# CSS ?

CSS stands for Cascading Style Sheet, là các tập tin định kiểu theo tầng. Chúng kiểm soát cách thể hiện HTML trên trang Web. Nó được hỗ trợ hầu như rất nhiều trình duyệt hiện nay.

# CSS Syntax

CSS chỉ đơn giản là tập hợp các quy luật dùng để áp dụng style cho các tag hoặc element của HTML. Mỗi quy luật sẽ chọn một hoặc nhiều tag, element để định dạng style cho nó. Ví dụ ta có quy luật định dạng các text của mọi **paragraph tag** trong một trang Web sẽ có font là Times New Roman hoặc chúng ta còn có quy luật định dạng các tag có chứa ID cụ thể nào đó cần định dạng gạch dưới. Nói chung, CSS có nhiệm vụ chọn các thành phần HTML và áp dụng các visual properties cho nó.

Ta đã biết CSS là tập hợp các quy luật, vậy mỗi quy luật sẽ có cú pháp bao gồm hai phần như sau:

**SELECTOR** {**PROPERTY** **DECLARATION**}

*#header*{

    color: green;

}

## Selector

Có ba dạng selector cơ bản là chọn một ID, một Class và một tag. Được minh họa theo thứ tự dưới đây

*#id* {delaration;}

*.class* {delaration;}

tag {delaration;}

## Property

Các declaration của property được đặt bên trong một cặp ngoặc nhọn và kết thúc bởi dấu chấm phẩy.

# Comment

Comment trong CSS được đặt đằng sau các ký tự **“/\*”** và kết thúc bởi **“\*/”.** Chúng ta cũng có thể xuống dùng chừng nào chưa kết thúc comment bằng **“\*/”**. Khác với Python và C/C++, comment trong CSS phải được đóng.

*/\*This is*

*a comment\*/*

# Targeting Elements

Để select một tag nào đó trong HTML thì chúng ta cứ dùng selector là tên của tag đó, theo sau đó là các property declaration là xong. Ví dụ dưới đây chọn *mọi* **anchor tag** có trong HTML.

a {

    text-decoration: underline;

    color:darksalmon;

}

# Targeting ID and Classes

Nhắc lại một lần nữa: Class có thể dùng nhiều lần trong một file HTML, tuy nhiên ID chỉ có thể dùng một lần, nó là unique.

## ID

Trong nhiều trường hợp chúng ta không muốn chọn hết tất cả các tag, ví dụ **division tag**. Chúng ta chỉ muốn chọn các **division tag** cụ thể nào đó. Muốn được như vậy, phải thêm ID hoặc class vào các **division tag** đó rồi dùng selector là các ID hoặc class vừa thêm.

<div *id*="header">

            <h1>Danh sách các loại cá mới "nổi"</h1>

            <ul>

                <li>Cá bơn</li>

                <li>Cá chình</li>

                <li>Cá mập</li>

                <li>Cá đuối</li>

                <li>Cá thu</li>

            </ul>

        </div>

Ví dụ có một đoạn HTML như trên đây, trong khi rất nhiều **division tag** khác. Ta sẽ thêm vào một ID độc nhất là **#header,** sau đó dùng selector là **#header** để gọi ra và định dạng cho **division tag** này.

*#header* {

    color:rgb(106, 6, 116);

    text-indent: 2*pt*;

}

Lúc này đây, ta đã chọn cái tag có chứa ID cần định dạng một cách riêng biệt mà không phải chọn hết tất cả các tag.

## Class

Trong trường hợp cần định dạng một lớp các tag (có thể nói là bao gồm nhiều ID), chúng ta sẽ dùng đến Class thay vì select mọi tag có thể có.

Ta thêm **class attribute** cho các tag và dùng selector có dạng **.class-name** để gọi ra. Giả sử chúng ta có một danh sách các loại hải sản như thế này, được chia thành hai loại là cá và giáp xác. Chúng ta muốn định dạng các tên loài cá khác với các loài giáp xác. Như vậy chúng ta chia chúng thành hai class là **“fish”** và **“crustacean”**. Sau đó dùng selector là **“.fish”** và **“.crustacean”** để gọi chúng ra và định dạng.

Text

Description automatically generated

CSS sẽ có dạng:

*.fish*{

    color:blue;

}

*.crustacean*{

    color: orange;

}

# Conflicts & the Cascade

## Problem

Trong nhiều trường hợp chúng ta sử dụng CSS bị trùng nhau, tức là có hai selector trùng tên, CSS sẽ xử lý như thế nào ? Hãy nhớ rằng CSS là tập tin định kiểu theo tầng, litteraly.

