codingninjas

Class 4:Developing Food Item Details Page & Recommendation Section

SESSION OVERVIEW:

By the end of this session, students will be able to:

- Understand the layout and structural requirements for developing a food item details page and its accompanying recommendation section, focusing on displaying key product information and related suggestions.
- Design a visually appealing and responsive food item details page using HTML and CSS, ensuring an organized presentation that includes sections for product images, descriptions, prices, and call-to-action buttons.
- Style the product details and recommendation components to maintain a clear and consistent look that enhances readability and user interaction across different screen sizes.
- Apply CSS techniques such as flexbox or grid to align and arrange elements within the details and recommendation sections effectively.
- Recognize the significance of a well-structured details page and recommendation section in user interface design, improving the overall browsing and selection experience for users.

SESSION TOPICS: Developing the Product Details Page with Recommendation Section

Understanding Product Details Page Layout:

- **Product details section:**
  - Understand the significance of structuring product details for a clear, user-friendly experience.
  - Include key elements such as product images, name, price, description, and rating for comprehensive item representation.
- **Recommendation Section Integration:**
  - Identify strategies for displaying related items or recommendations to enhance user engagement and browsing.

Designing a Responsive Layout:

- **Layout Techniques:**
  - Explore layout options using CSS Flexbox and Grid to structure the product details and recommendation sections effectively.
  - Ensure that the product details and recommendations adapt seamlessly to various screen sizes for an optimal viewing experience.

- **Responsiveness:**
    - Use responsive design techniques to make the page functional across desktops, tablets, and mobile devices.

Styling with CSS:

- **Product Details and Recommendation Styling:**
    - Use CSS to style sections for consistent and appealing visuals. Include features like borders, shadows, and hover effects for interactive user feedback.
- **Interactive Elements:**
    - Incorporate animations and hover transitions (e.g., scaling images or buttons) to engage users.

Enhancing User Interaction with Animations and Responsive Styling with Media Queries:

- **Applying Media Queries:**
    - Learn how to use media queries to adjust styles dynamically and ensure that both the product details and recommendations are visually appealing on all devices.
- **Layout Customization:**
    - Modify item layouts for different screen sizes, such as stacking or reordering elements on smaller screens for better usability.
- **CSS Keyframes and Transitions:**
    - Implement animations, such as `@keyframes`, for fade-in effects to create a smooth and engaging entry for the product details page.
- **Hover Effects:**
    - Use CSS transitions to create smooth hover effects that enhance the visual appeal of images and buttons.

Layout of the Product detail page:

Let's break down the implementation of a restaurant menu app product details page into the specified subtopics: basic Layout of the product details page: Navbar, Main Section, recommendation section and Footer.

1.HTML Basics:

First, let's create the basic HTML structure for our product details page. HTML structure that defines the overall layout of the product details.

HTML Code

```
<section class="product-details">

   <div class="product-image-container">
```

```
      <img src="https://placehold.co/400x500?text=ProductImage" alt="Product
Image" class="product-image">

    </div>


    <div class="product-info">

        <div>


            <h1 class="product-name">Product Name</h1>

            <p class="product-rating">★★★★☆ (4.0)</p>

        </div>


      <p class="product-description">

          This is a detailed description of the product, highlighting its key
features and unique qualities. Perfect for anyone looking for quality and value.

        </p>

        <div class="product-cta">


            <p class="product-price">$49.99</p>

            <div>


                <button class="add-to-cart-btn">Add to Cart</button>

                <button class="add-to-cart-btn">Order it</button>

            </div>
```

```
        </div>
    </div>
  </section>
```

CSS Style

```
/* Product Details Section */
.product-details {
    display: flex;
    max-width: 1000px;
    margin: 40px auto;
    padding: 20px;
    background-color: #fff;
    border-radius: 10px;
    box-shadow: 0 8px 20px rgba(0, 0, 0, 0.1);
    gap: 20px;
    animation: fadeIn 0.5s ease-in-out;
  }
```

