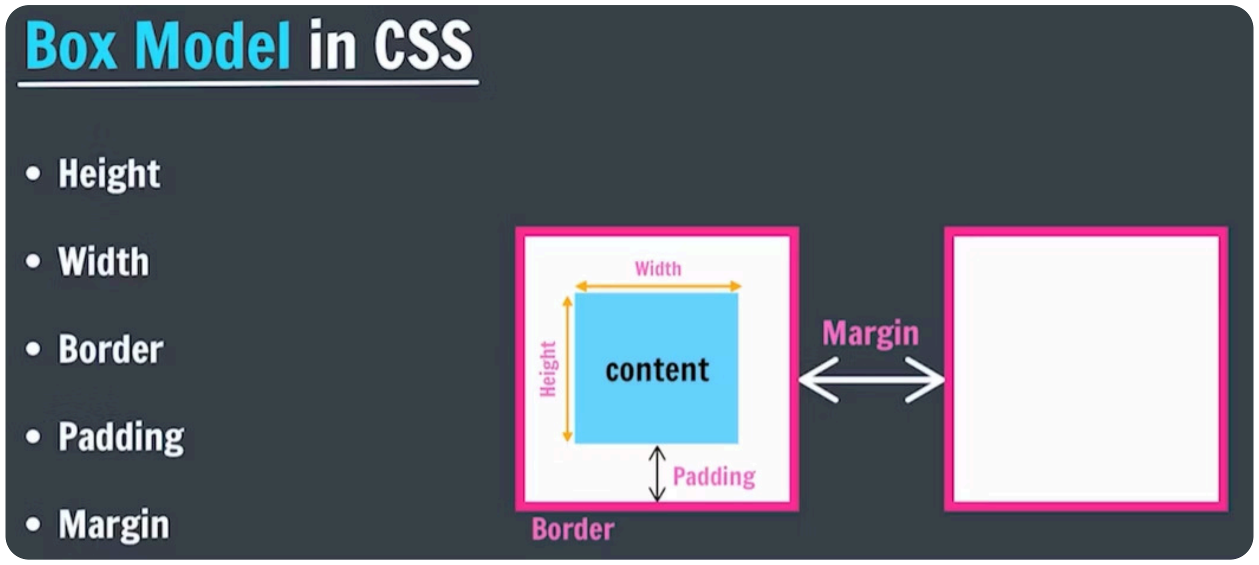
Understanding the CSS Box Model

Every HTML element rendered in the browser is interpreted as a rectangular box—this is the fundamental concept behind CSS layout. The box model defines how elements calculate their total space, incorporating content, padding, borders, and margins. Understanding this model is essential for precise layout control and troubleshooting spacing issues.



Height Property

By default, the height property sets the content area height of an element. This controls the vertical space available for the element's content.

```
div {
  height: 100px;
}
```

Width Property

By default, the width property sets the content area width of an element. This controls the horizontal space available for the element's content.

```
div {
 width: 300px;
}
```

BOX MODEL PROPERTIES

Styling Element Borders

The `border` property creates a visible outline around an element's content and padding. Borders add visual separation and definition, helping distinguish elements and create visual hierarchy in your layouts. You have complete control over border appearance through three key properties.

