**CSS Interview Questions**

1.selectors-In CSS, selectors are patterns used to select the element(s) you want to style.

2.>child element selector-The element>element selector is used to select elements with a specific parent.

Eg: div > p {   
    background-color: yellow;  
 }

With the above selector Elements that are not directly a child of the specified parent, are not selected.

3. Px and % how they work-PX--With sizing fonts in pixels, you are literally telling browsers to render the letters exactly that number of pixels in height. If you need fine-grained control, sizing fonts in pixel values (px) is an excellent choice. Pixels (px) are relative to the viewing device. For low-dpi devices, 1px is one device pixel (dot) of the display. For printers and high resolution screens 1px implies multiple device pixels.Pixels are fixed-size units that are used in screen media (i.e. to be read on the computer screen). One pixel is equal to one dot on the computer screen (the smallest division of your screen’s resolution). Many web designers use pixel units in web documents in order to produce a pixel-perfect representation of their site as it is rendered in the browser. One problem with the pixel unit is that it does not scale upward for visually-impaired readers or downward to fit mobile devices.

%-The percent unit is much like the “em” unit, save for a few fundamental differences. First and foremost, the current font-size is equal to 100% (i.e. 12pt = 100%). While using the percent unit, your text remains fully scalable for mobile devices and for accessibility.

4. combinations of selectors  
A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS3:

descendant selector (space)- The descendant selector matches all elements that are descendants of a specified element.

Use:

div p {  
    background-color: yellow;  
}

child selector (>)-The child selector selects all elements that are the immediate children of a specified element.

The following example selects all <p> elements that are immediate children of a <div> element:

div > p {  
    background-color: yellow;  
}

adjacent sibling selector (+)-selects all elements that are the adjacent siblings of a specified element.

The following example selects all <p> elements that are placed immediately after <div> elements:

div + p {  
    background-color: yellow;  
}

general sibling selector (~)-The general sibling selector selects all elements that are siblings of a specified element.

The following example selects all <p> elements that are siblings of <div> elements:

div ~ p {  
    background-color: yellow;  
}

Css image sprite- CSS Sprites are a means of combining multiple images into a single image file for use on a website, to help with performance. A web page with many images can take a long time to load and generates multiple server requests. Using image sprites will reduce the number of server requests and save bandwidth.

#home {  
    width: 46px;  
    height: 44px;  
    background: url(img\_navsprites.gif) 0 0;  
}

CSS Reset- A CSS Reset (or “Reset CSS”) is a short, often compressed (minified) set of CSS rules that resets the styling of all HTML elements to a consistent baseline. Using a [CSS Reset](http://www.cssreset.com/), CSS authors can force every browser to have all its styles reset to null, thus avoiding cross-browser differences as much as possible. reset, you can then go on to re-style your document, safe in the knowledge that the browsers’ differences in their default rendering of HTML.

With an external style sheet, you can change the look of an entire website by changing just one file.

Each page must include a reference to the external style sheet file inside the <link> element. The <link> element goes inside the <head> section:

<head>  
<link rel="stylesheet" type="text/css" href="mystyle.css">  
</head>

An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a .css extension.

Css3 transition- CSS3 transitions allows you to change property values smoothly (from one value to another), over a given duration. To create a transition effect, you must specify two things:

* the CSS property you want to add an effect to
* the duration of the effect.

div {  
    width: 100px;  
    height: 100px;  
    background: red;  
    -webkit-transition: width 2s; /\* Safari \*/  
    transition: width 2s;  
}

CSS3 gradients- let you display smooth transitions between two or more specified colors. CSS3 defines two types of gradients:

* Linear Gradients (goes down/up/left/right/diagonally)
* Radial Gradients (defined by their center)

Syntax: background: linear-gradient(*direction*, *color-stop1*, *color-stop2, ...*);

A radial gradient is defined by its center.

To create a radial gradient you must also define at least two color stops.

Syntax:: background: radial-gradient(shape size at position, start-color, ..., last-color);

With CSS3 you can add shadow to text and to elements.

he CSS3 text-shadow property applies shadow to text.

