# Coding grids

# Contents

# Bootstrap 3 grid layouts

## Introduction to grid layouts in Bootstrap 3

Empty for now!

Empty for now!

## Coding a grid layout with Bootstrap 3

# Flexbox grid layouts

## Introduction to grid layouts with flexbox

Empty for now!

Empty for now!

## Coding a grid layout with flexbox

# Float-based grid layouts

## Introduction to the 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 grd in HTML and CSS.

## What is 'float' positioning?

### CSS float positioning scheme

Learn about float positioning schemes in CSS.

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 more information about 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 elients 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** |
|---|---|
| *Value:* | none \| left \| right \| both \| inherit |
| *Initial:* | none |
| *Applies to:* | block-level elements |
| *Inherited:* | no |
| *Percentages:* | N/A |
| *Media:* | visual |
| *Computed value:* | as specified |

### CSS clear: both declaration

Enter short description.

Enter paragraph.

### Coding a clear-fix hack

In your CSS file, use the `clear` declaration inside an `:after` pseudo-element on a new `.cf` class to clear `both` right and left sides of the parent containing block.

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

## Coding 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  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 a new `.cf` class to clear `both` right and left sides of the parent containing block.

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

6. Apply the `.cf` clear-fix class to the parent containing block: `.grid`

```
<div class="grid cf">
  ...
</div>
```

# Pure.css grid layouts

## Introduction to grid layouts with Pure.css

Empty for now!

Empty for now!

## Coding a grid layout with Pure.css

# Index