DUMPSARENA

HTML5 Application Development

IT Specialist INF-306

Version Demo

Total Demo Questions: 9

Total Premium Questions: 91

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

Topic Break Down

Topic	No. of Questions
Topic 1, Manage the Application Life Cycle	5
Topic 2, Build the User Interface by Using HTML5	28
Topic 3, Format the User Interface by Using Cascading Style Sheets (CSS)	36
Topic 4, Code by Using JavaScript	22
Total	91

QUESTION NO: 1

You are drawing on the canvas defined by the white square.

At which x, y coordinate is the upper-left corner of the filled square?

A. 0,0

B. 0,50

C. 50,0

D. 50,50

ANSWER: D

Explanation:

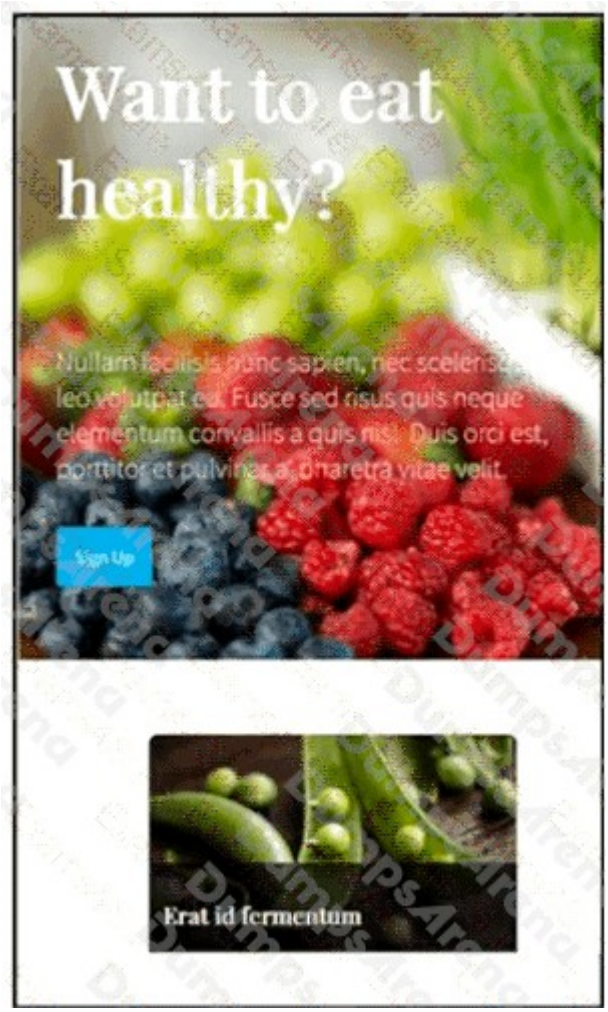
The coordinate **50,50** identifies a point 50 pixels to the right of the canvas's left edge and 50 pixels below its top edge. HTML canvas uses a two-dimensional coordinate system whose origin, `(0,0)`, is at the upper-left corner of the canvas drawing area. Horizontal position is represented by the x-coordinate and increases from left to right; vertical position is represented by the y-coordinate and increases from top to bottom. Therefore, when a filled rectangle is drawn with its upper-left corner inset equally from the top and left sides of the white canvas, its starting position is `(50,50)`. For example, the Canvas 2D API call `fillRect(50, 50, width, height)` begins the rectangle at that exact upper-left coordinate, then extends it to the right and downward according to its width and height. This follows the standard behavior of the `CanvasRenderingContext2D.fillRect()` method, which takes x and y values for the upper-left point of the rectangle. See the MDN fillRect() reference and the HTML Living Standard canvas specification.

QUESTION NO: 2 - (SIMULATION)

The following two images show a web page that uses a responsive layout displayed on a mobile device and on a P

C.

