Curser effect:-

To create a circle wave effect around the cursor when clicked in Windows 10, you'd need to use a scripting tool or programming language that interacts with the mouse events. Here's a simple way to achieve this effect using **HTML**, **CSS**, and **JavaScript** in a webpage.

**Steps:**

1. **HTML**: Create the structure.
2. **CSS**: Style the circle wave animation.
3. **JavaScript**: Detect the mouse click and trigger the wave effect.

Here’s a simple example that creates a circle wave effect around the cursor click:

**Code:**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Circle Wave on Click</title>

<style>

body {

margin: 0;

height: 100vh;

background-color: #f0f0f0;

overflow: hidden;

cursor: pointer;

}

.wave {

position: absolute;

border-radius: 50%;

background: rgba(0, 150, 255, 0.4);

pointer-events: none;

animation: wave-animation 1.5s ease-out;

}

@keyframes wave-animation {

0% {

transform: scale(0);

opacity: 1;

}

100% {

transform: scale(10);

opacity: 0;

}

}

</style>

</head>

<body>

<script>

document.body.addEventListener('click', function (e) {

let wave = document.createElement('div');

wave.classList.add('wave');

// Position the wave where the click happened

const size = Math.max(window.innerWidth, window.innerHeight);

const x = e.clientX - size / 2;

const y = e.clientY - size / 2;

wave.style.width = wave.style.height = `${size}px`;

wave.style.left = `${x}px`;

wave.style.top = `${y}px`;

// Append to the body and remove after animation

document.body.appendChild(wave);

setTimeout(() => {

wave.remove();

}, 1500); // Animation duration is 1.5s

});

</script>

</body>

</html>

**Explanation:**

1. **HTML**: The body is set to take the full viewport height with no margin.
2. **CSS**:
   * The .wave class creates a circle with border-radius: 50%.
   * The @keyframes animation gradually increases the size of the circle (from scale 0 to scale 10) while reducing its opacity, creating a fading wave effect.
3. **JavaScript**:
   * When the user clicks anywhere on the body, a new div with the class wave is created.
   * The circle is positioned at the mouse click location (e.clientX, e.clientY).
   * After the animation duration (1.5 seconds), the wave is removed from the DOM.

**How to Use:**

* Save the code as an .html file (e.g., circle-wave.html).
* Open the file in any web browser, and when you click anywhere on the page, you'll see the circle wave effect.

This method works on any webpage you open it in. If you want to implement it system-wide in Windows 10 (i.e., outside of a webpage), you'd need to develop a desktop application or script using a tool like **AutoHotKey** or **Electron**.

Scroll:-

To create a stylish scrollbar using **CSS** and **HTML**, you can customize the appearance of the scrollbar using CSS properties. These properties allow you to control the color, size, and style of the scrollbar, especially for webkit-based browsers like Chrome, Safari, and Edge.

Here’s an example of how to style a scrollbar with CSS:

**HTML and CSS Example for Stylish Scrollbar:**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Stylish Scrollbar</title>

<style>

/\* Apply styles to the entire page \*/

body {

font-family: Arial, sans-serif;

padding: 20px;

background-color: #f4f4f4;

height: 2000px; /\* Make the page long enough to show a scrollbar \*/

}

/\* Style the scrollbar \*/

::-webkit-scrollbar {

width: 12px; /\* Set the width of the scrollbar \*/

}

/\* Style the scrollbar track (background) \*/

::-webkit-scrollbar-track {

background: #f1f1f1; /\* Light grey track \*/

border-radius: 10px;

}

/\* Style the scrollbar thumb (the draggable part) \*/

::-webkit-scrollbar-thumb {

background: #888; /\* Grey thumb \*/

border-radius: 10px;

transition: background 0.3s ease;

}

/\* Change the thumb color when hovered \*/

::-webkit-scrollbar-thumb:hover {

background: #555; /\* Darker grey when hovered \*/

}

/\* Optional: Style the scrollbar corner \*/

::-webkit-scrollbar-corner {

background: #f1f1f1;

