

1. Find the Output

Problem

Submissions

Doubts

✓ Find the Output

Easy • Score: 20/20

Problem statement

[Send feedback](#)

Analyse the given HTML and CSS code and Find the correct output:

HTML:

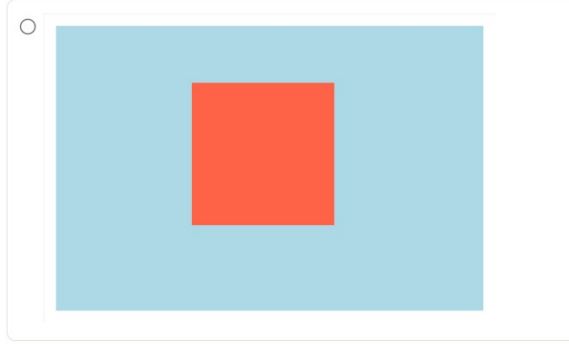
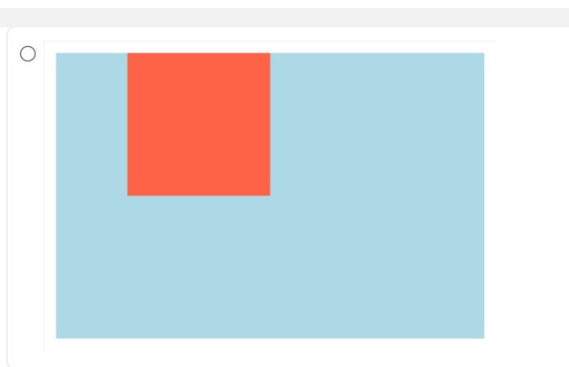
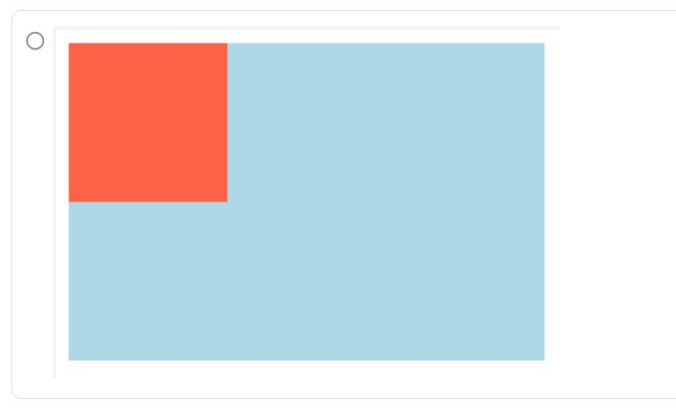
```
<div class="container">
  <div class="box">
  </div>
</div>
```

CSS:

```
.container {
  position: relative;
  width: 300px;
  height: 200px;
  background-color: lightblue;
}

.box {
  position: absolute;
  top: 50px;
  left: 50px;
  width: 100px;
  height: 100px;
  background-color: tomato;
}
```

Options: Pick one correct answer from below



Solution description

- The .container div has position: relative; which means any positioned elements within it will be positioned relative to it.
- The .box div has position: absolute; along with top: 50px; and left: 50px; This means it will be positioned 50px from the top and 50px from the left edge of its closest positioned ancestor, which in this case is the .container div.
- Therefore, the .box div will be positioned 50px from the top and 50px from the left edge of the .container div.

2. Find the Correct Code

Problem Submissions Doubts

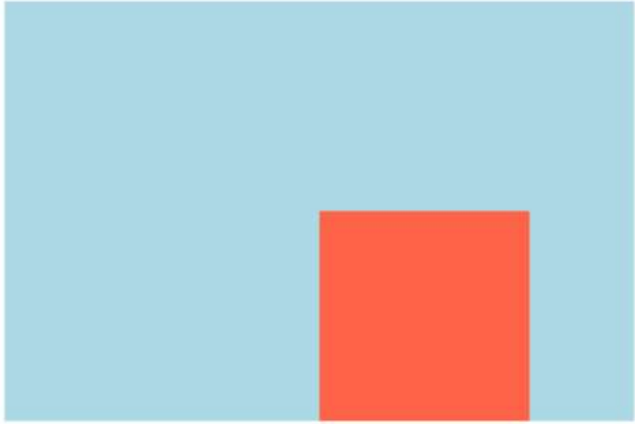
✓ Find the Correct Code

Easy • Score: 20/20

Problem statement [Send feedback](#)

Find the correct code by analysing the given output and the HTML code:

