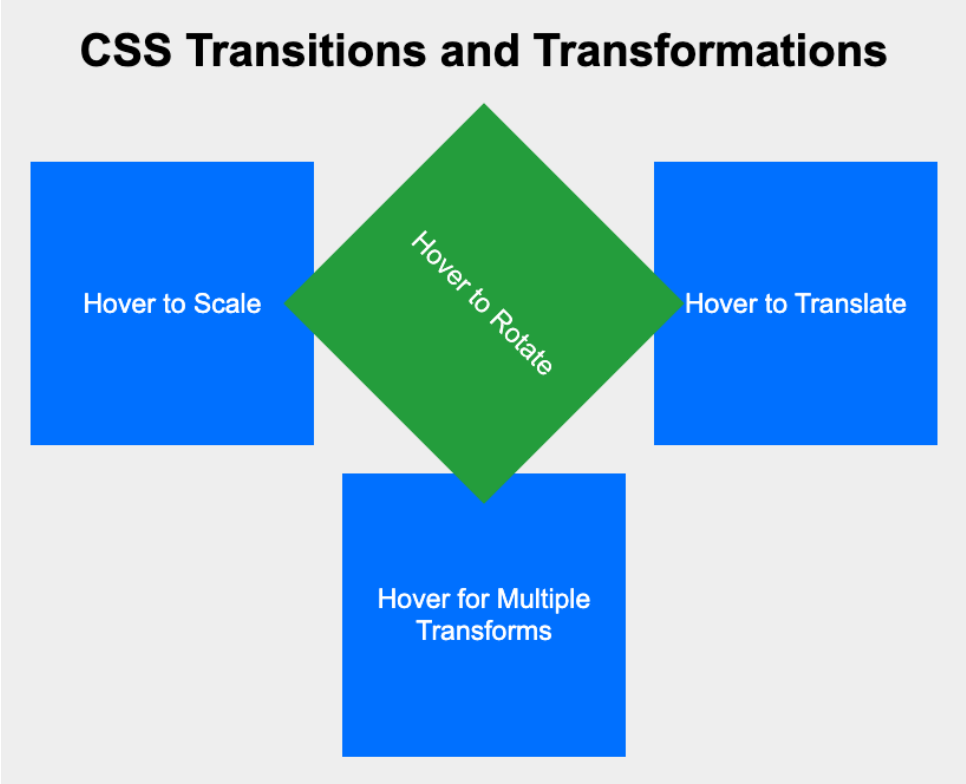
ADVANCED CSS ELEMENTS

FULL STACK SKILLS BOOTCAMP

ADVANCED CSS ELEMENTS

- **Lesson Overview:**
- In this lesson, we will be introduced to:

1. CSS position: Absolute, Fixed, Sticky, and Relative
2. Pseudo Elements
3. Pseudo Classes
4. Simple CSS Transitions and Transformations



CSS POSITION

- Definition:

  CSS position defines how an element is placed in relation to its normal document flow and its surrounding elements.

  position: relative

  Positions the element relative to its original (normal) position.
  Moves the element without affecting the layout of other elements.

```css
.relative-box {
  position: relative;
  top: 20px; /* Moves down 20px */
  left: 10px; /* Moves right 10px */
}
```

POSITION: ABSOLUTE

position: absolute

The element is positioned relative to its nearest positioned ancestor (other than static).

Removes the element from the normal document flow.

In this example, the box will be placed 50px from the top and 20px from the right of its container.

```
.absolute-box {
  position: absolute;
  top: 50px;
  right: 20px;
}
```

POSITION: FIXED

position: fixed

Positioned relative to the browser window.

Remains in the same position even when scrolling.

Useful for sticky navigation bars or fixed headers.

```
.fixed-header {
  position: fixed;
  top: 0;
  width: 100%;
}
```

POSITION: STICKY

position: sticky

Hybrid between relative and fixed positioning.

Element "sticks" to a defined position when scrolling until its containing block scrolls out of view.

