# 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;

}

# 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;

}

# 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.

# Attribute Selectors “[]”

Để chọn một tag nào đi kèm với attribute nhất định, chúng ta sẽ sử dụng cú pháp sau:

tag[attribute]{}

Ví dụ ta cần select thẻ div có chứa attribute là class dưới đây.

<div *class*="contact-method">

            <h4>Qua email: </h4>

            <p>Email công sở</p>

            <p>Email cá nhân</p>

        </div>

Ta sẽ select theo đúng cú pháp nêu trên.

div[*class*]{

    color: blue;

}

## Problem 1

Giả sử chúng ta cần select cụ thể hơn, select những thẻ div có attribute class có giá trị là “contact-method”. Chúng ta thêm giá trị cần select sau attribute của selector.

tag[attribute*=*"value"]{}

Giả sử ta cần chọn thẻ anchor có attribute href và giá trị là “index.html”.

<a *href*="index.html">Về trang chủ</a>

<a *href*="prices.html">Check giá</a>

## Solution

Thì CSS là:

a[*href=*"index.html"]{

    color:aquamarine;

}

## Problem 2

Trong trường hợp value của các attribute tương tự nhau, chẳng hạn như **“red light”, “green light”, “light”**,… Thì làm sao chúng ta select được hết các value đều có chữ “light” này ?

## Solution

Chúng ta sẽ sử dụng dấu **“~”** trước dấu bằng giữa attribute và value, với value là chữ chung trong tất cả các attribute, problem trên là chữ **“light”.** HTML có dạng:

    <div *class*="red light">

        Đèn đỏ

    </div>

    <div *class*="green light">

        Đèn xanh

    </div>

    <div *class*="light">

        Đèn

Thì CSS sẽ là:

div[*class~=*"light"]{

    color:blue;

}

## Problem 3

Cũng một trường hợp tương tự, chẳng hạn ta có ba value là **“hacker”**, **“coder”** và **“developer”**. Cả ba đều kết thúc với hai ký tự **“er”**. Chúng ta nhận ra điểm chung này và muốn select chúng, làm cách nào?

## Solution

Chúng ta sẽ dùng ký tự **“$”** trước dấu bằng, nằm giữa attribute và value, với value là các ký tự nằm cuối giống nhau giữa các attribute, problem trên sẽ là **“er”.** Với đoạn code mẫu dưới đây.

    <div *class*="hacker">

        Hacker

    </div>

    <div *class*="coder">

        Coder

    </div>

    <div *class*="Developer">

        Developer

    </div>

Ta sẽ có CSS là:

div[*class$=*"er"]

{

    color:blue;

}

Ý nghĩa của **“$”** là chọn các tag có attribute mang giá trị kết thúc bằng **“er”.**

Ngược lại, nếu chúng ta muốn chọn các attribute có value cùng các ký tự đầu, ta dùng ký tự **“^”** trước dấu bằng. Cú pháp khi đó sẽ là:

tag[attribute*^=*"same letter"]{}

Có nghĩa là chọn các tag có attribute mang giá trị bắt đầu là **“same value”.**

# Pseudo Selectors “:”

Các pseudo selectors là phần mở rộng của selector, chúng là các keywords được viết sau selector. Đa số các Pseudo Selector được chia ra làm hai lớp, Dynamic và Structural. Dynamic chính là select những phần tử có hành vi đặc biệt (ví dụ như hiệu ứng hover) và Structural là để select những phần tử có cấu trúc đặc biệt (ví dụ như quan hệ cha mẹ - con cái).

Ở lớp Dynamic, các Pseudo Selector cho phép chúng ta style cho các phần tử có tương tác với người dùng. Chẳng hạn như khi người dùng trỏ chuột (hover) vào link, bấm vào một nút hay tick vào một tickbox.

Còn ở lớp Structural, cho phép chúng ta select những thành phần có cấu trúc đặc biệt mà không thể select bằng CSS thông thường. Ví dụ như **list item tag** thứ 5 trong một list hoặc một tag mà không có children.

Cú pháp thông thường của Pseudo Selector có dạng như sau:

selector:keyword{declaration(s)}

Chúng ta sẽ bàn qua một số keyword thuộc lớp Dynamic dưới đây.

## Hover

Đây là keyword của hiệu ứng trỏ chuột (hover). Khi dùng keyword này, các style declaration sẽ được apply khi hover đến một element nào đó mà có selector trỏ tới.

Ví dụ là:

a*:hover*{

    color:aquamarine;

}

Đoạn code trên có ý nghĩa, khi trỏ chuột (hover) vào các dòng link (của anchor tag) thì màu sắc dòng link đổi thành màu aquamarine.

## Active

Còn đây là keyword dùng cho khi click vào một element nào đó mà selector trỏ tới.

Chẳng hạn như:

a*:active*{

    color: orange;

}

Thì khi click vào đoạn link, màu chữ sẽ đổi thành màu cam rồi mới chuyển hướng.