Output:



HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Desired Layout</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <div class="box"></div>
  </div>
</body>
</html>
```

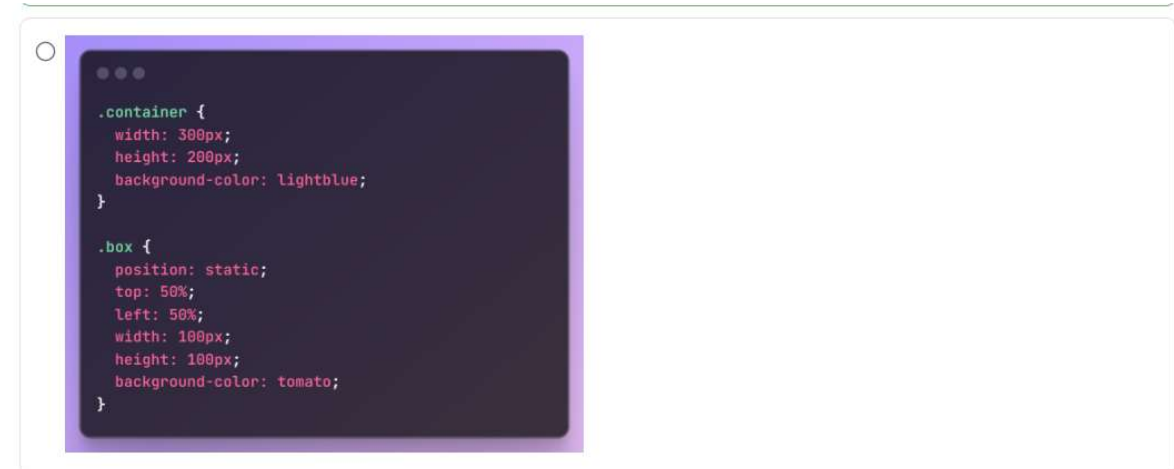
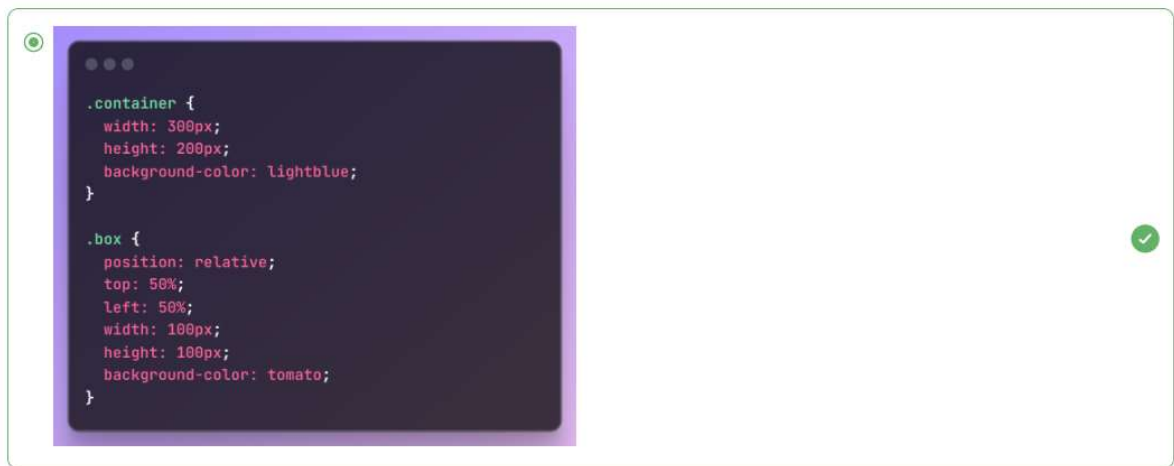
Options: Pick one correct answer from below

☐

```
.container {  
  width: 300px;  
  height: 200px;  
  background-color: lightblue;  
}  
  
.box {  
  position: absolute;  
  top: 50%;  
  left: 50%;  
  margin-top: -50px;  
  margin-left: -50px;  
  width: 100px;  
  height: 100px;  
  background-color: tomato;  
}
```

☐

```
.container {  
  width: 300px;  
  height: 200px;  
  background-color: lightblue;  
}  
  
