LECTURE 8

CSS3 Features

Rounded corners, gradients, shadows, transforms, and animation — visual polish with pure CSS.

CSC336 Web Technologies

OVERVIEW

In This Lecture

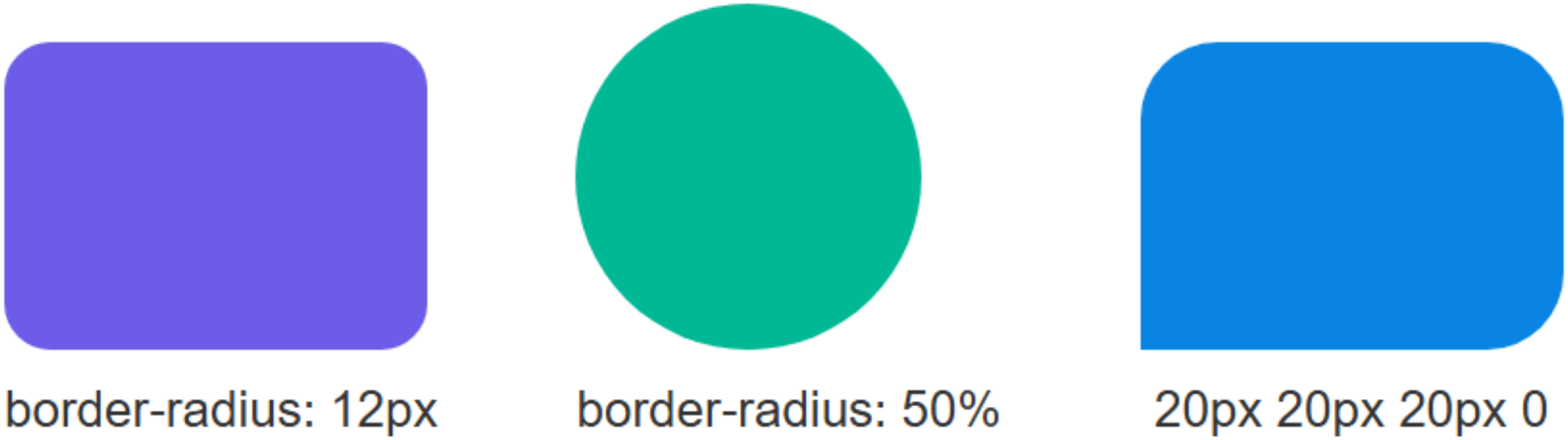
1. Rounded corners, gradients, shadows, and opacity
2. 2D/3D transforms, transitions, and keyframe animations
3. Web fonts, icon fonts, and CSS custom properties (variables)
4. Media queries, feature queries, and cross-browser compatibility
5. Vendor prefixes, and why compatibility still matters

border-radius

```
.card { border-radius: 12px; }
.avatar { border-radius: 50%; }  /* a perfect circle */
.bubble { border-radius: 20px 20px 20px 0; }
```

A pixel value rounds by a fixed amount; a percentage rounds relative to the element's own size.

border-radius, Rendered



Linear and Radial Gradients

```
.banner    { background: linear-gradient(to right, #4facfe, #00f2fe); }
.spotlight { background: radial-gradient(circle, #fff, #333); }
.diagonal  { background: linear-gradient(45deg, orange, red); }
```

Gradients, Rendered

Multiple Color Stops

```
.rainbow {
  background: linear-gradient(
    to right,
    red 0%, yellow 25%, green 50%, blue 75%, violet 100%
  );
}
```

box-shadow on Hover — Live

Genuinely live on the deployed page — shown here at both states.

```
.card {
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
}
.card:hover {
  box-shadow: 0 8px 20px rgba(0, 0, 0, 0.35);
}
```

text-shadow

```
h1 {
  text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
}
```