border-width

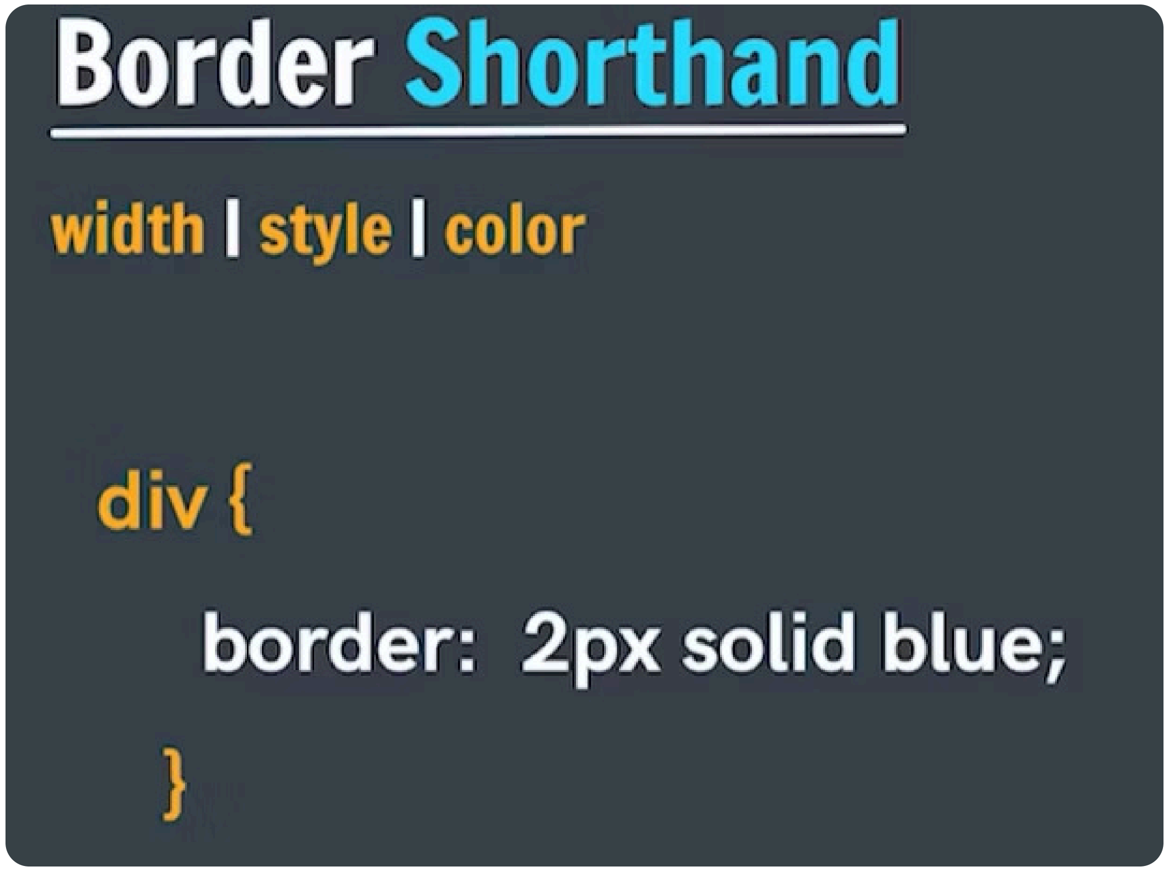
Controls the thickness of the border. Accepts values in pixels, ems, or keywords like `thin`, `medium`, and `thick`.

border-style

Defines the line pattern. Common values include `solid`, `dashed`, `dotted`, `double`, and `none`.

border-color

Sets the border color using hex codes, RGB values, or named colors. Can create striking visual effects.


Border Shorthand
width | style | color

```
div {
    border:  2px solid blue;
}
```

Controlling Individual Sides

You can target specific sides of an element's border using `border-left`, `border-right`, `border-top`, and `border-bottom`. This granular control allows you to create asymmetric designs and unique visual effects.

ADVANCED STYLING

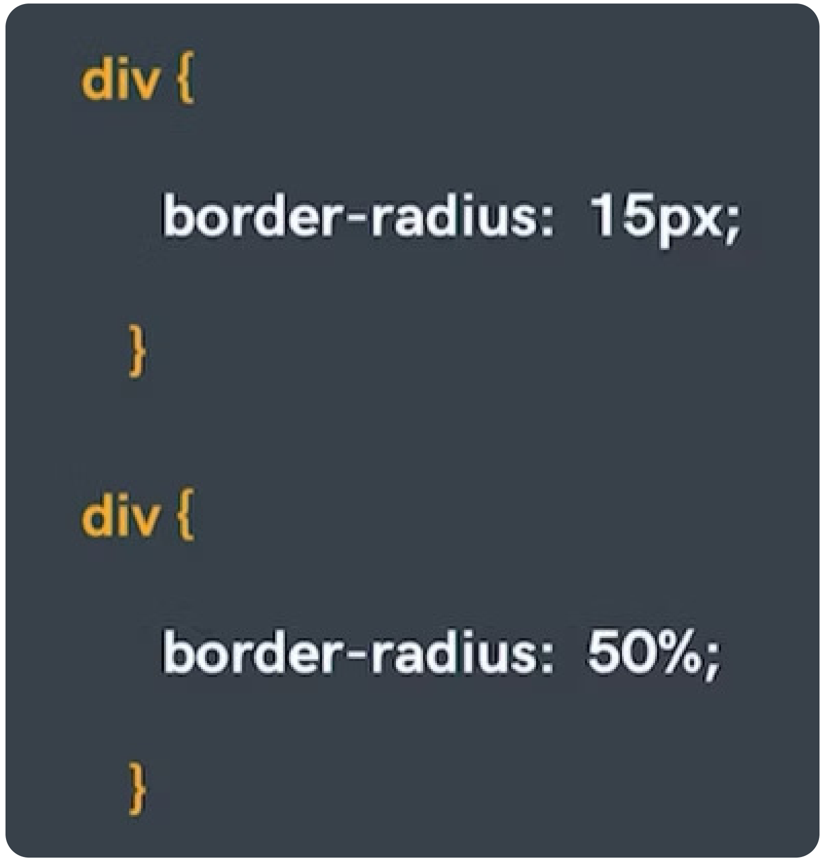
Creating Rounded Corners with Border Radius