## Visited

Mỗi lần chúng ta click vào một kết quả của google, sau khi trở lại trang tìm kiếm, thì kết quả đó thay đổi màu sắc. Chẳng hạn như:

Text

Description automatically generated

Đây gọi là hiệu ứng visited, đúng như tên gọi của nó thì khi một liên kết đã được “thăm hỏi” thì khi trở lại nó sẽ có một định dạng khác. Chẳng hạn đoạn code dưới đây sẽ làm những dòng link đã được click vào khi trở lại có màu cam.

a*:visited*{

    color: orange;

}

# First & Last Child Selector

## Problem

Làm sao để select được các phần tử con đầu tiên và cuối cùng của một thẻ nào đó mà không cần dùng class, id hay dùng Adjacent Selectors ?

## Solution

Chúng ta sẽ sử dụng Pseudo Selectors với các key word là **“first-child”** và “**last-child”.** Giả sử chúng ta có hai danh sách hải sản dưới đây.

        <div *id*="header">

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

            <ul *id*="fish-list">

                <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">

                    <li>Cua</li>

                    <li>Ghẹ</li>

                </ul>

            </div>

        </div>

Ta muốn chọn “Cá bơn” và “Cá thu” thì:

*#fish-list* li*:first-child*{

    color:orange;

}

*#fish-list* li*:last-child*{

    color:red;

}

Cú pháp tổng quát của nó là:

selector child-tag*:first-child*{}

selector child-tag*:last-child*{}

Nghĩa là sau tag con ta dùng thêm keyword **“first-child”** hoặc **“last-child”** cho chúng.

# First & Last of Type Selectors

## Problem

Giả sử ta muốn chọn một tag là tag đầu tiên trong dãy các tag cùng loại, nhưng nó không phải là con đầu tiên của tag cha. Ví dụ mẫu dưới đây:

<div *id*="header">

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

            <ul *id*="fish-list">

                <h2>Cá biển</h2>

                <li>Cá thu</li>

                <li>Cá mập</li>

                <li>Cá đuối</li>

                <h2>Cá sông</h2>

                <li>Cá bơn</li>

                <li>Cá lóc</li>

                <p>-------------------</p>

            </ul>

            <div>

                <h1>Danh sách các loại giáp xác</h1>

                <ul *id*="crustacean-list">

                    <li>Cua</li>

                    <li>Ghẹ</li>

                </ul>

            </div>

        </div>

Rõ ràng là <li> thứ nhất không là thẻ con đầu tiên của #fist-list, vì vậy ta không thể dùng “first-child”.

## Solution

Nhưng ta để ý rằng nó là <li> tag đầu tiên trong số các <li> thuộc #fish-list. Vì vậy muốn chọn “Cá thu”, ta có thể dùng keyword **“first-of-type”** của Pseudo Selector như sau:

*#fish-list* li*:first-of-type*{

    color:orange;

}

Dòng code này có nghĩa là chọn thẻ <li> là con của #fish-list nhưng là thẻ <li> đầu tiên trong số các thẻ <li> cùng loại.

Tương tự ta có thể chọn “Cá lóc”, thẻ này là thẻ cuối cùng trong số các thẻ <li> cùng loại. Ta sử dụng keyword “last-of-type”.

*#fish-list* li*:last-of-type*{

    color:red;

}

# nth Child Selectors

Giả sử ta muốn chọn một thẻ con thứ n của một thẻ cha nào đó, ta sẽ sử dụng cú pháp sau:

selector child-tag*:nth-child*(n){}

Tức là sau thẻ con, ta dùng keyword **“nth-child”** với tham số là **n,** là số thứ tự trong số các thẻ con đó. Ví dụ ta muốn chọn cá đuối, là thẻ <li> thứ 3 thuộc #fish-list.

            <ul *id*="fish-list">

                <li>Cá thu</li>

                <li>Cá mập</li>

                <li>Cá đuối</li>

                <li>Cá bơn</li>

                <li>Cá lóc</li>

            </ul>

Ta sử dụng cú pháp ở trên, tham số n = 3:

*#fish-list* li*:nth-child*(3)

{

    color:orange;

}

Cũng có thể chọn multiple:

*#fish-list* li*:nth-child*(3), li*:nth-child*(4)

{

    color:orange;

}

Tham số đầu vào của keyword **“nth-child”** cũng có thể là **“even”** (chẵn) hoặc **“odd”** (lẻ). Đoạn code dưới đây sẽ chọn các thẻ <li> theo thứ tự chẵn và lẻ.

