

بسم الله الرحمن الرحيم والصلاة والسلام على أشرف المرسلين سيدنا محمد صلى الله عليه وسلم

How CSS Works Behind The Scenes

أزاي ال CSS بتشتغل في ال browser عشان ت Display ال web Pages بالشكل اللي بتشوفه دا كدا

Table of Contents

Three Pillars of Writing Good HTML and CSS (Never Forget Them!)

How CSS Works Behind the Scenes : An Overview

How CSS is Parsed, Part 1: The Cascade and Specificity

Specificity in Practice

How CSS is Parsed, Part 2: Value Processing

How Units Are Converted From Relative To Absolute (px) - Value Processing

How CSS is Parsed, Part 3: Inheritance

How CSS Renders a Website: The Visual Formatting Model

الخاتمة وشوية فضفضة

اذكر الله ❤

الزموا ذكر الله ففيه العون على متاعب الحياة ♥

اللَّهُمَّ إِنِّي أَسْأَلُكَ عِلْمًا نَافِعًا، وَرِزْقًا طَيِّبًا، وَعَمَلًا مُتَقَبَّلًا.

اللَّهُمَّ أَعِنِّي عَلَى ذِكْرِكَ، وَشُكْرِكَ، وَحُسْنِ عِبَادَتِكَ.

اللهم صلّ وسلم وزد وبارك على نبينا محمد ﷺ

لا تنسونا من دعواتكم

Three Pillars of Writing Good HTML and CSS (Never Forget Them!)

خلينا في البداية نتكلم عن اهم **ثلاث مبادئ في ال CSS** واللي هما واللي لازم تكون علي وعي بيهم

Responsive Design -1

Maintainable and Scalable Code -2

Web Performance -3

Responsive Design

ودي ببساطة معناها أني أبني **Web Page** بشكل **Responsive** يعني قابل للعرض في جميع **مختلف الشاشات** يعني ع ال **Mobile** يتعرض بشكل **مناسب** و علي ال **Laptops** يتعرض بشكل مناسب و علي ال **Large Screens** يتعرض بشكل مناسب وهكذا ... تصميم ال **layout** يتناسب مع حجم الشاشة اللي هيتعرض فيها

جميل جدا ... هل في حاجات لازم اراعيها عشان استخدم أو اعمل **Responsive Design** بشكل سليم ؟
قالك ايون فيه حاجات تراعيها لدا زي مثلا

Fluid Layouts -1

Media Queries -2

Responsive Images -3

Correct Units -4

Desktop First vs Mobile First Strategy -5

Maintainable and Scalable Code

ودي من أهم النقاط والمبادئ اللي لازم تكون عارفها ومعني النقطة دي انك تكتب **Clean Code** وحاجات تانيه كتير لازم تعمل حسابها .. تيجي نشوف .. تعالي نشوف

Clean Code -1

Easy To Understand -2

Growth -3

Reusable -4

How To Organize Files -5

How To Name Classes -6

How To Structure HTML -7

Web Performance

في النقطة دي انت لازم تراعي ان ال **web App** بتاعك يكون علي اعلي مستوي من ال **performance** وعشان يكون علي اعلي مستوي لازم يكون سريع ويكون صغير من حيث الحجم

في حاجات كتير جدا بتأثر علي ال **Performance**

1- **Less HTTP Requests**

2- **Less Code**

3- **Compress Code**

4- **Use a CSS Performance**

5- **Less Images**

6- **Compress Images**

فكل ما تقلل حجم الصور وتستخدم **preprocess** زي ال **sass** مثلا وتكتب كود قليل وفعال فدا يساعدك تعلي من ال **Performance** بتاع الموقع

اذكر الله ♥

الزموا ذكر الله ففيه العون على متاعب الحياة ♥

اللهم صلّ وسلم وزد وبارك على نبينا محمد ﷺ

سبحان الله وبحمده عدد خلقه ورضا نفسه وزنة عرشه ومداد كلماته

لا إله الا انت سبحانك اني كنت من الظالمين

اللهم لك الحمد حمداً طيباً كثيراً مباركاً فيه كما ينبغي لجلال وجهك ولعظيم

سلطانك يا رب العالمين

لا حول ولا قوة الا بالله العلي العظيم

أَلَا بِذِكْرِ اللَّهِ تَطْمَئِنُّ الْقُلُوبُ

لا تنسوننا من دعواتكم

By: Amr Elsayed

How CSS Works Behind the Scenes: An Overview

اثناء تحميل صفحة الويب في حاجات كثير بتحصل خلف الكواليس زي **http request** و **right Domain** و **name Services** و حاجات تانيه كثير

بس احنا هنركز اكثر عن اللي بيحصل لل **Browser** علي ال **Computer** لما ال **user** بيفتح الموقع

أول حاجة بمجرد ما ال **page** تحمل بيحصل **load** لل **initial HTML File** يعني ملف ال **HTML** المبدئي كدا اول حاجة بتحمل

وبعد ما يتحمل بيحصل عمليه **parse** او تحليل **سطر بسطر** اثناء العملية دي ما بتحصل ال **Browser** يبيني

حاجة اسمها **Document Object Model (DOM)** واللي هي عبارة عن **Web Document** بيكون

فيها **family Tree** للعناصر يعني **div** جنبه **div** تاني وجوا الاثنين **Divs** دول فيهم عناصر والعناصر دي جواها عناصر دي ال **Tree** بتاعته

ببساطة بيتقال عليهم **Parents** واللي هما هنا ال **two divs** و ال **Childrens** واللي هما العناصر اللي جوا ال **Divs** دي و ال **Siblings** الاشقاء يعني العناصر اللي جنب بعض اللي راسها براس بعض

بعد ما بيحصل **parse** لل **HTML File** ال **Browser** بيلاحظ وجود لسطر كدا مكتوب فيه **تضمين** او **link** بملف **CSS**

اول ما بيلاحظ دا بيعمل للملف دا **load** يعني بي **load** ال **css File** وزي ما حصل لل **HTML File** برضو

ال **CSS File** دا بيحصله **Parse** او تحليل هو كمان ولكن ال **parse** دا بيكون مقعد بعض الشئ ودا اللي احنا عاملين الملخص دا عشان نناقشه اصلا

بس خليني اقولك ان اثناء عملية ال **parsing** دي في حاجتين او خطوتين اساسيتين بيحصلو

أول واحدة وهي **Conflicting CSS Declarations** بيحصلها **Resolving - حل** ودا هتناقشه بالتفصيل قدام.

و خليني أقولك ان العملية دي ليها مسمي وهو ال **Cascade** اول حرف في ال **CSS**

تاني حاجة بقا بتحصل اثناء عملية ال **parsing** دي وهي **Process Final CSS Values**

يعني ال **margins** بيتم قرأتها وتحويل قيمها ل **percentage units to px** يعني اي برضو ؟!

يعني تخيل معايا اني عامل **margin-left: 50%** مثلا ! بس ال **50%** دي مختلفه في الشاشات الصغيرة عن

الشاشات الكبيرة في عملية التحليل دي بيتم ترجمتها او تحليلها لل **px unit** لاستخدامها في مختلف ال **Devices**

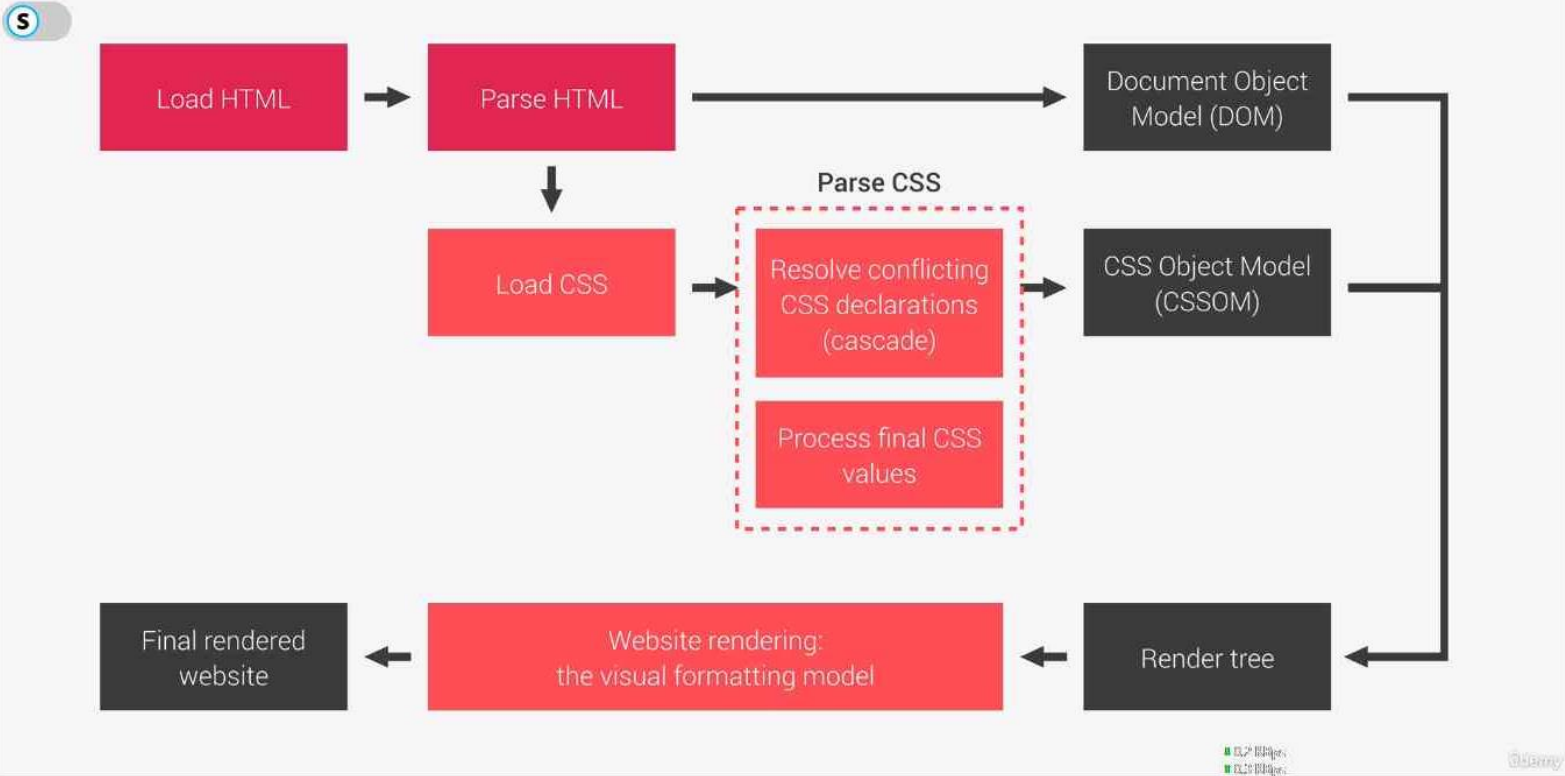
بعد ما ال **parsing** دا بيحصل لل **CSS** بيتم تخزينه في **CSS Object Model (CSSOM)** هي **tree** برضو زي

ال **HTML** كدا التخزين دا بيكون اسمه **CSSOM**

بعد ما التحليل دا كله بيحصل بقا بيحصل عملية اسمها **Rendering** لل **Trees** دي كلها مع بعض و عملية ال **Render** دي بتسمي بال **formatting** اللي ال **browser** بيستخدم فيها حاجة اسمها **Visual Formatting Model** وبرضو هنتكلم عنها قدام بالتفصيل اكثر

بعد ال **Formatting** دا ما يحصل بيحصل حاج اسمها **Final Rendered Website** وهنا انت بتقدر تشوف بقا الشكل بتاع الموقع بشكل فعلي علي المتصفح قدامك

WHAT HAPPENS TO CSS WHEN WE LOAD UP A WEBPAGE?