.box {  
  margin-top: 50px;  
  margin-left: 100px;  
  width: 100px;  
  height: 100px;  
  background-color: tomato;  
}
```





Solution description

- In Option C, the .box element has the CSS property position: relative; This property tells the browser to position the .box element relative to its normal position in the document flow.
- So, even though the .box element is inside the .container element, setting position: relative; allows us to control its position based on where it would naturally appear on the page.

3. Position Property

Problem Submissions Doubts

 **Position Property** 

Easy • Score: 20/20

Problem statement [Send feedback](#)

Consider a scenario where you want to create a navigation bar that stays fixed at the top of the page even when scrolling. Which positioning value would you use?

Options: Pick one correct answer from below

☐ Static

☒ Fixed



☐ Absolute

☐ Relative

Solution description

Fixed value ensures the element remains in the same position relative to the viewport, providing a consistent presence at the top of the screen regardless of scrolling activity.

4. Find the Correct Code II

Problem

Submissions

Doubts

 Find the Correct Code II



Easy • Score: 20/20

Problem statement

[Send feedback](#)

Choose the correct code based on the given output and it's HTML:

Output:

Sticky Box

HTML:

```
<div class="sticky-box">
  Sticky Box
</div>
<div class="content">
  <!-- Placeholder content to make scrolling
visible -->
</div>
```

Options: Pick one correct answer from below



```
.sticky-box {
  position: sticky;
  top: 50px;
  background-color: #e74c3c;
  color: #fff;
  padding: 10px;
}
.content {
  height: 2000px;
}
```



```
.sticky-box {
  position: fixed;
  top: 50px;
  background-color: #e74c3c;
  color: #fff;
  padding: 10px;
}
.content {
  height: 2000px;
}
```

```
.sticky-box {
  position: relative;
  top: 50px;
  background-color: #e74c3c;
  color: #fff;
  padding: 10px;
}
.content {
  height: 2000px;
}
```

```
.sticky-box {
  position: absolute;
  top: 50px;
  background-color: #e74c3c;
  color: #fff;
  padding: 10px;
}
.content {
  height: 2000px;
}
```

Solution description

The `.sticky-box` class has the property `position: sticky; set`. This property allows the element to act like a relatively positioned element until it reaches a specified scroll point, after which it becomes fixed, relative to the viewport.

5. Z-Index

Problem

Submissions

Doubts

✓ Z-Index



Easy • Score: 20/20

Problem statement

[Send feedback](#)

You have two overlapping elements. Element A has a z-index: 10, and element B has a z-index: of 5. In the final render, which element will be visible on top?

Options: Pick one correct answer from below

☒ Element A



☐ Element B

☐ Both elements will be partially visible.

☐ The order depends on the element type.

Solution description

- Elements are compared based on their z-index values.
- Since element A has a higher value (10), it takes precedence and is positioned visually on top of element B (with a value of 5).

6. Error Analysis

Problem

Submissions

Doubts

✓ Error Analysis



Easy • Score: 20/20

Problem statement

[Send feedback](#)

A developer is designing a webpage layout where he wants an image (#image) to overlap a paragraph (#paragraph). Despite these styles, he finds that the paragraph is covering the image. Which CSS property could be causing this issue?

```
#image {  
  position: absolute;  
  top: 50px;  
  left: 50px;  
  z-index: -1;  
}  
  
#paragraph {  
  position: relative;  
  z-index: 1;  
}
```

Options: Pick one correct answer from below

☐ position: absolute on the image is causing it to be positioned relative to the viewport rather than the paragraph.

☒ z-index: -1 on the image is placed below the stacking context of the paragraph.



☐ position: relative to the paragraph is creating a new stacking context that overrides the image's z-index.

☐ Both B and C are causing the issue due to conflicting stacking contexts and z-index values.

Solution description

The z-index property determines the stack order of positioned elements. In this case, the image has a negative z-index, placing it behind the stacking context of the paragraph, which has a higher z-index. This causes the paragraph to cover the image.

7. Float Property

Problem

Submissions

Doubts

✓ Float Property



Easy • Score: 20/20

Problem statement

[Send feedback](#)

Which scenario among the following necessitates the utilization of the float property in CSS?

Options: Pick one correct answer from below

☒ Aligning text alongside a floated image within a constrained container.



☐ Achieving a staggered layout where elements overlap for a parallax effect.

☐ Creating a dropdown menu with animated transitions upon hover.



☐ Positioning a fixed navigation bar at the top of the viewport during scrolling.

Solution description

The float property in CSS is specifically utilized to position elements horizontally within their parent container. Option A exemplifies a typical use case where text needs to wrap around a floated image within a limited space. By applying the float property to the image, the text can align beside it, creating a visually cohesive layout. This scenario succinctly demonstrates the primary function of the float property in CSS for aligning elements horizontally within a constrained area.

8. Overflow: Visible

Problem Submissions Doubts

 **Overflow: Visible** 

Easy • Score: 20/20

Problem statement [Send feedback](#)


If an element has the CSS rule **overflow: visible** what will happen when the content overflows?

Options: Pick one correct answer from below

☐ The content will be hidden.

☐ Scrollbars will appear to navigate the overflowed content.

☐ The overflow will be automatically resized to fit within the element.



☒ The content will be displayed outside the element's boundaries. 

Solution description

The overflow: visible; property allows the content to overflow the element's box and be displayed outside its boundaries. This may result in overlapping with other elements on the page.

9. Error Analysis II

Problem Submissions Doubts

 **Error Analysis II** 

Easy • Score: 20/20

Problem statement [Send feedback](#)

In the given scenario, the content inside the #content div is not fully visible due to the CSS properties applied. Your task is to determine which CSS property or combination of properties is causing the content to be hidden. Consider the following options and select the one that best describes why the content is clipped.

You have an HTML and CSS setup as follows:

HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Layout Issue</title>
  <link rel="stylesheet" href="styles.css" />
</head>
<body>
<div id="content">
  <p>This content is clipped due to overflow: hidden. Only part of this
paragraph is visible.</p>
  <p>This second paragraph is also clipped and not visible because overflow:
hidden hides content that exceeds the container's bounds.</p>
</div>
</body>
</html>
```

