DAY 2

Font Import

```
@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600&display=swap');
```

- Imports the **Poppins** font from Google Fonts.
- It includes weights: 400 (regular), 500 (medium), 600 (semi-bold).

Reset and Base Styling

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Poppins', sans-serif;
}
```

- margin/padding: 0: Removes default spacing of all elements.
- box-sizing: border-box: Includes padding and border in element's total width/height.
- font-family: Sets Poppins as the default font for all elements.

Body Styling

```
body {
  margin: 0;
  padding: 0;
  display: flex;
  padding: 0 10px;
  align-items: center;
  justify-content: center;
  flex-direction: column;
  background: linear-gradient(-45deg, #89f7fe, #11a6ff, #6a11cb, #ef0b7d);
  background-size: 400% 400%;
  animation: gradientBG 10s ease infinite;
}
```

- Uses **Flexbox** to center content vertically and horizontally.
- background: A **multi-color animated gradient**.
- background-size: 400%: Makes room for gradient movement.
- animation: gradientBG: Applies a smooth back-and-forth animation over 10 seconds.

Gradient Animation

```
@keyframes gradientBG {
  0% { background-position: 0% 50%; }
  50% { background-position: 100% 50%; }
  100% { background-position: 0% 50%; }
}
```

- Animates background gradient horizontally from left → right → left.
- Creates a smooth flowing background effect.

Heading (h1)

```
h1 {
  margin-top: 20px;
  font-size: 32px;
  color: #ffffff;
  text-align: center;
  margin-bottom: 10px;
}
```

- Adds spacing around the heading.
- Sets large font size and **white color** to pop on the gradient background.

Container Box

```
.container {
  width: 440px;
  border-radius: 7px;
  background: #fff;
  box-shadow: 0 10px 20px rgba(0,0,0,0.08);
}
```

- Fixed width card-like container.
- `border-radius`: rounded corners.
- `box-shadow`: adds subtle elevation for depth.

Container Title (h2)

```
.container h2 {
  font-size: 25px;
  font-weight: 500;
  padding: 16px 25px;
  border-bottom: 1px solid #ccc;
}
```

- `padding`: space inside the header.
- `border-bottom`: visual divider from content below.

Main Content Section

```
.container .content {
  margin: 25px 20px 35px;
}
```

- Adds margin around internal content for spacing.

Scrambled Word Display

```
.content .word {
  user-select: none;
  font-size: 33px;
  font-weight: 500;
  text-align: center;
  letter-spacing: 24px;
  margin-right: -24px;
  word-break: break-all;
  text-transform: uppercase;
}
```

- user-select: none: Prevents text from being selected (for better game UX).
- letter-spacing + margin-right: Equal letter spacing.
- uppercase: Makes the word all capital letters.

Hint & Details Text

```
.content .details {
  margin: 25px 0 20px;
}
.details p {
  font-size: 18px;
  margin-bottom: 10px;
}
.details p b {
  font-weight: 500;
}
```

- Controls spacing and font styling of hint/description details.

Input Field

```
.content input {
  width: 100%;
  height: 60px;
  outline: none;
  padding: 0 16px;
  font-size: 18px;
  border-radius: 5px;
  border: 1px solid #bfbfbf;
}
.content input:focus {
  box-shadow: 0px 2px 4px rgba(0,0,0,0.08);
}
.content input::placeholder {
  color: #aaa;
}
.content input:focus::placeholder {
  color: #bfbfbf;
}
```

- Styled for modern look.
- On focus, changes placeholder and adds shadow.

Buttons Container

```
.content .buttons {
  display: flex;
  margin-top: 20px;
  justify-content: space-between;
}
```

- Buttons are placed side-by-side with space between.

Individual Buttons

```
.buttons button {
  border: none;
  outline: none;
  color: #fff;
  cursor: pointer;
  padding: 15px 0;
  font-size: 17px;
  border-radius: 5px;
  width: calc(100% / 2 - 8px);
  transition: all 0.3s ease;
}
.buttons button:active {
  transform: scale(0.97);
}
```

- Uniform styling for all buttons.
- `transition`: Smooth hover/press effect.
- `:active`: Slight press-in animation.

Button Color Variants

```
.buttons .refresh-word {
  background: #6C757D;
}
.buttons .refresh-word:hover {
  background: #5f666d;
}
.buttons .check-word {
  background: #5372F0;
}
.buttons .check-word:hover {
  background: #2c52ed;
}
```

- Refresh is gray; Check is blue.
- On hover, background slightly darkens for feedback.

Responsive Media Query

