Mastering CSS Box Shadows & Background Images

Two essential CSS properties that elevate your web designs from basic to beautiful. Box shadows add depth and dimension, while background images create visual interest and atmosphere.

Box Shadow

Create depth with customizable shadows that make elements pop off the page.

Box Shadow Shorthand Syntax:
box-shadow: [horizontal offset] [vertical offset] [blur radius] [spread radius] [color];

```
.card {
  box-shadow: 2px 4px 8px 1px rgba(0, 0, 0, 0.2);
  /* horizontal | vertical | blur | spread | color */
}
```

Background Image

Transform layouts with images that scale, repeat, and position perfectly.

Background Size Options:

- cover: Scales image to cover entire container (may crop)
- contain: Scales image to fit inside container (no cropping)
- auto: Uses image's original size
- [width] [height]: Custom dimensions (e.g., 100px 200px, 50% 100%)

```
.hero {
  background-image: url('image.jpg');
  background-size: cover; /* or contain, auto, 100% 50% */
}
```

Position Property

The **position** CSS property sets how an element is positioned in a document. The top, right, bottom, and left properties determine the final location of positioned elements.

Position - Static

This is the default positioning. Elements are positioned according to the normal document flow. The top, right, bottom, left, and z-index properties have no effect.

Position - Absolute

The element is removed from the normal document flow, and no space is created for the element in the page layout. It is positioned relative to its closest positioned ancestor, if any; otherwise, it is placed relative to the initial containing block.

Position - Relative

The offset is relative to itself based on the values of top, right, bottom, and left.

Position - Fixed

The element is removed from the normal document flow, and no space is created for the element in the page layout. It is positioned relative to the initial containing block established by the viewport.

Flexbox

Flexbox is one dimensional layout method helps you to arrange items in row or column.

display: flex

- Activates flexbox on a container
- Direct children become flex items

```
.container {
  display: flex;
}
```

flex-direction

- Defines the main axis direction
- Values: row, row-reverse, column, column-reverse

```
.container {
  display: flex;
  flex-direction: row; /* default */
}
```

justify-content

- Tells how space should be distributed between and around content items strictly along main axis.
- Values: flex-start, flex-end, center, space-between, space-around, space-evenly

```
.container {
  display: flex;
  justify-content: space-between;
}
```

align-items

- Aligns items along the cross axis
- Values: flex-start, flex-end, center, baseline

```
.container {
  display: flex;
  align-items: center;
}
```

flex-wrap

- Controls whether items are forced onto one line or can wrap onto multiple lines
- Values: nowrap, wrap, wrap-reverse

```
.container {
  display: flex;
  flex-wrap: wrap;
}
```

flex (shorthand for items)

- Controls how flex items grow and shrink
- Shorthand for flex-grow, flex-shrink, flex-basis

```
.item {
  flex: 1; /* grows to fill space */
}
```

The Flex Model

Understanding the flex model is crucial to mastering flexbox layouts. The flex model consists of two axes: the main axis and the cross axis. The direction of these axes is determined by the flex-direction property.

Main Axis

- The primary axis along which flex items are laid out
- Direction set by flex-direction property
- justify-content aligns items along this axis

```
.container {
  display: flex;
  flex-direction: row; /* main axis = horizontal */
}
```

Cross Axis

- The axis perpendicular to the main axis
- If main axis is horizontal, cross axis is vertical
- align-items aligns items along this axis

```
.container {
  display: flex;
  align-items: center; /* aligns on cross axis */
}
```

Flex Direction Impact

- row/row-reverse: main axis is horizontal
- column/column-reverse: main axis is vertical
- Changing direction swaps which axis is main/cross

```
.container {
  display: flex;
  flex-direction: column; /* main axis = vertical */
}
```