s;
      /* Preserve the effect of the animation at ending */
      animation-fill-mode: forwards;
    }

    @keyframes slide-right {

      from {
        margin-left: 0px;
      }

      50% {
        margin-left: 110px;
        opacity: 0.5;
      }

      to {
        margin-left: 200px;
        opacity: 0.2;
      }
    }

    /* ]]> */
  </style>

  <script>
    /* <![CDATA[ */

    function DoAnimation() {
      var targetElement = document.getElementById("target");
      targetElement.className = "animate";
    }

    function reset() {
      var targetElement = document.getElementById("target");
      targetElement.className = "none";
    }
    /* ]]> */
  </script>
</head>
<body>
  <h1>CSS Animations</h1>
  <div id="target">Super div</div>
  <button onclick="DoAnimation();">Go</button>
  <button onclick="reset();" type="reset">Reset</button>
</body>
</html>
```

C:\Users\John\Documents\Sda\Code\web\Examples\CssExamples\css2017\css-animations.xhtml
(W3C The latest, "CSS Animations Level 1", https://www.w3.org/TR/css-animations-1/), https://www.w3.org/TR/css-animations-1/

Css Techniques

Box Shadows

Material design box shadows.

```
    tr {
      /* border: 1px solid black; */
      background-color: white;
      border-radius: 8px;
      /* Material Design Box Shadows, A Pen By Samuel Thornton, Card 2,
https://codepen.io/sdthornton/pen/wBZdXq */
      box-shadow: 0 3px 6px rgba(0,0,0,0.16), 0 3px 6px rgba(0,0,0,0.23);
      padding: 0.5em;
      margin-bottom: 2em;
    }
```

*(Thorton 2019, *Material Design Box Shadows*, https://codepen.io/sdthornton/details/wBZdXq)*
https://codepen.io/sdthornton/pen/wBZdXq

References (Zotero)

|CSS Fonts 3| W3C. 2013. "CSS Fonts Module Level 3". https://www.w3.org/TR/css-fonts-3/. 2013-10-03.

Bentley, John. 2018. "CSS 2.1 Core - Quick Reference". \\Atlas\Users\John\Documents\Sda\Info\Web\KB\Css\QuickReference\CSS2.1Core-QuickReference.docx. 2018.

Coyier, Chris. 2011. "Multiple Attribute Values". CSS-Tricks. https://css-tricks.com/multiple-attribute-values/. 2011-12-19.

Coyier, Chris. 2018. "A Complete Guide to Flexbox". CSS-Tricks. https://css-tricks.com/snippets/css/a-guide-to-flexbox/. 2018-11-15.

Google. 2017. "Get Started with the Google Fonts API | Google Fonts". Google Developers. https://developers.google.com/fonts/docs/getting_started. 2017-06-02.

MDN. 2018. "Controlling Ratios of Flex Items Along the Main Axis". MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Controlling_Ratios_of_Flex_Items_Along_the_Main_Ax. 2018.

———. 2018. "Web Technology for Developers > CSS: Cascading Style Sheets > Display". MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/CSS/display. 2018-12.

Mozilla. 2018. "MDN Web Docs: Transform-Origin". MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/CSS/transform-origin. 2018.

———. n.d. "Basic Concepts of Grid Layout". MDN Web Docs. Accessed 2018-04-15. https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout/Basic_Concepts_of_Grid_Layout. n.d.

Thorton, Samuel. 2019. *_Material Design Box Shadows_*. https://codepen.io/sdthornton/details/wBZdXq. 2019.

W3C. 2015. "CSS Custom Properties for Cascading Variables Module Level 1". https://www.w3.org/TR/css-variables-1/. 2015-12-03.

———. 2016. "CSS Values and Units Module Level 3". https://www.w3.org/TR/css-values-3/. 2016-09-29.

———. 2017. "CSS Grid Layout Module Level 1". https://www.w3.org/TR/css-grid-1/. 2017-12-14.

———. 2019. "Techniques for WCAG 2.1". https://www.w3.org/WAI/WCAG21/Techniques/. 2019-01-11.

———. The latest. "Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification". https://www.w3.org/TR/CSS22/. The latest.

———. The latest. "CSS Animations Level 1". https://www.w3.org/TR/css-animations-1/. The latest.

———. The Latest. "CSS Box Alignment Module Level 3". https://www.w3.org/TR/css-align-3/#valdef-align-content-space-around. The Latest.

———. The Latest. "CSS Cascading and Inheritance Level 4". https://www.w3.org/TR/css-cascade-4/. The Latest.

———. The Latest. "CSS Display Module Level 3". https://www.w3.org/TR/css-display-3/. The Latest.

———. The Latest. "CSS Flexible Box Layout Module Level 1". https://www.w3.org/TR/css-flexbox-1/. The Latest.

———. The Latest. "CSS Intrinsic & Extrinsic Sizing Module Level 3". https://www.w3.org/TR/css-sizing-3/. The Latest.

———. The Latest. "CSS Media Queries". https://www.w3.org/TR/css3-mediaqueries/. The Latest.

———. The Latest. "CSS Snapshot - The Latest". https://www.w3.org/TR/CSS/#css. The Latest.

———. n.d. "CSS Timing Functions Level 1". Accessed 2018-09-06. https://drafts.csswg.org/css-timing/. n.d.

———. The latest. "CSS Transforms Module Level 1". https://www.w3.org/TR/css-transforms-1/. The latest.

———. The latest. "CSS Transitions". https://www.w3.org/TR/css-transitions-1/. The latest.

———. The Latest. "Selectors Level 3". https://www.w3.org/TR/selectors-3/. The Latest.

———. The latest. "Selectors Level 4". https://www.w3.org/TR/selectors-4/. The latest.

Document Licence