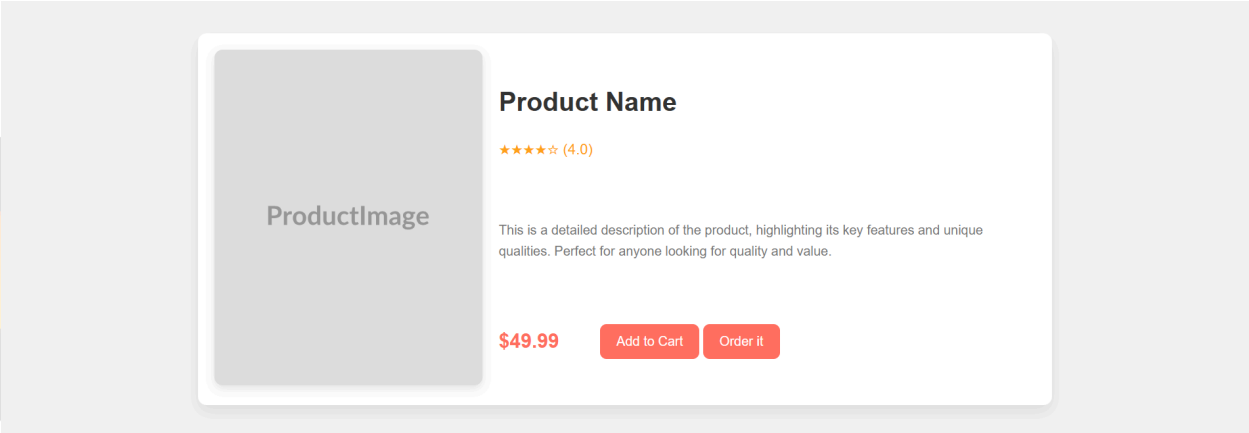
```css
.product-image-container {
  flex: 1;
}

.product-image {
  width: 100%;
  border-radius: 10px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
  transition: transform 0.3s ease;
}

.product-image:hover {
  transform: scale(1.05);
}

.product-info {
  flex: 2;
  display: flex;
  flex-direction: column;
  justify-content: space-around;
  gap: 10px;
}
```

```
.product-name {
  font-size: 2rem;
  /* margin-bottom: 0px; */
  color: #333;
  /* margin-bottom: 5px; */
}

.product-rating {
  font-size: 1.1rem;
  color: #ffa41b;
  /* margin-top: 0px; */
  margin-bottom: 15px;
}

.product-description {
    font-size: 1rem;
    color: #7a7a7a;
    line-height: 1.6;
    margin: 10px 0;
  }
  .product-price {
    font-size: 1.5rem;
```

```
    color: #ff6f61;
    font-weight: bold;
    /* margin: 10px 0; */
  }

.product-cta{
  display: flex;
  justify-content: flex-start;
  gap: 50px;
  align-items: center;
}

.add-to-cart-btn {
  background-color: #ff6f61;
  color: #fff;
  border: none;
  padding: 12px 20px;
  font-size: 1rem;
  cursor: pointer;
  border-radius: 8px;
  transition: transform 0.2s, background-color 0.3s;
}
```

```
.add-to-cart-btn:hover {
  background-color: #e55a4f;
  transform: translateY(-2px);
}

@keyframes fadeIn {
  from {
    opacity: 0;
    transform: translateY(20px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}
```

Output:



Explanation:

`.product-details`: A flex container to layout the product section with shadow and rounded corners for style.

`.product-image`: Styled to fill its container, with hover scaling for interactivity.

`.product-info`: Flex container with spacing for product details.

`.product-name`, `.product-rating`, `.product-description`, `.product-price`: Styled to represent the product's name, rating, details, and price.

`.add-to-cart-btn`: Button with hover and transition effects for user interaction.