The `border-radius` property transforms sharp rectangular boxes into softer, rounded shapes by curving the outer border edges. This simple property has become a cornerstone of modern web design, enabling everything from subtle softening of interface elements to perfectly circular avatars and buttons.

Border radius values can be specified in pixels or percentages. Using `50%` on a square element creates a perfect circle, while smaller values produce gentle curves. You can even specify different radius values for each corner to create unique organic shapes.

```
div {
  border-radius: 10px;
}

.circle {
  border-radius: 50%;
}
```



```
div {
    border-radius: 15px;
  }

div {
    border-radius: 50%;
  }
```

> Border radius revolutionized web aesthetics, replacing the harsh geometric web of the early 2000s with the softer, more approachable interfaces we expect today.

SPACING CONTROL

Understanding Padding

Padding creates internal spacing between an element's content and its border. This transparent space ensures content doesn't crowd against the edges, improving readability and visual breathing room. Padding is crucial for creating comfortable, well-proportioned layouts that feel professional and polished.

padding-left

Adds space on the left side between content and border

padding-right

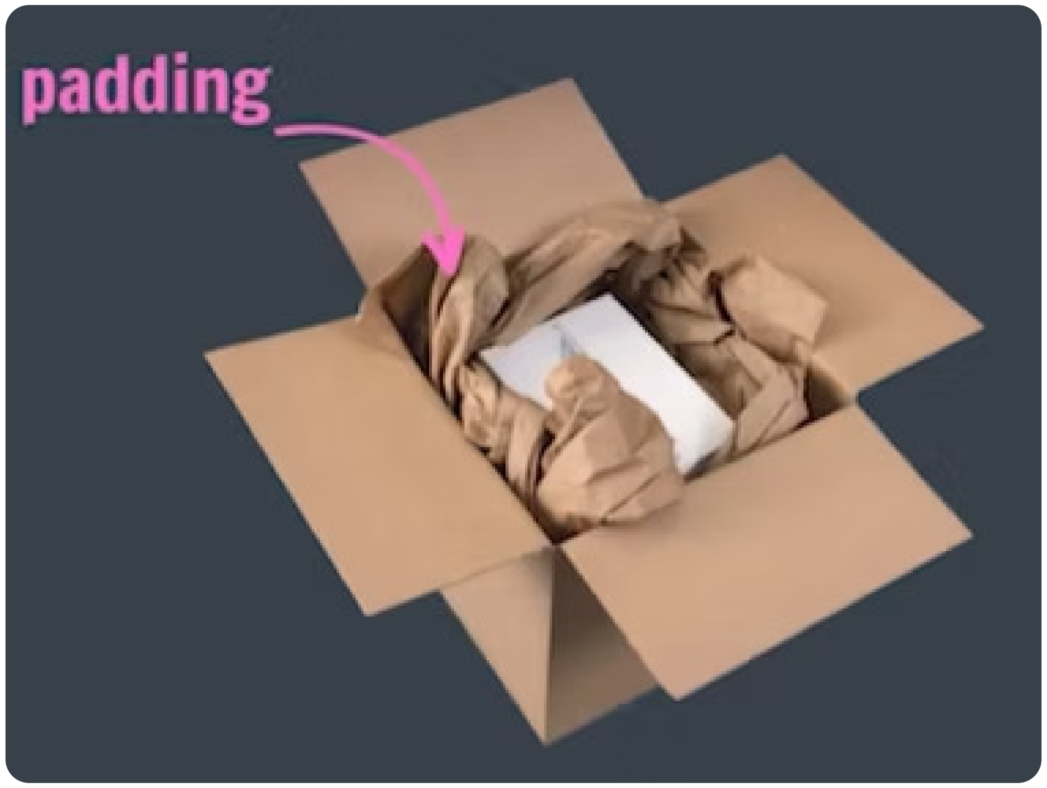
Adds space on the right side between content and border