Incorrect Output:

This content is clipped due to
overflow: hidden. Only part of
this paragraph is visible.

This second paragraph is also

Expected Output:

This content is clipped due to
overflow: hidden. Only part of
this paragraph is visible.

This second paragraph is also
clipped and not visible because
overflow: hidden hides content
that exceeds the container's
bounds.

Options: Pick one correct answer from below

- ☐ The width property of #content is too narrow to display the full content, causing it to be clipped.
- ☒ The height property of #content is too small, and the overflow: hidden; property hides any content that exceeds this height. ✓
- ☐ The padding property of #content adds extra space inside the container, which causes the content to be hidden
- ☐ The background-color property of #content is too light, making it difficult to see the content.

Solution description

The height property of the #content div is set to 20vh, which is a relative height based on the viewport height. If the content inside the div exceeds this height, overflow: hidden; ensures that any overflow beyond this height is clipped and not visible. This is why part of the content is not visible.

10. Writing-Mode

Problem Submissions Doubts

✓ Writing-Mode +

Easy • Score: 20/20

Problem statement

[Send feedback](#)

Which of the following option describes the primary purpose of the writing-mode property in CSS?

Options: Pick one correct answer from below



- ☐ Change the font size of an element.
- ☐ Specify the margins of an element.
- ☒ Control the layout and orientation of text within an element. ✓
- ☐ Set the background color of an element.

Solution description

writing-mode specifically targets the way text is displayed within an element, allowing control over its direction and flow.

11. Text-Direction

Problem Submissions Doubts

 **Text-Direction** 

Easy • Score: 20/20

Problem statement [Send feedback](#)


Which value of the writing-mode property indicates that text is laid out vertically from top to bottom and progresses from right to left?

Options: Pick one correct answer from below

☐ horizontal-tb

☐ vertical-lr

☐ horizontal-rl



☒ vertical-rl 

Solution description

The value "vertical-rl" of the writing-mode property specifies that text is laid out vertically from top to bottom, with each line progressing from right to left along the right side of the container. This is commonly used in languages such as Mongolian. Options: horizontal-tb, horizontal-rl are for horizontal text layout, while vertical-lr is used for vertical text layout with the text progressing from top to bottom along the left side of the container.

12. Object-Fit Property

Problem Submissions Doubts

 **Object-Fit Property** 

Easy • Score: 20/20

Problem statement [Send feedback](#)

If an image with dimensions 1000px × 500px is placed inside a container with **object-fit: cover**, which of the following statements will be true?

Options: Pick one correct answer from below

☐ The image will be stretched to fill the entire container, distorting its aspect ratio.

☒ The image will maintain its aspect ratio and cover the entire container, potentially cropping parts of the image. 

☐ The image will be displayed at its original size, centred within the container.

☐ The image will be scaled down to fit entirely within the container, preserving its aspect ratio.

Solution description

- object-fit: cover; instructs the image to resize while maintaining its aspect ratio (2:1 in this case) and fills the container as much as possible.
- The container has no dimension (it doesn't matter if it's square or rectangular).