Event Handlers

Event handlers in JS can be used if I need something to happen after an event (like a click or submit) happens.

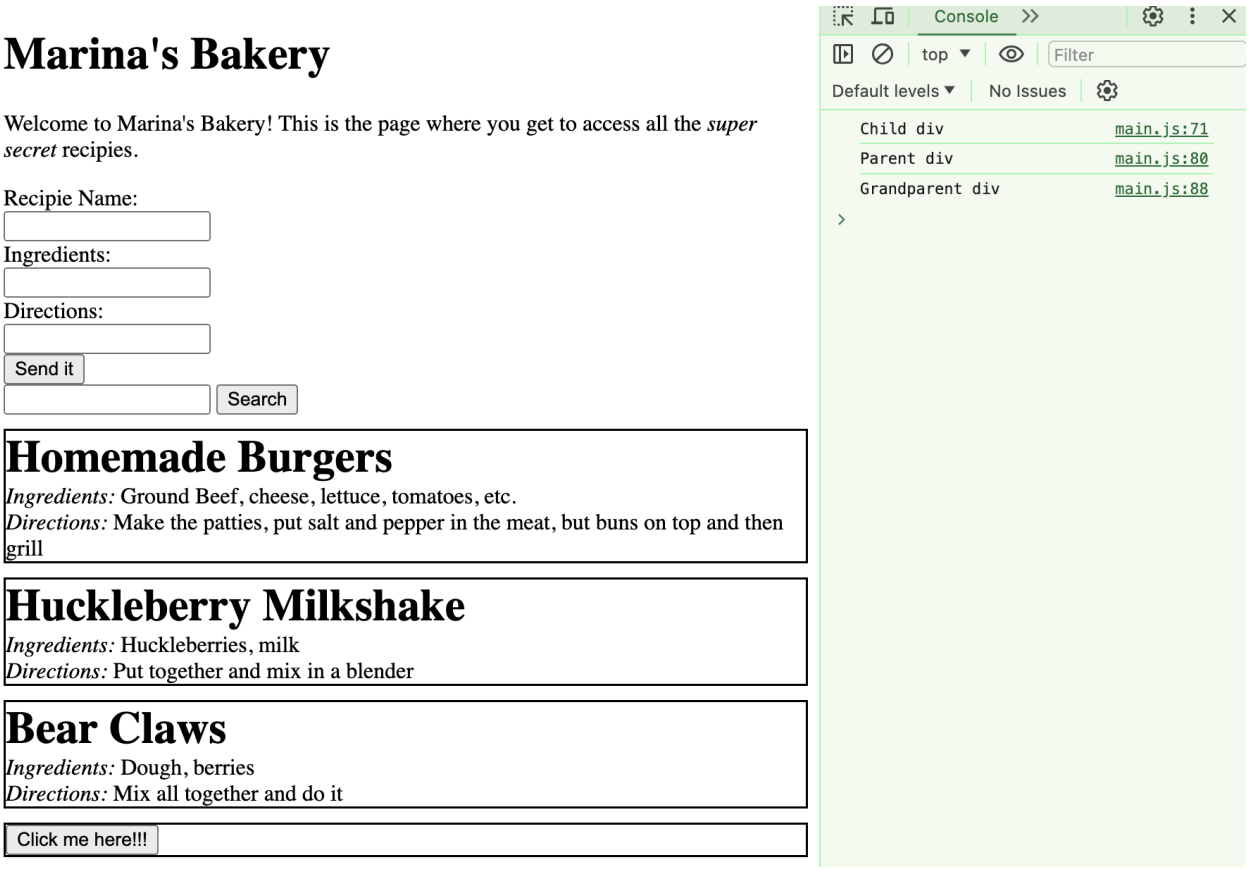
In order to listen to an event, I need to get the object (like a button, div, etc.) and addEventListener to it.

```
const searchBar = document.getElementById('search-bar');
const handleForm = (event) => event.preventDefault();
searchBar.addEventListener('submit', handleForm);
```

The addEventListener function takes in a few parameters:
1. The event type (submit, click, etc.)
2. What I want the listener to do after hearing the event type
3. In the case if there's a need for event bubbling or event catching, I could set catch: true

Event Bubbling

There are times when there are divs on top of divs and there could be a button on top. If I add event listeners to all of the divs, and "touch" the top button, all of the divs will be touched, starting from the button all the way down to the grandparent.



If I add console.log statements to all the listeners in that section that holds the lower button and divs, then the "bubbles" will go down to the grandparent

Event Catching

Event catching has the opposite effect, going from the grandparent element, to the child. All that is needed is {capture: true} for the third parameter.

```
anotherRandomDiv.addEventListener(
  'click',
  () => {
    console.log('Grandparent div');
  },
  { capture: true }
);
```

If **only** the grandparent has this, then the result looks like this:

```
Grandparent div          main.js:88
Child div                main.js:71
Parent div               main.js:80
```

The grandparent gets captured, and then everything goes back to bubbling from the child down. If all of them will be set like that, then the catching would go from grandparent to child.

Stop Propogation

If I won't want this effect, inside the function that is inside the second parameter, I could add stopPropogation().