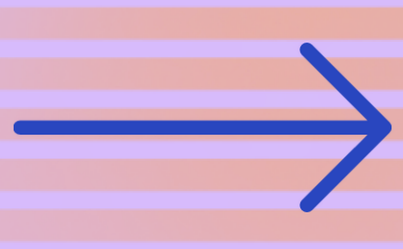
# Intersection Observer API

## Make Visibility Tracking Easy

JS

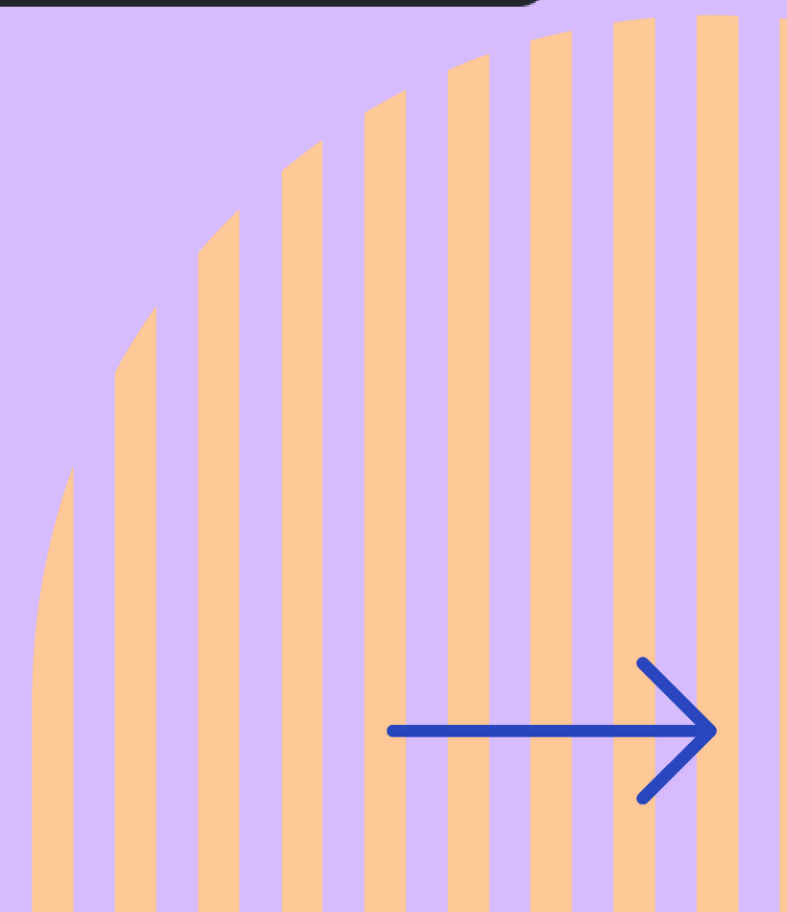


# How Intersection Observer Works

**Core Idea:**
Tracks element visibility relative to a container (default: viewport).

javascript

```javascript
// Create an observer
const observer = new IntersectionObserver((entries) => {
  entries.forEach(entry => {
    if (entry.isIntersecting) {
      console.log(`${entry.target.id} is visible`);
    }
  });
});

// Observe an element
const target = document.querySelector("#element");
observer.observe(target);
```