## Solution

Vì vậy những selector nào gọi sau sẽ được áp dụng thay vì cái gọi trước. Cứ tưởng tượng CSS là một cái thác nước, nó sẽ đi từ trên xuống dưới và cái dưới cùng sẽ là thứ có độ ưu tiên cao hơn.

Giả sử ta có code CSS thế này:

*.fish*{

    color:blue;

}

*.fish*{

    color:red;

}

Thì các class “fish” sẽ được định dạng màu đỏ thay vì xanh, bởi vì định dạng màu đỏ được declare cuối cùng.

Điều này cũng được áp dụng trong các trường hợp inline và internal CSS. Chúng ta cứ xét theo thứ tự từ trên xuống. Thông thường external và internal được declare ở phần head, chúng sẽ theo thứ tự mà apply. Sau phần head sẽ đến phần body, chúng ta sẽ có inline CSS. Lúc này inline CSS sẽ lại chồng lên external và internal CSS và được apply.

Graphical user interface, text

Description automatically generated with medium confidence

Đoạn code trên đây dùng external trước cho class “fish”, sau đó dùng intenal. Ở dưới còn có inline CSS apply cho dòng “Cá bơn”. Vậy toàn bộ các class fish sẽ được apply màu xanh lá. Riêng “Cá bơn” sở hữu inline CSS, có độ ưu tiên cao hơn nên sẽ có màu cam.

Dưới đây là kết quả hiển thị trên trang Web:

Text

Description automatically generated

Kết luận rằng: giữa hai selector trùng tên, cái nào declare sau thì cái đó được apply.

# Inheritance

Giả sử chúng ta có một đoạn code nhỏ sau đây:

        <div *id*="header">

            <h1>Danh sách các loại cá mới "nổi"</h1>

            <ul>

                <li>Cá bơn</li>

                <li>Cá chình</li>

                <li>Cá mập</li>

                <li>Cá đuối</li>

                <li>Cá thu</li>

            </ul>

        </div>

Thì **division tag** có chứa hai loại tag khác là **h1 tag** và **unordered list tag**. Hai tag này được gọi là tag con của **division tag**. Tương tự trong **unordered list tag** còn có các **list item tag**, chúng cũng là các tag con của **unordered list tag**.

Vậy khi chúng ta dùng CSS cho **division tag**, ví dụ set màu chữ là màu đỏ, mọi tag con bên trong nó sẽ có màu chữ cũng là màu đỏ. Đây là sự kế thừa. Tương tự, chúng ta dùng CSS cho **unordered list tag** để set chữ màu xanh, thì các **list item tag** sẽ có màu chữ là màu xanh vì chúng kế thừa từ tag cha là **unordered list tag**. Chúng ta sẽ có CSS thế này:

*#header*{

    color:red;

}

*#header* ul{

    color:blue;

}

Kết quả hiển thị ở trang Web là:

A picture containing graphical user interface

Description automatically generated

Ngoài ra còn một vấn đề nữa, liên quan đến style mặc định của Web Browser.

Giả sử ta có đoạn code thế này:

        <div *id*="ref">

            <a *href*= "contact.html">Ib dảk deal ngay</a>

            <a *href*= "prices.html">Mua hàng không nhìn giá</a>

        </div>

Và ta CSS cho nó:

*#ref*{

    color:green;

}

Rõ ràng là các **anchor tag** sẽ được kế thừa lại style của **division tag** có chứa id **“ref”**. Chữ của các liên kết này sẽ có màu xanh lá cây. Tuy nhiên khi hiển thị trên trang Web lại là:



Điều này là do, mặc dù chúng ta đã CSS cho nó rồi, nhưng style mặc định của trình duyệt lại là màu xanh và có gạch dưới. Đó chính là những thứ mặc định mà không thể dùng CSS bằng cách kế thừa. Nói như vậy có nghĩa là ta vẫn có thể CSS cho nó một cách bình thường thông qua selector đến chính các **anchor tag** này.

*#ref* a{

    color:green;

}

Kết quả đã ra đúng:



# Selector Specificity

Chúng ta đã biết nếu hai selector cùng tên nhau thì cái nào declare sau thì cái đó sẽ thắng. Chúng ta đã nhớ rằng inline CSS luôn nằm trong body và internal + external CSS luôn nằm trong head. Vì vậy mà inline luôn chiến thắng internal + external CSS, cái này rút ra từ hệ quả của việc ví CSS như thác nước ở trên mà thôi.

## Problem 1

Tuy nhiên chúng ta vẫn có những trường hợp đặc biệt. Giả sử cả hai đều là external, và một cái gọi đến một tag kèm theo ID và một cái thì không. Bạn nghĩ cái nào sẽ thắng ?

