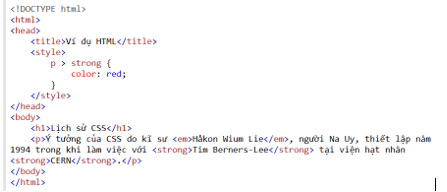
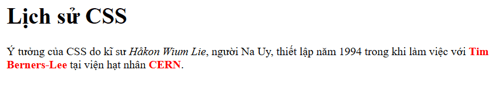
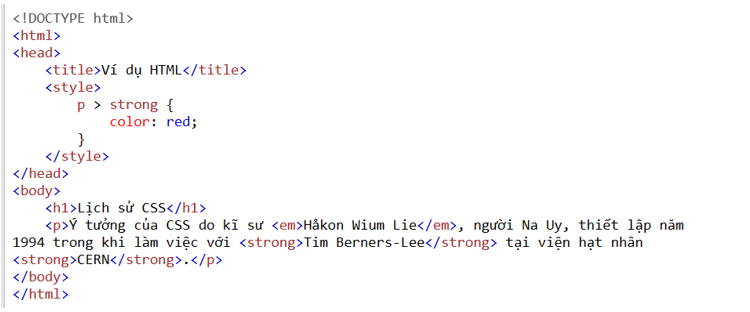
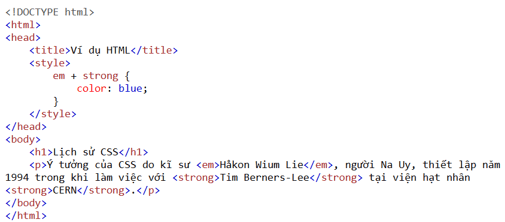
# Bài 15: Tạo màu cho chữ và nền

**Giải Tin học 12 Bài 15: Tạo màu cho chữ và nền**  
**Khởi động trang 83 Tin học 12**: Các định dạng sau có thiết lập cùng một màu hay không? Em có nhận xét gì về thiết lập này?  
  
**Lời giải:**  
Các định dạng sau đều thiết lập cùng một màu, đó là màu tím.  
a) p {color: rgb(128,0,128);} sử dụng giá trị RGB để thiết lập màu. Trong trường hợp này, giá trị RGB là (128, 0, 128), tương ứng với màu tím.  
b) p {color: #800080;} sử dụng mã màu HEX để thiết lập màu. Mã màu HEX "#800080" cũng tương ứng với màu tím.  
c) p {color: hsl (300,100%,25.1%);} sử dụng giá trị HSL để thiết lập màu. Trong trường hợp này, giá trị HSL là (300, 100%, 25.1%), tương ứng với màu tím.  
Nhận xét: Cả ba định dạng đều thiết lập cùng một màu tím, tuy nhiên, cách biểu diễn màu sử dụng các hệ thống màu khác nhau. RGB sử dụng giá trị đỏ (Red), xanh lá cây (Green) và xanh dương (Blue), mã màu HEX sử dụng mã hexa và HSL sử dụng giá trị màu (Hue), độ bão hòa (Saturation) và độ sáng (Lightness).  
  
**Hoạt động 1 trang 83 Tin học 12**: Cùng thảo luận và tìm hiểu hệ màu RGB hỗ trợ bởi HTML và CSS  
**Lời giải:**  
Hệ màu RGB (Red, Green, Blue) là một hệ màu được hỗ trợ bởi HTML và CSS để biểu diễn các màu sắc. Trong hệ màu RGB, mỗi màu được đại diện bởi một giá trị từ 0 đến 255, tức là một số nguyên 8-bit. Kết hợp các giá trị của ba màu cơ bản này, chúng ta có thể tạo ra hàng triệu màu khác nhau.  
Trong HTML và CSS, có ba cách để biểu diễn màu theo hệ màu RGB:  
- Sử dụng hàm rgb(): Ta có thể sử dụng hàm rgb() để chỉ định giá trị đỏ (red), xanh lá cây (green) và xanh dương (blue) của một màu. Cú pháp là rgb(red, green, blue), trong đó red, green và blue là các giá trị từ 0 đến 255. Ví dụ: rgb(255, 0, 0) đại diện cho màu đỏ, rgb(0, 255, 0) đại diện cho màu xanh lá cây.  
- Sử dụng hàm rgba(): Hàm rgba() tương tự như hàm rgb(), nhưng nó cung cấp thêm một tham số alpha để xác định độ trong suốt của màu. Tham số alpha có giá trị từ 0.0 (hoàn toàn trong suốt) đến 1.0 (hoàn toàn không trong suốt). Ví dụ: rgba(255, 0, 0, 0.5) đại diện cho màu đỏ có độ trong suốt là 50%.  
- Sử dụng mã hexa: Mã hexa là một cách khác để biểu diễn màu RGB bằng cách sử dụng một chuỗi sáu ký tự 16-hexa. Mỗi cặp ký tự hexa đại diện cho giá trị của một màu cơ bản trong hệ màu RGB. Ví dụ: #FF0000 đại diện cho màu đỏ, #00FF00 đại diện cho màu xanh lá cây.  
Hệ màu RGB cho phép chúng ta định rõ màu sắc của các yếu tố trên trang web bằng cách sử dụng các giá trị RGB hoặc mã hexa tương ứng.  
**Câu hỏi 1 trang 84 Tin học 12**: Các màu cơ bản red, yellow, green, blue, magentam cyan được thể hiện bằng hàm hsl như thế nào?  
**Lời giải:**  
Red (đỏ): HSL(0, 100%, 50%)  
Yellow (vàng): HSL(60, 100%, 50%)  
Green (xanh lá cây): HSL(120, 100%, 50%)  
Blue (xanh dương): HSL(240, 100%, 50%)  
Magenta (đỏ tím): HSL(300, 100%, 50%)  
Cyan (xanh lam): HSL(180, 100%, 50%)  
  
