**9.4.3** **Relative positioning**

Once a box has been laid out according to the [normal flow](https://www.w3.org/TR/CSS21/visuren.html#normal-flow) or floated, it may be shifted relative to this position. This is called *relative positioning*. Offsetting a box (B1) in this way has no effect on the box (B2) that follows: B2 is given a position as if B1 were not offset and B2 is not re-positioned after B1's offset is applied. This implies that relative positioning may cause boxes to overlap. However, if relative positioning causes an 'overflow:auto' or 'overflow:scroll' box to have overflow, the UA must allow the user to access this content (at its offset position), which, through the creation of scrollbars, may affect layout.

A relatively positioned box keeps its normal flow size, including line breaks and the space originally reserved for it. The section on [containing blocks](https://www.w3.org/TR/CSS21/visuren.html#containing-block) explains when a relatively positioned box establishes a new containing block.

For relatively positioned elements, 'left' and 'right' move the box(es) horizontally, without changing their size. 'Left' moves the boxes to the right, and 'right' moves them to the left. Since boxes are not split or stretched as a result of 'left' or 'right', the used values are always: left = -right.

If both 'left' and 'right' are 'auto' (their initial values), the used values are '0' (i.e., the boxes stay in their original position).

If 'left' is 'auto', its used value is minus the value of 'right' (i.e., the boxes move to the left by the value of 'right').

If 'right' is specified as 'auto', its used value is minus the value of 'left'.

If neither 'left' nor 'right' is 'auto', the position is over-constrained, and one of them has to be ignored. If the 'direction' property of the containing block is 'ltr', the value of 'left' wins and 'right' becomes -'left'. If 'direction' of the containing block is 'rtl', 'right' wins and 'left' is ignored.

**9.5.1** **Positioning the float: the** [**'float'**](https://www.w3.org/TR/CSS21/visuren.html#propdef-float) **property**

**'float'**

|  |  |
| --- | --- |
| *Value:* | left | right | none | [inherit](https://www.w3.org/TR/CSS21/cascade.html#value-def-inherit) |
| *Initial:* | none |
| *Applies to:* | all, but see [9.7](https://www.w3.org/TR/CSS21/visuren.html#dis-pos-flo) |
| *Inherited:* | no |
| *Percentages:* | N/A |
| *Media:* | [visual](https://www.w3.org/TR/CSS21/media.html#visual-media-group) |
| *Computed value:* | as specified |

This property specifies whether a box should float to the left, right, or not at all. It may be set for any element, but only applies to elements that generate boxes that are not [absolutely positioned](https://www.w3.org/TR/CSS21/visuren.html#absolutely-positioned). The values of this property have the following meanings:

User agents may treat float as 'none' on the root element.

Here are the precise rules that govern the behavior of floats:

1. The left [outer edge](https://www.w3.org/TR/CSS21/box.html#outer-edge) of a left-floating box may not be to the left of the left edge of its [containing block](https://www.w3.org/TR/CSS21/visuren.html#containing-block). An analogous rule holds for right-floating elements.
2. If the current box is left-floating, and there are any left-floating boxes generated by elements earlier in the source document, then for each such earlier box, either the left [outer edge](https://www.w3.org/TR/CSS21/box.html#outer-edge) of the current box must be to the right of the right [outer edge](https://www.w3.org/TR/CSS21/box.html#outer-edge) of the earlier box, or its top must be lower than the bottom of the earlier box. Analogous rules hold for right-floating boxes.
3. The right [outer edge](https://www.w3.org/TR/CSS21/box.html#outer-edge) of a left-floating box may not be to the right of the left [outer edge](https://www.w3.org/TR/CSS21/box.html#outer-edge) of any right-floating box that is next to it. Analogous rules hold for right-floating elements.
4. A floating box's [outer top](https://www.w3.org/TR/CSS21/box.html#outer-edge) may not be higher than the top of its [containing block](https://www.w3.org/TR/CSS21/visuren.html#containing-block). When the float occurs between two collapsing margins, the float is positioned as if it had an otherwise empty [anonymous block parent](https://www.w3.org/TR/CSS21/visuren.html#anonymous-block-level) taking part in the flow. The position of such a parent is defined by [the rules](https://www.w3.org/TR/CSS21/box.html#collapsed-through) in the section on margin collapsing.
5. The [outer top](https://www.w3.org/TR/CSS21/box.html#outer-edge) of a floating box may not be higher than the outer top of any [block](https://www.w3.org/TR/CSS21/visuren.html#block-boxes) or [floated](https://www.w3.org/TR/CSS21/visuren.html#floats) box generated by an element earlier in the source document.
6. The [outer top](https://www.w3.org/TR/CSS21/box.html#outer-edge) of an element's floating box may not be higher than the top of any [line-box](https://www.w3.org/TR/CSS21/visuren.html" \l "line-box) containing a box generated by an element earlier in the source document.
7. A left-floating box that has another left-floating box to its left may not have its right outer edge to the right of its containing block's right edge. (Loosely: a left float may not stick out at the right edge, unless it is already as far to the left as possible.) An analogous rule holds for right-floating elements.
8. A floating box must be placed as high as possible.
9. A left-floating box must be put as far to the left as possible, a right-floating box as far to the right as possible. A higher position is preferred over one that is further to the left/right.