}

</style>

</head>

<body>

<h1>Stylish Scrollbar Example</h1>

<p>Scroll down to see the effect. This page has a custom scrollbar style!</p>

<p>Keep scrolling to view the full effect. You can change the width and colors of the scrollbar to suit your design.</p>

<p>The scrollbar will have a smooth transition effect when hovered over.</p>

</body>

</html>

**Explanation:**

* **::-webkit-scrollbar**: This targets the scrollbar itself. The width property controls the width of the scrollbar in a vertical layout or the height for horizontal scrollbars.
* **::-webkit-scrollbar-track**: This targets the track or the background area of the scrollbar. You can apply styles like color and border-radius to make the track more stylish.
* **::-webkit-scrollbar-thumb**: This is the draggable part of the scrollbar. By styling it, you can change its color, shape, and even add hover effects (as seen in the example with background: #555 on hover).
* **::-webkit-scrollbar-corner**: If your page has both vertical and horizontal scrollbars, this targets the corner where they intersect. You can style this as well, although it's not always visible unless both scrollbars are active.

**Key Notes:**

1. This approach works on webkit-based browsers (like Chrome, Safari, Edge).
2. The styles applied to the scrollbar do not work in all browsers. For example, Firefox has its own way of styling scrollbars, and it requires different properties (see below for Firefox styles).
3. The example has a height of 2000px on the body to ensure the page is long enough to activate scrolling.

**Firefox Custom Scrollbars:**

For **Firefox**, you can use the following pseudo-elements to style the scrollbar:

css

Copy code

/\* Firefox Scrollbar \*/

scrollbar {

width: 12px;

}

/\* Track \*/

scrollbar-track {

background: #f1f1f1;

border-radius: 10px;

}

/\* Thumb \*/

scrollbar-thumb {

background: #888;

border-radius: 10px;

}

scrollbar-thumb:hover {

background: #555;

}

**Conclusion:**

By using the above code and properties, you can create a more visually appealing and custom scrollbar for your website. The styles can be further adjusted to match your website's theme, making it more user-friendly and engaging.

Curser:-

To make the cursor stylish using **CSS** and **HTML**, you can customize the cursor in several ways. This includes changing the cursor to a custom image or styling it with built-in CSS cursor types.

Here’s an example of how to create a stylish cursor using **CSS**:

**1. Default Stylish Cursor:**

You can customize the cursor using basic CSS cursor property by setting it to one of the built-in values or a custom image.

**Example 1: Custom Cursor Style with CSS**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Stylish Cursor</title>

<style>

/\* Make the body fill the entire viewport and add some padding \*/

body {

margin: 0;

height: 100vh;

background-color: #f0f0f0;

cursor: url('https://cdn.iconscout.com/icon/free/png-256/brush-278-458146.png'), auto; /\* Custom cursor \*/

}

/\* Add hover effects for links or interactive elements \*/

a {

text-decoration: none;

font-size: 20px;

color: #007bff;

font-weight: bold;

transition: color 0.3s;

}

a:hover {

color: #0056b3;

}

/\* Apply styles to the entire page for demonstration \*/

h1 {

text-align: center;

font-size: 3rem;

color: #333;

margin-top: 20%;

}

</style>

</head>

<body>

<h1>Stylish Cursor Example</h1>

<p style="text-align:center;">Click around and see the custom cursor in action!</p>

</body>

</html>

**Explanation:**

* **Custom Image Cursor**: In the cursor property, I used url('image-path') to specify the custom cursor image. In this example, I used an icon URL (a paintbrush) from an online source.

css

Copy code

cursor: url('https://cdn.iconscout.com/icon/free/png-256/brush-278-458146.png'), auto;

This applies a paintbrush image as the cursor. If the image doesn't load, it defaults to the system's default cursor (auto).