اذكر الله ❤️

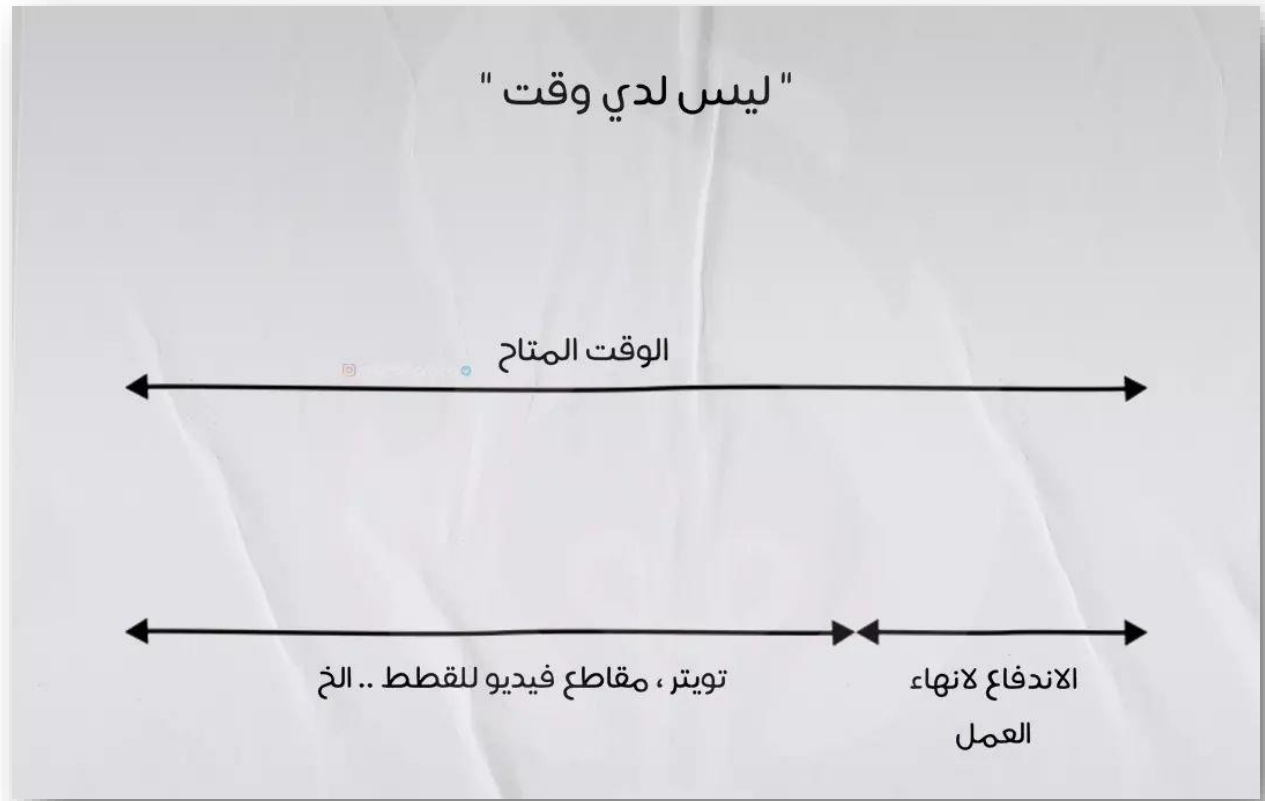
الزموا ذكر الله ففيه العون على متاعب الحياة ❤️

اللهم إني أستغفرك واتوب إليك، لا إله الا انت سبحانك إني كنت من الظالمين.

حسبي الله لا إله الا هو عليه توكلت وهو رب العرش العظيم.

ألا بذكر الله تَظْمَنُ الْقُلُوبُ

♥ راقث لى ♥



قبل ما آقول معنڊىش وقت شوف فعلىا الوقت المآآ قءامك اى وشوف انت بآقضىه ازاي هآلاآظ ان كل وقتك او اهم فآرات الوقت رايآة فى الآرفىه وآزء قليل آءا لأعمالك المهمة

اذكر الله ♥

سبحان الله وبآمءه .. سبحان الله العظىم

أستغفر الله العظىم الذى لا اله الا هو الـهى القىوم و أآوب إله

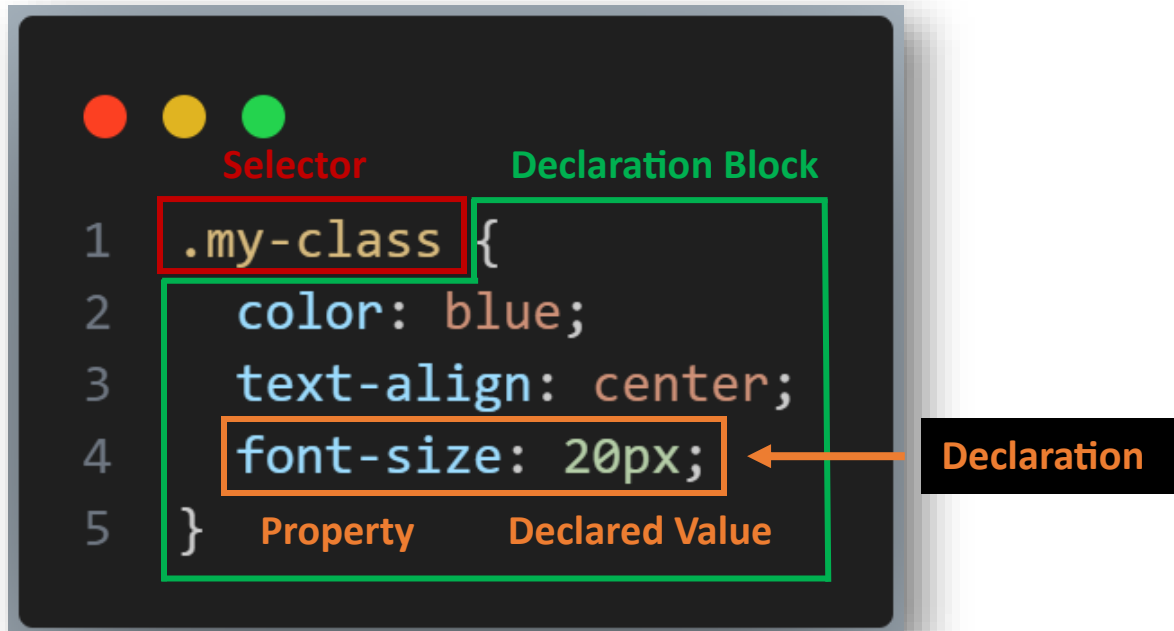
آسبى الله لا اله الا هو علىه آوكلآ وهو رب العرش العظىم

لا آنسونا من آعواآكم

By: Amr Elsayed

How CSS is Parsed, Part 1: The Cascade and Specificity

في الدرس اللي فات خدنا **overview** عن ازي ال **HTML** و ال **CSS** بيحصلهم **Process** في المتصفح
في الدرس دا هنشوف عملية تحليل ملف ال **CSS** بتتم ازي
خلينا متفقين كلنا ان ال **Css** بيكون ليه **Rules** معينة في طريقة الكتابة زي مثلا



والمقصود بال **rules** دي هو ان طريقة كتابة ال **css** بتكون معتمدة علي وجود **Selector** اللي هو في حالتنا دي ال **.my-class**. واللي بكتبه عشان احدد اي عناصر ال **HTML** اللي عندي اللي هديها التنسيقات اللي هتكتب كمان معتمد علي وجود ال **Declaration Block** واللي هي الخواص المكتوبة والقيم بتاعتها (تنسيقات العنصر) فزي ما انت شايف كدا

اول حاجة في عملية ال **parsing** لل **CSS** لو تفتكر لما قولت فوق وهي ال **Conflicting CSS Declarations**
وتاني حاجة في عملية ال **Parsing** لل **CSS** لو تفتكر برضو قولت فوق وهي ال **Process Final CSS Values**

CASCADE

Process of combining different stylesheets and resolving conflicts between different CSS rules and declarations, when more than one rule applies to a certain element.

ال **cascade** هي العملية الي بيتتم فيها **دمج** بين جميع ملفات ال **CSS** المختلفة وحل ال **conflicts** او التعارض الموجود بين كل ال **Declarations** الموجودة في كل الملفات دي لو كانت متكررة لعنصر واحد

يعني ببساطة شديدة جدا لو انت عندك **عنصر** معين مسكته في ال **CSS** و حطيت فيه بعض التنسيقات وروحت لملف تاني او في نفس الملف برضو تحت شوية مسكت نفس العنصر واديتة نفس التنسيقات دي تاني ال **cascade** بقا بيكون عامل زي **مراقب** كدا بيبص علي كل عنصر وتنسيقاته ولو متكررة بياخد منها نسخة واحدة بس ويطبقها علي العنصر الي تم اختياره دا

لنفترض انك مسكت class اسمه **my-class**. مثلا و حطيتله **font-size: 20px** ونزلت تحت شوية مسكت نفس العنصر دا و حطيتله نفس ال **font-size: 20px** او حتي غيرت قيمته فاللي بيحصل هنا عملية اسمها **Conflicting CSS Declarations** او **Cascade** بتصلح الاختلافات الي حصلت دي ويطبق آخر حاجة اتكتب

ال **CSS** ممكن يجي من أكثر من مصدر .. طيب يعني اي بقا معني الكلام دا ؟ واي هي المصادر دي ؟! اقولك انا

اول مصدر:

بيكون ال **Declarations** الي اتكتب من خلالك انت ك **Developer** والمصدر دا بيتسمي **Author Declarations**

تاني مصدر:

اللي بتيجي من ال **User** علي سبيل المثال لما **User** معين يغير ال **Default font-size** من ال **Browser**

وتالت مصدر:

وهو ال **Default Browser Declaration** والي هي التنسيقات الي بتتحط بشكل تلقائي من المتصفح نفسه

علي بعض العناصر والمصدر دا بيكون اسمه ال **user agent stylesheet** علي سبيل المثال ال **Anchor tag** `<a>` مش بيكون ليه تنسيقات من المتصفح زي خط تحته ولون **ازرق**؟!

بس فيه سؤال هنا ازاي فعليا ال **Cascade** دا بيتم او بيحصل او بيعمل اي عشان يعالج ال **Conflicts** دي كلها؟! ويتري في **قواعد** معينه بيمشي عليها عشان يحل **الاختلافات** دي ولا لا؟!

اول حاجة ببص عليها ال **cascade** دا اول ما يجي يشتغل وهي ال **!important flag** بعد كدا ال **specificity** بعد كدا ال **Source order (conflicts)** الي بتحصل منك انت ك **Developer** بالترتيب كدا

طيب تعالي نشوف ال **!Important flag** دا عبارة عن اي

بص معايا الكود دا كدا



```
1 <div class="box">Box</div>
```



```
1 .box {  
2   background-color: blue !important;  
3   color: white;  
4 }  
5  
6 .box {  
7   background-color: green;  
8 }
```

يتري اي **Style** في دول الي هيتطبق هل كلمة **Box** هيكون ال **Background** بتاعها لونها **أزرق** ولا المفروض هيكون لونها **أخضر** عشان انا اخر حاجة كتبتها لل **Box** دا `background-color: green;` وبعدين مش المفروض آخر حاجة اكتبها للعنصر هي الي تتطبق؟! تعالي نشوف

Box

```
.box {  
  background-color: green;  
}  
  
.box {  
  background-color: blue !important;  
  color: white;  
}
```

style.css:206
style.css:201

اي الي حصل ! دا طبق أول خاصية اللون الازرق مع ان الأخضر اكتب تحته هنا دا اول حاجة بيص عليها ال Cascade وهو بيحلل ملف ال CSS بيشف العناصر وتنسيقها ويبص ع ال **important Flag** ويطبقه طيب لو ال **important Flag** دا متطبق في العنصرين **حدد ازي انهي خاصية الي هتشتغل !؟**

وهنا بيجي دور ثاني مرحلة والي هي ال **specificity** أو **الأقوي** بص معايا ع الكود دا كدا



```
1 <div id="parent">
2   <div class="box">Box</div>
3 </div>
```



```
1 #parent .box {
2   background-color: green !important;
3 }
4 .box {
5   background-color: blue !important;
6   color: white;
7 }
```

دلوقتي انا عندي العنصرين بتوعي فيه **Background-color** واللاتين مطبق عليهم ال **important Flag** يتري مين فيهم الي هيشغل أكيد الاخير الي هو ال **background-color: blue** لانه آخر حاجة اكتب وهو الي هيعمل **overwrite** علي الي قبله !... تعالي نشوف

Box

```
#parent .box {
  background-color: green !important;
}

.box {
  background-color: blue !important;
  color: white;
}
```

اي الي حصل !؟ دا اللون الاخضر الي فوق هو الي اطبق !؟ ودا حصل بسبب **specificity** الاول أقوى عشان فيه **ID** و **Class** مع بعض لكن ال **box** بس لوحدها وهي **Class** كدا اضعف من الحالة الأولى لذلك هو طبق اللون الأخضر بسبب قوة ال **specificity** مع ان اللون الأزرق جاي بعد الأخضر والمفروض هو الي ي **overwrite** عليه لكن دا محصلش بسبب ال **specificity** زي ما قولنا

ولاحظ ان ال **specificity** هتكون أقوى حاجة لو ال **style** معمول **inline-style** يعني انت حاطط التنسيقات في ال **HTML** نفسه مع العنصر في حالة انك مش عامل **important** لتنسيق معين يعني عندك