Displayed on a mobile device:



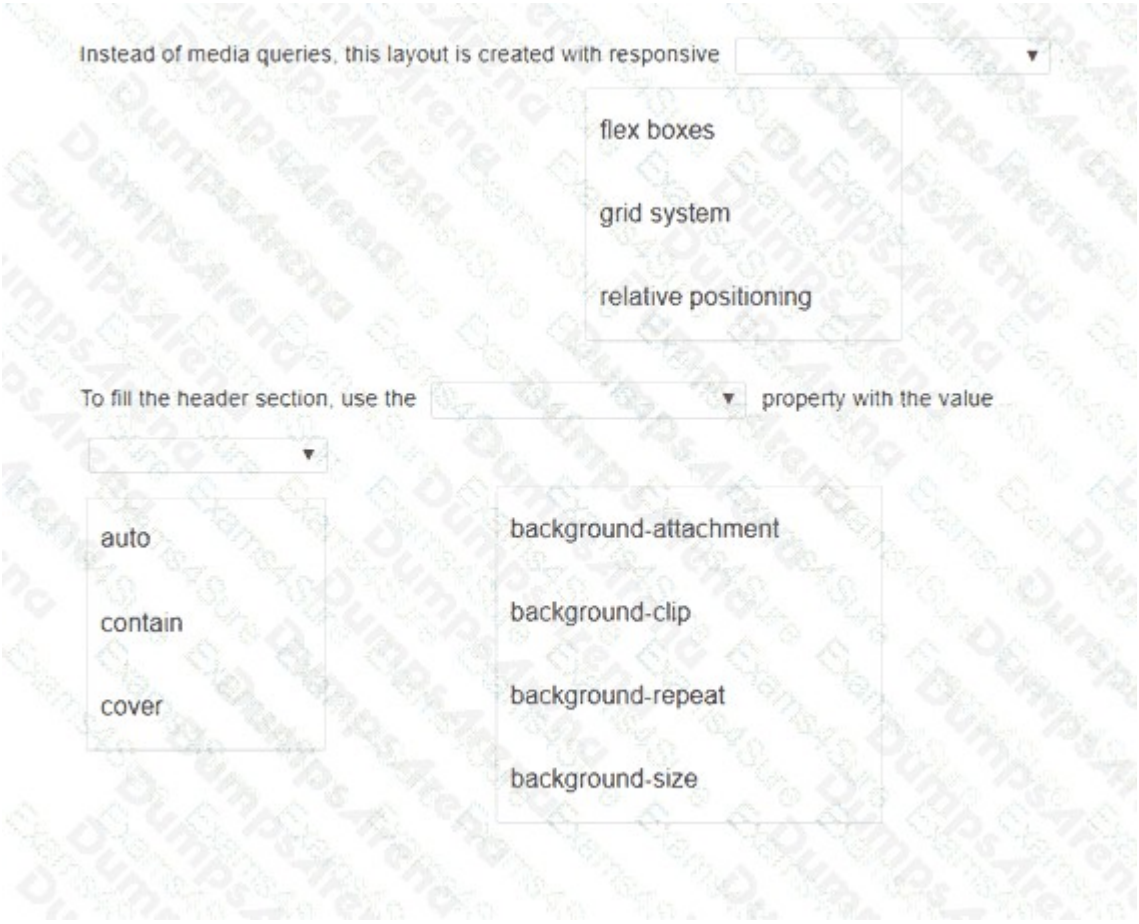
The layout stacks the content vertically.

Displayed on a computer monitor:

The layout displays the content cards horizontally in multiple columns.

Review the images on the left.

Complete the statements by selecting the correct option from each drop-down list.



ANSWER: See the explanation for the answer

Explanation:

Select:

Instead of media queries, this layout is created with responsive **flex boxes**.
To fill the header section, use the `background-size` property with the value `cover`.

The resulting CSS concept is:

```
background-size: cover;
```

Flexbox allows the cards to wrap from a horizontal multi-column desktop arrangement into a vertically stacked mobile arrangement as available space decreases. `background-size: cover` scales the background image to cover the entire header, cropping overflow if necessary rather than leaving blank space.