```
@media screen and (max-width: 470px) {
  .container h2 {
```

```
    font-size: 22px;
    padding: 13px 20px;
  }
  .content .word {
    font-size: 30px;
    letter-spacing: 20px;
    margin-right: -20px;
  }
  .container .content {
    margin: 20px 20px 30px;
  }
  .details p {
    font-size: 16px;
    margin-bottom: 8px;
  }
  .content input {
    height: 55px;
    font-size: 17px;
  }
  .buttons button {
    padding: 14px 0;
    font-size: 16px;
    width: calc(100% / 2 - 7px);
  }
}
```

- Adjusts sizing and spacing for small screens like mobile.
- Ensures readability and usability on phones.

JAVASCRIPT

DOM Selection

```
const wordText = document.querySelector(".word"),
    hintText = document.querySelector(".hint span"),
    timeText = document.querySelector(".time b"),
    inputField = document.querySelector("input"),
    refreshBtn = document.querySelector(".refresh-word"),
    checkBtn = document.querySelector(".check-word");
```

This block selects elements from the HTML and stores them in variables:

Variable	Selects	Used for
wordText	.word element	To display the scrambled word
hintText	.hint span	To display the hint
timeText	.time b	To display the countdown timer
inputField	<input> field	Where user types the guessed word
refreshBtn	.refresh-word button	To reshuffle and restart the game
checkBtn	.check-word button	To check if user guessed correctly

Timer Initialization

```
let correctWord, timer;

const initTimer = maxTime => {
   clearInterval(timer); // clear any previous timer
   timer = setInterval(() => {
      if(maxTime > 0) {
         maxTime--;
         return timeText.innerText = maxTime;
      }
      clearInterval(timer);
      alert(`Time off! ${correctWord.toUpperCase()} was the correct word`);
      initGame(); // Restart game after time runs out
   }, 1000);
}
```

Purpose: This function starts a **countdown from 30 seconds** and updates it every second.

- `setInterval()` runs code every 1000ms (1 sec).
- If time runs out (`maxTime === 0`), the timer stops and shows the correct word.
- Game restarts automatically.

Game Initialization

```
const initGame = () => {
   initTimer(30); // Start a 30-second timer

   let randomObj = words[Math.floor(Math.random() * words.length)];
   // Get a random word-hint object

   let wordArray = randomObj.word.split(""); // Split word into letters
   // Fisher–Yates shuffle
   for (let i = wordArray.length - 1; i > 0; i--) {
      let j = Math.floor(Math.random() * (i + 1));
      [wordArray[i], wordArray[j]] = [wordArray[j], wordArray[i]];
   }

   wordText.innerText = wordArray.join(""); // Display shuffled word
   hintText.innerText = randomObj.hint; // Display hint
   correctWord = randomObj.word.toLowerCase(); // Store correct word
   inputField.value = ""; // Clear previous input
   inputField.setAttribute("maxlength", correctWord.length); // Limit input length
}
```

- Selects a **random word and its hint** from a `words` array (not shown in your snippet).
- Shuffles the word using **Fisher-Yates algorithm**.
- Updates UI with the scrambled word and hint.
- Prepares the input field for new entry.

Checking User Input

```
const checkWord = () => {
```

```
  let userWord = inputField.value.toLowerCase(); // Get user's input
  if(!userWord) return alert("Please enter the word to check!");

  if(userWord !== correctWord)
    return alert(`Oops! ${userWord} is not a correct word`);

  alert(`Congrats! ${correctWord.toUpperCase()} is the correct word`);
  initGame(); // Start new game after correct answer
}
```

- Gets the word the user typed.
- If it's empty, prompts them to enter something.
- If incorrect → shows an alert.
- If correct → shows congratulations and starts a new round.

Event Listeners

```
refreshBtn.addEventListener("click", initGame);
checkBtn.addEventListener("click", checkWord);
```

- When **"Refresh Word"** button is clicked → `initGame()` is called (reshuffles word and resets timer).
- When **"Check Word"** button is clicked → `checkWord()` is called.

Auto Start on Page Load

```
initGame();
```

- Starts the game **automatically when the page loads** by calling `initGame()` once.

Enter Key Function

```
inputField.addEventListener("keydown", (event) => {

  if (event.key === "Enter") {

    checkWord();

  }

});
```

- It listens for any key press **while typing** in the input field.
- If the key is `"Enter"` → it automatically calls the `checkWord()` function.