يعني انا قصدي التالي بص ع الكود دا عشان تعرف مين ال **specificity** الأقوي



```
1 <div id="parent">
2   <div class="box" style="background-color: red">Box</div>
3 </div>
```



```
1 #parent .box {
2   background-color: green;
3 }
4 .box {
5   background-color: blue;
6   color: white;
7 }
8
```

Box

```
element.style {
  background-color: red;
}
#parent .box {
  background-color: green;
}
.box {
  background-color: blue;
  color: white;
}
```

الاقوي هنا هو ال **inline-style**



```
1 <nav id="nav">
2   <div class="pull-right">
3     <a href="#" class="button">Don't Click Here</a>
4   </div>
5 </nav>
```



```
1 .button {
2   font-size: 20px;
3   color: white;
4   background-color: blue;
5 }
6
7 nav#nav div.pull-right.button {
8   background-color: green;
9 }
10
11 a {
12   background-color: purple;
13 }
14
15 #nav a.button:hover {
16   background-color: yellow;
17 }
```

ياتري انهو **background-color** في الاربعة دول الي هتطبق ع العنصر بتاعي

ال **specificity** بتتسبب بالترتيب كدا الاولية بتكون لل **inline-style** ثم ال **ID** ثم ال **class** ثم ال **element**
في العنصر الاول مفيش غير **class** واحد بس يبقى كدا المجموع (0 inline-style, 0 id, 1 class, 0 element)

في العنصر الثاني عندي ال **nav element** و عندي **ID #nav** وعندي **Element** كمان الي هو ال **div** وعندي **class** الي هو **pull-right**. وعندي **class** كمان الي هو ال **.button**.
يبقى المجموع (0 inline-style, 1 id, 2 classes, 2 elements)

في العنصر الثالث عندي **element** واحد بس الي هو ال **a** يبقى المجموع
(0 inline-style, 0 id, 0 class, 1 element)

في العنصر الرابع عندي **#nav id** وعندي **a element** وعندي **button class** وعندي **hover: class** وبقي المجموع (0 inline-style, 1 id, 2 classes, 1 element)

لو بصيت ع المجموع بتاع كل **Selector** هتلاحظ ان الثاني هو أقوى واحد فيهم هو اللي ال **specificity** بتاعته أعلى لذلك اللي هيطبق هو اللون الاخضر

Don't Click Here

```
nav#nav div.pull-right .button { style.css:216
  background-color: green;
}
.button { style.css:210
  font-size: 20px;
  color: white;
  background-color: blue;
}
a { style.css:220
  background-color: purple;
}
```

طيب لو كل حاجة متساوية كل ال **Selectors** نفس ال **specificity** وكلهم نفس ال **important Flag** ومفيش اي **inline-style** هنا بقا يجي اخر حاجة يبص عليها ال **Cascade** و اللي هي ال **Source order**

ال Source order

ودي ببساطة ال **conflicts** اللي بتحصل منك لو كل ال **Selectors** زي بعض في كل حاجة هنا ال **Cascade** هيشوف آخر حاجة اتكتب ويطبقها يعني آخر حاجة مكتوبه هتعمل **overwrite** علي الباقيين

يبقي ملخص سريع جدا ع اللي فات كله

اول حاجة ببص عليها ال **cascade** دا اول ما يجي يشتغل وهي ال **important flag** بعد كذا ال **specificity** بعد كذا ال **Source order** (conflicts) اللي بتحصل منك انت ك **Developer**

لو عندي عنصر في ال **HTML** حطيت جواه **inline-style** باللون الأحمر ومسكت نفس العنصر دا في ال **CSS** ادितه لون أخضر وحطيت معاه ال **important Flag** يبقي اللون الاخضر هو اللي هيتطبق لانه أقوى حاجة ودي أول حاجة ببص عليها ال **Cascade** ويطبقها ودي مرحلة ال **important Flag**

لو عندي عنصر في ال **HTML** وملهوش اي **inline-style** وانا مش مستخدم ال **important Flag** بس في ال **CSS** انا محدد العنصر دا بأكثر من طريقة ال **Cascade** هيشوف انهو **تحديده الأقوي** فيهم ويطبقها علي العنصر وفوق في المثال الكبير عرفنا ازي بيحدد انهو **Selector** الأقوي ودي ثاني مرحلة بعد ال **Important** وهي ال **specificity**

لو عندي عنصر في ال **HTML** ليه **inline-style** او ملوش وانا مستخدم ال **important Flag** وكلهم زي بعض في ال **specificity** ال **Cascade** هياخد آخر حاجة اتكتب لل **Selector** دا ويطبقها علي العنصر ودي المرحلة اللي بيكون اسمها ال **Source order** (conflicts)

يبقي الأولوية ديما بتروح لل **important**

ثم ال **specificity** واللي ببص فيها لو فيه **Inline-style** لو مفيش ببص علي ال **ID** لانه بيكون ثاني اقوي حاجة بعد ال **inline-style** وبعدها ببص ع ال **class** ودا بيكون ثالث اقوي حاجة وبعدين ببص ع ال **element** اللي بيكون رابع اقوي حاجة

ثم لو كل حاجة زي بعض في كل حاجة نفس ال **important** ونفس ال **specificity** يروح لأخر حاجة واللي هي ال **Source order** (conflicts) بمعني انه هيشوف آخر حاجة اتكتب لل **Selector** ويطبقها ويكون ليها الأولوية في الظهور

ملحوظة مهمة جدا: خلي أستخدامك لل **important** في أضيق الحدود الممكنة لو بس بتصلح كود بتاع شخص ثاني وتنسيقاتك مش راضيه تتطبق لسبب ما استخدم ال **Important** لان زي ما قولنا بيكون اول واحد ليه الأولوية في التطبيق أن وجد

Some Notes in English

- CSS declarations marked with !important have the highest priority;
- But only use !important as a last resource. It's better to use correct specificities – more maintainable code!
- Inline styles will always have priority over styles in external stylesheets;
- A selector that contains 1 ID is more specific than one with 1000 classes;
- A selector that contains 1 class is more specific than one with 1000 elements;
- The universal selector * has no specificity value (0 ,0 ,0 ,0)
- Rely more on specificity than on the order of selectors;
- But, Rely on Order When using 3rd-party stylesheets – always put your author stylesheet last.



اذكر الله ❤

الزموا ذكر الله ففيه العون على متاعب الحياة ❤

سبحان الله وبحمده سبحان الله العظيم

أَلَا بِذِكْرِ اللَّهِ تَطْمَئِنُّ الْقُلُوبُ

By: Amr Elsayed

Specificity in Practice

تعالى ناخذ تمرين على ال **Specificity** عشان نعلمق فهمه اكتر ونوضح بعض حاجات برضو

شوف الكود دا معايا

```
1 <nav id="nav">
2   <div class="pull-right">
3     <a href="#" class="button">Don't Click Here</a>
4   </div>
5 </nav>
```

```
1 .button {
2   font-size: 20px;
3   color: white;
4   background-color: blue;
5 }
6
7 a {
8   background-color: purple;
9 }
10
11 #nav div.pull-right a.button {
12   background-color: orangered;
13 }
14
15
16 #nav a.button:hover {
17   background-color: yellow;
18 }
```

مين الأقوي هنا والي هيطبق ال **Declarations** اللي فيه ؟!

تعالى نحسبها تاني بحسبة بسيطة جدا

تخيل ان عندي اربع حاجات هما اطراف المقارنة وهما (**inline-style, ID, Class, Element**)
يبقى الاربعة حاجات اللي فوق دا هبص علي كل **Block** عندي فوق واشوف ال **Selector** بيحتوي علي
ايه من الاربعة حاجات دي

قبل ما ابدأ المقارنة انا معنديش اي **inline-style** لذلك هو دوما هيكون بصفر **0** في المقارنة دي

اول Selector عندي اللي هو ال **.button**. عبارة عن **class** واحد يبقى المقارنة كالتالي :
(**0** inline-style, **0** id, **1** class, **0** element)

تاني Selector عندي اللي هو ال **a** عبارة عن **Element** واحد يبقى المقارنة كالتالي :
(**0** inline-style, **0** id, **0** class, **1** element)

تالت Selector عندي فيه التالي

- 1- #nav** ادي كدا أول **ID**
- 2- div Element** ادي كدا أول **Element**
- 3- .pull-right** ادي كدا أول **Class**
- 4- a Element** ادي كدا ثاني **Element**
- 5- .button** ادي كدا ثاني **Class**

يبقى المقارنة كالتالي :

(**0** inline-style, **1** id, **2** class, **2** element)

يبقى بشكل مبدئي كدا اللون ال **orangered** في تالت **Selector** هو اللي هيطبق
حتي لو نقلت ال **Selector** دا او ال **Block** دا كدا كله علي بعضه فوق هو برضو اللي هيطبق وقولنا ليه ؟!
عشان ال **Specificity** بتاعته أقوى واحد فيهم

طيب ليه لما جيت أعمل **Hover** تأثير ال **Hover** مظهرش؟! واتحول ل اللون **الاصفر**؟!!

Don't Click Here

عشان برضو هنا تأثير ال **Specificity** في ال **block** بتاع ال **Hover** أضعف من الي الي فوق هتقولي ازاي؟! هقولك تعالي نعمل مقارنة ثاني

ثالث Selector عندي فيه التالي

1. **#nav** ادي كدا أول ID
2. **div Element** ادي كدا أول Element
3. **.pull-right** ادي كدا أول Class
4. **a Element** ادي كدا ثاني Element
5. **.button** ادي كدا ثاني Class

يبقي المقارنة كالتالي :

(0 inline-style, 1 id, 2 class, 2 element)

رابع Selector عندي فيه

- 1- **#nav** ادي كدا أول ID
- 2- **a Element** ادي كدا أول Element
- 3- **.button** ادي كدا أول Class
- 4- **:hover** ادي كدا ثاني Class

يبقي المقارنة كالتالي :

(0 inline-style, 1 id, 2 class, 1 element)

وبالمناسبة ال **Pseudo Class** تصنف ك **Class** وبيتم احتسابها في مقارنة ال **Specificity**

كدا مازال **ثالث Selector** أقوى حتي ال **Hover** مش هيتم تطبيقه غير لو بقا مساوي **لثالث Selector** في القوة



```
1 #nav div.pull-right a.button {  
2   background-color: orangered;  
3 }  
4  
5 #nav div.pull-right a.button:hover {  
6   background-color: yellow;  
7   color: black;  
8 }
```

كدا كافة المقارنة رجحت لحساب **Selector** بالتالي تأثير ال **Hover** هيتم تطبيقه

Don't Click Here

لاحظ انك ممكن لو استخدمت ال **!important** هيكون ليه الأولوية في التنفيذ والتطبيق لأنه زي ما قولنا فوق في الدرس الي قبل دا أول حاجة فوق خالص ان ال **Cascade** أول حاجة بيي بص عليها أو بيديها الأولوية هي ال **!important Flag** لو تفتكر انا برضو هجبلك اسكرين من الجزء دا عشان تفتكر

اول حاجة بيي بص عليها ال **cascade** دا اول ما يجي يشتغل وهي ال **!important flag**
بعد كدا ال **specificity** بعد كدا ال **Source order** (**conflicts**) الي بتحصل منك انت ك **Developer**

بس انا واي حد عموما مش بيفضل استخدام ال **!important Flag** غير في ظروف طارئة جدا والسبب في دا بيرجع لان الكود بتاعك يكون **maintainable** أو قابل للصيانة ديما مع استخدامك لـ **!important Flag** بيكون الوضع صعب شوية خصوصا لو التطبيق كبير جدا وعندك اكثر من عنصر ليهم نفس ال **Specificity**
فبلاش استخدام ال **!important Flag** كتير خصوصا لو انت ك **Developer** الي بتكتب ال **Code** بايدك

صورة مختصرة للمقارنة بشكل توضيحي أفضل فيها بعض الاختلافات عن الكود اللي فوق بس ال **Concept** يعني

Don't click here!		
	(inline, ID, classes, elements)	
<code>.button { font-size: 20px; color: white; background-color: blue; }</code>	(0, 0, 1, 0)	✗
<code>nav#nav div.pull-right .button { background-color: green; }</code>	(0, 1, 2, 2)	✓
<code>a { background-color: purple; }</code>	(0, 0, 0, 1)	✗
<code>#nav a.button:hover { background-color: yellow; }</code>	(0, 1, 2, 1)	✗

اذكر الله

الزموا ذكر الله ففيه العون على متاعب الحياة ♥

لا إله الا الله وحده لا شريك له .. له الملك وله الحمد وهو على كل شيء قدير.

استغفروا الله العظيم الذي لا إله الا هو الحي القيوم واتوب اليه.

اللهم صلّ وسلم وزد وبارك على نبينا محمد ﷺ.

لا إله الا انت سبحانك إني كنت من الظالمين.

