What happen to css when we load the page?

Loading Css ->parse css

How HTML parse?

Load page 🡪 parse HTML 🡪 DOM object model

Parse css:

* Resolving conflict css declaration ->cascade
* Process final css values like px,em % based on screen size

The parse returns the Css Object model(CSSOM)

Both css and html are rendered with render tree 🡪 visual formatting model

How css parsing works?

Css rule

.my-class 🡪seelctor

{

Color:red; -🡪declaration block

}

**Step 1:**

1.cascade 🡪process of combining different stylesheet and resolving conflict b/t different css rules and declarations,when more than one rule applied to certain element

2.it looks for importance of selector based on specification and apply style based on it

Importance -🡪 specificity -🡪source order

**Specificity :**

* Importance
* Inline
* ID
* Class,pseudo class,attribute
* Element ,pseudo-elements
* When 2 selector specificity is same the last added selector color will be added to UI
* Universal selector have 0 specificity
* Relay more on specificity rather than order of source element

How values are parsed in css?

* We need to know all the units are converted to px

How css values calculated

.section{

Font-size:1.5rem;

Width :280px;

}

P{

Width:280px;

Background:green;

}

.amazing{

Width:66%

}

<div class=”section”>

<p class=”amazing”>test</p>

</div>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Width(P)** | **Padding(P)** | **Font-size(root)** | **Font-size(section)** | **Font-size(p)** |
| **Declared value**(author declared) | 140px,66% | - | - |  |  |
| **Cascaded value**(value got after checking specificity and cascaded) | Based on cascade class take precedence so 66% | - | 16px(bowser default) | 1.5rem | Inherit the value of parent so 24px |
| **Specific value**(default if there is no cascaded value) | 66% | 0px(initial value) | 16px | 1.5rem | 24px |
| **Computed value**(convert relative relative value to absolute) | 66% | 0px | 16px | 24px(16\*1.5) | 24px |
| **Used value**(final based on layout) | Based on layout and parent 184.8px 🡺parent ->280\*66% => 184.8 | 0px | 16px | 24px | 24px |
| **Actual value**(browser restriction) | 185px | 0px | 16px | 24px | 24px |

How units are calculated from relative and absolute

|  |  |  |  |
| --- | --- | --- | --- |
|  | example | How convert to pixel | Result in pixel |
| %(font) | 150% | x%\*computed font size of **parent** | 24px(150% of 16px) |
| %(length) | 10% | X%\* parent computed width | 100px if parent width is 1000px |
| Em(font) | 3em | x\*computed font size of **parent** | 3\*24=>72 |
| Em(length)r | 2em | X\*current element width2 | 2em |
| Rem(font) | 1rem | X\*computed font of root element |  |
| vw |  | Browser view port width |  |
| vh |  | Browser view port height |  |

Inheritance in css:

* It passes some values from parent to child eg: font size from parent can be inherited
* **Note : The computed size gets inherited not actual value i.e eg;2rem is not inherited rather 32px is inherited**
* If some values are not inherited then the specified value will be considered for element
* **We can use inherit keyword to inherit the value of parent**
* Only when no values are given then inherit value is used

**Using rem and em:**

Html{

Font-size:10px ; 🡺 so 1rem = 10px .. rem depends on root element i.e html

}

How css render a website? 🡪visual formatting model

Algorithm that calculate boxes and determine the layout of these boxes,and finally tells how the element should appear in DOM

It is determined by below:

* Dimension of box : box modal
* Box type:inline,inline-block,block
* Positioning of box:floating and positioning
* Stacking contexts
* Other elements in render tree
* View size port and dimensions

Box model:

1.content

2.**padding**: transparent area inside a box

3.**Border:**b/t padding and content

4**.margin :** space b/t boxes

5.**filled area**:background image and images are applied to entire area

Height and width:

Total width : right border+right padding+specified width + left padding+left border

Total height: top border+top padding +specified height+bottom padding+bottom border

**Box-sizing:border-box** 🡪the height and width will be added to content area along with border and padding so no separate px wil be added eg: 100px 🡪 includes border and padding

**Different blockes in css**

**1.block level:**

Display:block,flex and table are block by default

It occupies the max width of box and create new line breaks

**2.inline level:**

Heigth and width cannot be used and it wont add to new line 🡪**padding and margin only to left and right**

**3.inline-block ..>**

it wont create new line but we can add height and width to it

**Positioning in css:**

**Normal :**

* Element are laid according to source code if no position is mentioned

**Float:**

* **Float : left or right** move to either left or right of content
* **Text and inline** elements will be wrapped with floated element
* **The container will not fix to height**
* Element removed form normal flow

**Absolute positioning:**

* **Element removed from normal**
* Position : absolute 🡪 position absolute to its parent element if no parent element then it is relative to root parent
* **Relative :** The element relative to root
* **Fixed:**The element is fixed to root

**Stacking context:**

* **Z-index 🡪**overlapping the elements with z-index ..>one with more z-index is
* **Opactity** also create stacking context

**Css architecture ,components and BEM**

* Clean
* Modular
* Resuable
* Growth

Think and build : think the layout before and then build it

* Component driven design:
  + Modular building blocks
  + Reusable
  + Independent ->they should not depend on other component
* Build:
  + BEM 🡪 clean system -> Block Element Modifier

Eg: .block{} , .block\_\_element{} , .block\_\_element—modifier{}

* Block 🡪 standalone element on its own
* Element ->part of block that has no standalone meaning
* Modifier ->a different version of block eg:rounded,border

**Architect**:

* **Seven one pattern 🡪 7 different folders for partial SASS files**
* **One Sass file is imported to all into a compiled css stylesheet**