A Visual Guide to References in Javascript

↳ Reassigning a function parameter will only affect the local variable, not the original one that was passed in.

Primitive types in JS: • string, number, boolean, symbol, undefined, and null
• they are immutable (a.k.a. read-only).
→ when a variable holds one of these primitive types, you can't modify the value itself — only reassign that variable to a new value.

objects, arrays, functions, and other data structures (e.g, Map and Set):

↳ they are all objects and mutable.

```
document.addEventListener('click', () => console.log('clicked'));
   "    .removeEventListener(           "                       );
```

↳ this code will never remove the event listener because those two arrow functions aren't referentially equal. They aren't the same function.

↳ Every time you write an arrow function or a regular one, that creates a new object.

↳ Fix:

```
const onClick = () => console.log('clicked');
document.addEventListener('click', onClick);
   "    .removeEventListener(    "     );
```

```
const newArray = [...array].sort();
let copy = {...book}
```

↳ we can use the spread operator to make object, array copies.