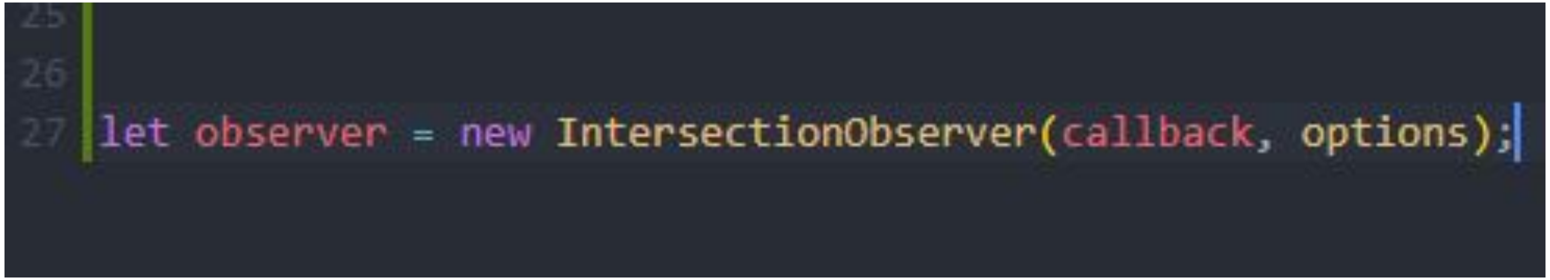
Intersection Observer

Javascript API

User cases

- Lazy-loading content
- Infinite scroll
- Animating target based on screen position

Components of the Intersection Observer



```
let observer = new IntersectionObserver(callback, options);
```

Initiate Intersection Observer as a new object

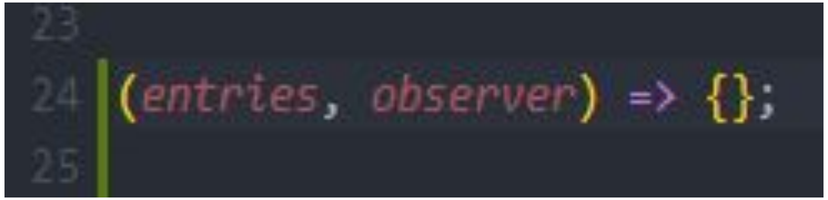
- Callback
- Options

Callback

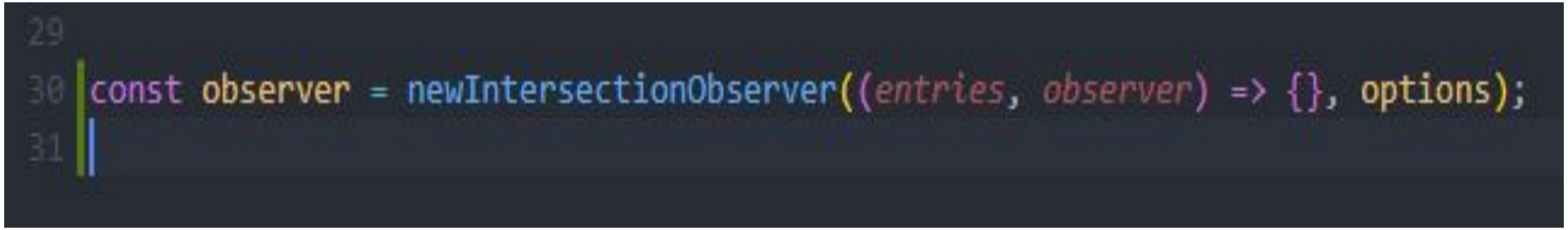
- Callback takes 2 arguments. Last argument is optional.

- Entries is for each target reporting its change.

- Last argument is used if you want to un-observe a target.



```
(entries, observer) => {};
```



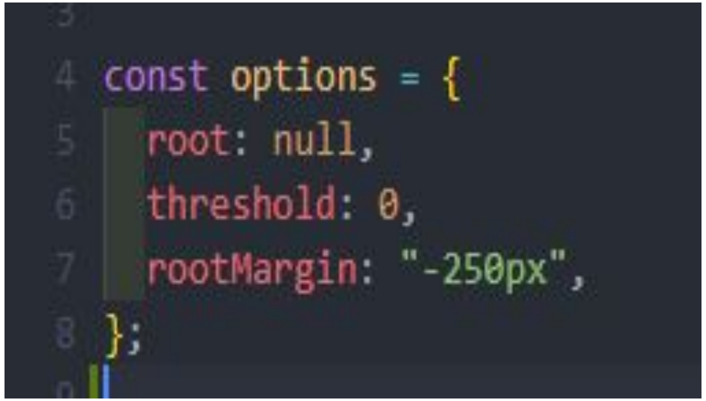
```
const observer = newIntersectionObserver((entries, observer) => {}, options);
```

Options

root: element that is being considered the viewport. If left empty or set to null, defaults to browser.

threshold: 0-1 value representing what percent of target visibility will trigger the callback.

