Vendor Prefixes in CSS -

Definition

Vendor prefixes are special keywords added to CSS properties or values to ensure **compatibility with specific browsers** during the time a feature is experimental or not yet fully standardized.

They allow developers to **use new CSS features before they are officially supported across all browsers**.

Common Vendor Prefixes

Prefix	Browser Vendor	Example
-webkit-	Google Chrome, Safari, newer Opera	-webkit-transition: all 0.5s;
-moz-	Mozilla Firefox	-moz-transition: all 0.5s;
-o-	Old Opera (Presto engine)	-o-transition: all 0.5s;
-ms-	Microsoft Internet Explorer & Edge (legacy)	-ms-transform: rotate(45deg);

Syntax

```
selector {
   -webkit-property: value; /* Chrome, Safari */
   -moz-property: value;    /* Firefox */
   -o-property: value;      /* Old Opera */
   -ms-property: value;     /* IE / old Edge */
   property: value;         /* Standard property */
}
```

Always put the **standard property last** so it overrides if supported.

Example – Using Vendor Prefixes

```
.box {
   -webkit-border-radius: 10px; /* Chrome, Safari */
   -moz-border-radius: 10px;    /* Firefox */
   border-radius: 10px;         /* Standard */
}
```

When to Use Vendor Prefixes

- For **new or experimental CSS properties** (e.g., backdrop-filter, clip-path in early days).
- When supporting **older browsers** that required prefixes.
- Until the feature is fully standardized and widely supported.

Modern Examples

1. **Flexbox (Old Syntax Support)**

```
.container {
   display: -webkit-flex; /* Safari older versions */
   display: flex;
}
```

1. **Gradient**

```
background: -webkit-linear-gradient(left, red, blue);
background: -moz-linear-gradient(left, red, blue);
background: linear-gradient(to right, red, blue);
```

Advantages

- Enables use of cutting-edge CSS features earlier.
- Improves cross-browser compatibility.

Limitations

- Increases CSS file size and maintenance complexity.
- Often unnecessary for modern browsers as features become standardized.
- Can cause bugs if not updated when the standard syntax changes.

Tips

- Use **Autoprefixer** (a PostCSS tool) to automatically add/remove prefixes based on browser support:

npm install autoprefixer postcss-cli

- Check browser support using Can I Use before adding prefixes.
- Always keep the **standard property** as the last fallback.