

Understanding CSS Grids

Contents

- Understanding CSS Grids..... 3**
 - CSS Grid Tutorials..... 3
 - Creating a float-based, responsive grid..... 3
 - Common CSS Grid Properties..... 4
 - Float positioning..... 4
- Index.....6**

Understanding CSS Grids

CSS Grid Tutorials

Creating a float-based, responsive grid

Coding a float-based grid in CSS

Website grids can be easily coded with a simple structure of HTML containing blocks and CSS float and width values.

Every person who begins to design web pages should understand the basic components to float-based, responsive grids. In this [tutorial](#), you will learn how to write a float-based, responsive grid in HTML and CSS.

How to code a float-based, responsive grid

The key to writing a responsive, float-based grid is developing the 2 following structures of containing blocks: 1) initial grid context, and 2) CSS classes devoted to defining column widths within the former element.

Grid context

1. Write a wrapper <div> element with a class named "grid"


```
<div class="grid">
  <!-- 100% wide -->
</div>
```

Making column widths

2. In the HTML file, write a testable combination of columns within the containing block `.grid`.

The example below presents a common use case, creating a main content block with a sidebar to its right.

```
<!-- Example of a common use case -->
<div class="grid">
  <div class="col-2-3">
    Main Content
  </div>
  <div class="col-1-3">
    Sidebar
  </div>
</div>
```

3.  **Note:** The class names correspond to the desired output goal to divide the containing block context into columns.

In your CSS file, divide up your 100% grid context by writing classes to create your desired column widths.

```
.col-3-4 {
  width: 75%;
}
.col-2-3 {
  width: 66.66%;
}
.col-1-2 {
  width: 50%;
}
.col-1-3 {
  width: 33.33%;
}
```

```

}
.col-1-4 {
  width: 25%;
}

```

4.  **Note:** Normal flow of elements stacks these div blocks on each other.

In order for the columns to assemble in rows, write a regular expression selector to float all of the column classes.

```

[class*='col-'] {
  float: left;
}

```

Clearing the parent containing block context

5. In your CSS file, use the *clear* declaration inside an *:after* pseudo-element on *.grid* to clear *both* right and left sides of the block.

```

.grid:after {
  content: "";
  display: table;
  clear: both;
}

```

Common CSS Grid Properties

Float positioning

CSS float positioning scheme

Writing CSS involves learning about the different ways you can manipulate the 'box model' and its positioning schemes. One important writing and design method is working with the float positioning scheme.

According to the CSS Working Group, a float positioning scheme defines the behavior of a box. A box can be "floated" left, right, both, or none within its current position and relation to its parent containing block and other inline elements.

For a detailed review of float positioning scheme behavior, see the W3 specification: <http://www.w3.org/TR/CSS2/visuren.html#floats> [outbound link].

The 'clear' property: How to position boxes next to floats

The CSS 'clear' property helps web designers control block-level elements in relationship to 'floated' boxes.

According to the *W3 specification on CSS2*, the 'clear' property "indicates which sides of an element's box(es) may *not* be adjacent to an earlier floating box." In other words, the 'clear' property only applies to block-level elements -- not floated elements within itself or in other block formatting contexts.

'clear'	entry
<i>Value:</i>	none left right both inherit
<i>Initial:</i>	none
<i>Applies to:</i>	block-level elements
<i>Inherited:</i>	no
<i>Percentages:</i>	N/A
<i>Media:</i>	visual

'clear'	entry
Computed value:	as specified

CSS clear: both declaration

Enter short description.

Enter paragraph.

Clear-fix hack

Website grids can be easily coded with a simple structure of HTML containing blocks and CSS float and width values.

In your CSS file, use the *clear* declaration inside an *:after* pseudo-element on *.grid* to clear *both* right and left sides of the block.

```
.grid:after {  
  content: "";  
  display: table;  
  clear: both;  
}
```

Index

C

clear fix [5](#)
containing blocks [3](#)
CSS [3](#), [5](#)

F

float [3](#)

G

grids [3](#)

H

HTML [3](#), [5](#)