أَلَا بِذِكْرِ اللَّهِ تَطْمَئِنُّ الْقُلُوبُ

أفكرونا بدعوة طيبة

How CSS is Parsed, Part 2: Value Processing

في الدرس دا هنفهم ازي ال **Values** بتاعت ال **Properties** بيحصلها تحليل في ال **CSS** كمان هنعرف ازي ال **Percentage %** و ال **CSS Units** زي و ال **em** و ال **rem** و ال **vh** بترجم او بتتسبب ازي في ال **CSS**

طيب في سؤال حلو انا سامعه من الاستاذ اللي واقف هناك دا بيقولي ليه انا محتاج اعرف ال **units** دي بتتسبب ازي؟!

عشان في كل مرة بتستخدم فيها **unit** غير ال **px** وليكن بتستخدم ال **rem** و ال **em** تبقي فاهم ان ال **units** دي بتتحول لل **px** برضو في النهاية بس السؤال هنا بقا المهم جدا هو ازي بتتسبب و ازي بتتحول لل **px** ودا اللي هنناقشه في الدرس دا .. يلا بينا

بص معايا كدا علي الأكواد دي وفي **آخر الدرس** انا هعملك **جدول** لامم كل اللي هيتشرح لحد ما قبل الجدول دا

```
1 <div class="section">
2   <p class="amazing">CSS is absolutely amazing</p>
3 </div>
```

```
1 .section {
2   font-size: 1.5rem;
3   width: 280px;
4   background-color: orangered;
5 }
6 p {
7   width: 140px;
8   background-color: green;
9 }
10 .amazing {
11   width: 66%;
12 }
```

CSS is absolutely
amazing

بالنظر كدا لل **CSS** هلاقي انه عندي **Declaration 6** الي هما **font-size** و **width** و **background-color** و **width** ثاني و **background-color** ثاني و **width** تالت مرة

تعالى نشوف بقا ازاى بيحصلهم **analysis** فى ال **CSS**

تعالى نبدأ بال **p** ونشوف ال **width** الي فيه

بشكل مبدأى كدا انا عندي هنا **conflict** حاصل بسبب انى عملت **Select** لل **p** وعملت **Select** لل **amazing**.
الي هو اصلا **Class** ال **P** فى ال **HTML**

انهو **width** الي هيطبق ؟! لو قولت ال **amazing**. عشان هو آخر حاجة ال **css** هتشوفها وتترجمها هقولك صح بس لو انا نقلت ال **class** دا فوق ال **p** هل هتكون دي اجابتك برضو ؟! اعتقد لا
لان فى سبب تاني لو ركزت شوية هتلاحظ هنا ان ال **specificity** بتاعت ال **class** اقوي من ال **element** لو تفتكر
يبقى ال **66% width** هو الي هيطبق على ال **P** مين الي حدد دا هل انا او انت ؟! لا ال **Cascade** بال **Rules** بتاعته
حدد مين الأقوي ومين ليه الأولوية فى التطبيق

طيب ال **66%** دي زي ما اتفقنا هتتحول لل **px** ازاى دا بيتم بقا .. دا بيتم ع اكر من مرحلة او فى مراحل فى النص كدا
بتحصل زي اى

1- **Declared Value** (Author Declarations)

2- **Cascaded Value** (After The Cascade)

3- **Specified Value** (Defaulting Value If There Is No Cascaded Value)

4- **Computed Value** (Converting Relative Values To Absolute)

5- **Used Value** (Final Calculations, Based On Layout)

6- **Actual Value** (Browser And Device Restrictions)

يبقى دي كل ال **process** الي ال **CSS** بتمشي عليها عشان تحول أى **unit** لل **pixel** فى النهاية وطبعا هنفصصهم
واحدة واحدة

أول مرحلة: ال **Declared Value** ودي المرحلة الي بيكون فيها القيمة الي انت ك **author** او ك **Developer**
كتبتها وفى حالتنا دي هي ال **66%** فى ال **amazing class**. و ال **140px** فى ال **p**

ثاني مرحلة: ال **Cascaded Value** ودي المرحلة الي ال **CSS** بتحل فيها ال **Conflicts** واول ما تشوف **Conflict**
حصل بين ال **amazing**. و ال **p** بتشوف مين الأقوي وتختارة وبناءا عليه ال **66%** هتتطبق

لحد كدا انت معايا وسليم وتمام وكله زي الفل ؟! فاهم صح ؟ تمام

ثالث مرحلة: ال Specified Value

في المرحلة دي لو مفيش اي **Declared Value** بيتحط حاجة اسمها **initial Value** وهنعرف ازاي دي بتيجي او بتيجي منين لما نخلص الخاصية دي بمراحلها ال 6 ونشوف خاصية تانية

في حالتنا وبما ان انا حطيت **value** ك **Developer** يبقى ال 66% هتتطبق مدام فيه **Cascaded Value** هيبقي هتطبق وهتبقى ال **Specified Value** ب 66% هي كمان

رابع مرحلة: ال Computed Value

في المرحلة دي ال **values** اللي بتكون **Relative** او نسبية ببتحول لل **px** بحيث يتم توريثها وهنتكلم بالتفصيل عن التوريث او ال **inheritance** .. كمان في المرحلة دي ال **CSS keywords** زي **orange, bolder** واللي زيتها بيحصل **Compute** وبتتبدل في المرحلة دي

بس عشان ال % تكنيكالي مش **unit** ايون زي ما بقولك كدا ففي الحالة دي هتفضل 66% زي ما هي برضو وهتبتلك حالا ان ال % مش **unit** فعلية

Example:

```
div {width:5%}
```

Percentage is not technically considered a "length unit".

[لينك](#)

[لينك آخر](#)

خامس مرحلة: ال Used Value

ودي بقا المرحلة اللي بيحصل فيها تحويل فعلي من ال % لل **px** في حالتنا احنا عندنا ال **section**. الأب في الكود فوق كان فيه **width** بالقيمة **280px** لذلك ال **amazing**. هي 66% من ال **280px** دول يعني بتساوي **184.8px** هي استمدت قيمتها من ال **Parent Element** الي هو ال **section**. الأب

سادس مرحلة: ال Actual Value

في المرحلة دي ال **CSS** بتكون محددة جدا او **specific** بتقرب القيمة دي لل **185px**

يبقي ال **185px** هي القيمة الفعلية اللي هتتخط في النهاية

تعالى نشوف خاصية تانيه ونشوف ال **value** بتاعتها بيحصلها **parse** ازاى

وليكون تعالى نشوف ال **padding** فى المثال بتاعنا هتقولى استنى بس انا مكتبتش **padding** اصلا هقولك صحيح بس انا عايز تعرف ان فى خواص انت مش بتكتبها بيتحطها قيم معينة فعليا بشكل **default** فتعالى نعدي ال **padding** اللى مش موجود دا اصلا على ال 6 مراحل اللى عندي فوق

السبب فى الكلام دا ان كل **element** فى الصفحة بيكونله كل **CSS property** محتاجه يكون ليها **Value** حتى لو انت معملتهاش **Declaration**

فتعالى نشوف ال **padding** اللى احنا متكتبتهوش دا مع ال 6 مراحل بتوعنا

أول مرحلة: ال **Declared Value** مفيش خالص اى **Declared Value**

تاني مرحلة: ال **Cascaded Value** مفيش خالص اى **Cascaded Value** لان مفيش **declaration** حصل

ثالث مرحلة: ال **Specified Value** انا هبقى هنا بقا **Specified Value** او ال **default** ودا بسبب ان كل واي **CSS property** بيكون ليها **initial value** وببساطة دي بتكون ال **Value** الفعلية لو مفيش **Cascaded Value** ودا اللى انا قولتلك هنجيلها كمان شوية فوق لما كنت بتكلم فى الخاصية اللى فاتت فى الحالة دي (ال **initial Value** هتكون بـ **0px**) وبينى وبينك التوريث بيكون ليه عامل هنا بس هنتعرف على دا بالتفصيل قدام

رابع مرحلة: ال **Computed Value** بشكل فعلى ال **0px** هي **absolute unit** يعنى هو **0px** وال **unit** بتاعته **px** فعليا

لذلك المرحلة دي واللى بعدها هتفضل **0px** او خلاص مش محتاجينهم

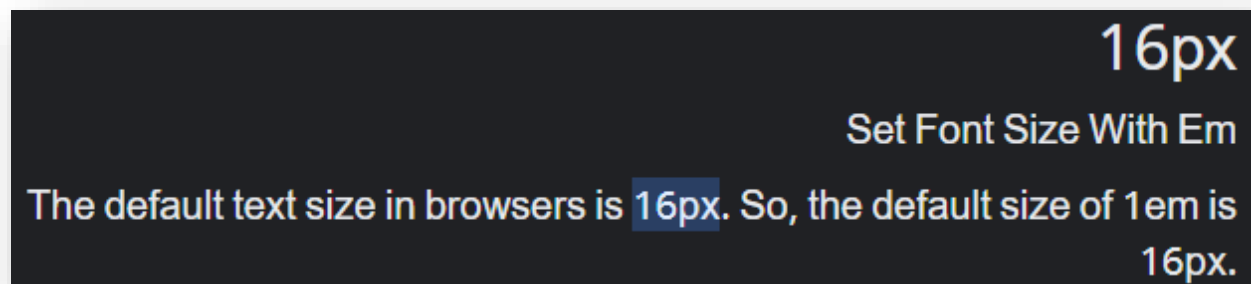
خامس مرحلة: ال **Used Value** هتكون **0px**

سادس مرحلة: ال **Actual Value** هتكون **0px**

تعالى نشوف ال **font-size** اللى موجود فى ال **section**. واللى هو **1.5rem** بس قبل ما نشوف دي تعالى نناقش حاجة تانيه وهنرج لل **font-size** اللى موجود فى ال **section**.

احنا محتاجين نعرف ال **font-size** بتاع ال **root** بتاعنا بتكون **16px** بشكل تلقائي زي ما انا وانت والدنيا كلها عارفه ان ال **Browsers** بتحط **Default Font-Size** بالقيمة **16px**

يبقى ال **root** ال **font-size** بتاعه بيبكون **16px** ودا برضو هيعدي ع ال **6** مراحل بتوعنا



أول مرحلة: ال **Declared Value** مفيش خالص اي **Declared Value** لانها **16px** من ال **browser** نفسه

تاني مرحلة: ال **Cascaded Value** هنا الحالة دي مختلفة عن ال **padding** هنا فيه **Cascaded Value** طيب هتقولي ازاي ؟!

هقولك فاكرك لما قولنا ان ال **CSS** بتيجي من أكثر من مصدر ؟ من ضمن المصادر دي اللى هو ال **user agent stylesheet** واللى هي هنا فرضت ال **font-size** لل **page** ب **16px** ك **default Value**

ثالث مرحلة: ال **Specified Value** بتتحط ك **initial value** لو مفيش **Cascaded value** وبما ان فيه **Cascaded Value** جاية من ال **user agent stylesheet** يبقى ال **Specified Value** هتكون ب **16px** هي كمان

رابع مرحلة: ال **Computed Value** هتكون **16px**

خامس مرحلة: ال **Used Value** هتكون **16px**

سادس مرحلة: ال **Actual Value** هتكون **16px**

تعالى بقا نشوف ال **font-size** اللى موجود فى ال **section**. واللى هو **1.5rem** بعدنا ما عرفنا ال **font-size** لل **root**

```
.section {  
  font-size: 1.5rem;  
}
```

يلا نعديه ع ال 6 مراحل

أول مرحلة: ال **Declared Value** واللى هي **1.5rem** و ال **rem** هي **Relative unit** لذلك هتتحول لل **px**

ثاني مرحلة: ال **Cascaded Value** واللى هي هيتم قرأتها ك **1.5rem** برضو

ثالث مرحلة: ال **Specified Value** هتكون **1.5rem** مدام فيه **Cascaded Value** يبقى ال **Specified value** هتكون زي ال **Cascaded Value**

رابع مرحلة: ال **Computed Value** هتكون **24px** ... اذااي بقا .. تعالي اقولك مش انت سألتني ازاي سيبيني أجابو بقا سعتك: "D"

ال **rem unit** بتكون نسبية او **Relative** لل **root font-size** يعني هي بتستمد قيمتها من ال **root** وبما ان ال **root** له **Font-size** بشكل **default** بـ **16px**

يبقى ال **section**. ال **Font-size** بتاعه هيكون **1.5 * 16px** بتساوي **24px**

خامس مرحلة: ال **Used Value** هتكون **24px**

سادس مرحلة: ال **Actual Value** هتكون **24px**

تعالى نشوف ال **font-size** اللي موجود في ال **p** هتقولي بس انا محطتش **font-size** لل **p Element** ولا حتي لل **class** اللي في ال **p** اللي هو ال **amazing**. هقولك انا ثانية واحدة بس انت سألت نفسك الأول ازاى ال **p** ال **text** اللي فيه ليه **font-size** او ظاهر اصلا مع انك فعليا مكتبتش اي **Font-size**؟! صح مش كدا ولا اي؟!