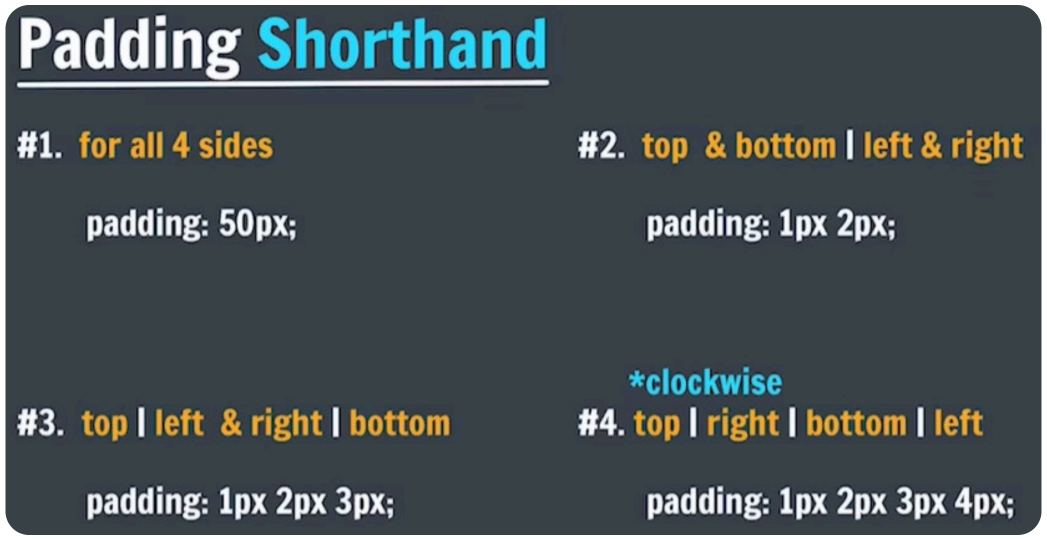
padding-top

Adds space at the top between content and border

padding-bottom

Adds space at the bottom between content and border





Padding Shorthand

#1. for all 4 sides

padding: 50px;

#2. top & bottom | left & right

padding: 1px 2px;

#3. top | left & right | bottom

padding: 1px 2px 3px;

*clockwise

#4. top | right | bottom | left

padding: 1px 2px 3px 4px;

Shorthand Syntax: You can use `padding: 10px 20px 15px 5px;` to set all four sides at once (top, right, bottom, left). Or use `padding: 10px 20px;` for vertical and horizontal values.

EXTERNAL SPACING

Controlling Layout with Margin

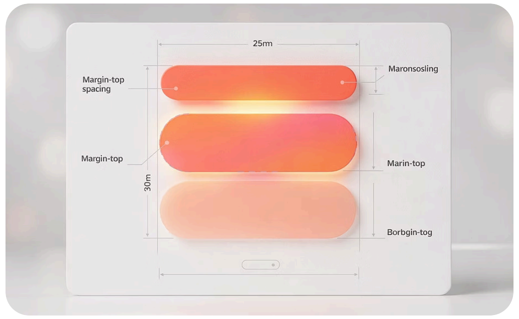
While padding creates internal spacing, margin controls the **external space** between elements—the transparent buffer zone that separates one element from its neighbors. Margins are essential for controlling layout flow, creating visual groupings, and preventing elements from colliding with each other.

margin-left

Creates space to the left of an element, pushing it away from adjacent elements or the container edge.

margin-right

Creates space to the right of an element, controlling the distance to elements that follow.



margin-top

Adds vertical space above an element, separating it from elements positioned above.

margin-bottom

Adds vertical space below an element, creating separation from subsequent elements.

Key Difference: Padding is *inside* the border and affects background colors, while margin is *outside* the border and is always transparent. Margins can also collapse vertically, while padding never does.