## 9.7 Relationships between 'display', 'position', and 'float'

The three properties that affect box generation and layout — ['display'](https://www.w3.org/TR/CSS21/visuren.html#propdef-display), ['position'](https://www.w3.org/TR/CSS21/visuren.html#propdef-position), and ['float'](https://www.w3.org/TR/CSS21/visuren.html#propdef-float) — interact as follows:

1. If ['display'](https://www.w3.org/TR/CSS21/visuren.html#propdef-display) has the value 'none', then ['position'](https://www.w3.org/TR/CSS21/visuren.html#propdef-position) and ['float'](https://www.w3.org/TR/CSS21/visuren.html#propdef-float) do not apply. In this case, the element generates no box.
2. Otherwise, if ['position'](https://www.w3.org/TR/CSS21/visuren.html#propdef-position) has the value 'absolute' or 'fixed', the box is absolutely positioned, the computed value of ['float'](https://www.w3.org/TR/CSS21/visuren.html#propdef-float) is 'none', and display is set according to the table below. The position of the box will be determined by the ['top'](https://www.w3.org/TR/CSS21/visuren.html#propdef-top), ['right'](https://www.w3.org/TR/CSS21/visuren.html#propdef-right), ['bottom'](https://www.w3.org/TR/CSS21/visuren.html#propdef-bottom) and ['left'](https://www.w3.org/TR/CSS21/visuren.html#propdef-left) properties and the box's containing block.
3. Otherwise, if 'float' has a value other than 'none', the box is floated and 'display' is set according to the table below.
4. Otherwise, if the element is the root element, 'display' is set according to the table below, except that it is undefined in CSS 2.1 whether a specified value of 'list-item' becomes a computed value of 'block' or 'list-item'.
5. Otherwise, the remaining ['display'](https://www.w3.org/TR/CSS21/visuren.html#propdef-display) property values apply as specified.

|  |  |
| --- | --- |
| **Specified value** | **Computed value** |
| inline-table | table |
| inline, table-row-group, table-column, table-column-group, table-header-group, table-footer-group, table-row, table-cell, table-caption, inline-block | block |
| others | same as specified |

**9.9.1** **Specifying the stack level: the** [**'z-index'**](https://www.w3.org/TR/CSS21/visuren.html#propdef-z-index) **property**

For a positioned box, the ['z-index'](https://www.w3.org/TR/CSS21/visuren.html#propdef-z-index) property specifies:

1. The stack level of the box in the current stacking context.
2. Whether the box establishes a stacking context.

Values have the following meanings:

**[<integer>](https://www.w3.org/TR/CSS21/syndata.html" \l "value-def-integer)**

This integer is the stack level of the generated box in the current stacking context. The box also establishes a new stacking context.

**auto**

The stack level of the generated box in the current stacking context is 0. The box does not establish a new stacking context unless it is the root element.

*In this section, the expression "in front of" means closer to the user as the user faces the screen.*

In CSS 2.1, each box has a position in three dimensions. In addition to their horizontal and vertical positions, boxes lie along a "z-axis" and are formatted one on top of the other. Z-axis positions are particularly relevant when boxes overlap visually. This section discusses how boxes may be positioned along the z-axis.

The order in which the rendering tree is painted onto the canvas is described in terms of stacking contexts. Stacking contexts can contain further stacking contexts. A stacking context is atomic from the point of view of its parent stacking context; boxes in other stacking contexts may not come between any of its boxes.

Each box belongs to one *stacking context*. Each positioned box in a given stacking context has an integer *stack level*, which is its position on the z-axis relative other stack levels within the same stacking context. Boxes with greater stack levels are always formatted in front of boxes with lower stack levels. Boxes may have negative stack levels. Boxes with the same stack level in a stacking context are stacked back-to-front according to document tree order.

The root element forms the root stacking context. Other stacking contexts are generated by any positioned element (including relatively positioned elements) having a computed value of 'z-index' other than 'auto'. Stacking contexts are not necessarily related to containing blocks. In future levels of CSS, other properties may introduce stacking contexts, for example '[opacity](http://www.w3.org/TR/css3-color/#transparency)' [[CSS3COLOR]](https://www.w3.org/TR/CSS21/refs.html#ref-CSS3COLOR).

Within each stacking context, the following layers are painted in back-to-front order:

1. the background and borders of the element forming the stacking context.
2. the child stacking contexts with negative stack levels (most negative first).
3. the in-flow, non-inline-level, non-positioned descendants.
4. the non-positioned floats.
5. the in-flow, inline-level, non-positioned descendants, including inline tables and inline blocks.
6. the child stacking contexts with stack level 0 and the positioned descendants with stack level 0.
7. the child stacking contexts with positive stack levels (least positive first).

Within each stacking context, positioned elements with stack level 0 (in layer 6), non-positioned floats (layer 4), inline blocks (layer 5), and inline tables (layer 5), are painted as if those elements themselves generated new stacking contexts, except that their positioned descendants and any would-be child stacking contexts take part in the current stacking context.

This painting order is applied recursively to each stacking context. This description of stacking context painting order constitutes an overview of the detailed normative definition in [Appendix E.](https://www.w3.org/TR/CSS21/zindex.html)

Bonus:

The **Element.getBoundingClientRect()** method returns the size of an element and its position relative to the viewport.

httpRequest:  
 Functie de url si functor; fie un XMLHttpRequest care pe onreadystatechange are hook care cand readyState e 4 daca status e 200 apeleaza functorul pe .responseText, altfel eroare. cu asta, deschid open cu get url si true si trimit un null.

JSOO:   
 D::ctr(...):B(...) -> constructor(...){super(...);...};  
 Subclase: n.p = Obj.create(o.p) + copiere metode + extinderi / extends  
 Object.getPrototypeOf(a)