perplexity

explain to me step by step

```
const form = document.querySelector("form"),
emailField = form.querySelector(".email-field"),
emailInput = emailField.querySelector(".email"),
passField = form.querySelector(".create-password"),
passInput = passField.querySelector(".password"),
cPassField = form.querySelector(".confirm-password"),
cPassInput = cPassField.querySelector(".cPassword");
// Email Validtion
function checkEmail() {
const emaiPattern = /[ ]+@[^ ]+.[a-z]{2,3}!%*?&])[A-Za-z\d@!/;
if (!passInput.value.match(passPattern)) {
return passField.classList.add("invalid"); //adding invalid class if password input value do not
match with passPattern
}
passField.classList.remove("invalid"); //removing invalid class if password input value matched
with passPattern
}
// Confirm Password Validtion
function confirmPass() {
if (passInput.value !== cPassInput.value || cPassInput.value === "") {
return cPassField.classList.add("invalid");
}
cPassField.classList.remove("invalid");
}
// Calling Funtion on Form Sumbit
form.addEventListener("submit", (e) => {
e.preventDefault(); //preventing form submitting
checkEmail();
createPass();
confirmPass();
//calling function on key up
emailInput.addEventListener("keyup", checkEmail);
passInput.addEventListener("keyup", createPass);
cPassInput.addEventListener("keyup", confirmPass);
if (
!emailField.classList.contains("invalid") &&
!passField.classList.contains("invalid") &&
!cPassField.classList.contains("invalid")
) {
location.href = form.getAttribute("action");
```

```
}
})
```

Absolutely! Let’s break down your JavaScript code step by step so you understand how it works and what each part does.

Step-by-Step Explanation

1. Selecting Elements from the DOM

```
const form = document.querySelector("form"),
  emailField = form.querySelector(".email-field"),
  emailInput = emailField.querySelector(".email"),
  passField = form.querySelector(".create-password"),
  passInput = passField.querySelector(".password"),
  cPassField = form.querySelector(".confirm-password"),
  cPassInput = cPassField.querySelector(".cPassword");
```

- **What it does:**
  - Selects the main form and various input fields (email, password, confirm password) and their parent containers.
  - This makes it easy to access and manipulate these elements later.

2. Email Validation Function

```
function checkEmail() {
  const emaiPattern = /^[^ ]+@[^ ]+\.[a-z]{2,3}$/;
  if (!emailInput.value.match(emaiPattern)) {
    return emailField.classList.add("invalid");
  }
  emailField.classList.remove("invalid");
}
```

- **What it does:**
  - Checks if the email input matches a basic email pattern.
  - If not, adds the class "invalid" to highlight the error.
  - If valid, removes the "invalid" class.

3. Show/Hide Password Feature

```
const eyeIcons = document.querySelectorAll(".show-hide");
eyeIcons.forEach((eyeIcon) => {
  eyeIcon.addEventListener("click", () => {
    const pInput = eyeIcon.parentElement.querySelector("input");
```

```
    if (pInput.type === "password") {
      eyeIcon.classList.replace("bx-hide", "bx-show");
      return (pInput.type = "text");
    }
    eyeIcon.classList.replace("bx-show", "bx-hide");
    pInput.type = "password";
  });
});
```

- **What it does:**
  - Selects all elements with the class "show-hide" (usually eye icons).
  - When clicked, toggles the password field between visible (`type="text"`) and hidden (`type="password"`).
  - Also toggles the icon class for visual feedback.

4. Password Validation Function

```
function createPass() {
  const passPattern =
    /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$/;
  if (!passInput.value.match(passPattern)) {
    return passField.classList.add("invalid");
  }
  passField.classList.remove("invalid");
}
```

- **What it does:**
  - Checks if the password meets certain criteria (at least 8 characters, upper/lowercase, number, special character).
  - Adds "invalid" class if not valid, removes it if valid.

5. Confirm Password Validation

```
function confirmPass() {
  if (passInput.value !== cPassInput.value || cPassInput.value === "") {
    return cPassField.classList.add("invalid");
  }
  cPassField.classList.remove("invalid");
}
```

- **What it does:**
  - Checks if the confirm password input matches the original password.
  - Adds "invalid" class if they don't match or if confirm password is empty.

6. Form Submission Handling

```
form.addEventListener("submit", (e) => {
  e.preventDefault();
  checkEmail();
  createPass();
  confirmPass();
  emailInput.addEventListener("keyup", checkEmail);
  passInput.addEventListener("keyup", createPass);
  cPassInput.addEventListener("keyup", confirmPass);
  if (
    !emailField.classList.contains("invalid") &&
    !passField.classList.contains("invalid") &&
    !cPassField.classList.contains("invalid")
  ) {
    location.href = form.getAttribute("action");
  }
})
```

- **What it does:**
  - Prevents the default form submission.
  - Runs all three validation functions.
  - Adds real-time validation as the user types (on keyup).
  - If all fields are valid (none have the "invalid" class), it redirects to the URL specified in the form's `action` attribute (simulating a successful submission).

Summary Table

Functionality	How it works
Email Validation	Checks format, adds/removes "invalid" class
Show/Hide Password	Toggles input type and icon class
Password Validation	Checks complexity, adds/removes "invalid" class
Confirm Password	Ensures match with password, adds/removes "invalid" class
Form Submission	Prevents default, validates, enables real-time validation, redirects

What Happens When You Use This Code?

1. User fills out the form.
2. On submit, all fields are validated.
3. If a field is invalid, it gets visually marked (with "invalid" class).
4. As the user types, validation happens in real time.
5. If all fields are valid, the page redirects to the form's action URL.

Let me know if you want a visual example or further breakdown of any part!