LAYOUT FUNDAMENTALS

The Display Property: Block vs Inline

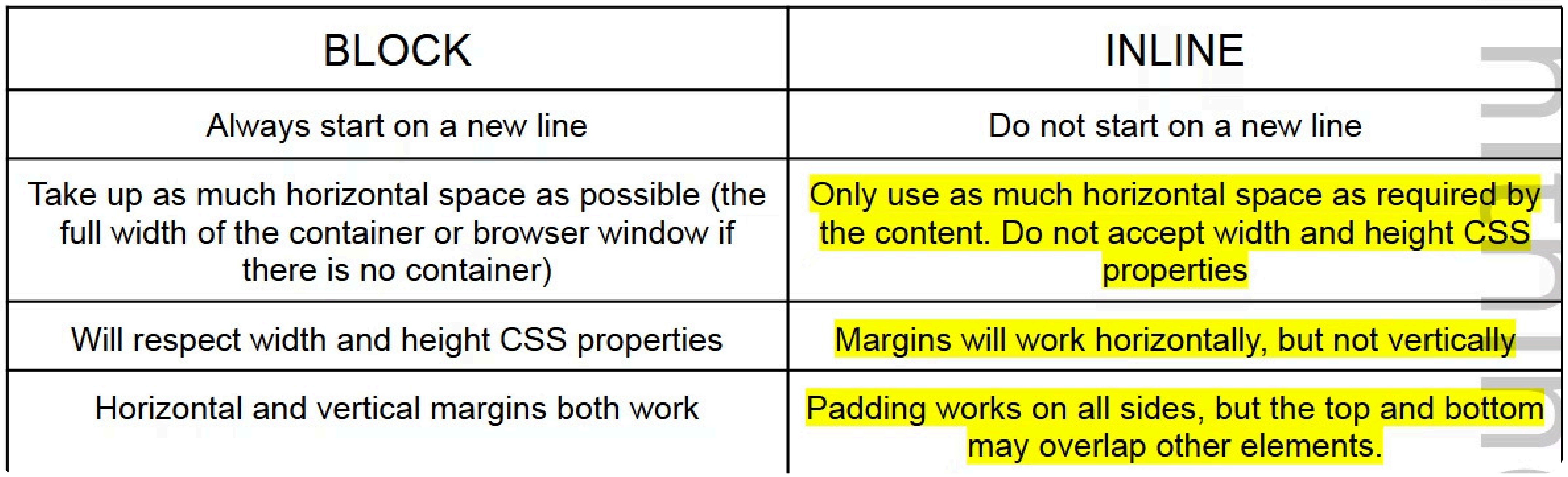
The display property is one of the most fundamental CSS properties, determining how an element behaves in the document flow. It controls whether an element is treated as a block-level or inline element, which dramatically affects how it interacts with surrounding content and how much space it occupies.

display: block

Block-level elements start on a new line and stretch to fill the full width of their container (unless a width is specified). They stack vertically and respect all box model properties including width, height, margins, and padding. Common block elements include <div>, <p>, <h1>-<h6>, and <section>.

display: inline

Inline elements flow within text content and only take up as much horizontal space as their content requires. They do not start on a new line and sit alongside other inline elements. Width and height properties have no effect, and vertical margins are ignored. Common inline elements include <span>, <a>, <strong> etc



BLOCK	INLINE
Always start on a new line	Do not start on a new line
Take up as much horizontal space as possible (the full width of the container or browser window if there is no container)	Only use as much horizontal space as required by the content. Do not accept width and height CSS properties
Will respect width and height CSS properties	Margins will work horizontally, but not vertically
Horizontal and vertical margins both work	Padding works on all sides, but the top and bottom may overlap other elements.

Other Display Values

Beyond block and inline, CSS offers display: inline-block (combines features of both), display: flex (for flexible layouts), display: grid (for two-dimensional layouts), and display: none (completely removes the element from the document flow).

DISPLAY VALUES

Display: Inline-Block & None

While block and inline are the foundational display values, CSS provides additional options that offer more flexibility. Two particularly useful values are inline-block, which combines the best features of both block and inline, and none, which completely removes elements from the document flow.

display: inline-block

Inline-block elements flow inline like inline elements but respect width, height, and all margin/padding properties like block elements. This hybrid behavior makes them perfect for creating navigation menus, button groups, and grid-like layouts without using floats or flexbox. Elements sit side-by-side but can be sized precisely.

```
.button {
  display: inline-block;
  width: 150px;
  padding: 10px 20px;
  margin: 5px;
}
```

display: none

Setting display to none completely removes the element from the document flow as if it never existed. The element takes up no space, and surrounding elements collapse to fill the gap. This differs from visibility: hidden, which hides the element but preserves its space. Commonly used for toggling content visibility with JavaScript.