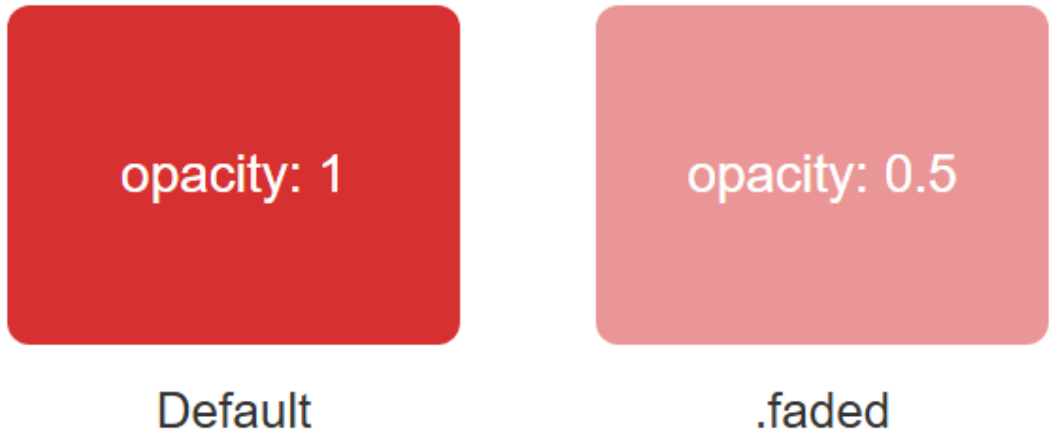
Multiple Shadows

```
box-shadow: 0 1px 2px #000, 0 0 20px gold;
/* stack shadows by separating them with commas */
```

opacity

opacity fades EVERYTHING inside, including text; use rgba() for a transparent background with solid text.

```
.faded {
  opacity: 0.5; /* 50% transparent */
}
```

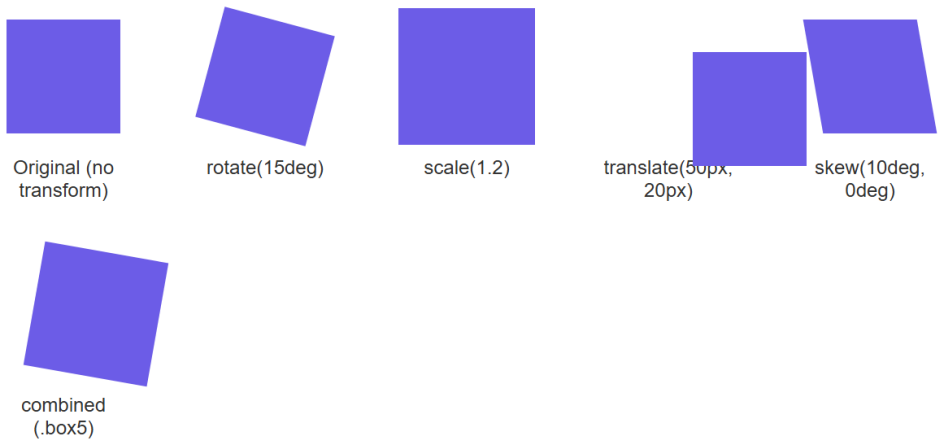


2D and 3D Transform Functions

Function	What It Does
translate(x, y) / translateZ(z)	Moves an element
rotate(deg) / rotateX/Y/Z(deg)	Spins an element
scale(n) / scaleX/Y/Z(n)	Resizes an element
skew(deg)	Slants an element

2D Transforms, Compared

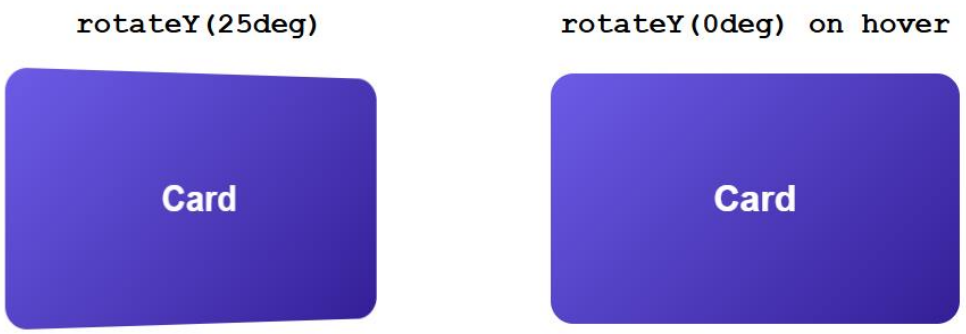
```
.box  { transform: rotate(15deg); }
.box2 { transform: scale(1.2); }
.box3 { transform: translate(50px, 20px); }
.box4 { transform: skew(10deg, 0deg); }
.box5 { transform: translate(20px, 0) rotate(10deg) scale(1.1); }
```