في الحالة دي بقا دا لا هو فيه **default value** محطوة لل **p** عشان يظهر كدا ولا فيه اي **initial Value** هتقولي الله!! انا كدا احتارت اومال ال **p** دا واخد **Font-size** وظاهر كدا منين وليه هو ال **font-size** بتاعه اللي محطوطله مخليه شكله كدا؟! يعني ليه هو مش اصغر من كدا وليه مش أكبر من كدا ليه هو شكله كدا تحديدا؟! دا حصل بسبب ميكانيزم تاني اسمه ال **inheritance** او التوريث ال **p** دا يا صديقي ورث **حجم الخط** اللي هو فيه دا من الأب بتاعه اللي ال **section**. اللي هو قيمته بـ **1.5rem** اللي فعليا بتساوي **24px** اكبر دليل علي كلامي روح علي ال **p** دا حطله **font-size** بالقيمة **24px** من عندك مش هتلاقي اي تغير حصل ليه؟ عشان هو ورث **حجم الخط** دا من الأب بتاعه اللي هو ال **section**.

طيب ليه بيحصل الميكانيزم دا؟! لان ببساطة تخيل انك عندك اكر من **p** جوا الأب بتاعهم انك تديهم كلهم **font-size** حاجة مش عملية خالص ورخمة لذلك هي **بترث من الاب** اللي هي فيه وهنتكلم عن التوريث بالتفصيل وازاي بتم برضو متقلقش

ما تيحي نبص ع ال 6 مراحل بتوعنا

أول مرحلة: ال **Declared Value** مفيش اي **Declared Value**

ثاني مرحلة: ال **Cascaded Value** مفيش اي **Declared value** عشان يحصلها **Cascade** بالتالي مفيش اي **Cascaded Value**

ثالث مرحلة: ال **Specified Value** هتكون **24px** بطريقة ال **inheritance** من الأب

رابع مرحلة: ال **Computed Value** نفس ال **Value** بـ **24px**

خامس مرحلة: ال **Used Value** نفس ال **Value** بـ **24px**

سادس مرحلة: ال **Actual Value** نفس ال **Value** بـ **24px**

تعالى بقا أعملك الجدول اللى هملك فيه كل الكلام اللى فات دا

	width (paragraph)	padding (paragraph)	font-size (root)	font-size (section)	font-size (paragraph)
1- Declared Value (Author Declarations)	140px 66%	-	-	1.5rem	-
2- Cascaded Value (After The Cascade)	66%	-	16px (Browser default)	1.5rem	-
3- Specified Value (Defaulting Value If There Is No Cascaded Value)	66%	0px (initial value)	16px	1.5rem	24px
4- Computed Value (Converting Relative Values To Absolute)	66%	0px	16px	24px (1.5 * 16px)	24px
5- Used Value (Final Calculations, Based On Layout)	184.8px	0px	16px	24px	24px
6- Actual Value (Browser And Device Restrictions)	185px	0px	16px	24px	24px

INHERTANCE

وخليني الخصلك كل اللى فات دا في كلمتين كدا هنا برضو

- كل **Property** بيكون ليها **initial value** بيتم اللجوء اليها لو مفيش اي **Declared values** او مفيش اي **inheritance**

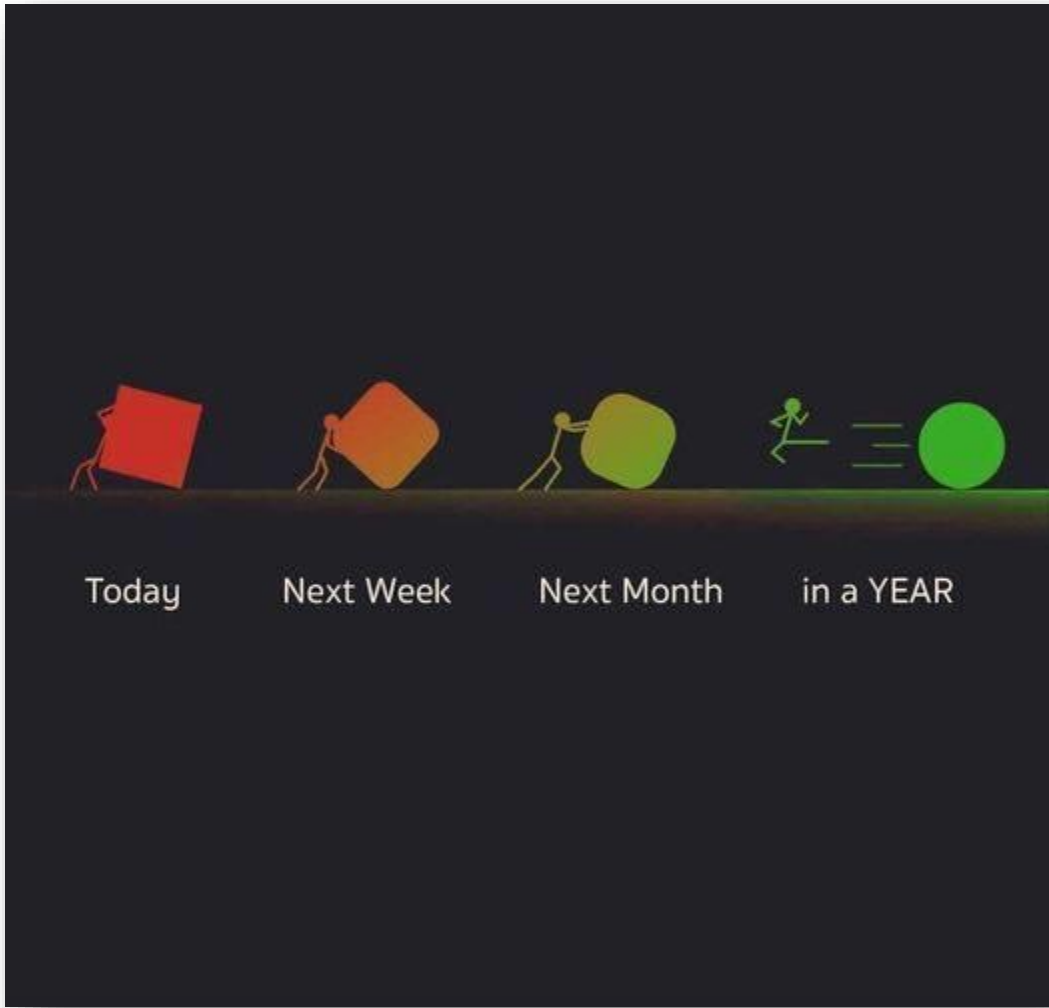
- ال **Browsers** بتخصص **Default font-size** لل **Page** بالقيمة **16px**

- النسبة المئوية % وكل ال **الواحدات النسبية** او ال **Relative Units** بيتم تحويلها لل **pixels**

أفتكرونا بدعوة طيبة

By: Amr Elsayed

❤ عبارات تحفيزية ❤



لو أن الناس كلما استصعبوا أمرا تركوه ماقام للناس دنيا ولادين

-عمر بن عبدالعزيز

❤ اذكر الله

❤ الزموا ذكر الله ففيه العون على متاعب الحياة

سبحان الله و الحمد لله و لا اله الا الله و الله أكبر و لا حول ولا قوة الا بالله

اللهم صل وسلم وزد وبارك على نبينا محمد ﷺ

اللهم إني أستغفرك واتوب إليك، لا إله الا انت سبحانك إني كنت من الظالمين

ألا بِذِكْرِ اللَّهِ تَطْمَئِنُّ الْقُلُوبُ

By: Amr Elsayed

How Units Are Converted From Relative To Absolute (px) - Value Processing

تعالى بقا نفهم بالتفصيل الشديد ازاي ال **Engine** بتاع ال **CSS** بيحول ال **Relative Units** زي ال **ems** وال **rem** وال **vw** وال **vh** لل **Pixels** في المرحلة الرابعة (ال **Computed Value**) و المرحلة الخامسة (ال **Used value**)

بص معايا ع الكود دا كدا

```
1 html, body {
2   font-size: 16px;
3   width: 80vw;
4 }
5
6 header {
7   font-size: 150%;
8   padding: 2em;
9   margin-bottom: 10rem;
10  height: 90vh;
11  width: 1000px;
12}
13
14.header-child {
15  font-size: 3em;
16  padding: 10%;
17}
```

بشكل مبدأى عايز أقولك انه فيه فرق بين استخدام **النسب المئوية % للخطوط** و استخدامها لقياسات الطول والعرض والمسافات وهنهم دا حالا بالتفصيل الشديد

وخليني افكر ان ال **%** مش **unit** بشكل تيكنيكال هي مش **unit** بس عايزك تعتبرها كدا المرادي عشان الشرح

اول حاجة خليني امسك ال **header** انا فيه عملت **declaration** لل **font-size** بـ 150% ودا معناه انه ال **header** هيبص ع الأب بتاعه اللي هو ال **body** و ال **html** هلاقي ال **font-size** بـ 16px

فدا معناه ان $150\% * 16px$ بيساوي 24px

يبقي ال **header** ال **font-size** بتاعه هيكون 24px

ولو تفتكر من **خمس ست سطور** قولتلك انه فيه فرق بين استخدام **النسب المئوية % للخطوط fonts** واستخدامها **لقياسات الطول والعرض** ففي ال **header** انا استخدمت النسبة المئوية مع ال **fonts**

تعالى نشوف الفرق بقا اثناء استخدامها للطول والعرض

ملحوظة: لما بنستخدم ال **%** مع ال **length** زي ال **height** وال **width** وال **padding** وال **margin** بيكون التعامل هنا مختلف عن استخدام ال **%** مع ال **fonts** الفرق هنا بيكون كالتالي

ال **length** لما بتستخدم فيه ال **%** وهي بتحسب ال **computed Value** بيكون المرجع بتاعها ال **width** بتاع ال **parent - الاب** اللي ال **Element** دا نفسه موجود فيه

اكيد انت مش فاهم حاجة وانا هشرحك دلوقتي بالتفصيل

تعالى نشوف ال **padding** اللي موجود في ال **header-child**. اللي هو بالقيمة 10%

```
.header-child {  
  padding: 10%;
```

هيحسب قيمة ال **padding** دا من ال **width** بتاع ال **Parent** بتاعه اللي هو ال 1000px

```
header {  
  width: 1000px;
```

فدا معناه ان $10\% * 1000px$ بيساوي 100px

ومرة ثانية بأكد فيه فرق بين استخدامك لل **%** وال **units** كلها عموما في ال **fonts** زي ال **font-size** وبين استخدامك ليهم في ال **length** زي ال **padding** وال **margin** وال **height** وهكذا هتفهم اكرر قدام

ال **ems** وال **rems** هما **font-based units** يعني ديما بتبص ع حجم ال **font-size** هو دا مرجعها

اي الفرق بين ال **em** وال **rem**

ال **em** بيكون ال **reference** بتاعها ال **element** نفسه في بعض الحالات او ال **parent** بتاع ال **element** دا
ال **rem** بيكون ال **reference** بتاعها ال **root font-size**

عندي في الكود اللي اكتب فوق فيه **em** موجودة في ال **font-size** بتاع ال **header-child**.

```
.header-child {  
  font-size: 3em;
```

ال **css** هتحتها ازاى؟! اتفقنا ان ال **em** بيكون المرجع بتاعها يا **العنصر نفسه** يا **الاب اللي هو فيه**
وبما ان الاب اللي العنصر دا فيه ال **font-size** بتاعه **150%** اي **24px** يبقى ال **3em** هتكون كالتالي
24px * 3em بيساوي **72px** يبقى ال **header-child** ال **font-size** بتاعه ب **72px**

فيه **em** موجودة في ال **padding** بتاع ال **header**

```
header {  
  padding: 2em;  
  font-size: 150%;
```

ال **padding** دا **length** هنا بتكون فيه حسابات تانيه عن ال **fonts** هنا ال **padding** ب **2em** هياكون المرجع بتاعه
هو ال **font-size** بتاع ال **element** دا نفسه هنا المرجع بتاعها هياكون العنصر نفسه تحديد ال **font-size** بتاع
العنصر

وبما ان العنصر دا فيه ال **font-size** ب **150%** اي **24px** يبقى ال **2em** هتكون كالتالي
24px * 2em بيساوي **48px**

هتقولي طيب افرض ال **header** دا مكنش فيه ال **font-size** ب **150%** الي هي **24px** كان ال **2em** دي هتتحتسب
ازاي؟!