`.product-cta`: Arranges the call-to-action items flexibly.

Recommendation Section:

Styling and functionality of our recommendation section using CSS.

HTML CODE :

```
  <!-- Recommended Section (for additional products) -->
  <div class="Recommended-section">
    <div class="Recommended-header">
      <h3>Recommended</h3>
    </div>
    <div class="Recommended-items">
      <div class="recommended-card">
        <img src="https://placehold.co/200x100?text=BestSeller" alt="Best Seller
Image">
```

codingninjas

```
        <div class="recommended-card-content">
          <h4>Best Seller Item 1</h4>
          <p>Description of the best seller item.</p>
          <span>Price: <strike class="strike-price">$10.99</strike> $8.99 <span
style="color:rgb(243, 57, 57); font-size: 13px;">10% off</span></span>
        </div>
        <div class="">
          <button class="cta-button">Add to Cart</button>
        </div>
      </div>
      <div class="recommended-card">

        <img src="https://placehold.co/200x100?text=BestSeller" alt="Best Seller
Image">
        <div class="recommended-card-content">
          <h4>Best Seller Item 1</h4>
          <p>Description of the best seller item.</p>
          <span>Price: <strike class="strike-price">$10.99</strike> $8.99 <span
style="color:rgb(243, 57, 57); font-size: 13px;">10% off</span></span>
        </div>
        <div class="">
          <button class="cta-button">Add to Cart</button>
        </div>
      </div>
      <div class="recommended-card">
        <img src="https://placehold.co/200x100?text=BestSeller" alt="Best Seller
Image">
        <div class="recommended-card-content">
          <h4>Best Seller Item 1</h4>
          <p>Description of the best seller item.</p>
          <span>Price: <strike class="strike-price">$10.99</strike> $8.99 <span
style="color:rgb(243, 57, 57); font-size: 13px;">10% off</span></span>
        </div>
        <div class="">
          <button class="cta-button">Add to Cart</button>
        </div>
      </div>
      <div class="recommended-card">
```

```
        <img src="https://placehold.co/200x100?text=BestSeller" alt="Best Seller
Image">
        <div class="recommended-card-content">
          <h4>Best Seller Item 1</h4>
          <p>Description of the best seller item.</p>
          <span>Price: <strike class="strike-price">$10.99</strike> $8.99 <span
style="color:rgb(243, 57, 57); font-size: 13px;">10% off</span></span>
        </div>
        <div class="">
          <button class="cta-button">Add to Cart</button>
        </div>
      </div>
      <div class="recommended-card">
        <img src="https://placehold.co/200x100?text=BestSeller" alt="Best Seller
Image">
        <div class="recommended-card-content">
          <h4>Best Seller Item 1</h4>
          <p>Description of the best seller item.</p>
          <span>Price: <strike class="strike-price">$10.99</strike> $8.99 <span
style="color:rgb(243, 57, 57); font-size: 13px;">10% off</span></span>
        </div>
        <div class="">
          <button class="cta-button">Add to Cart</button>
        </div>
      </div>

      </div>
      </div>
```

CSS Style:

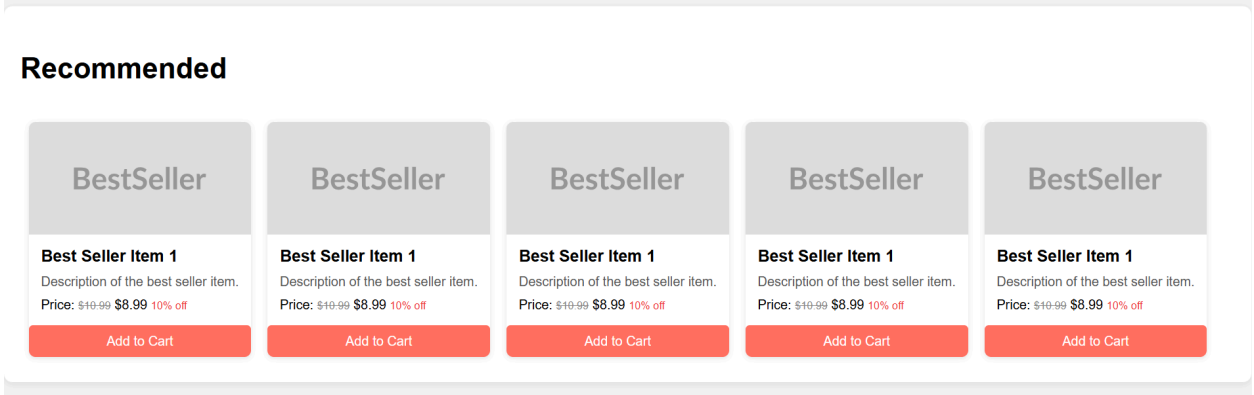
```
  /* Recommended Section */
  .Recommended-section {
    background-color: #ffffff;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
    margin-bottom: 20px;
  }
```

codingninjas

```css
.Recommended-header {
  /* text-align: center; */
  font-size: 30px;
  margin-bottom: 20px;
}
.Recommended-items {
  display: flex;
  flex-wrap: wrap;
  /* margin-left: 30px; */
  /* justify-content: space-between; */
}
.recommended-card {
  background-color: #ffffff;
  border-radius: 8px;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
  margin: 10px;
  transition: transform 0.5s ease-out;
  display: flex;
  flex-direction: column;
  overflow: hidden;
}
.recommended-card:hover {
  transform: scale(1.05);
}
.recommended-card img {
  width: 100%;
  height: auto;
  border-top-left-radius: 8px;
  border-top-right-radius: 8px;
}
.recommended-card-content {
  padding: 15px;
}
.recommended-card-content h4 {
  margin: 0 0 10px;
  font-size: 1.25em;
}
.recommended-card-content p {
  margin: 0 0 10px;
```

```
  color: #555;
}
.strike-price {
  text-decoration: line-through;
  color: #888;
}
.cta-button {
  background-color: #ff6f61;
  color: #ffffff;
  border: none;
  padding: 10px;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s;
  font-size: 1em;
  margin-top: auto; /* Align button to the bottom */
}
.cta-button:hover {
  background-color: #e55a4f;
}
```

Output



Explanation:

Container Styling (`.Recommended-section` and `.Recommended-header`):

- The section has a white background, rounded corners, shadow for subtle depth, and padding to create space inside.
- The header is styled to be prominent with a larger font size.

Item Layout (`.Recommended-items`):

- Uses `flex` layout to arrange items and `flex-wrap: wrap` so items break onto new lines if necessary.

Card Design (`.recommended-card`):

- Each card has a white background, rounded edges, shadow, and a hover effect (`transform: scale(1.05)`) for interaction.
- Cards use a `flex` column layout to arrange their content vertically.

Image Styling (`.recommended-card img`):

- Images are set to take the full width with rounded top corners to match the card's shape.

Card Content (`.recommended-card-content`):

- Contains title (`h4`), description (`p`), and price details styled for clarity.
- The `.strike-price` style indicates a previous price with a line-through effect.

CTA Button (`.cta-button`):