QUESTION NO: 3 - (SIMULATION)

You are creating a dietary request form for an upcoming conference. The form must include the following elements:

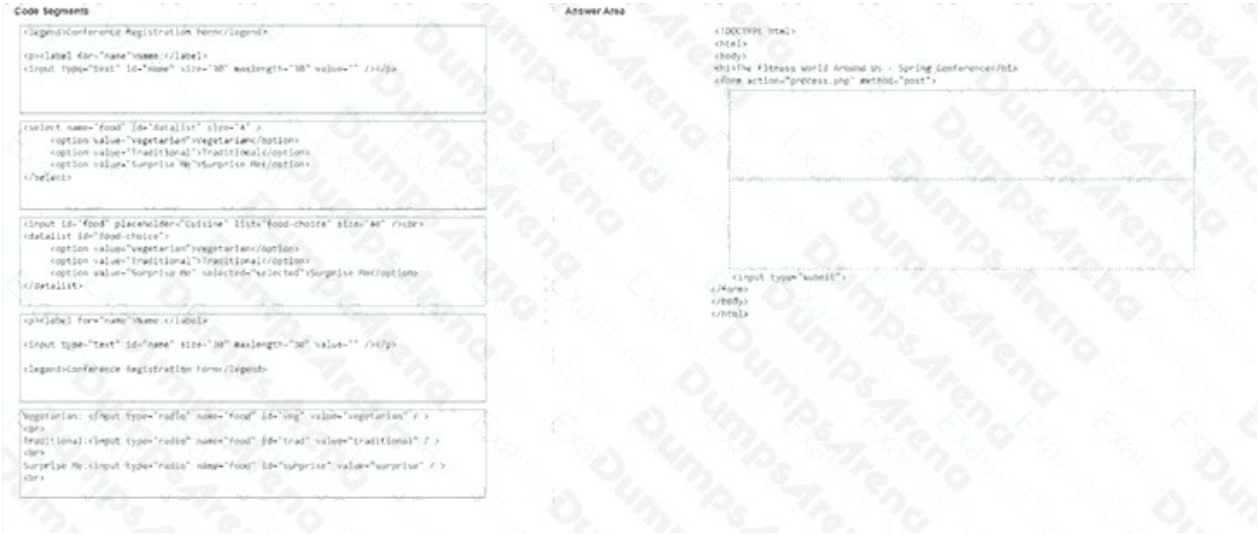
• The title " Conference Registration Form "

• A Name input field for the attendee ' s name

• A list of food options the attendee can select by using the mouse or typing

Complete the code by moving the appropriate code segments from the list on the left to the correct locations on the right. You may use each code segment once, more than once, or not at all.

Note: You will receive partial credit for each correct response.



ANSWER: See the explanation for the answer

Explanation:

Place the code segment containing the form title and Name field in the first blank (immediately after the opening `<form>` tag):

```
<legend>Conference Registration Form</legend>
<p><label for="name">Name:</label>
<input type="text" id="name" size="30" maxlength="30" value="" /></p>
```

Place the `datalist` segment in the second blank, before the Submit input:

```
<input id="food" placeholder="Cuisine" list="food-choice" size="40" /><br>
<datalist id="food-choice">
  <option value="Vegetarian">Vegetarian</option>
  <option value="Traditional">Traditional</option>
  <option value="Surprise Me" selected>Surprise Me</option>
</datalist>
```

The `label` is associated with the Name text field through matching `for="name"` and `id="name"` attributes. The input's `list` attribute connects it to the `datalist`, allowing attendees either to select a suggested food option with the mouse or type a value.

QUESTION NO: 4 - (DRAG DROP)

You need to identify the form elements.

Move the appropriate semantic elements from the list on the left to the correct locations on the right.

Note: You will receive partial credit for each correct response.

ANSWER:

Explanation:

The mapping correctly identifies the semantic purpose of each marked area in the form. The Club Points display is a visual indicator of a value within a defined range. That is the intended use of the HTML `<meter>` element: it represents a scalar measurement, such as points, ratings, storage usage, or a score. The HTML specification describes `<meter>` as a gauge for a known-range value; see the MDN meter element reference.

The Order Status wording is placed as a caption attached to the top edge of its form grouping. A `<legend>` supplies the caption for a fieldset, giving the related controls a meaningful group label. The outlined Shipping Information region is correspondingly a `<fieldset>`, which semantically groups related form content and controls. This grouping structure improves the meaning of the form and provides useful context to assistive technologies. The relationship between these elements is documented in the MDN fieldset reference.