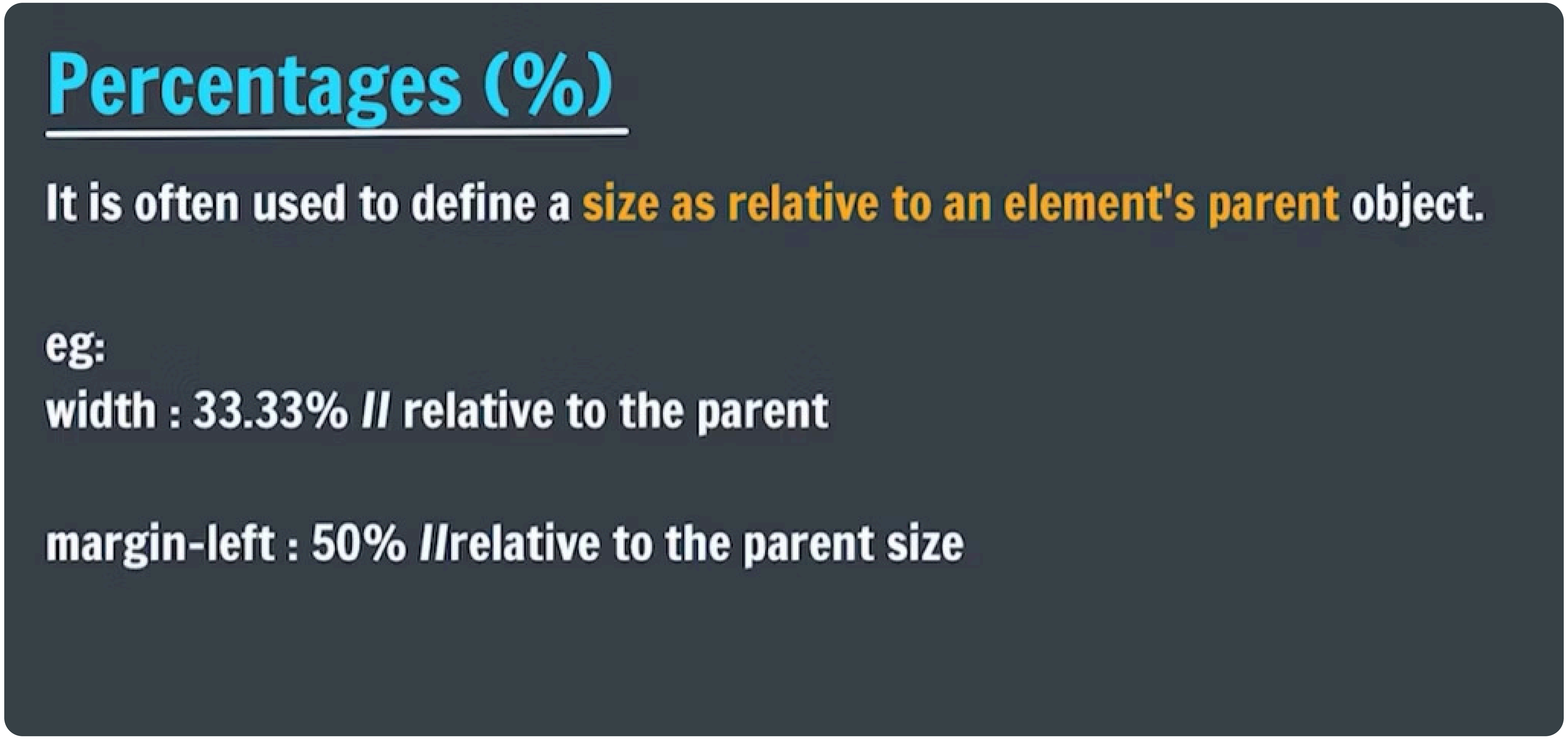
```
.hidden {
  display: none;
}

.modal.closed {
  display: none;
}
```

Inline-Block Gotchas

Inline-block elements are affected by whitespace in your HTML—spaces and line breaks between elements create small gaps. This can be fixed by removing whitespace in markup, using font-size: 0 on the parent, or using modern layout methods like flexbox.

Percentage unit



Percentages (%)

It is often used to define a size as relative to an element's parent object.

eg:
width : 33.33% // relative to the parent

margin-left : 50% //relative to the parent size