CSS Diagnostic Test

Section 1: Fundamentals (Basic Understanding)

1. What is CSS and what problem does it solve in web development?

CSS is Cascading Style Sheets and the problem it solves is styling an otherwise basic HTML file without compromising functionality.

2. What are the three ways to apply CSS to an HTML file? Explain when you would use each method and mention at least one advantage and disadvantage of each.

- With a .css file and reference it in the head tag.
- Directly in the head tag styles="" attribute
- If using something like tailwindcss, you can use className directly in the html tag you want to modify

3. What is a CSS rule? Identify and explain each part of the following CSS rule:

```
css
.container > p {
  color: blue;
  margin: 10px;
}
```

A CSS rule is a style that you want to apply to a tag in HTML.

The CSS shown is targeting a class named container that inherits from p tag CSS rules.
The .container will appear blue and have a margin of 10 pixels.
These rules will override the rules p tag had if set previously.

Section 2: Selectors & Specificity (Intermediate)

4. Explain the difference between these selectors and provide an example use case for each:

- Class selector
    -
    - Class is a general selector that can be reused.
    - In CSS rules they are identified by .className
- ID selector

    -
    - ID selector is unique to the tag you ID'd
    - In CSS rules they are identified by **#**idName
- Attribute selector
    -
    - Depending on the tag, can change. Usually to set the functionality of the tag.
- Pseudo-class selector
    - <div class="pseudoClass"r>
    - Change the behavior of a given tag when different interactions occur
    - In CSS rules they are identified by pseudoClass**::hover** or other actions

5. What is CSS specificity? Calculate the specificity of these selectors and order them from lowest to highest:

```
css
a) div p
b) .container #main
c) nav > ul li.active
d) p
e) [type="text"]
```

CSS specificity is how we target certain nested tags or tag variations.

D, A, B, C, E

6. When and why should you use `!important`? What are the potential problems with overusing it?

!important is used when we want to make sure a rule is followed regardless of later changes in the CSS. The main potential problem with overusing it is that conflict resolution gets messy since many things want priority and the CSS then doesn't produce the right styling.

Section 3: Box Model & Layout (Core Concepts)

7. Explain the CSS Box Model. What's the difference between `box-sizing: content-box` and `box-sizing: border-box`?

The box model is how the content of the HTML is designed and can be changed with CSS rules.

It includes:

- content = the main content that is displayed
- padding = the distance between the content and the border
- margin = the distance between the border and display

- border = border of the content

box-sizing: content-box is when you want the content to be the relative center when interacting with other objects

box-sizing: border-box is when you want the border to be the relative center when interacting with other objects

8. What's the difference between `margin` and `padding`?

padding = the distance between the content and the border

margin = the distance between the border and display limit

9. Explain the difference between `display: none`, `visibility: hidden`, and `opacity: 0`.

display: none = Do not display the object and do not interact with the other objects

visibility: hidden = Hide the object until a condition is met

opacity: 0 = make the object invisible but still present and interactable

Section 4: Modern Layout Systems (Practical Application)

10. Explain what Flexbox is designed for. Then write code that:

- Creates a horizontal navigation bar with items evenly spaced
- Vertically centers content in a container

Flexbox is used to simplify the CSS styling of the box model.

```
.nav {
  display: flex,
  justify-items: even,
}
.container {
  display: flex,
  align-content: center
```

}

11. Explain what CSS Grid is designed for and when you'd choose it over Flexbox. Then write code that creates a basic 3-column, 2-row layout with a header spanning all columns.

Grid is designed for items that you'd want to be in rows and columns. You'd choose it over Flexbox because you can set the Grid to repeat the same rules for each item in the rows and columns. In Flexbox you'd have to set the position for each item individually.

```
#grid {

  display: grid,

  grid-template-rows: 2,

  grid-template-columns: 3,

}
```

12. What is the difference between `justify-content` and `align-items` in Flexbox? What about `justify-items` and `align-content` in Grid?

The difference between justify and align in both Grid and Flexbox is that justify affects the horizontal spacing whether that be items or content, while align affects the vertical spacing whether that be items or content. In the context of Grid it also affects where the rows/cols go.

Section 5: Responsive Design & Media Queries

13. What are media queries used for? Write a media query that:

- Changes a 3-column grid to a single column on mobile devices
- Explain what "mobile-first" approach means in this context

They are used for changing the CSS in response to a change in the display size. Mobile-first means that the application was made with mobile users being the main audience and access is easy for them.

```
@media mobile-screensize-example #grid {

  grid-template-columns: 1

}
```

14. What's the difference between using `em`, `rem`, `px`, `%`, and `vh/vw` units? When would you use each?

em and rem both are relative sizes to the box model being used. rem is also relative to the modifications of the box models(margin/padding)

px is used when you know the exact measurements you need

% is used when you want the size to be relative to some object

vh/vh is used to have a size relative to the current display

Section 6: Advanced Concepts (Going Deeper)

15. What is CSS inheritance? Give examples of properties that inherit and properties that don't.

CSS inheritance is when you pass the rules of the parent to the child and then the child can modify certain things belonging to the parents without changing the parent.

16. Explain the CSS cascade. If multiple rules target the same element, how does CSS determine which styles to apply?

The closer to EOF the rules are the more priority are given to them. If rules target the same element, it will prioritize !important and which comes later in the style sheet.

17. What are CSS custom properties (variables)? Write an example showing how to define and use them, including how to provide a fallback value.

They are a property that can be used to give a value of your choice.

.classA {

- - fallback_color: #000000,

- - custom_color_property: var(#FFFFFF, - -fallback_color)

}

18. What's the difference between `position: relative`, `position: absolute`, `position: fixed`, and `position: sticky`? Provide a practical use case for each.

relative: displays next to another element which it uses as an anchor (p tag)

absolute: overlaps other elements if necessary and within the display (hero)

fixed: remains at the given position regardless of other elements (footer)

sticky: remains at the same position on the display regardless of other elements (nav)

19. Explain what pseudo-elements are. What's the difference between `::before` and `:hover`? Provide a code example using `::before` or `::after`.

Pseudo-elements are a subset of the element when different actions happen when interacting with it.

The difference between ::before and ::hover is that ::before is present before any interaction happens to the element. ::hover is only active when the user is hovering over the element.

```
.classA::before {

  color: black

}
```

20. What are CSS transitions and animations? Write a simple example of each and explain when you'd use one over the other.

CSS transitions are when there is a variable affecting the visuals when an action happens. We would use this for a literal transition between state A and state B.

```
.transition_class::after {

  transition: translate(10px)

}
```

CSS animations are when the visuals are always acting in accordance with a given rule. We would use this when we want an animation to run continuously.

```
@keyframes animation_name {
  // some change
}
.animation_class {
  animate: animation_name linear infinite
}
```