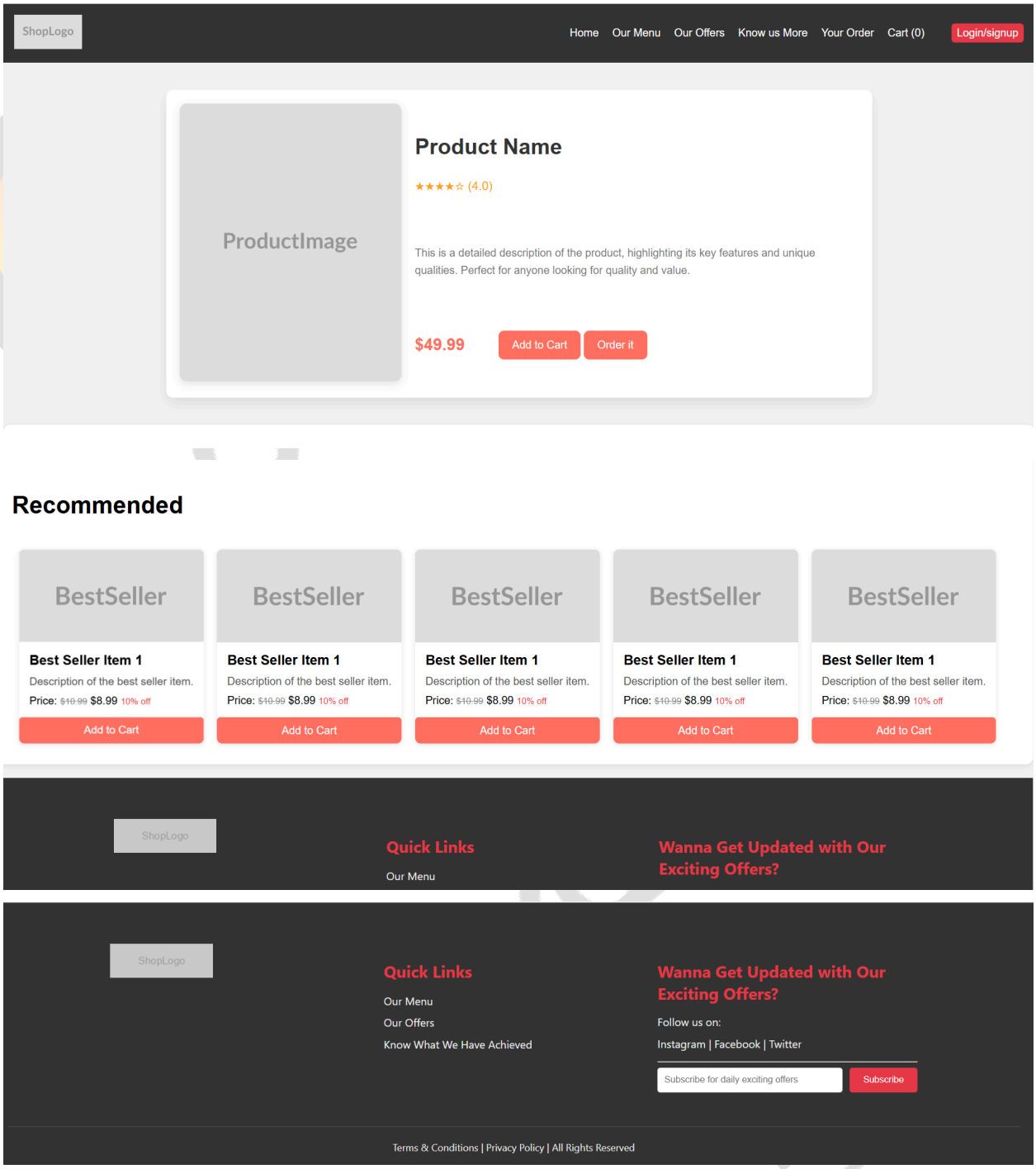
- Styled with a background color that changes on hover and rounded edges.
- Positioned at the bottom using `margin-top: auto` for better card structure.

Note:
Now Add the Header section and footer section code to complete your product page. Also, include media queries to ensure your page is responsive and adapts well to different screen sizes. This way, you can maintain a consistent design and functionality across various devices.