In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px):

h1 {  
    text-shadow: 2px 2px;  
}

The CSS3 box-shadow property applies shadow to elements.

div {  
    box-shadow: 10px 10px;  
}

The display property is the most important CSS property for controlling layout. The display property specifies if/how an element is displayed. Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline. A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

Examples of block-level elements:

* <div>
* <h1> - <h6>
* <p>
* <form>
* <header>
* <footer>
* <section>

An inline element does not start on a new line and only takes up as much width as necessary.

* This is an inline <span> element inside a paragraph.
* Examples of inline elements:
* <span>
* <a>
* <img>
* visibility:hidden; also hides an element.
* However, the element will still take up the same space as before. The element will be hidden, but still affect the layout:
* h1.hidden {  
      visibility: hidden;  
   }
* The color property specifies the color of text.-
* body {  
      color: red;  
  }  
    
  h1 {  
      color: #00ff00;  
  }  
    
  p {  
      color: rgb(0,0,255);  
  }

syntax::: color: color|initial|inherit;

overflow property- The overflow property specifies what happens if content overflows an element's box.

This property specifies whether to clip content or to add scrollbars when an element's content is too big to fit in a specified area.

div {  
    width: 150px;  
    height: 150px;  
    overflow: scroll;  
}

syntax:: overflow: visible|hidden|scroll|auto|initial|inherit;

|  |  |  |
| --- | --- | --- |
| visible | The overflow is not clipped. It renders outside the element's box. This is default | [Play it »](http://www.w3schools.com/CSSref/playit.asp?filename=playcss_overflow) |
| hidden | The overflow is clipped, and the rest of the content will be invisible | [Play it »](http://www.w3schools.com/CSSref/playit.asp?filename=playcss_overflow&preval=hidden) |
| scroll | The overflow is clipped, but a scroll-bar is added to see the rest of the content | [Play it »](http://www.w3schools.com/CSSref/playit.asp?filename=playcss_overflow&preval=scroll) |
| auto | If overflow is clipped, a scroll-bar should be added to see the rest of the content | [Play it »](http://www.w3schools.com/CSSref/playit.asp?filename=playcss_overflow&preval=auto) |
| initial | Sets this property to its default value. [Read about initial](http://www.w3schools.com/CSSref/css_initial.asp) | [Play it »](http://www.w3schools.com/CSSref/playit.asp?filename=playcss_overflow&preval=initial) |
| inherit | Inherits this property from its parent element. [Read about inherit](http://www.w3schools.com/CSSref/css_inherit.asp) |  |

Padding- The CSS padding properties are used to generate space around content.

The padding clears an area around the content (inside the border) of an element.

With CSS, you have full control over the padding. There are CSS properties for setting the padding for each side of an element (top, right, bottom, and left).

CSS has properties for specifying the padding for each side of an element:

padding-top

padding-right

padding-bottom

padding-left

All the padding properties can have the following values:

length - specifies a padding in px, pt, cm, etc.

% - specifies a padding in % of the width of the containing element

inherit - specifies that the padding should be inherited from the parent element

offset- The outline-offset property adds space between an outline and the edge or border of an element.

Outlines differ from borders in two ways:

* An outline is a line drawn around elements, outside the border edge
* A outline do not take up space
* An outline may be non-rectangular

Syntax: outline-offset: length|initial|inherit;

The orientation [CSS](https://developer.mozilla.org/en-US/docs/Web/CSS) descriptor controls the orientation of a document defined by [@viewport](https://developer.mozilla.org/en-US/docs/Web/CSS/@viewport).

For a UA/device where the orientation is changed upon tilting the device, an author can use this descriptor to inhibit the orientation change.

orientation: auto;

orientation: portrait;

orientation: landscape;

Media Queries: Media query is a CSS technique introduced in CSS3.It uses the @media rule to include a block of CSS properties only if a certain condition is true.