*#fish-list* li*:nth-child*(even)

{

    color:orange;

}

*#fish-list* li*:nth-child*(odd)

{

    color:blue;

}

Cũng có thể sử dụng các công thức để thay thế cho even và odd:

*#fish-list* li*:nth-child*(2n)

{

    color:orange;

}

*#fish-list* li*:nth-child*(2n+1)

{

    color:blue;

}

Các công thức cũng có thể là “3n+1”, “4n+1”,…

# nth of Type Selectors

## Problem

Ví dụ ta có đoạn code HTML dưới đây:

Text

Description automatically generated

Chúng ta muốn select tag <dt> mang giá trị là “Cá đuối”, mà không sử dụng Class, Id hay Adjacent. Có thể sử dụng nth Child Selector, tuy nhiên như vậy thì chúng ta phải sử dụng tham số là các công thức hoặc số n cụ thể. Bởi vì có các tag <hr> xen kẽ, nên even hoặc odd cũng không khả thi.

## Solution

Cũng tương tự First & Last of Type Selectors, chúng ta cũng có thể select các tag theo thứ tự n trong số các tag cùng loại.

Giả sử ở trên ta muốn chọn cá đuối, là tag con thứ 8 thuộc <dl> nhưng lại là tag <dt> thứ 4 thuộc tag <dl>. Ta sẽ sử dụng keyword **“nth-of-type”**, với tham số đầu vào là thứ tự, ở đây n = 4.

dl dt*:nth-of-type*(4)

{

    color:orange;

}

Tương tự, ta cũng có thể sử dụng odd, even và công thức làm tham số đầu vào. Giả sử ở trên t muốn select “Cá chình”, “Cá đuối”, “Cua”. Ta sử dụng CSS là even, vì đây là các tag <dt> có thứ tự chẵn trong số các tag <dt> cùng loại.

dl dt*:nth-of-type*(even)

{

    color:orange;

}

# Combining Selectors

Giả dụ ta có một đoạn code HTML dưới đây.

Text

Description automatically generated

Trong trường hợp một tag có đi kèm với một class nào đó cụ thể. Tag và class này đều *cùng cấp* nên ta không thể dùng dấu cách để select tag đó được. Ở trên thì không thể dùng:

dt *.fish*{

    color: blue;

}

Vì “.fish” không phải con của dt. Mà ta sẽ dùng Combining Selectors, các selector kết hợp cùng cấp, như sau:

dt*.fish*{

    color: blue;

}

Chúng ta chỉ cần bỏ dấu cách là xong!

Ngoài ra, giả dụ trường hợp có hai tag cùng class thì cũng có thể dùng Combining Selectors. Nếu HTML có dạng như thế này.

A screenshot of a computer

Description automatically generated with medium confidence

Thì CSS sẽ là:

dt*.crustacean*{

    color:orange;

}

Hoặc

p*.crustacean*{

    color:red;

}

# Universal Selector “\*”

Ở trên ta có nhắc một chút về độ ưu tiên của một selector đặc biệt, là **“\*”**. Selector này là Universal Selector, nó sẽ chọn tất cả các tag có trong tệp HTML để apply style. Tuy nhiên do độ ưu tiên thấp nhất, nên nó cũng thường bị các selector khác có độ ưu tiên cao hơn đè lên. Vì vậy mà khi dùng Universal Selector, ta thường dùng các style nào áp dụng chung nhất cho tất cả các tag.

Cần phân biệt giữa selector **“\*”** và selector **“body”**. Khi ta dùng **“\*”**, mọi thứ đều được select và apply style. Còn khi dùng **“body”**, mọi thứ bên trong thẻ body sẽ được kế thừa lại. Nghĩa là về ý tưởng thì mọi thứ hiển thị trên trang Web đều tương tự nhau. Tuy nhiên, ta nhớ rằng thẻ **anchor tag** chứa link, được trình duyệt set mặc định là màu xanh dương đậm. Và cái rule này của trình duyệt sẽ chồng lên cái mà nó kế thừa từ body. Vì vậy khi ta dùng selector là **“body”,** mặc dù toàn bộ nội dung hiển thị một màu (chẳng hạn cam), thì cái link lại có màu xanh,. Ví dụ dưới đây sẽ minh họa cho điều đó.

\*{

    color: orange;

}

Sẽ có kết quả là:

Text

Description automatically generated

Trong khi đó, nếu dùng selector là **“body”,** kết quả sẽ là

Text

Description automatically generated with medium confidence

Có thể thấy rằng các đường liên kết không bị đổi màu, do cái rule trình duyệt apply đã chồng lên cái rule kế thừa từ body.

Vì vậy, mặc dù Universal Selector có độ ưu tiên thấp hơn selector **“body”**, nhưng nó vẫn có thể chồng lên trong một số trường hợp cụ thể như ở trên. Có thể nói rằng Universal là chọn mọi tag một cách trực tiếp, trong khi selector **“body”** là chọn **body tag** và mọi tag khác kế thừa từ nó.

## CSS Reset

Đây là một cách để loại bỏ tất cả các rule của trình duyệt để có thể select các element như ý muốn, ta gọi cách này là CSS *Reset*. Lúc này thì không cần đi vào chi tiết, nhưng một dạng cơ bản của CSS Reset sẽ là thế này:

\*{

    margin: 0; //Viền

    padding: 0; //Đệm giữa các box

    color:black; //Màu chữ

}