Final Output:

Productdetails.css

productdetails.html


ShopLogo
Home
Our Menu
Our Offers
Know us More
Your Order
Cart (0)
Login/signup
ProductImage
Product Name
★★★★☆ (4.0)
This is a detailed description of the product, highlighting its key features and unique qualities. Perfect for anyone looking for quality and value.
$49.99
Add to Cart
Order it
Recommended
BestSeller
Best Seller Item 1
Description of the best seller item.
Price: $10.99 $8.99 10% off
Add to Cart
ShopLogo
Quick Links
Our Menu
Wanna Get Updated with Our Exciting Offers?
ShopLogo
Quick Links
Our Menu
Our Offers
Know What We Have Achieved
Wanna Get Updated with Our Exciting Offers?
Follow us on:
Instagram | Facebook | Twitter
Subscribe for daily exciting offers
Subscribe
Terms & Conditions | Privacy Policy | All Rights Reserved

codingninjas

Concept Indepth:

Animation in CSS

Animations in CSS are a powerful way to add interactivity and visual interest to a webpage. They help bring elements to life and can create a more engaging user experience.

Below, we'll break down the basics of CSS animations, how to create them, and provide an example for better understanding.

---

1. What is CSS Animation?

CSS animations allow you to gradually change the style of an element from one state to another. These changes can include properties like color, size, position, or any other CSS property that can be transitioned.

Why Use CSS Animations?

- To make a page visually appealing.
- To draw attention to important elements.
- To create smooth, interactive experiences without needing JavaScript.

---

2. How CSS Animations Work

Animations in CSS are created using two main components:

- Keyframes: Define what styles the element will have at different times during the animation.
- Animation Properties: Apply the animation to an element and control how it behaves (e.g., duration, timing, number of repetitions).

Key Properties Explained:

- **`animation-name`:** Specifies the name of the `@keyframes` block.
- **`animation-duration`:** Sets how long the animation should take to complete one cycle.
- **`animation-timing-function`:** Determines the speed curve of the animation (e.g., `ease`, `linear`, `ease-in-out`).
- **`animation-iteration-count`:** Specifies how many times the animation should repeat (e.g., `1`, `infinite`).

---

3. Example of a Simple Animation

Imagine you have a button that you want to animate so that it scales up when hovered over. Here's how to create that effect:

HTML Structure:

html

```
<button class="animate-btn">Hover Over Me!</button>
```

CSS Code:

```
  /* Keyframes to scale the button */
@keyframes scaleUp {
  0% {
    transform: scale(1);
  }
  100% {
    transform: scale(1.2);
  }
}
/* Apply the animation to the button */
.animate-btn {
  background-color: #3498db;
  color: #fff;
  border: none;
  padding: 15px 30px;
  font-size: 16px;
  border-radius: 5px;
  cursor: pointer;
  animation-name: scaleUp;
  animation-duration: 0.5s;
  animation-timing-function: ease-in-out;
  animation-iteration-count: 1;
}

/* Trigger animation on hover */
.animate-btn:hover {
  animation-name: scaleUp;
}
```

Explanation:

- The `@keyframes` block defines the animation steps. In this case, the button starts at its original size (`scale(1)`) and scales up to `scale(1.2)`.
- The `animation-duration` is set to `0.5s`, so the animation completes in half a second.
- The `animation-timing-function` is `ease-in-out`, making the animation start and end smoothly.
- The `animation-iteration-count` of `1` ensures the animation only runs once when triggered.

---

4. Understanding `@keyframes`

The `@keyframes` rule is essential for defining the stages of your animation. Here's how to use it effectively:

Structure of `@keyframes`:

css

```
@keyframes animationName {
  0% {
    /* Initial state */
  }
  50% {
    /* Midway state */
  }
  100% {
    /* Final state */
  }
}
```

---

5. Tips for Effective Animations

- Keep It Simple: Overusing animations can make your site feel cluttered.
- Use Meaningfully: Ensure animations add value and are not just for decoration.
- Test Performance: Complex animations can impact site performance, so test them on different devices.

Understanding `@keyframes` in CSS Animation

What Are Keyframes?