## Solution

Câu trả lời chính là selector có kèm theo ID. Lý do đơn giản thôi, bởi vì nó specific hơn, nó cụ thể hơn. Trong khi selector còn lại chọn toàn bộ n tag thì selector kèm theo ID sẽ chỉ chọn một số tag nhất định trong n tag đó có chứa ID cần dùng.

Một cách dễ phân biệt hơn, ta đánh điểm số cho các loại CSS

1. Inline: 1000
2. External & Internal:
   1. ID: 100
   2. Class: 10
   3. Tag: 1
   4. \* - Chọn tất cả các tag: 0

Trong ví dụ dưới đây, rõ ràng selector vừa chứa ID vừa chứa tag có nhiều điểm hơn selector chỉ chứa tag (101 > 1) nên nó sẽ được apply.

*#ref* a{

    color: green;

}

a {

    color: yellow;

}

## Problem 2

Tiếp theo giả sử ta có đoạn code HTML & CSS như thế này:

//HTML

<div *id*="ref">

   <a *href*= "contact.html">Ib dảk <strong>deal</strong> ngay</a>

   <a *href*= "prices.html">Mua hàng không nhìn giá</a>

//CSS

</div>

*#ref* a{

    color: green;

}

strong {

    color: orange;

}

Chúng ta nhận thấy selector đầu tiên có 101 điểm, nó sẽ select cho toàn bộ **anchor tag** bên trong tag có chứa ID là **“ref”**. Trong khi đó selector strong chỉ có 1 điểm, nó select **strong tag** bên trong **anchor tag**. Theo lẽ thường, 101 điểm sẽ chiến thắng 1 điểm và chữ “deal” bên trong **strong tag** sẽ có màu xanh lá.

## Solution

Tuy nhiên sự việc không đơn giản như vậy, mặc dù selector 101 điểm có chọn **anchor tag**, nhưng nó không trực tiếp chọn **strong tag**. Style của **strong tag** chỉ đơn giản là kế thừa lại style của **anchor tag** mà selector chọn. Vì vậy khi ta chọn **strong tag** một cách trực tiếp, 1 điểm này vẫn win được 101 điểm kia.

Nói cách khác, selector 101 điểm chỉ là chọn **strong tag** một cách ngầm định (implicitly), trong khi selector 1 điểm lại chọn **strong tag** một cách rõ ràng (explicitly). Vì lẽ đó mà chữ “deal” của chúng ta vẫn sẽ có màu cam.

Logo

Description automatically generated with medium confidence

# Descendant Selectors

## Problem

Giả sử ta có đoạn code dưới đây

        <div *id*="ref">

            <a *href*= "contact.html">Ib dảk <strong>deal</strong> ngay</a>

            <a *href*= "prices.html">Mua hàng không nhìn giá</a>

            <p>Kết thúc <strong>chương trình</strong> truyền hình đến đây là hết</p>

        </div>

Nếu ta chỉ muốn select **strong tag** bên trong **anchor tag**, mà không phải gọi **strong tag** bên trong **paragtaph tag** thì sao?

## Solution

Chúng ta sẽ dùng cách gọi selector từ phân cấp lớn nhất rồi đến nhỏ nhất, cách nhau bởi khoảng trắng.

*#ref* a strong{

    color: green;

}

Giả sử **strong tag** bên trong **anchor tag** có id là “deal”. Lúc này selector của chúng ta chỉ đơn giản là “#deal”.

//HTML <div *id*="ref">

         <a *href*= "contact.html">Ib dảk <strong *id*="deal">deal</strong> ngay</a>

         <a *href*= "prices.html">Mua hàng không nhìn giá</a>

         <p>Kết thúc <strong>chương trình</strong> truyền hình đến đây là hết</p>

       </div>

*//CSS #deal*{

    color:green;

}

# Child Selectors

## Problem

Giả sử ta cần chọn các **unordered list tag** bên trong **division tag** có ID là “fish-list”, nhưng không bao gồm các **unordered list tag** bên trong **division tag** có ID là “crustacean-list”. Cho đoạn code mẫu dưới đây.