هقولك يا معلم ركز ال **em** بيعتمد ع ال **font-size** بس في **حالتين** الحالة الأولى لو فيه ال **font-size** في ال **element**
نفسه يبقى هو هيعتمد عليه لو مفيش يروح يعتمد ع ال **font-size** بتاع ال **parent** بتاعه

لو بقا ال **header** مفهوش **font-size** كان ال **2em** دي هتروح تشوف ال **root** او ال **Body** ال **font-size** فيه بكام
وتحسب علي اساسه

وفيه **rem** موجودة في ال **margin-bottom** بتاع ال **header**

```
.header {  
  margin-bottom: 10rem;  
}
```

زي ما قولنا ال **rem** بيكون ال **reference** بتاعه ال **font-size** بتاع ال **root** في الحالة دي هو **16px**
إذا $10\text{rem} * 16\text{px}$ بيساوي **160px**

ال **ems** وال **rems** هما **font-based units** يعني ديما بتبص ع حجم ال **font-size** هو دا مرجعها

اي الفرق بين ال **em** و ال **rem**

ال **em** بيكون ال **reference** بتاعها ال **element** نفسه في بعض الحالات او ال **parent** بتاع ال **element** دا

فاكر لما قولتلك **السطرين اللي فوق** دول انا هنا كان قصدي اي؟! تعالي اديك مثال عشان تفهم قصدي
لنفترض ان هغير ال **padding** اللي في ال **header-child**. دا من **10%** ل **2em**

```
.header-child {  
  font-size: 3em;  
  padding: 2em;  
}
```

ال **padding** هنا هيتحسب ازاي؟!

انا قولتلك اي .. قولتلك حاجتين ال **ems** بتعتمد ع ال **font-size** وقولتلك ان المرجع بتاعها بيكون يا ال **element** نفسه في حالة وجود **font-size** فيه ... يال **parent** .. هنا مدام فيه **font-size** في ال **element** اللي هي فيه هتبص علي ال **font-size** بتاع ال **element** نفسه

يبقي ال **padding** هنا هيكون كالتالي

ال **font-size** بتاع ال **element** هو **3em** و ال **3em** بتاعته دي في الحالة دي اعتمدت ع ال **parent** اللي هو **150%** اي **24px** فال **Font-size** هنا كان $24\text{px} * 3\text{em}$ ب **72px**

وبما ان ال **font-size** بتاع ال **element** ب **72px** يبقي ال **padding** بال **em** هيكون المرجع بتاعها ال **font-size** دا
يبقي ال $72\text{px} * 2\text{em}$ بيساوي **144px**

عارف انا لو شيلت ال **font-size** اللي في ال **element** دا ال **padding** هيتسحب ازاي؟! فكر ثواني كدا

ال **padding** بالقيمة **2em** هيكون المرجع بتاعه ال **font-size** بتاع ال **parent** اللي هو **150%** اي **24px**
فهتكون الحسبة كالتالي $24\text{px} * 2\text{em}$ بتساوي **48px** ... انت فاهم صح اكيد فاهم مفياش نقاش

انت دلوقتي ممكن تكون بتسأل نفسك سؤال مهم جدا ليه استخدام ال **ems** و ال **font-size** وهي معتمدة علي ال **Font-Size**؟! ... عشان في ال **layouts المعقدة والكبيرة** بمجرد ما تغير ال **font-size** لعنصر ما ال **padding** و ال **margin** و ال **length** بتاعه هيتغير بشكل تلقائي لانه معتمد ع ال **font-size** ودي بيخلي التصميم بتاعك فيه **flexibility** كبيرة جدا وجامد جدا ومبني كله علي بعضه واي تغير بتعمله بيخلي التصميم مرن

واخيرا عندنا اخر اثنين **Relative units** اللي هما ال **vh** و ال **vw** بشكل مبدئي ال **1vh** بيساوي **1%** من ال **viewport height** بتاع الشاشة

```
.header {  
  height: 90vh;
```

فدا معناه ان ال **90vh** دي هي بمثابة **90%** من ال **current viewport height**
90vh * 1% بيساوي **90%**

وال **1vw** بيساوي **1%** من ال **viewport width** بتاع الشاشة

```
html, body {  
  width: 80vw;
```

فدا معناه ان ال **80vw** دي هي بمثابة **80%** من ال **current viewport width**
80vw * 1% بيساوي **80%**

ال **viewport** دا بيتحدد من ال **browser** نفسه

HOW UNITS ARE CONVERTED FROM RELATIVE TO ABSOLUTE (PX)

S

	Example (x)	How to convert to pixels	Result in pixels
Font-based	% (fonts)	150%	→ 24px
	% (lengths)	10%	→ 100px
	em (font)	3em	→ 72px (3 * 24)
	em (lengths)	2em	→ 48px
	rem	10rem	→ 160px
Viewport-based	vh	90vh	→ 90% of the current viewport height
	vw	80vw	→ 80% of the current viewport width

```
html, body {
  font-size: 16px;
  width: 80vw;
}

header {
  font-size: 150%;
  padding: 2em;
  margin-bottom: 10rem;
  height: 90vh;
  width: 1000px;
}

.header-child {
  font-size: 3em;
  padding: 10%;
}
```

Shamy

وخليني الخصلك كل اللي فات دا في كلمتين كدا هنا برضو

- كل **Property** بيكون ليها **initial value** بيتم اللجوء اليها لو مفيش اي **Declared values** او مفيش اي **inheritance**
- ال **Browsers** بتخصص **Default font-size** لل **Page** بالقيمة **16px**
- **النسبة المئوية %** وكل ال الواحدات النسبية او ال **Relative Units** بيتم تحويلها لل **pixels**
- **النسبة المئوية %** في ال **child** لو بتستخدمها في ال **font-size** بيكون قياسها **relative** لل **Font-size** بتاع الأب
- **النسبة المئوية %** بتكون **Relative** لل **Width** بتاع الأب لو بتستخدمها مع اي **lengths** زي ال **padding** مثلا
- ال **em** بيتم قياسها نسبةً لل **font-size** بتاع الأب لو بتستخدمها في خاصية ال **font-size** لل **child**
- ال **em** بيتم قياسها نسبةً لل **current font-size** بتاع ال **element** نفسه لو بتستخدمها مع ال **lengths**
- ال **rem** ديموا وابدأ بتكون **Relative** او بيتم قياسها نسبةً لل **font-size** بتاع ال **root element**
- ال **vh** و ال **vw** هما ببساطة ال **view port's height and width respectively** يعني بيتم تحديدهم بناءً علي ال **viewport** بتاع الشاشة نفسها والمتصفح اللي بيحدها

How CSS is Parsed, Part 3: Inheritance

اتكلمنا عن ال **inheritance** في الدروس الي فاتت وجيه الوقت عشان نشوف ال **css** بتعامل ازاى مع ال **inherit** ال **inheritance** أو التوريث هي طريقة بيتم فيها توريث بعض الخصائص والقيم من الأباء للأبناء

شوف معايا الكود دا كدا وتعالى نشوف ال **line-height** ال **child**. هيوثره ازاى من ال **parent**.

```
1 .parent {  
2   font-size: 20px;  
3   line-height: 150%;  
4 }  
5  
6 .child {  
7   font-size: 25px;  
8 }
```

في كذا سيناريو محتاجين ندرسهم الأول

كل **CSS Property** لازم يكون ليها **Value**

السيناريو الأول وهو ان ال **CSS** بتسأل سؤال هل فيه **Cascaded Value** ؟!

لو أه <= فيه يبقى ال **specified Value** هتكون هي ال **Cascaded Value**
وخلي بالك ال ال **specified Value** اللي هي بتكون ال **init value** لو مفيش **Cascaded Value**
وال **Cascaded Value** بتتوجد لما يكون فيه **Declared Value** منك انت ك **Developer**

لو لأ <= هيحصل سيناريوهات ثانية

السيناريو الأول: لو مفيش **Cascaded Value** هل ال **Property** دي **inherited** ؟!

لو أه <= يبقى ال **specified Value** هتبقى ال **Computed Value** بتاعت ال **Parent**

لو لأ <= يبقى ال **specified Value** هتتخط بال **init value**

طيب في المثال بتاعنا في الكود الي فوق
ال CSS هتسأل فيه Cascaded Value هنا لا line-height في ال child. ؟!

لا مفيش

يبقي هنسأل سؤال ثاني هو ال line-height دا خاصية من الخواص الي ممكن تتورث عادي ؟!
لو الاجابة أه وهي أه فعلا

يبقي ال child. دا هيكون ال line-height بتاعه ب 30px

ودا حصل من خلال أن ال line-height بتاع الاب الي ب 150% لما يتحول لا px

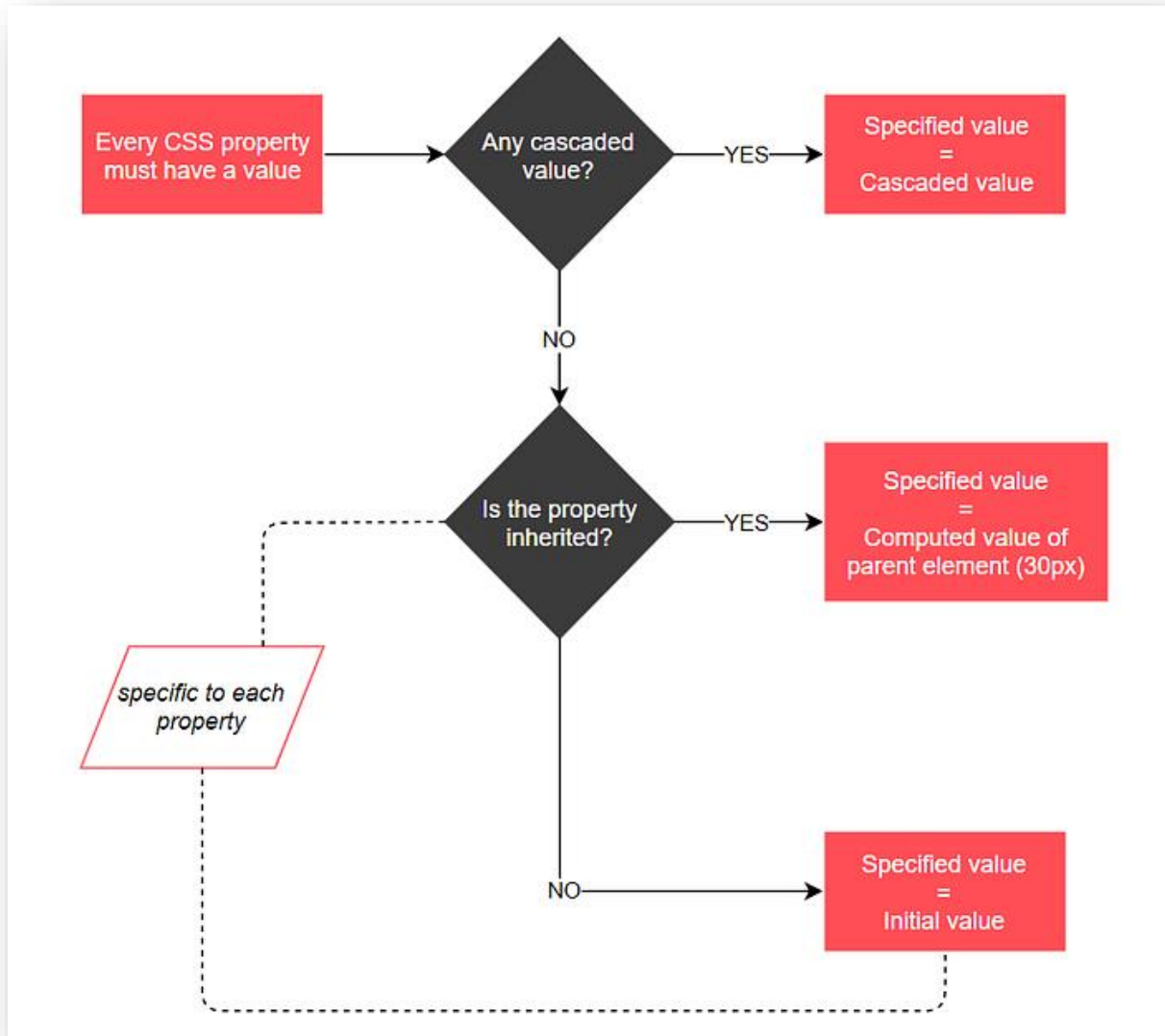
هيتعمد علي ال Font-size بتاع الاب الي هو 20px

فيبقي المعادلة كتالي $20px * 150\%$ بتساوي 30px

يبقي ال line-height بتاع ال parent. ب 30px

وال line-height من الخواص الي بيتم توريثها عادي وال child. هيورثها يبقي ال line-height بتاع ال child.
هيكون هو كمان ب 30px