`@keyframes` is a rule in CSS used to create animations. It allows you to define how an element should look at specific points during the animation timeline. This means you can specify different styles at different percentages of the animation's duration.

How Does It Work?

You define an animation with keyframes by specifying a series of "frames" (points) at which the element's properties will change. The browser smoothly transitions between these points, creating a continuous animation.

Basic Example:

Imagine you want to make a box move from the left to the center of the screen and change its color as it moves:

```
@keyframes moveAndColor {
  0% {
    transform: translateX(0);
    background-color: red;
  }
  50% {
    transform: translateX(50%);
    background-color: yellow;
  }
  100% {
    transform: translateX(100%);
    background-color: green;
  }
}
```

```
.box {
  width: 100px;
  height: 100px;
  animation: moveAndColor 2s ease-in-out;
}
```

Explanation:

- **0%**: The element starts at its initial position (`translateX(0)`) with a red background.
- **50%**: Halfway through, the element moves to the middle (`translateX(50%)`) and turns yellow.
- **100%**: At the end, it reaches the final position (`translateX(100%)`) and becomes green.

The `animation` property in `.box` applies the animation for 2 seconds (`2s`) with an `ease-in-out` easing for a smooth start and end.

Example from Code:

```
@keyframes fadeIn {
    from {
      opacity: 0;
      transform: translateY(20px);
    }
    to {
      opacity: 1;
      transform: translateY(0);
    }
  }
```

- **`from`:** The element is initially invisible (`opacity: 0`) and shifted 20 pixels down.
- **`to`:** The element becomes fully visible (`opacity: 1`) and returns to its original position.

Application: Adding `animation: fadeIn 0.5s ease-in-out;` to an element will make it fade in smoothly within 0.5 seconds.

Why Use Keyframes?

Keyframes make it easy to create complex animations that enhance user experience. They allow for:

- Transitions between multiple states.
- Custom control over timing and effects.
- Flexibility in defining precise styles at different points in time

Concept Brush-ups for Product Details Page Design:

HTML Basics:

1. Explain the Difference Between `<div>` and `<span>` in HTML
   `<div>` is a block-level element typically used to structure the page layout, while `<span>` is an inline element used for styling parts of text within a block-level element.
2. The Purpose of `<meta>` Tags
   The `<meta>` tag provides metadata about the document, like its character encoding (`<meta charset="UTF-8">`), viewport settings for mobile responsiveness, or descriptions that help with SEO.

CSS Grid and Flexbox for Layouts:

1. Advantages of CSS Grid
   CSS Grid is superior for complex, two-dimensional layouts. For example, it allows for precise placement of elements like a product image, product description, and price in separate rows and columns, ensuring they align perfectly across different screen sizes.
2. CSS Grid Lines and Tracks
   CSS Grid's concept of grid lines and grid tracks allows for precise placement of elements within the grid. Grid lines are the vertical and horizontal dividing lines, while tracks are the space between them. These help define the layout structure of product pages effectively.
3. Flexbox vs. Grid
   While CSS Grid excels at two-dimensional layouts, Flexbox is best for simpler, one-dimensional arrangements like a product image gallery or pricing section where items should be laid out in a row or column.

CSS Animations and Interactivity:

- **Hover Animations for Buttons**
  Adding a hover effect to Add to Cart buttons (e.g., changing the background color or scaling the button) enhances interactivity. Example:

```
button:hover {
  background-color: #f8f8f8;
  transform: scale(1.05);
  transition: transform 0.3s ease-in-out;
}
```

- **Image Transitions for Product Images**
  Smooth transitions on hover can be used for product images, such as a zoom effect when hovering over a product image.
  Example:

```
.product-image {
  transition: transform 0.5s ease;
}
.product-image:hover {
  transform: scale(1.1);
}
```