        <div *id*="header">

            <h1>Danh sách các loại cá mới "nổi"</h1>

            <ul *id*="fish-list"> //Con trực tiếp

                <li>Cá bơn</li>

                <li>Cá chình</li>

                <li>Cá mập</li>

                <li>Cá đuối</li>

                <li>Cá thu</li>

            </ul>

            <div>

                <ul *id*="crustacean-list"> //Con gián tiếp

                    <li>Cua</li>

                    <li>Ghẹ</li>

                </ul>

            </div>

        </div>

## Solution

Ta nhận thấy rằng **unordered list tag** bên trong **division tag** chứa ID “fish-list” là các tag con trực tiếp của **division tag** chứa ID “header”. Trong khi đó các **unordered list tag** bên trong **division tag** chứa ID “crustacean-list” lại là các tag con gián tiếp của **division tag** chứa ID “header”. Vậy selector để chọn các tag con trực tiếp sẽ có dạng như sau:

*#header* *>* ul{

    color:red;

}

Chúng ta sử dụng dấu “>” để truy xuất đến các tag con trực tiếp. Kết quả ở trang Web:

Text

Description automatically generated

# Adjacent Selectors

## Problem

Giả sử chúng ta có 100 **division tag**, mỗi **division tag** chứa 1 **h1 tag** và **10 paragraph tag**. Chúng ta muốn format cho **paragraph tag** đầu tiên, sau **h1 tag**, thì có những cách nào ?

Chúng ta có thể nghĩ đến việc dùng class cho lượng lớn các paragraph đầu tiên đó, rồi dùng selector gọi class đó ra. Tuy nhiên, việc này sẽ gây khó khăn khi update vì phải chỉnh sửa tận 100 class.

## Solution

Cách để giải quyết vấn đề này là dùng Adjacent Selectors, tức là selector kề cận. Rõ ràng chúng ta thấy h1 tag kề với paragraph đầu tiên. Vậy chỉ cần dùng Adjacent Selectors mà không cần đặt class dài dòng.

Giả sử ta có một đoạn code HTML như sau:

    <div *id*="contact">

        <h3> Liên hệ với chúng tôi tại </h3>

        <div *class*="contact-method">

            <h4>Qua email: </h4>

            <p>Email công sở</p>

            <p>Email cá nhân</p>

        </div>

        <div *class*="contact-method">

            <h4>Qua điện thoại: </h4>

            <p>Điện thoại bàn</p>

            <p>Điện thoại di động</p>

        </div>

        <div *class*="contact-method">

            <h4>Qua fb: </h4>

            <p>Fanpage</p>

            <p>FB cá nhân</p>

        </div>

        <div *class*="contact-method">

            <h4>Qua tax: </h4>

            <p>Qua tax1 </p>

            <p>Qua tax2 </p>

        </div>

    </div>

Và chúng ta muốn các paragraph tag theo sau h4 tag đều sẽ được in đậm. Chúng ta sử dụng cú pháp Adjacent Selectors như sau:

*.contact-method* h4 *+* p{

    text-decoration: underline;

}

Ta sẽ viết tag cận kề trước rồi thêm dấu cộng (“+”), theo sau đó là tag muốn target.

Kết quả sẽ là:

Text

Description automatically generated

# Important Declaration

Có một cách khai báo CSS mà không selector nào có thể override (chồng lên) nó. Đó là sử dụng cú pháp

!important

Theo sau giá trị của một property nào đó trong CSS. Ví dụ.

ul {

    color:blue;

}

*#header* ul{

    color: green;

}

Ta thấy rõ ràng selector thứ hai có 101 điểm, nó sẽ override cái selector thứ nhất, theo Selector Specificity. Tuy nhiên, nếu ta muốn “color: blue” là thứ override lên tất cả mọi thứ, ta sẽ thêm Important Declaration phía sau nó. Kiểu như sau:

ul {

    color:blue *!important*;

}

*#header* ul{

    color: green;

}

Lưu ý thêm là Important Declaration này có số điểm lớn hơn 1000, tức là nó có thể override cả inline CSS nếu muốn. Đồng thời Important Declaration không phải là một Quick Fix khi có conflicts giữa các CSS selector khác nhau, hãy dùng đúng nơi đúng lúc.

# Targeting Multiple Elements

## Problem

Nếu như khách hàng yêu cầu chúng ta cho mọi text trên trang Web là như nhau về format, ví dụ font-size là 14, font-family là Arial, color là red. Thì thông thường chúng ta sẽ sử dụng các nhiều selector trỏ đến nhiều tag, nhưng lại có các properties declaration giống hệt nhau. Ví dụ:

p {

    color:red;

    font-size: 12*px*;

    font-family: Arial;

}

dt {

    color:red;

    font-size: 12*px*;

    font-family: Arial;

}

dd {

    color:red;

    font-size: 12*px*;

    font-family: Arial;

}

## Solution

Để tối ưu CSS, vừa gọn code vừa tiết kiệm thời gian khi cần update property, thì chúng ta nên gom lại thành một declaration cho nhiều selector. Các selector chung một declaration cách nhau bởi dấu phẩy. Kết quả như thế này:

p,dt,dd {

    color:red;

    font-size: 12*px*;

    font-family: Arial;

}

# Attribute Selectors