A 3D Hover Flip — Live

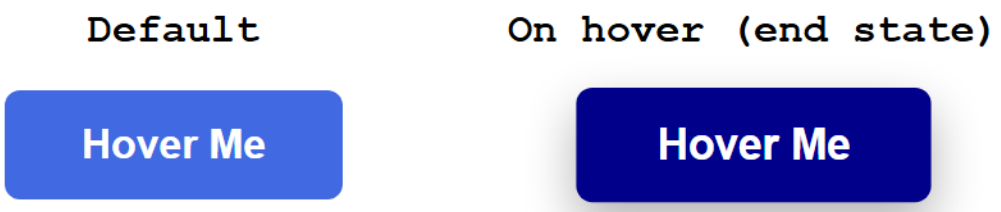
```
.scene { perspective: 800px; }

.card-3d {
  transform: rotateY(25deg);
  transition: transform 0.4s ease;
}
.card-3d:hover { transform: rotateY(0deg); }
```



A Smooth Hover Transition — Live

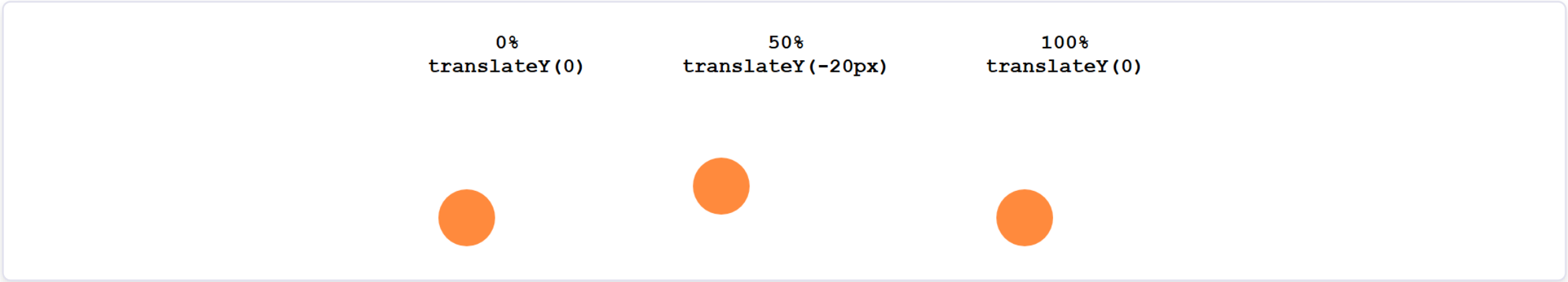
```
button {
  background-color: royalblue;
  transition: background-color 0.3s ease, transform 0.3s ease;
}
button:hover {
  background-color: darkblue;
  transform: scale(1.05);
}
```



@keyframes — Live and Looping

```
@keyframes bounce {
  0%   { transform: translateY(0); }
  50%  { transform: translateY(-20px); }
  100% { transform: translateY(0); }
}

.ball { animation: bounce 1s ease-in-out infinite; }
```



Timing Functions

Value	Feel
linear	Constant speed, start to finish — feels mechanical
ease (default)	Starts slow, speeds up, ends slow — the most natural-feeling default
ease-in / ease-out	Slow start only / slow finish only
cubic-bezier(...)	A fully custom easing curve for a specific feel

transition vs. animation

TIP

Use a transition for simple state changes (hover, focus, a toggled class). Use a keyframe animation for anything that plays on its own, loops, or needs more than a start and end state.

Web Fonts and Icon Fonts

- Web fonts (Google Fonts, or @font-face) let every visitor see the exact same typeface
- Always end a font-family list with a generic fallback: sans-serif, serif, monospace
- Icon fonts let you use icons like text — resizable, colorable — though many projects now prefer inline SVG

CSS Variables

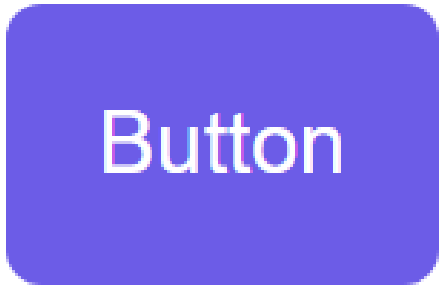
```
:root {
  --main-color: #6c5ce7;
  --spacing-unit: 8px;
}

.button {
  background-color: var(--main-color);
  padding: calc(var(--spacing-unit) * 2);
}
```

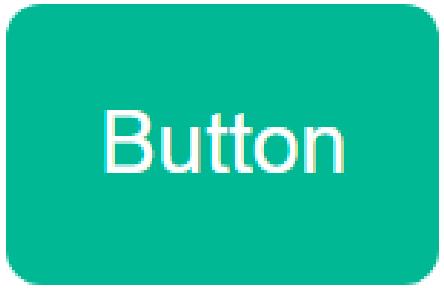
A live, real value in the browser — change it once in :root and every var(--main-color) usage updates instantly.

