CSS Visibility

A property called *visibility* allows you to hide an element from view.

- ***Visible*** - The box and its contents are shown to the user.
- ***Hidden*** - The box and its content are made invisible, although they still affect the layout of the page (but still takes up space)
- ***Collapse -*** Only for table rows (<tr>), row groups (<tbody>), columns (<col>), column groups (<colgroup>). This value removes a row or column, but it does not affect the table layout. The space taken up by the row or column will be available for other content.

  Note:- If collapse is used on other elements, it renders as "hidden"
- ***Initial -*** Sets this property to its default value.
- ***Inherit -*** Inherits this property from its parent element.

CSS Specificity

If there are two or more conflicting CSS rules that point to the same element, the browser follows some rules to determine which one is most specific and therefore wins out

- The universal selector (*) has low specificity, while ID selectors are highly specific!

Note: Specificity is a common reason why your CSS-rules don't apply to some elements, although you think they should.

CSS Specificity Hierarchy

Every selector has its place in the specificity hierarchy. There are four categories which define the specificity level of a selector:

- **Inline styles** - An inline style is attached directly to the element to be styled. Example: <h1 style="color: #ffffff;">.
- **IDs** - An ID is a unique identifier for the page elements, such as #navbar.
- **Classes, attributes and pseudo-classes** - This category includes .classes, [attributes] and pseudo-classes such as :hover, :focus etc.
- **Elements and pseudo-elements** - This category includes element names and pseudo-elements, such as h1, div, :before and :after.

How to Calculate Specificity?

Start at 0, add 1000 for style attribute, add 100 for each ID, add 10 for each attribute, class or pseudo-class, add 1 for each element name or pseudo-element.

A: h1
B: #content h1
C: `<div id="content"><h1 style="color: #ffffff">Heading</h1></div>`

The specificity of A is 1 (one element)
The specificity of B is 101 (one ID reference and one element)
The specificity of C is 1000 (inline styling)

Since 1 < 101 < 1000, the third rule (C) has a greater level of specificity, and therefore will be applied.

Specificity Rules

- **Equal specificity: the latest rule counts** - If the same rule is written twice into the external style sheet, then the lower rule in the style sheet is closer to the element to be styled and applied.
- **ID selectors have a higher specificity than attribute selectors**
- **Contextual selectors are more specific than a single element selector -** The embedded style sheet is closer to the element to be styled
- **A class selector beats any number of element selectors** - a class selector such as .intro beats h1, p, div, etc:
- **The universal selector and inherited values have a specificity of 0** - *, body * and similar have a zero specificity.

CSS !important

!important rule in CSS is used to add more importance to a property/value than normal, it will override ALL previous styling rules for that specific property on that element.

- !important helps if you have to override a style that cannot be overridden in any other way

CSS Scrollbars

- There may be a case when an element's content might be larger than the amount of space allocated to it. For example, given width and height properties do not allow enough room to accommodate the content of the element.
- CSS provides a property called *overflow* which tells the browser what to do if the box's contents is larger than the box itself.

- Visible -Allows the content to overflow the borders
- Hidden – no scrollbars is visible
- Scroll – scrollbars added to allow user to scroll
- Auto – same as scroll, but scrollbars shows when required only

CSS Positioning

CSS helps you to position your HTML element. You can put any HTML element at whatever location you like. You can specify whether you want the element positioned relative to its natural position in the page or absolute based on its parent element.

- ***Relative*** - Relative positioning changes the position of the HTML element relative to where it normally appears.
- ***Absolute*** - An element with **position: absolute** is positioned at the specified coordinates relative to your screen top-left corner
- ***Fixed*** - Fixed positioning allows you to fix the position of an element to a particular spot on the page, regardless of scrolling.

CSS Layers

- CSS gives you opportunity to create layers of various divisions. The CSS layers refer to applying the *z-index* property to elements that overlap with each other.
- The z-index property is used along with the *position* property to create an effect of layers.

Pseudo Elements

- CSS pseudo-elements are used to add special effects to some selectors. You do not need to use JavaScript or any other script to use those effects

```
selector:pseudo-element {property: value}
```

1. **:first-line -** Use this element to add special styles to the first line of the text in a selector.
2. **:first-letter -** Use this element to add special style to the first letter of the text in a selector.
3. **:before -** Use this element to insert some content before an element.
4. **:after -** Use this element to insert some content after an element.

CSS Gradients

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

- Linear Gradients (goes down/up/left/right/diagonally)

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

- Radial Gradients (defined by their center)

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

CSS Shadows

With CSS you can add shadow to text and to elements

- *Text-shadow*

  The CSS text-shadow property applies shadow to text. In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px)

- *Box-shadow*

  The CSS box-shadow property applies shadow to elements.

CSS Transitions

CSS transitions allows you to change property values smoothly, 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

- transition
- transition-delay
- transition-duration
- transition-property
- transition-timing-function

CSS Animations

CSS allows animation of HTML elements without using JavaScript or Flash!

- An animation lets an element gradually change from one style to another.
- You can change as many CSS properties you want, as many times as you want.
- To use CSS animation, you must first specify some keyframes for the animation.
- Keyframes hold what styles the element will have at certain times.

CSS Animations (@keyframes)

- When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.
- To get an animation to work, you must bind the animation to an element.

```
@keyframes example {
 from {background-color: red;}
 to {background-color: yellow;}
}
div {
 width: 100px;
 height: 100px;
 background-color: red;
 animation-name: example;
 animation-duration: 4s;
}
```

CSS Variables

- The var() function is used to insert the value of a CSS variable.
- CSS variables have access to the DOM, which means that you can create variables with local or global scope, change the variables with JavaScript, and change the variables based on media queries.
- Good way to use CSS variables is when it comes to the colors of your design. Instead of copy and paste the same colors over and over again, you can place them in variables.