The delivery-frequency control is intended to let a user enter or select from predefined values. A `<datalist>` provides the suggested values for an associated input while retaining input flexibility, which matches this type of delivery-frequency selector. The applicable behavior is described in the MDN datalist reference. Thus, the displayed placements correctly associate the points gauge with `<meter>`, the Order Status caption with `<legend>`, the delivery-frequency suggestions with `<datalist>`, and the Shipping Information grouping box with `<fieldset>`.

QUESTION NO: 5

Which two code segments declare a JavaScript method? Choose 2.

A. var funct = (a);

B. Score: function() { ... }

C. this.Score = function() { ... }

D. var a = Score();

ANSWER: B C

Explanation:

`Score: function() { ... }` declares a method when used as a property definition in an object literal, such as `const game = { Score: function() { ... } };`. In JavaScript, methods are functions associated with an object, and an object-literal property can hold a function expression. The function is then invoked through the containing object, for example `game.Score()`. This remains a common and valid method-definition pattern, although modern JavaScript also supports concise method syntax.

`this.Score = function() { ... }` assigns a function to the `Score` property of the current object. Because the assigned value is a function and is accessible through that object, it creates a method. This pattern is especially common in constructor functions, where each constructed instance receives its own callable `Score` member. For example, inside a

constructor, `this.Score = function() { return 100; };` enables callers to use `instance.Score()`. MDN defines methods as functions that are properties of objects and documents assigning function expressions to properties. See MDN: Defining methods and MDN: function expression.

QUESTION NO: 6

You write the following markup to create a page. Line numbers are included for reference only.

01 < !DOCTYPE html >

02 < html >

03 < head >

04 < style >

05

10 < /style >

11 < /head >

12 < body >

13 < svg 500 " 500 " >

14 < defs >

15 < filter id= " f2 " x= " 0 " y= " 0 " 200% " 200% " >

16 < feOffset result= " offOut " in= " SourceGraphic " dx= " 20 " dy= " 20 " / >

17 < feGaussianBlur result= " blurOut " in= " offOut " stdDeviation= " 10 " / >

18 < feBlend in= " SourceGraphic " in2= " blurOut " mode= " normal " / >

19 < /filter >

20 < /defs >

21 < text x= " 10 " y= " 100 " style= " fill:red; " > Blur Me! < /text >

22 Sorry, your browser does not support inline SVG.

23 < /svg >

24 < /body >

25 < /html >

An SVG blur filter is defined in the markup on the left. You need to apply the SVG blur filter to the text element on the page.

Which CSS code should you insert at line 05?

A. text { font: 48px arial bold; filter: #blur;}

B. text { filter: url(" #f2 "); font-size: 50pt; color: red;}

C. text { font: 48px arial bold; fill: blur;}

D. text { font: 48px arial bold; filter: url(blur);}

ANSWER: B

Explanation:

`text { filter: url( " #f2 " ); font-size: 50pt; color: red;}` is correct because the SVG filter is defined in the document's `<defs>` section with the identifier `f2`. A CSS `filter` declaration applies an SVG filter by using a URL reference to that identifier, written as `url(#f2)`. This fragment reference connects the rendered SVG `<text>` element to the filter pipeline declared by `<filter id="f2">`.

The referenced filter first offsets the source graphic, then applies `<feGaussianBlur>` to the offset result, and finally blends that result with the original source graphic. Applying the filter to the `text` selector therefore affects the displayed "Blur Me!" text. The font-size and color declarations are unrelated presentation properties, but they do not prevent the filter from being applied; the inline SVG `fill:red` supplies the text paint color. SVG filter references can be used through the CSS `filter` property when the target filter is identified by a URL fragment. See the MDN documentation for the CSS filter property and the SVG filter element.

QUESTION NO: 7

You have created custom error messages for a form. When a user attempts to submit the form with invalid data, the data must remain in the form and error messages must be displayed. Which Event property or method should you use?

A. defaultPrevented

B. preventDefault

C. abort

D. cancelable

ANSWER: B

Explanation:

`preventDefault()` is the appropriate Event method because it cancels the browser's normal action for a cancelable event. For a form's `submit` event, the normal action is submitting the form, commonly resulting in navigation to the form's action URL or a page reload. Calling `event.preventDefault()` within the submit-event listener keeps the user on the current page, so the values they entered remain available in the form controls while the application renders its custom validation messages.