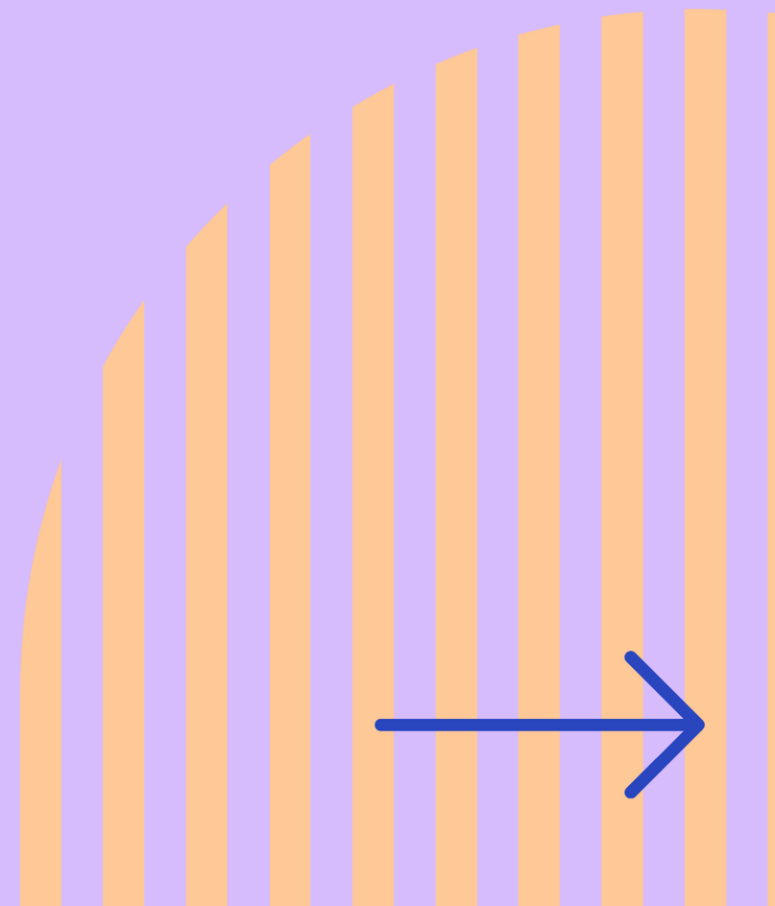


# Lazy Loading Images

**Optimize Performance with Lazy Loading:**
Only load images when they are about to enter the viewport.

javascript

```javascript
const lazyLoad = new IntersectionObserver((entries, observer) => {
  entries.forEach(entry => {
    if (entry.isIntersecting) {
      const img = entry.target;
      img.src = img.dataset.src; // Load image
      observer.unobserve(img);   // Stop observing
    }
  });
});

// Observe all images with data-src
document.querySelectorAll("img[data-src]")
.forEach(img => lazyLoad.observe(img));
```

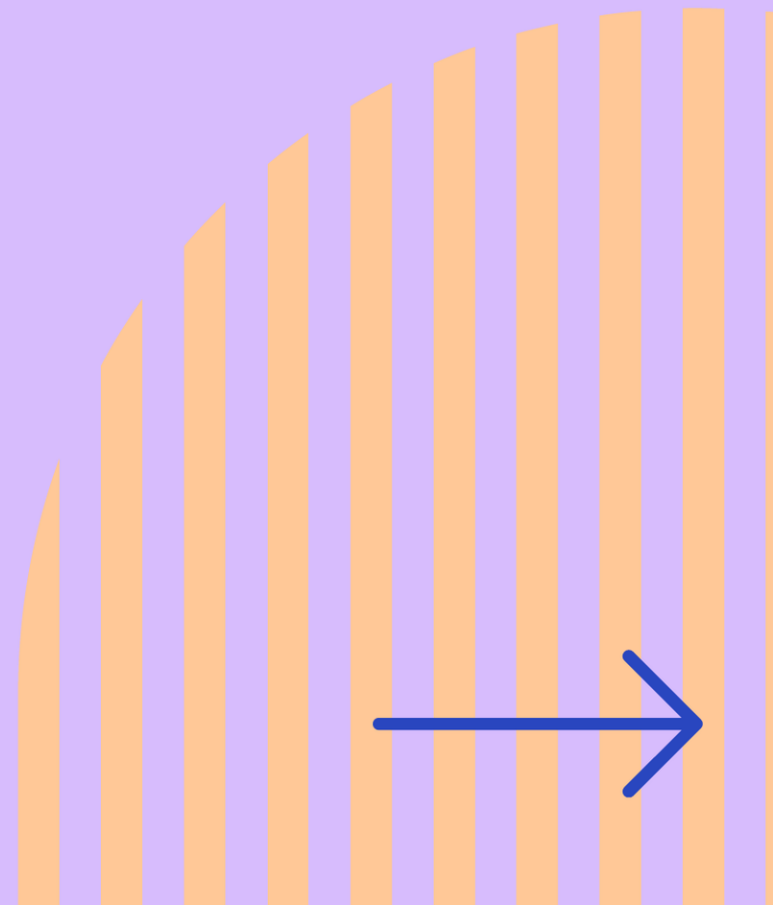


# Infinite Scrolling

**Dynamically Load More Content:**
Automatically fetch more data when the user scrolls to the bottom.

javascript

```javascript
const loadMore = new IntersectionObserver((entries) => {
  entries.forEach(entry => {
    if (entry.isIntersecting) {
      fetchMoreData(); // Load more content
    }
  });
});

// Observe the loader at the bottom
const loader = document.querySelector("#loader");
loadMore.observe(loader);
```