CSS Media

One of the most important features of style sheets is that they specify how a document is to be presented on different media: on the screen, on paper, with a speech synthesizer, with a braille device, etc.
We have currently two ways to specify media dependencies for style sheets −

- Specify the target medium from a style sheet with the @media or @import at-rules.
- Specify the target medium within the document language

Media queries can be used to check many things, such as:

- width and height of the viewport
- width and height of the device
- orientation (is the tablet/phone in landscape or portrait mode?)
- resolution

CSS Printing

- You can use CSS to change the appearance of your web page when it's printed on a paper. You can specify one font for the screen version and another for the print version.
- @media- This rule allows you to specify different style for different media. So, you can define different rules for screen and a printer.

Also, if you are defining your style sheet in a separate file, then you can also use the media attribute when linking to an external style sheet

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

CSS Mutli columns

- The CSS multi-column layout allows easy definition of multiple columns of text - just like in newspapers

CSS Multi-column Properties

- *column-count*
- *column-gap*
- *column-rule-style*
- *column-rule-width*
- *column-rule-color*
- *column-rule*
- *column-span*
- *column-width*

CSS Flexbox layout

The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning

- The direct child elements of a flex container automatically becomes flexible (flex) items.
- Flex item properties are :

1. Order
2. Flex-grow
3. Flex-shrink
4. Flex-basis
5. Flex
6. Align-flex

CSS Responsive

- Responsive web design makes your web page look good on all devices.
- Responsive web design uses only HTML and CSS.
- Responsive web design is not a program or a JavaScript.

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.
- Before tablets and mobile phones, web pages were designed only for computer screens, and it was common for web pages to have a static design and a fixed size.

CSS Responsive

HTML5 introduced a method to let web designers take control over the viewport, through the <meta> tag.

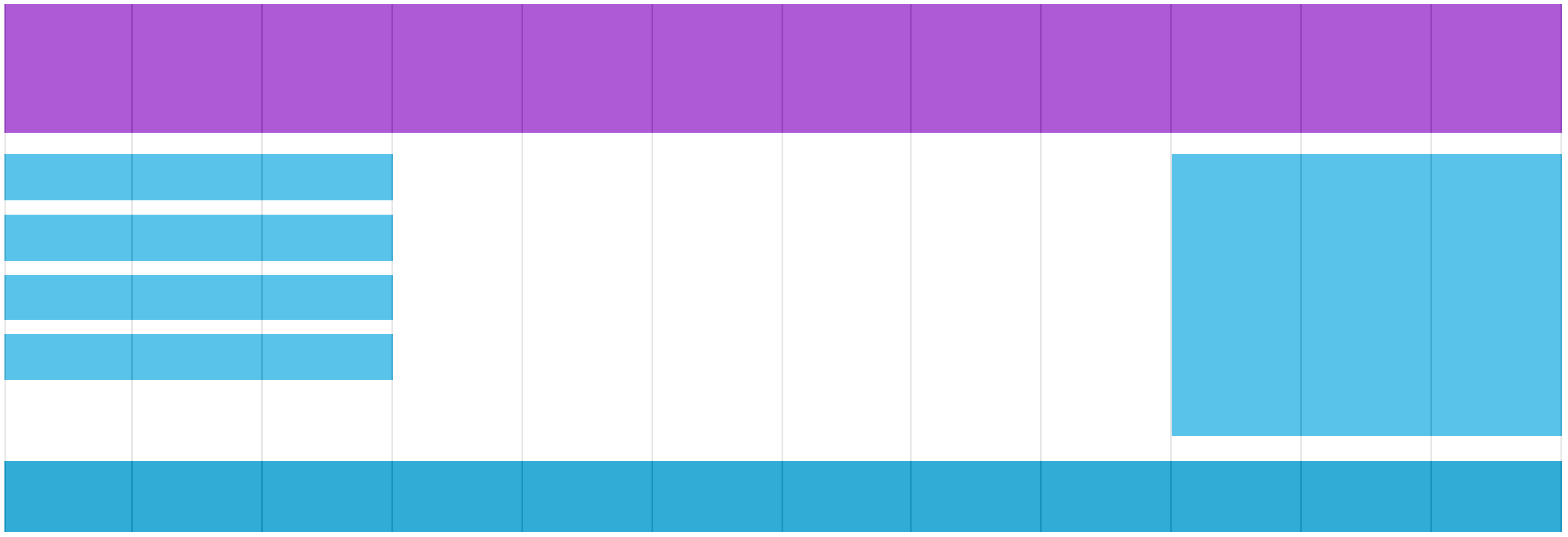
- You should include the following <meta> viewport element in all your web pages:

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

- 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.

CSS GridView

- Many web pages are based on a grid-view, which means that the page is divided into columns.
- A responsive grid-view often has 12 columns, and has a total width of 100%, and will shrink and expand as you resize the browser window.



Sass

- SASS (Syntactically Awesome Stylesheet) is a CSS pre-processor, which helps to reduce repetition with CSS and saves time.
- It is more stable and powerful CSS extension language that describes the style of document structurally.

WHY?

1. It is a pre-processing language which provides indented syntax (its own syntax) for CSS.
2. It is a super set of CSS,
3. It provides some features, which are used for creating stylesheets that allows writing code more efficiently and is easy to maintain.

Bootstrap

- Bootstrap is a sleek, intuitive, and powerful, mobile first front-end framework for faster and easier web development. It uses HTML, CSS and Javascript.

WHY?

1. **Mobile first approach** – Bootstrap 3, framework consists of Mobile first styles throughout the entire library instead them of in separate files.
2. **Browser Support**.
3. **Responsive Design.**
4. **Easy to get started.**