A typical implementation validates the relevant fields in the submit handler and, when validation fails, calls `event.preventDefault()` before displaying feedback near the affected controls. This method does not itself perform validation or generate error text; rather, it gives the application control over the submission flow so custom validation UI can be shown without losing the entered data. The Event API defines `preventDefault()` as the mechanism for telling the user agent that the event's default action should not occur. See the MDN Event.preventDefault() reference and the MDN submit event reference.

QUESTION NO: 8

Which two CSS segments are valid filter properties? Choose 2.

A. filter: opacity(25%)

B. filter: box-shadow(16px 24px 24px green)

C. filter: blur(25deg)

D. filter: drop-shadow(16px 16px 16px red)

ANSWER: A D

Explanation:

`filter: opacity(25%)` is a valid CSS declaration because `opacity()` is a supported CSS filter function. The function accepts either a number or percentage that represents the amount of the original image or element that remains visible after the filter is applied. A percentage value of `25%` is therefore valid and produces a rendered result with reduced opacity. The `filter` property can apply one or more graphical effects to an element, and filter functions are evaluated as part of its visual rendering.

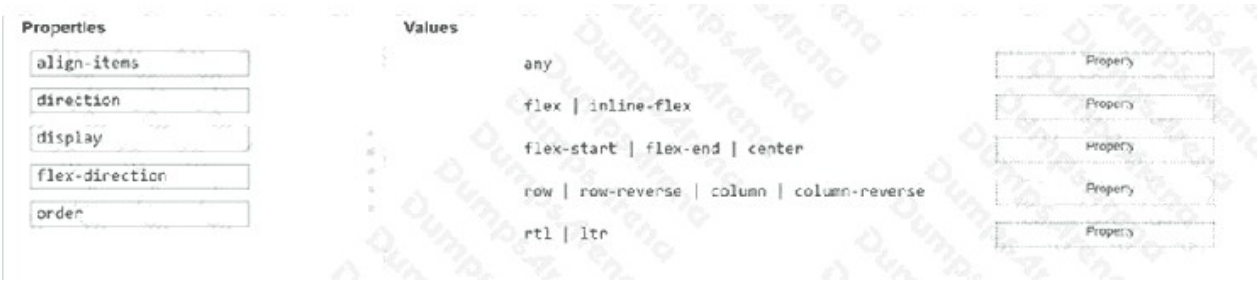
`filter: drop-shadow(16px 16px 16px red)` is also valid. The `drop-shadow()` filter function accepts horizontal and vertical offsets, an optional blur radius, and an optional color. Here, the first two `16px` values specify the shadow offsets, the third specifies a blur radius, and `red` provides the shadow color. Unlike a rectangular box-oriented shadow, this filter derives the shadow from the visible alpha shape of the rendered content, making it particularly useful with transparent images and irregular shapes. These declarations follow the syntax defined for the CSS `filter` property and its filter functions. See MDN's filter reference and MDN's drop-shadow() reference.

QUESTION NO: 9 - (HOTSPOT)

Match each property to its corresponding value for creating the CSS flexible box layout.

Move each property from the list on the left to the correct value on the right.

Note: You will receive partial credit for each correct match.



ANSWER:

Explanation:

A flexible box layout begins when an element is made a flex container through the `display` property. Using `flex` creates a block-level flex container, while `inline-flex` creates an inline-level flex container. In either case, the element's direct children participate as flex items. The `flex-direction` property then establishes the main-axis direction used for laying out those items. Its relevant values are `row`, `row-reverse`, `column`, and `column-reverse`, which define horizontal or vertical flow and whether that flow is reversed. The MDN reference for basic Flexbox concepts describes this container, item, and axis model.

Cross-axis alignment is controlled by `align-items`. The values `flex-start`, `flex-end`, and `center` place flex items at the start, end, or center of the cross axis. The `direction` property controls the base text and inline direction, so its listed values are `ltr` for left-to-right and `rtl` for right-to-left. Finally, `order` is applied to individual flex items to set their ordinal position in the flex layout. It accepts integer numbers, represented by "any" in the matching exercise, enabling visual ordering independently of source order. The MDN order documentation confirms that this property uses an integer value. These pairings correctly distinguish container setup, axis direction, cross-axis alignment, writing direction, and item ordering.