One Rule, Two Colors

Nothing changed except the value of --main-color in scope — proof the variable controls the rendered color.



--main-color: #6c5ce7 (default)



--main-color overridden to #00b894 -- same .button rule, new rendered color

Media and Feature Queries

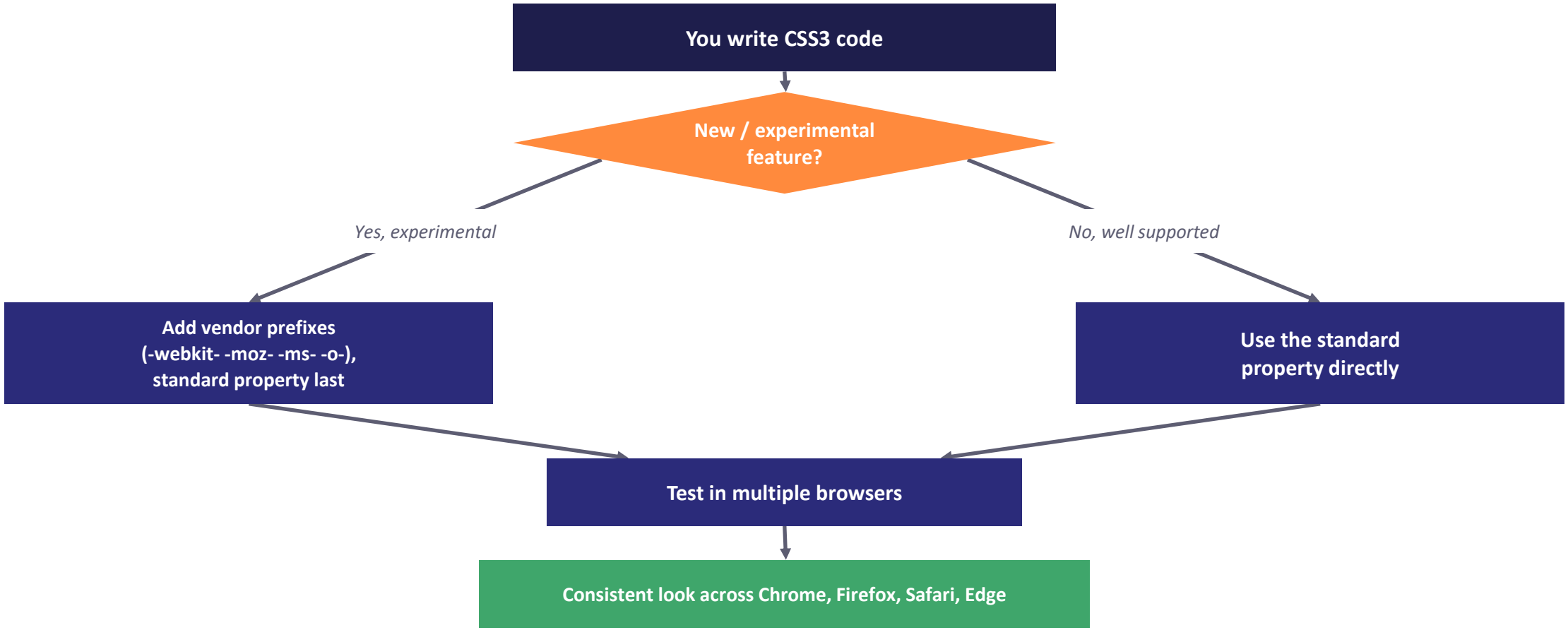
- @media (min-width: 768px) { ... } adapts styles to screen size — the foundation of responsive design
- @supports (display: grid) { ... } applies CSS only if the browser actually supports a feature
- Both check something real about the current browser before applying their block

Vendor Prefixes

Prefix	Browser Engine
-webkit-	Chrome, Safari, newer Edge
-moz-	Firefox
-ms-	old Internet Explorer / Edge
-o-	old Opera

Always place the unprefixed, standard property LAST — Autoprefixer automates this in real projects.

The Vendor-Prefix Workflow



KEY TAKEAWAYS

Lecture 8 in Six Points

1. border-radius, gradients, box-shadow/text-shadow, and opacity build modern visual effects with no images.
2. transform repositions/rotates/resizes without disturbing layout; 3D transforms need perspective on a parent.
3. transition animates between two states; @keyframes + animation define multi-step, looping animations.
4. CSS custom properties (--name, read with var(--name)) power theming and dark mode from one place.
5. Media queries adapt to screen size; feature queries (@supports) adapt to what the browser can render.
6. Vendor prefixes exist for experimental features — always place the unprefixed property last.