@media only screen and (max-width: 500px) {  
    body {  
        background-color: lightblue;  
    }  
}

Viewport: The viewport is the user's visible area of a web page. The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen.

Setting the viewport:

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

A <meta> viewport element gives the browser instructions on how to control the page's dimensions and scaling.

The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

Margin-auto: You can set the margin property to auto to horizontally center the element within its container. The element will then take up the specified width, and the remaining space will be split equally between the left and right margins:

div {  
    width: 300px;  
    margin: auto;  
    border: 1px solid red;  
}

CSS order of precedence:

### *Browser Style Sheets*

All browsers have a built-in default style sheet. The cascade starts with these styles. If you have no styles defined, the browser styles will be used. Note that defaults are slightly different in each browser.

**Note**: It is a good idea to examine a page with no styles in each browser in order to see any difference.

### *External style sheets*

Browser default styles are overridden by the rules in external style sheets. If there are several external style sheets, they apply in the order listed.

### *Embedded styles*

Styles embedded in the page header are the next in order of precedence. If there are many rules, typically they are placed in an external style sheet.

### *Inline styles*

Finally, since inline styles are closest to the content, they take **top precedence**.

## Specific rules

The more specific a rule is, the greater its precedence.

Any rule on a tag is the least specific. A**class rule** is *more specific* and overrides the tag rule. Since an **ID rule** can be used only once in a web page, it is the*most specific*, overriding the other two.

For example, if the style sheet defines **<h1>** as being centered, then **<h2 class="left">** takes precedence and **<h2 id="right">** takes precedence to both rules.

## Later rules take precedence

Finally, sort the rules by the order they were defined. **Rules that are defined later in the document tree have higher precedence** than those defined earlier. And rules from an imported style sheet are considered before rules directly in the style sheet.

Selectors: #-id,.class, \*-all elements.

Box Model: All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

* **Content** - The content of the box, where text and images appear
* **Padding** - Clears an area around the content. The padding is transparent
* **Border** - A border that goes around the padding and content
* **Margin** - Clears an area outside the border. The margin is transparent

div {  
    width: 300px;  
    border: 25px solid green;  
    padding: 25px;  
    margin: 25px;  
}

Display property: The display property specifies if/how an element is displayed. Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.

## **Block-level Elements**

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The <div> element is a block-level element.

Examples of block-level elements:

* <div>
* <h1> - <h6>
* <p>
* <form>
* <header>
* <footer>
* <section>

Inline elements: An inline element does not start on a new line and only takes up as much width as necessary.

This is an inline <span> element inside a paragraph.

Examples of inline elements:

* <span>
* <a>
* <img>

Display:none-display: none; is commonly used with JavaScript to hide and show elements without deleting and recreating them. Take a look at our last example on this page if you want to know how this can be achieved.

The <script> element use display: none; as its default.

visibility:hidden; also hides an element.

Pseudo classes: A pseudo-class is used to define a special state of an element.

For example, it can be used to:

* Style an element when a user mouses over it
* Style visited and unvisited links differently
* Style an element when it gets focus

Syntax: selector:pseudo-class {  
    property:value;  
}

/\* unvisited link \*/  
a:link {  
    color: #FF0000;  
}  
  
/\* visited link \*/  
a:visited {  
    color: #00FF00;  
}  
  
/\* mouse over link \*/  
a:hover {  
    color: #FF00FF;  
}  
  
/\* selected link \*/  
a:active {  
    color: #0000FF;  
}

Margin property: CSS has properties for specifying the margin for each side of an element:

* margin-top
* margin-right
* margin-bottom
* margin-left

p {  
    margin-top: 100px;  
    margin-bottom: 100px;  
    margin-right: 150px;  
    margin-left: 80px;  
}

Margin Short-hand property:

p {  
    margin: 100px 150px 100px 80px;  
}

Inherit value: This example lets the left margin be inherited from the parent element:

div.container {  
    border: 1px solid red;  
    margin-left: 100px;  
}  
  
p.one {  
    margin-left: inherit;  
}