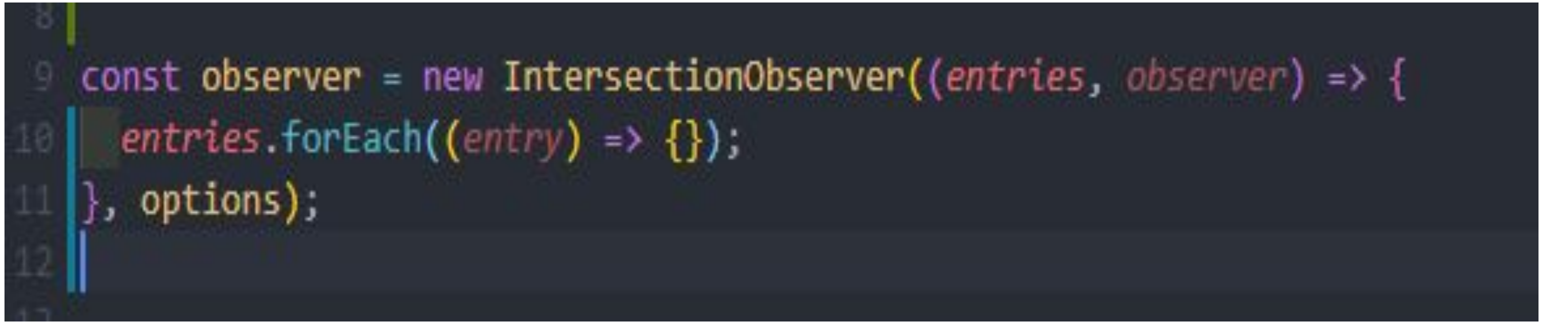
rootMargin: the margin around the root that will activate. Works like html margin.



```
const options = {
  root: null,
  threshold: 0,
  rootMargin: "-250px",
};
```

Unwrapping Notes

- Entries will come back as an array
- Unwrap with a forEach loop

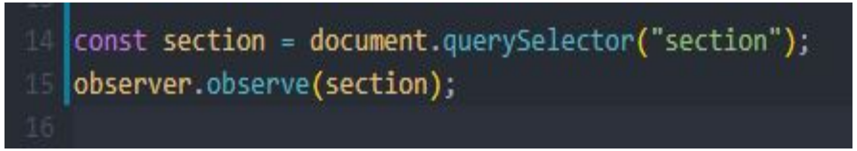


```
const observer = new IntersectionObserver((entries, observer) => {
  entries.forEach((entry) => {});
}, options);

```

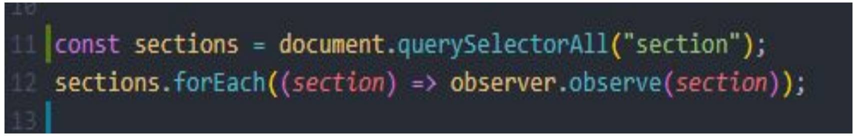
Targeting an element for the observer

Setting a single observer on an element.



```
const section = document.querySelector("section");
observer.observe(section);
```

Setting multiple observers on an array of elements.

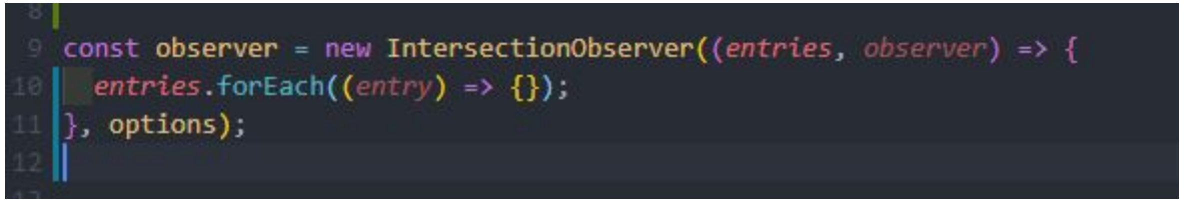


```
const sections = document.querySelectorAll("section");
sections.forEach((section) => observer.observe(section));
```

Actions to entry

isIntersecting: true or false value based on observed place in the viewport. Will help us determine when to execute actions on target

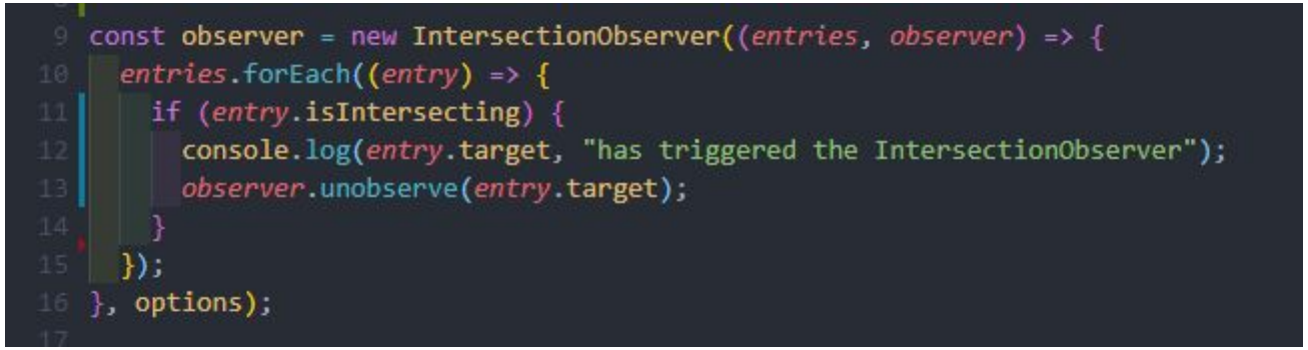
target: the target html property That is being observed. Will be needed to get or change any properties within the entry



```
const observer = new IntersectionObserver((entries, observer) => {
  entries.forEach((entry) => {});
}, options);
```

Observer Argument

- The observer argument is used if you want to stop observing an element



```
const observer = new IntersectionObserver((entries, observer) => {
  entries.forEach((entry) => {
    if (entry.isIntersecting) {
      console.log(entry.target, "has triggered the IntersectionObserver");
      observer.unobserve(entry.target);
    }
  });
}, options);
```