**2. Using Built-in Cursors:**

You can use the default CSS cursor property for built-in cursor styles like pointer, crosshair, move, etc.

**Example 2: Built-In Cursor Styles**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Cursor Styles</title>

<style>

/\* Apply a pointer cursor for clickable elements \*/

button {

cursor: pointer;

}

/\* Apply a move cursor for draggable elements \*/

.draggable {

cursor: move;

}

/\* Apply a crosshair cursor on hover \*/

.crosshair {

cursor: crosshair;

}

/\* Apply a text cursor inside text input \*/

input, textarea {

cursor: text;

}

/\* Apply the default cursor for other elements \*/

.default {

cursor: default;

}

</style>

</head>

<body>

<h1>Try Different Cursor Styles</h1>

<button>Click Me (Pointer Cursor)</button>

<div class="draggable" style="width: 200px; height: 100px; background-color: lightblue; margin-top: 20px;">Drag Me (Move Cursor)</div>

<div class="crosshair" style="width: 200px; height: 100px; background-color: lightcoral; margin-top: 20px;">Hover for Crosshair Cursor</div>

<input class="default" type="text" placeholder="Default Cursor Inside Text Box" style="margin-top: 20px; padding: 10px; width: 100%;">

</body>

</html>

**Explanation:**

* **Pointer Cursor**: Used for buttons or clickable elements.

css

Copy code

cursor: pointer;

* **Move Cursor**: This is commonly used for draggable items.

css

Copy code

cursor: move;

* **Crosshair Cursor**: Often used when indicating precision or selection.

css

Copy code

cursor: crosshair;

* **Text Cursor**: Used inside text fields for typing.

css

Copy code

cursor: text;

* **Default Cursor**: This is the default cursor for general content (like text or images).

css

Copy code

cursor: default;

**3. Animation Effect on Cursor (Optional):**

To make the cursor more dynamic, you can apply an animation effect to the cursor itself, for instance, making the cursor enlarge on hover.

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Cursor Animation</title>

<style>

body {

margin: 0;

height: 100vh;

background-color: #f4f4f4;

cursor: url('https://cdn.iconscout.com/icon/free/png-256/brush-278-458146.png'), auto;

transition: cursor 0.2s ease;

}

/\* Hover effect: Make the cursor larger \*/

body:hover {

cursor: url('https://cdn.iconscout.com/icon/free/png-256/brush-278-458146.png') 16 16, auto;

}

</style>

</head>

<body>

<h1>Stylish Animated Cursor</h1>

<p>Hover and watch the cursor size change dynamically!</p>

</body>

</html>

**Conclusion:**

By using the above methods, you can create stylish and dynamic cursors on your website. You can either use a built-in cursor style or upload a custom cursor image to make it unique. Additionally, combining animations or transitions can make the cursor more engaging for users.

Top of Form



Bottom of Form

Keyboard:-

In **CSS** and **HTML**, you cannot directly create keyboard shortcuts in the same way you can with programming languages like JavaScript, because HTML and CSS are primarily for structure and styling. However, you can **create keyboard shortcuts** for specific interactions using **JavaScript** combined with **HTML**.

Here are a few steps on how to use JavaScript to implement keyboard shortcuts:

**1. Using JavaScript to Handle Keyboard Shortcuts**

You can add event listeners in JavaScript that listen for specific key combinations. When the user presses certain keys, JavaScript will trigger actions like opening a menu, focusing an element, or triggering an event.