مخطط توضيحي للفكرة الي نقشناها فوق



- ال **child**. سيكون فيه كل خواص ال **css** عادي بس بال **init value** بتاعتها وفي المثل بتاعنا ال **child**. موجود فيه ال **line-height** بال **init value** بـ 0 بس عشان ال **line-height** بقا من الخواص اللي بتتورث عادي من الاباء للأبناء فال **line-height** هيتم توريثته لل **child**. فهنا ال **init value** اللي بـ 0 هتتشال ويتحط مكانها نفس القيمة اللي في ال **parent**. اللي هي **30px** وحسبناها فوق
- في خواص كتير مبيتمش توريثها زي ال **padding** مثلا وخصائص تانية كتير وتقدر تشوف الخواص اللي بيتم توريثتها والخصائص اللي لا
- ال **inheritance** هو امداد الابناء ببعض الخواص من الأباء لو الخواص دي قابلة انها تتورث ودا بشكل أو بآخر بيخليك تكتب كود اقل وبيحقق شرط من الشروط اللي اتكلمنا عليها في أول صفحة من الملخص وهي ان الكود بتاعك يكون **more maintainable code**
- كل ال **properties** اللي متعلقة بال **text** بتكون **inherited** زي ال **font-size** وال **color** وال **font-family** وكدا وكل الخواص زي ال **padding** وال **margins** مش بيتم توريثها لانه فقط تخيل انك عندك **Section** مديله **padding** بـ **60px** فتخيل لو كل عنصر جو ال **section** دا ورث ال **padding** دا !! شئ فظيع جدا طبعا: "D"
- خلي بالك ان ال **computed value** اللي القيمة بتاعتها بال **px** في شكلها النهائي هي اللي بيتم توريثها وليس ال **declared values**
- ال **inheritance** **بيشتغل** في الابناء لو مفيش **Declared value** للخاصية اللي تم توريثها دي يعني لو انت كتبت في ال **parent** خاصية بيتم توريثها وروحت للابن كتبت نفس الخاصية بقيم تانية هنا ال **inheritance** **مش هيشغل**
- ممكن تستخدم ال **inherit keyword** عشان **تجبر** اي خاصية مكتوبة في الابن انها **تورث** من الأب
- ممكن تستخدم ال **initial keyword** عشان تعمل **reset** لأي **property** لل **init value** بتاعتها

أفكروننا بدعوة طيبة

♥ راقّت لي ♥



Hasan Mattar ✓
@HasanmMattar

...

جرب أدوات الذكاء الاصطناعي المجانية هذه:

1. دردشة ChatGPT - للبحث
2. أداة QuillBot - للتدقيق الإملائي والنحوي
3. أداة StoryLab - لكتابة العناوين والخطوط العريضة
4. أداة Hemingway - للإيجاز والوضوح
5. أداة Tweet Hunter - لإنشاء المحتوى
6. أداة playgroundai لتحويل النص إلى صور
7. أداة tutorai مدرس شخصي لأي شيء تريده

اذكر الله ♥

الزموا ذكر الله ففيه العون على متاعب الحياة ♥

لا إله إلا الله وحده لا شريك له .. له الملك وله الحمد وهو على كل شيء قدير

أَلَا بِذِكْرِ اللَّهِ تَطْمَئِنُّ الْقُلُوبُ

لا تنسونا من دعواتكم

Never Forget That (**Always Search**)

By: Amr Elsayed

How CSS Renders a Website: The Visual Formatting Model

بعد ما اتكلمنا عن ال **Parsing Phase** او مرحلة تحليل ال **CSS** لل **code** دلوقتي نتكلم فيه عن ال **Website Render Phase** أو مرحلة ظهور ال **Website** بشكله المرحلة دي بيتم استخدام فيه آلية عمل اسمها **Visual Formatting Model**

ماهو ال **Visual Formatting Model**

DEFINITION

Algorithm that calculates boxes and determines the layout of these boxes, for each element in the render tree, in order to determine the final layout of the page.

هو ببساطة **algorithm** بيستخدم لحساب ال **Boxes** و تحديد ال **layout** لكل ال **boxes** دي لكل عنصر بيحصله **render** بالاضافة لتحديد ال **final layout of the page** يعني هو اللي بيحسب قياسات ال **box-model** وال **Floats** وال **flex** وال **positions** وال **layout** بشكل عام

In summary, the Visual Formatting Model provides the foundation for rendering and positioning elements on a web page, enabling developers to create visually appealing and well-structured web layouts.

The layout of these boxes is governed by:

- **Box dimensions** (the box model)
- **Box type** (inline, block, inline-block)
- **Positioning scheme** (normal flow, float, and absolute positioning).
- **Stacking-context**
- **Relationships between elements in the document tree.**
- **External information** (e.g., viewport size, intrinsic dimensions of images, etc.).

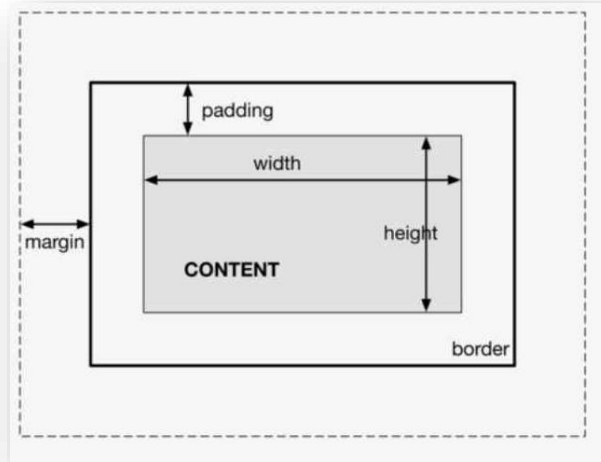
يبقى ال **Visual Formatting Model** بوظيفة هو حساب كل النقاط اللي اتذكرت فوق دي وهنتعرف عليهم واحد واحد دلوقتي

Box dimensions (the box-model)

ال **Box-model** هو واحد من أهم مبادئ لغة ال **CSS** وهو حاجة لازم تتقنها لتصميم صفحة ويب.

ال **Box-model** هو أحد العوامل الي بتحدد ازاى بيتم عرض العناصر على صفحة الويب وكيفية تحديد حجمها. وفقاً لل **Box-model**، يمكن اعتبار كل عنصر على صفحة الويب عبارة عن مربع مستطيل.

ويمكن لكل مربع أن يحتوي على **width** و **height** و **padding** و **margins** و **border**



ومع ذلك، لاحظ أن كلهم ممكن يتوجدو بشكل اختياري، عشان ممكن يكون فيه عناصر او **boxes** بدون **margins** أو **padding** او **border**

ومن خلال فهم واستخدام ال **Box-model** ، يمكنك إنشاء **layouts** متجاوبة وجذابة من الناحية البصرية. عشان بيمكنك من التحكم في حجم العناصر وتحديد مواقعها وتباعدها، وتنفيذ **layouts** مرنة بتتكيف مع مقاسات الشاشات المختلفة.

أزاى ال **Box-model** بيحسب ال **width** و ال **height** بتاع العنصر

total width = right border + right padding + specified width + left padding + left border

total height = top border + top padding + specified height + bottom padding + bottom border

Example: calculate height

=

0px top border + 20px top padding + 100px specified height +
20px bottom padding + 0px bottom border

=

140px

مش ملاحظ حاجة اننا لو حددنا **width** معين أو **height** معين زي ال **100px** اللي في المثال بنلاحظ ان ال **padding** و ال **border** ييزودو من حجم ال **height** دا وييتحسبوا معاه؟! بس دا بشكل ما مش حاجة كويسة

عشان كذا الحل اللي بنسخدمه عشان نمنع الموضوع دا هو خاصية ال **box-sizing** بالقيمة **border-box** لانك لما تحدد **height** معين وليكن **100px** وتزود عليه **padding-top** و **padding-bottom** بـ **20px** لكل واحد

لما بتستخدم ال **box-sizing** هو بخلي العنصر دا دا ال **height** بتاعه ككل بال **100px** لانه بيخلي ال **height** الفعلي بـ **60px** وبيحسب ال **padding** بـ **40px** فيكون المجموع الكلي للعنصر دا **100px** فقط

Example: **calculate height after using box-sizing**

$$\begin{aligned} &= \\ &0\text{px top border} + \cancel{20\text{px top padding}} + \boxed{100\text{px specified height}} + \\ &\quad \cancel{20\text{px bottom padding}} + 0\text{px bottom border} \\ &= \\ &100\text{px} \end{aligned}$$

Box type (inline, block, inline-block)

ال **type** او نوع ال **box** بيتحدد من خلال خاصية ال **display** وكل عنصر ال **html** بيكون عليه **default display** زي ال **div** بيكون **block** وزى ال **span** بيكون **inline** وهكذا

وطبعا انت مدام بتقرا الملخص دا فانت شخص فاهم **CSS** كويس وبتزود معلوماتك مفيش داعي اشرحلك الفرق بينهم انا هكتفي بصورة فقط فيها مقارنة بينهم ابقى أقرأها يعني

Block

- بياخد ال **full-width** لو لم يتم تحديد **width**
- بيعمل خط فاصل بينه وبين العناصر اللي هتيجي تحته عشان يمنعها تيجي جنبه
- بيحترم ال **padding** - و ال **margin** - و ال **Width** - و ال **height** بمعنى انه بينفذ أي أمر ليهم

Inline

- لا يحترم ال **Width** و ال **Height**
- لا يضيف خط فاصل بينه وبين العناصر اللي تحته .. يعني أي عنصر هيجي جنب الثاني
- بيحترم ال **padding** و ال **margin** [من اليمين و الشمال فقط]
- بيسمح بوجود أي عنصر قبله وبعده

Inline-Block

- بيحترم ال **padding** - و ال **margin** - و ال **Width** - و ال **height** بمعنى انه بينفذ أي أمر ليهم
- بيسمح بوجود أي عنصر قبله وبعده

2. BOX TYPES: INLINE, BLOCK-LEVEL AND INLINE-BLOCK

S

Block-level boxes

- Elements formatted visually as blocks
- 100% of parent's width
- Vertically, one after another
- Box-model applies as showed

```
display: block
(display: flex)
(display: list-item)
(display: table)
```

Inline-block boxes

- A mix of block and inline
- Occupies only content's space
- No line-breaks
- Box-model applies as showed

```
display: inline-block
```

Inline boxes

- Content is distributed in lines
- Occupies only content's space
- No line-breaks
- No heights and widths
- Paddings and margins only horizontal (left and right)

```
display: inline
```

Positioning scheme (normal flow, float, and absolute positioning).

normal flow

البيكون ال **default flow** بتاع ال **element** اللي هو **position relative** او **position static** مش بيكون **Floated** ومش بيكون **absolute** ال **element** نفسه بيظهر بشكل الطبيعي من المصدر

Floats

ال **element** مع ال **Float** بتشال من حالة ال **normal flow** بتاعته ال **text** و ال **inline elements** بتكون متحاوطة حولين ال **floated element** ال **floated element** مش بيكون عندي وعي بحساب ال **height** المناسب ليه او انه يفصل بينه وبين العناصر اللي تحته وفوقه لذلك بنستخدم ال **clear property** بالقيمة **both**

absolute positioning

ال **element** مع ال **absolute positioning** بتشال من حالة ال **normal flow** بتاعته مش بياثر علي المحتوى ولا علي العناصر المحيطة بيه بتقدر تستخدم معاه ال **top** و ال **bottom** و ال **left** و ال **right** عشان تحدد موقع ال **element** والحاجات دي بتحدد نسبة لكون الاب بتاعه **relative**

3. POSITIONING SCHEMES: NORMAL FLOW, ABSOLUTE POSITIONING AND FLOATS

S

Normal flow

- Default positioning scheme;
- NOT floated;
- NOT absolutely positioned;
- Elements laid out according to their source order.