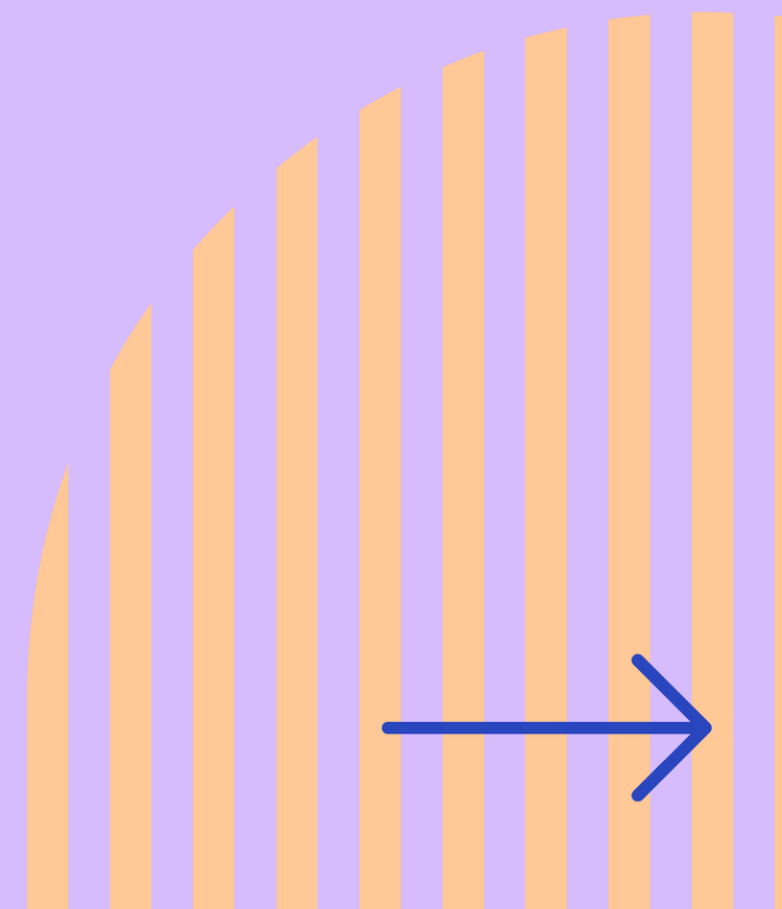
# Triggering Animations

## Animate Elements When They Enter the Viewport:

Add animations dynamically as elements appear in view.

javascript

```javascript
const animateOnScroll = new IntersectionObserver((entries) => {
  entries.forEach(entry => {
    if (entry.isIntersecting) {
      entry.target.classList.add("animate");
    }
  });
});

// Observe elements to animate
document.querySelectorAll(".hidden")
.forEach(el => animateOnScroll.observe(el));
```



# Customizing the Observer

## Configuring Options:

Fine-tune the observer with root, rootMargin, and threshold.

javascript

```javascript
const options = {
  root: document.querySelector("#scroll-container"), // Parent container
  rootMargin: "20px", // Margin around the root
  threshold: [0.25, 0.5, 0.75] // Trigger at different visibility levels
};

const observer = new IntersectionObserver(callback, options);
observer.observe(document.querySelector("#target"));
```

# Best Practices

## Write Performant and Clean Code

javascript

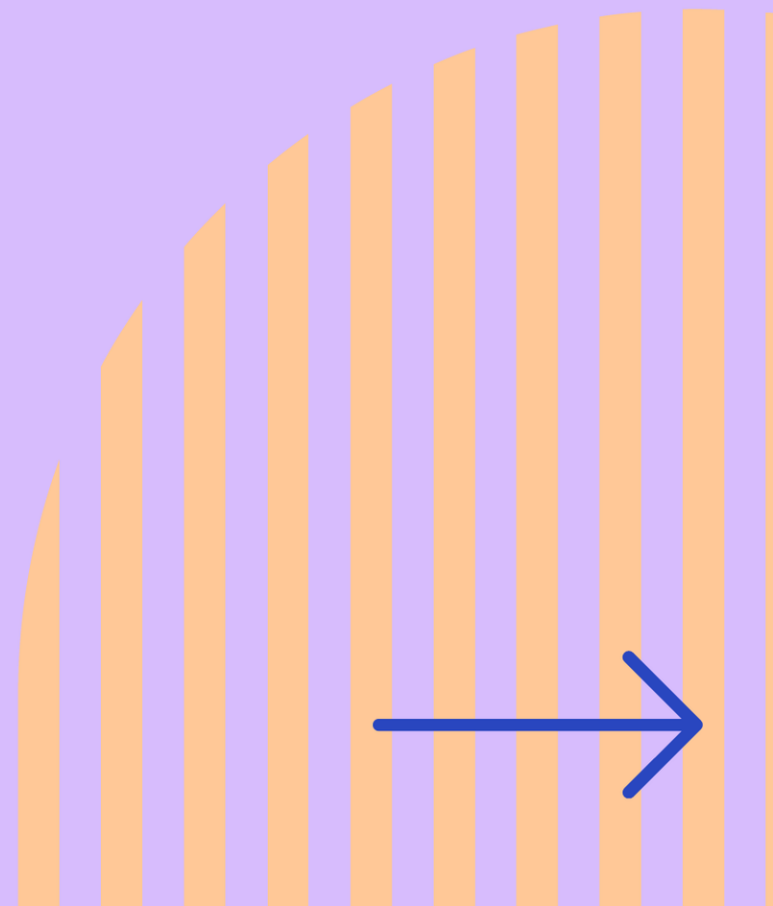
```javascript
// Stop observing elements when they're no longer needed
const observer = new IntersectionObserver((entries, observer) => {
  entries.forEach(entry => {
    if (entry.isIntersecting) {
      observer.unobserve(entry.target);
    }
  });
});

// Always unobserve elements to avoid memory leaks!
observer.observe(document.querySelector("#element"));
```

# Conclusion

Why Learn Intersection Observer?

Use **Intersection Observer** for: 1. Lazy Loading 2. Infinite Scrolling 3. Scroll Animations 4. Visibility Tracking

```javascript
console.log("Smarter UIs start here!");
```



# HAPPY CODING