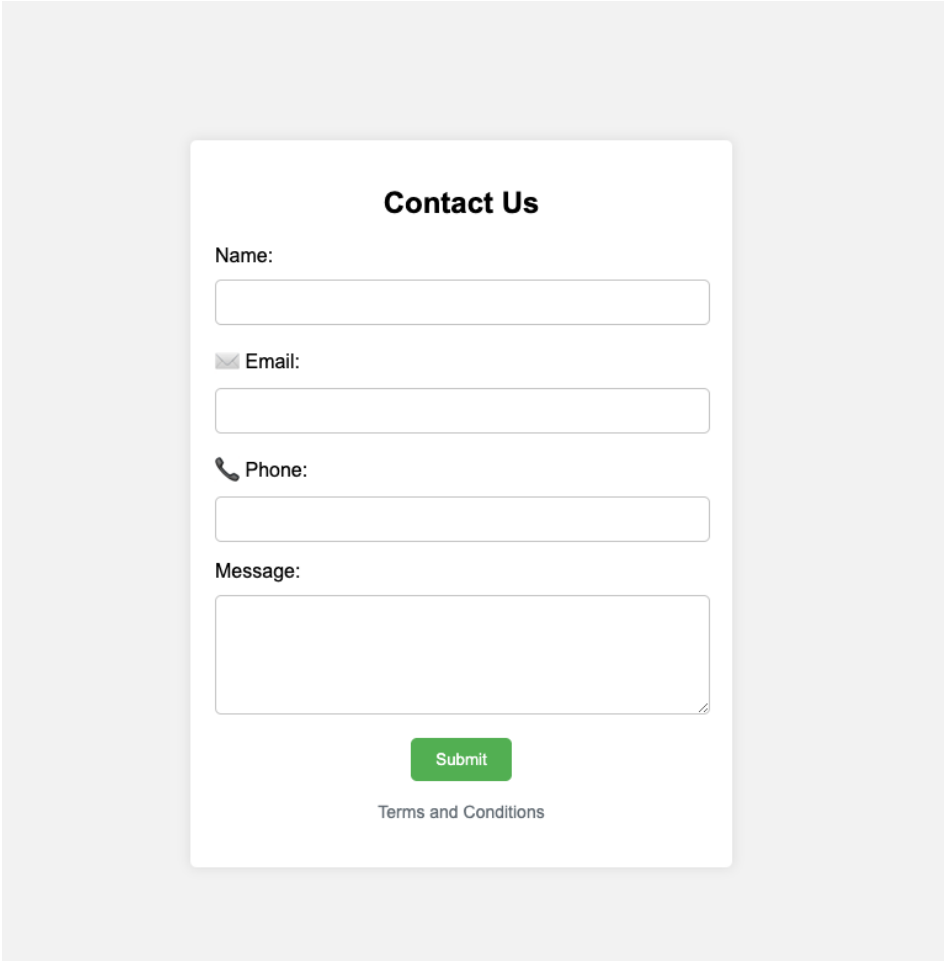
Stays 10px from the top of the viewport when scrolling.

Example...

```
.sticky-element {
  position: sticky;
  top: 10px;
}
```

PSEUDO ELEMENTS

- **Why use pseudo elements?**
- Decorative elements without adding extra HTML.
- Create complex layouts and styles with minimal code.
- **Example**: Adding icons, quotes, or shapes with ::before and ::after.



PSEUDO CLASSES

- Pseudo classes target elements based on their state, position, or interaction with the user.
- For example:
- **:hover**: Styles an element when the user hovers over it.
  :focus: Styles an element when it has keyboard or mouse focus.
  :nth-child(n): Targets the nth child of a parent element.
- This example changes the button background when hovered.

  Example...

```
button:hover {
    background-color: blue;
}
```

ADVANCED PSEUDO CLASSES

- **:first-child**: Targets the first child of a parent element.
- **:last-child**: Targets the last child of a parent.
- **:not(selector)**: Selects all elements that do not match the given selector.

This example selects all <p> elements that don't have the highlight class.

```
p:not(.highlight) {
  color: gray;
}
```

CSS TRANSITIONS

- CSS transitions allow smooth changes between styles when an element's state changes.

  With this example; when hovered, the button smoothly transitions to the new background color over 0.3 seconds.

```
button {
  transition: background-color 0.3s ease;
}

button:hover {
  background-color: coral;
}
```

CSS TRANSITION PROPERTIES

- **transition-property**: The CSS property to animate (e.g., background-color).
- **transition-duration**: How long the transition lasts (e.g., 0.3s).
- **transition-timing-function**: The speed curve (ease, linear, etc.).
- **transition-delay**: Delay before the transition starts.

CSS TRANSFORMATIONS

- CSS transform allows you to move, rotate, scale, and skew elements.

  In this example, the element grows 20% larger when hovered.

```
.box:hover {
  transform: scale(1.2);
}
```

COMMON TRANSFORM FUNCTIONS

- **scale(n)**: Enlarges or shrinks the element.
- **rotate(deg)**: Rotates the element by a specified degree.
- **translate(x, y)**: Moves the element along the X and Y axis.
- **skew(x, y)**: Skews the element along the X and Y axis.

COMBINING TRANSITIONS AND TRANSFORMS

- You can combine transitions with transforms for smooth animations.

  In this example, the element rotates and scales when hovered, and the transition makes it smooth.

```css
.box {
  transition: transform 0.3s ease;
}

.box:hover {
  transform: rotate(15deg) scale(1.1);
}
```

RECAP

- **CSS Positioning**: Control where elements appear in relation to others.
- **Pseudo Elements and Classes**: Style elements dynamically based on state or parts of the element.
- **Transitions and Transformations**: Create smooth animations and dynamic effects with minimal effort.

- **MDN Web Docs – CSS Box Model:** https://developer.mozilla.org/en-US/docs/Web/CSS/position
- **CSS-Tricks – CSS Box Model:** https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements

Step8Up Academy

QUESTIONS?