- **CSS Animations with Keyframes**
  For more complex interactions, CSS animations using @keyframes can animate elements across a timeline, such as rotating a product image or creating a smooth fade-in effect when the page loads.
  Example:

```
@keyframes fadeIn {
  0% { opacity: 0; }
  100% { opacity: 1; }
}
.product-details {
  animation: fadeIn 1s ease-out;
}
```

CSS Animation: Key Concepts and Trends (2024)

1. Performance Optimization in CSS Animations

- GPU Acceleration: For smoother animations, it's recommended to animate properties like transform and opacity, which are processed by the GPU. This avoids performance bottlenecks that occur when properties like width, height, or left are animated, which trigger expensive repaints in the browser
- Avoid Layout Thrashing: Animations that affect layout should be avoided when possible. Using properties that do not trigger layout recalculations helps ensure smoother performance, especially on mobile devices.

2. CSS Shorthand for Keyframes

- Using @keyframes allows for more concise, reusable animations. Complex animations can be broken down into multiple steps and triggered by events like hover or scroll.

Example of a simple slide-in animation:

```
@keyframes slidein {
  0% { transform: translateX(-100%); }
  100% { transform: translateX(0); }
}
```

3. Interactive Animations

- CSS animations are increasingly used to create interactive web experiences. Animations triggered by pseudo-classes like :hover, :focus, and :active allow elements to respond dynamically to user inputs.
- Scroll-based animations are gaining popularity, allowing elements to animate as users scroll through the page. Libraries like AOS (Animate On Scroll) help implement these types of animations with minimal JavaScript.

4. CSS Variables in Animations

- The use of CSS custom properties (variables) is on the rise, enabling dynamic changes to values in response to user interactions or media queries. This makes animations more flexible and easier to maintain.

Example of CSS variable usage in animation:

```
  :root {
```

```
  --main-color: #f00;
}

.animate {
  animation: colorChange 2s infinite;
}

@keyframes colorChange {
  0% { color: var(--main-color); }
  100% { color: #00f; }
}
```

5. SVG Animations

- SVG (Scalable Vector Graphics) animations are becoming more common, especially for logos, icons, and illustrations. SVG allows for high-quality, resolution-independent animations.
- Techniques like animating stroke-dasharray and stroke-dashoffset are used to animate paths and lines in a dynamic way
  .

6. Motion Path Animation

- The motion-path CSS property lets elements follow a custom path defined by an SVG path or a simple geometric shape, making animations more natural and organic.

Example:

```
.move {

  animation: moveAlongPath 4s infinite;

  motion-path: path('M0,0 C0,50 100,50 100,0');

}
```

7. Micro-Animations for UX

- Micro-animations (small, subtle animations) are being widely used to enhance user experience. These can be animations like buttons bouncing on hover, or input fields showing validation feedback. The key is to keep them light and non-intrusive while still improving usability.
- These are often used in combination with hover or focus states.

8. New CSS Features and Tools

- Subgrid: A part of CSS Grid, allows elements within a grid container to align more easily with the parent grid's rows and columns, improving layout consistency.
- @property: This CSS Houdini feature lets developers register CSS properties that can be animated, making it possible to create more complex and customizable animations**.**
- **Animista:** A tool for experimenting with pre-built CSS animations and customizing them for specific use cases**.**

Additional Interview Tips for Product Details Pages:

Master Responsive Design
Ensure product details pages look great on all devices by using media queries and fluid layouts. Example for mobile-first design:

```
@media (max-width: 768px) {
  .product-container {
    flex-direction: column;
  }
}
```

1. Use Browser DevTools
   Inspect and manipulate your page styles live to experiment with layout changes, which is essential for debugging and fine-tuning the design.
2. Accessibility Considerations
   Ensure the product details page is accessible to all users by using proper ARIA attributes (e.g., `aria-labelledby`, `aria-hidden`) and testing keyboard navigation for all interactive elements.

Follow tech blogs like Smashing Magazine, CSS-Tricks, and Dev.to to stay current with industry trends.