**Example 1: Basic Keyboard Shortcuts with JavaScript**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Keyboard Shortcuts Example</title>

<style>

body {

font-family: Arial, sans-serif;

padding: 20px;

background-color: #f4f4f4;

}

#message {

font-size: 20px;

color: #333;

}

</style>

</head>

<body>

<h1>Keyboard Shortcuts Example</h1>

<p id="message">Press "Ctrl + S" to trigger a shortcut, or "Ctrl + D" to change this message.</p>

<script>

// Function to show a message when Ctrl + S is pressed

document.addEventListener('keydown', function(e) {

// Detect if "Ctrl + S" is pressed (Ctrl key + S key)

if (e.ctrlKey && e.key === 's') {

e.preventDefault(); // Prevent the default browser behavior (Save dialog)

alert("You pressed Ctrl + S! This shortcut does nothing here.");

}

// Detect if "Ctrl + D" is pressed (Ctrl key + D key)

if (e.ctrlKey && e.key === 'd') {

document.getElementById('message').innerText = "You pressed Ctrl + D! This is a new message.";

}

});

</script>

</body>

</html>

**Explanation:**

1. **Event Listener**: The keydown event is used to detect when a key is pressed.
2. **Key Combinations**: We check if the user presses the **Ctrl** key (e.ctrlKey) combined with other specific keys, like **S** or **D** (e.key).
3. **Prevent Default Behavior**: e.preventDefault() is used to stop the browser's default behavior (like opening the Save dialog when pressing Ctrl + S).

**2. Example 2: Triggering an Action on Keyboard Shortcut**

Let’s create a simple example where pressing **Ctrl + 1** focuses an input field, and pressing **Ctrl + 2** opens an alert.

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Keyboard Shortcuts for Actions</title>

<style>

body {

font-family: Arial, sans-serif;

padding: 20px;

background-color: #f4f4f4;

}

input {

padding: 10px;

font-size: 16px;

margin-top: 20px;

}

</style>

</head>

<body>

<h1>Try Keyboard Shortcuts</h1>

<p>Press "Ctrl + 1" to focus on the input field.</p>

<p>Press "Ctrl + 2" to see an alert message.</p>

<input type="text" id="inputField" placeholder="Focus me with Ctrl + 1">

<script>

// Event listener for keyboard shortcut actions

document.addEventListener('keydown', function(e) {

// Check for Ctrl + 1 to focus the input field

if (e.ctrlKey && e.key === '1') {

document.getElementById('inputField').focus();

}

// Check for Ctrl + 2 to display an alert

if (e.ctrlKey && e.key === '2') {

alert("You pressed Ctrl + 2!");

}

});

</script>

</body>

</html>

**Explanation:**

1. **Ctrl + 1**: When the user presses **Ctrl + 1**, the focus is set to the input field (inputField).
2. **Ctrl + 2**: When **Ctrl + 2** is pressed, an alert box pops up.

**3. Handling Function Keys (F1, F2, etc.)**

You can also use the **function keys** (like F1, F2) for keyboard shortcuts.

**Example 3: Function Key Shortcuts (F1, F2)**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Function Key Shortcuts</title>

<style>

body {

font-family: Arial, sans-serif;

padding: 20px;

background-color: #f4f4f4;

}

p {

font-size: 18px;

}

</style>

</head>

<body>

<h1>Function Key Shortcuts Example</h1>

<p>Press F1 to show a help message, and F2 to change the background color.</p>

<script>

document.addEventListener('keydown', function(e) {

// F1 Key for Help Message

if (e.key === 'F1') {

e.preventDefault(); // Prevent default help dialog

alert('This is your help message!');

}

// F2 Key for Changing Background Color

if (e.key === 'F2') {

document.body.style.backgroundColor = "#add8e6"; // Change background to light blue

}

});

</script>

</body>

</html>

**Explanation:**

1. **F1**: When **F1** is pressed, it shows a help message (alert).
2. **F2**: When **F2** is pressed, the background color changes to light blue.

**Conclusion:**

While **HTML** and **CSS** are not designed to handle keyboard shortcuts directly, **JavaScript** provides an easy way to detect key presses and bind them to actions or events on the webpage. With JavaScript, you can create keyboard shortcuts for various interactions like form focus, opening a dialog, or changing styles on the page.