Default
position: relative

Floats

- Element is removed from the normal flow;
- Text and inline elements will wrap around the floated element;
- The container will not adjust its height to the element.

float: left
float: right

=

≠

Absolute positioning

- Element is removed from the normal flow
- No impact on surrounding content or elements;
- We use top, bottom, left and right to offset the element from its relatively positioned container.

position: absolute
position: fixed

MDN Web Docs
MDN Web Docs

Scrimba

في ال **absolute positioning** في عناصر بتيجي فوق عناصر فال **CSS** بتستخدم حاجة اسمها **Stacking-context**

Stacking-context

ال **Stacking-context** ببساطة شديدة هو الي بيحدد ازاي ال **Elements** بتظهر فوق بعض ازاي ودا تعريفه بشكل دقيق

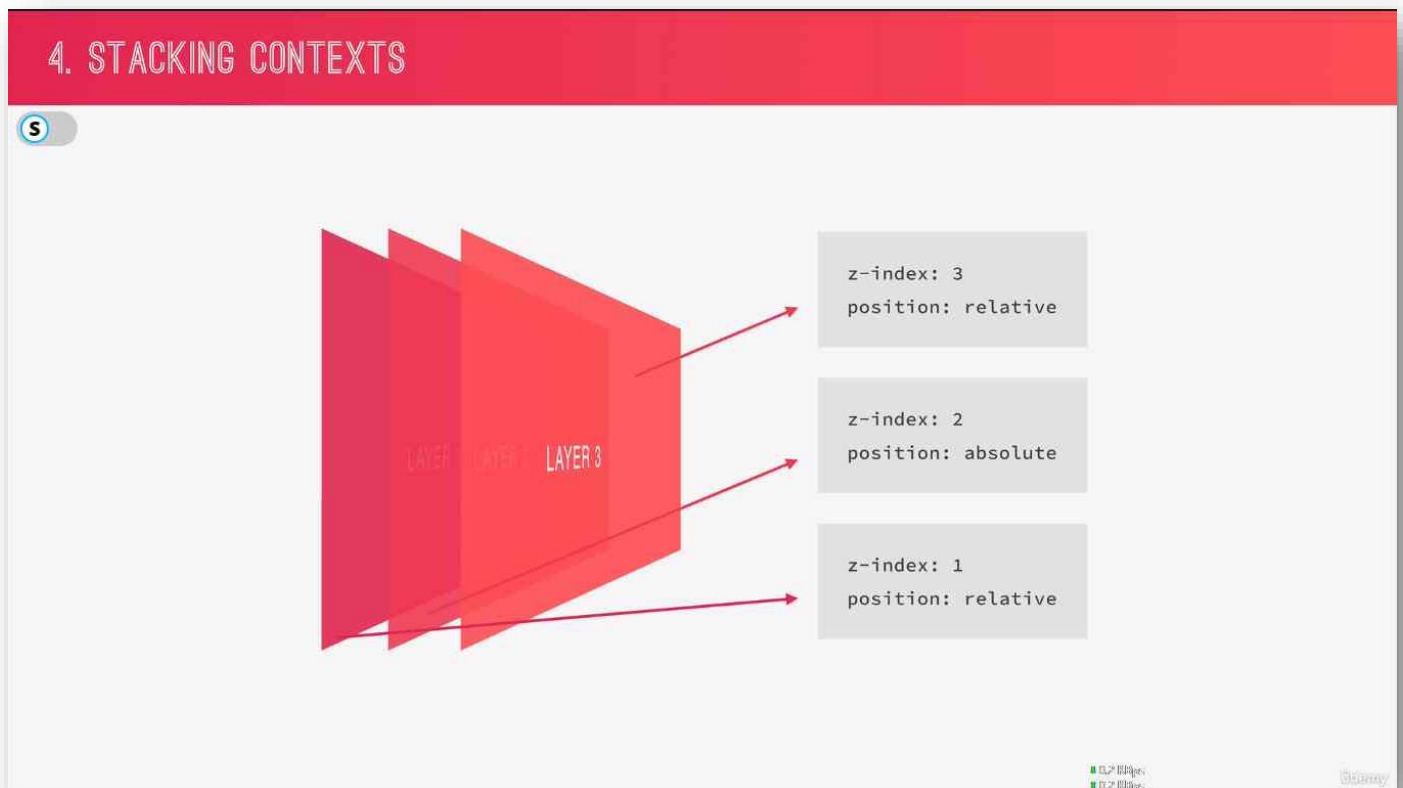
determines how elements are layered and displayed on top of each other within the document. It manages the order, visibility, and appearance of overlapping elements. Stacking contexts are created when specific CSS properties are applied to an element or its descendants.

استخدامك بقا لا **Stacking-context** بيخليك تعمل **control** علي شكل ظهور العناصر دي فوق بعض زي استخدامك لا **z-index** مثلا

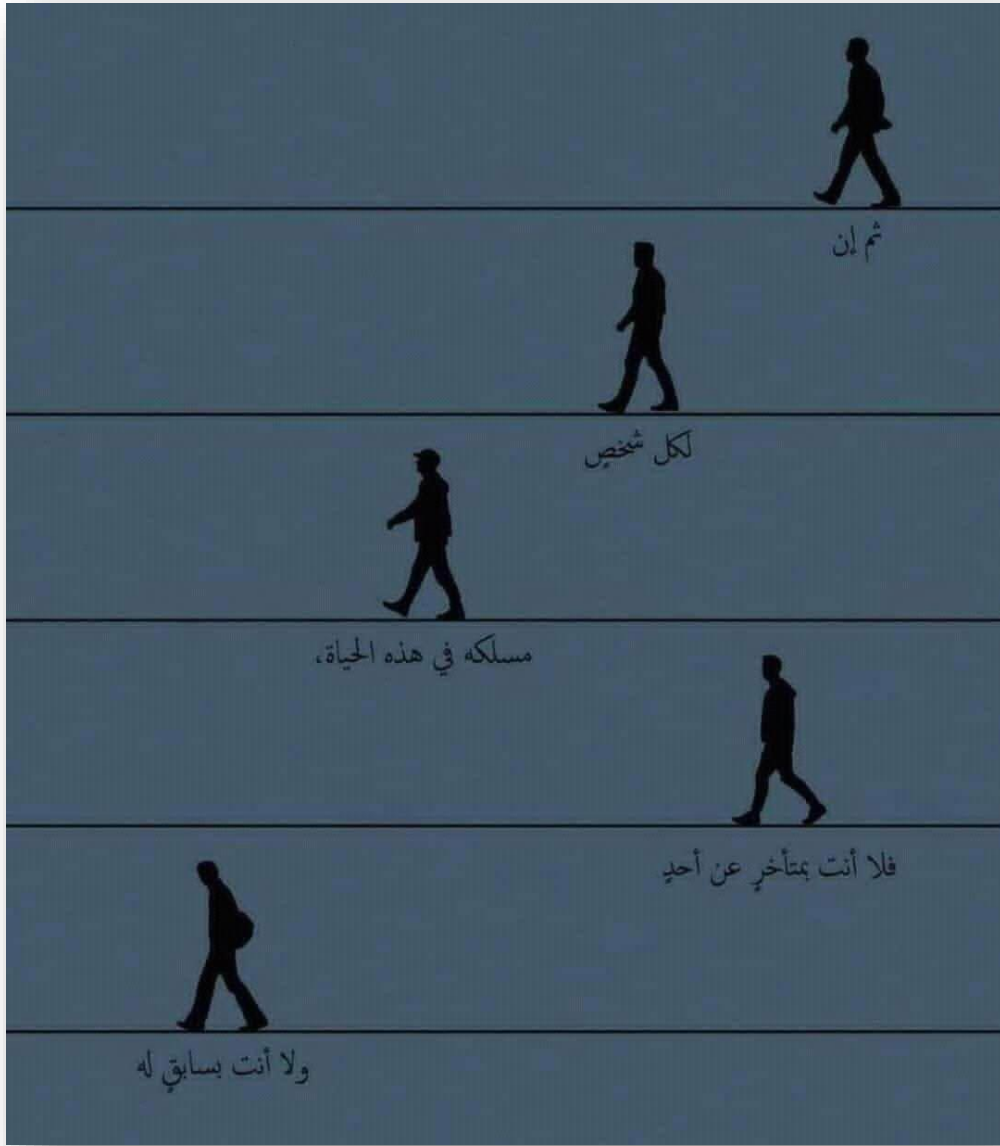
ودا تلخيص سريع ليها في تلت نقاط عن بتساعدنا ازاي وبتحل اي

Stacking contexts help solve several problems in CSS layout and rendering, including:

- 1- **Z-index control:** Stacking contexts allow you to control the vertical stacking order of elements using the z-index property. Elements within a stacking context can be layered above or below other elements based on their assigned z-index values.
- 2- **Controlling overlap and visibility:** Stacking contexts allow you to control the visibility and appearance of overlapping elements. Elements within different stacking contexts can be positioned and displayed independently, preventing unintended overlap or interference.
- 3- **Isolation and encapsulation:** Stacking contexts provide a level of isolation and encapsulation for elements and their descendants. Styles applied within a stacking context do not affect elements outside of that context, preventing unintended style cascades and conflicts.



❤ عبارات تحفيزية ❤



❤ اذكر الله

❤ الزموا ذكر الله ففيه العون على متاعب الحياة
أَلَا بِذِكْرِ اللَّهِ تَطْمَئِنُّ الْقُلُوبُ

اللهم لك الحمد حمدًا طيبًا كثيرًا مباركًا فيه كما ينبغي لجلال وجهك ولعظيم سلطانك يا رب العالمين
اللهم اني أسألك علمًا نافعا ورزقًا طيبًا وعملاً متقبلًا

لا تنسونا من دعواتكم

By: Amr Elsayed

الخاتمة وشوية فضفضة

بسم الله الرحمن الرحيم والصلاة والسلام علي أشرف المرسلين سيدنا محمد

- ❖ استكمالا للمشوار وبعد تلخيص ال **HTML** و ال **CSS** و ال **SASS** و ال **JS** وفقنا الله لعمل هذا الملخص تحت عنوان **How CSS Works Behind The Scenes** .. ارجوا من الله عز وجل ان يكون ذا نفع للجميع .. و ارجوا من الله عز وجل ان يجعل هذا العمل خالصا لوجهه الكريم وان يجعله في ميزان حسناتنا جميعا – [جميع الملخصات هنا .. أضغط](#)
- ❖ الملخص دا مينفعش تشوفه لو مش دارس **CSS** كويس وفاهم خصائص كتير فيها وبتعرف تستخدمها بشكل كويس لازم تكون **فاهم CSS** وطبقت بيها مشاريع كتير عشان تبقي فاهم حاجات كتير بتتكتب في النص
- ❖ الملخصات كلها طبعا مجانية بالكامل بس لو حابب تدعمني ولو مبلغ بسيط او زي ما بيقولو تعزمني ع كوباية قهوة فدا رقمي **فودافون كاش 01005074554** تقدر لو حابب وعندك قدرة تدعمني بالمبلغ اللي تحبه
- ❖ اي **دعم مادي** هيجيلي باذن الله **هخصص** منه جزء **يخرج كصداقات** عني وعن الي دعموني والجزء الباقي هيكون في **تطوير الحاجات** الي بشتغل عليها **وشراء كورسات** اكثر **للتعلم وتلخيصها** للجميع باذن الله
- ❖ باذن الله الملخصات الجاية هتكون عن ال **Bootstrap B** بالتفصيل و ال **Tailwind** و بروضو هنلخص ال **Command line** و ال **Git and GitHub** و طبعا طبعا يعني ال **TypeScript TS** و ال **React js** باذن الله
- ❖ بعذر عن وجود أي أخطاء املائية او أخطاء في أي كود فالكمال لله وحده ولو لقيت أي خطأ في الشرح تواصل معايا
- ❖ خصصت في الملخص دا تلت أجزاء .. الجزء الأول **لذكر الله** عشان متخليش حاجه ابدأ تشغلك عن ذكر الله اللي قادر ينجيك ويفتحلك ابواب الرزق ويرزقك ويسهل عليك الفهم و جزئين **العبارات التحفيزية** و جمل وعبارات **راقت لي** اه هما يعتبروا سبب رئيسي في تكبير حجم الملخص، ولكن والله حبيت انهم يكسروا ملل المذاكرة والقراءة وربما يكون فيهم نفع او الهام لحد
- ❖ الملخص دا كان مصدره كورس المحاضر **Jonas Schmedtmann** – من **Udemy** واي **صور** فيه كانت من ال **Course** كورس بعنوان **Advanced CSS and Sass: Flexbox, Grid, Animations and More!**
- ❖ مش عايزك تتضايق لاي سبب لو **مفهمتش أي جزء** انا حولت ابسط المفاهيم قدر المستطاع وان شاء الله هتفهم يعني هتفهم فبقترح عليك انك تعيد قراءة الحاجة دي وتجرب تكتب **code** بإيدك وتجرب بنفسك
- ❖ لأي حد هيوصله الكلام دا مش طالب منك غير طلب واحد وامانة عليك لازم تنفذه و هو إنك **تفتكرني بدعوة طيبة** لوجه الله أنا وأهل بيتي

[LinkedIn Account](#)

[GitHub Account](#)

[Codepen Account](#)

[Twitter Account](#)

[Mail Me](#)

[Facebook Account](#)

افتكروني بدعوة طيبة لوجه الله واشوفكم ان شاء الله مع حاجه أفضل وأفضل مستقبلا

By: Amr Elsayed