**Câu hỏi 2 trang 84 Tin học 12**: Trong hệ màu rgb có bao nhiêu màu thuộc màu xám?  
**Lời giải:**  
Trong hệ màu RGB, màu xám được đại diện bởi các giá trị RGB giống nhau. Cụ thể, nếu giá trị đỏ (red), xanh lá cây (green) và xanh dương (blue) đều bằng nhau, chúng ta sẽ có một màu xám. Vì mỗi kênh màu có 256 giá trị từ 0 đến 255, nên có tổng cộng 256 màu xám khác nhau trong hệ màu RGB.  
  
**Hoạt động 2 trang 84 Tin học 12**: Cùng thảo luận về cách thiết lập màu trong các mẫu CSS và trả lời các câu hỏi sau:  
1. Cách thiết lập định dạng màu chữ trong CSS như thế nào?  
2. Cách thiết lập màu nền cho các phần tử CSS như thế nào?  
**Lời giải:**  
- Để thiết lập định dạng màu chữ trong CSS, bạn có thể sử dụng thuộc tính "color". Có thể áp dụng giá trị màu bằng tên màu (ví dụ: "red", "blue"), mã hex (ví dụ: "#FF0000" cho màu đỏ) hoặc giá trị RGB (ví dụ: "rgb(255, 0, 0)" cho màu đỏ).  
- Để thiết lập màu nền cho các phần tử CSS, bạn có thể sử dụng thuộc tính "background-color". Tương tự như màu chữ, bạn có thể áp dụng giá trị màu bằng tên màu, mã hex hoặc giá trị RGB.  
**Câu hỏi 1 trang 85 Tin học 12**: Sửa lại CSS trong ví dụ trên, định dạng màu nền và khung viền cho cụm từ Tim Berners-Lee với màu khác biệt.  
**Lời giải:**  
em {  
 background-color: lightgreen;  
 border: 1px solid red;  
}  
p {  
 color: blue;  
}  
   
/\* Thêm CSS cho cụm từ Tim Berners-Lee \*/  
.tim-berners-lee {  
 background-color: yellow; /\* Màu nền khác biệt \*/  
 border: 2px solid green; /\* Viền màu và dày khác biệt \*/  
 padding: 5px; /\* Thêm đệm để viền nhìn rõ hơn \*/  
}  
  
**Câu hỏi 2 trang 85 Tin học 12**: Sửa lại CSS trên, thay đổi định dạng khung viền cho phần tử p. Em hãy kiểm tra xem tính chất này có kế thừa cho các phần tử con không.  
**Lời giải:**  
em {  
 background-color: lightgreen;  
 border: 1px solid red;  
}  
p {  
 color: blue;  
 border: 2px dashed purple; /\* Thêm định dạng khung viền cho phần tử p \*/  
 padding: 10px; /\* Thêm padding để khung viền không dính vào chữ \*/  
}  
Tim-berners-lee {  
 background-color: yellow;  
 border: 2px solid green;  
 padding: 5px;  
}  
Về việc kế thừa, trong CSS, có một số tính chất có thể được kế thừa từ phần tử cha xuống phần tử con, như color và n, tính chất border không được kế thừa. Điều này có nghĩa là, nếu áp dụng một khung viền cho một phần tử p, các phần tử con bên trong p (ví dụ như em, span, hoặc strong) sẽ không tự động kế thừa khung viền đó. Mỗi phần tử sẽ chỉ có khung viền nếu bạn một cách rõ ràng định nghĩa nó trong quy tắc CSS của mình cho phần tử đó.  
  
**Hoạt động 3 trang 85 Tin học 12**: Quan sát, trao đổi và thảo luận về 4 trường hợp bộ chọn là tổ hợp các phần tử, nêu ý nghĩa và sự khác biệt giữa các trường hợp này: E F, E > F, E + F và E ~ F.  
**Lời giải:**  
- E F (Descendant selector): Chọn tất cả các phần tử F nằm bên trong phần tử E, bất kể cấp độ lồng nhau của chúng. Ví dụ: div p sẽ chọn tất cả các phần tử <p> nằm trong các phần tử <div>.  
- E > F (Child selector): Chọn các phần tử F là con trực tiếp của phần tử E. Điều này chỉ chọn các phần tử F nằm trực tiếp bên trong phần tử E, không bao gồm các phần tử F ở các cấp độ lồng nhau sâu hơn. Ví dụ: ul > li sẽ chọn tất cả các phần tử <li> là con trực tiếp của phần tử <ul>.  
- E + F (Adjacent sibling selector): Chọn các phần tử F ngay sau phần tử E và cùng cấp với E. Điều này chỉ chọn phần tử F đầu tiên sau phần tử E. Ví dụ: h2 + p sẽ chọn phần tử <p> đầu tiên ngay sau một phần tử <h2>.  
- E ~ F (General sibling selector): Chọn tất cả các phần tử F cùng cấp với phần tử E, ngay sau và lồng nhau. Điều này tương tự như trường hợp E + F, nhưng khác biệt là E ~ F có thể chọn nhiều phần tử F, không chỉ phần tử đầu tiên. Ví dụ: h3 ~ p sẽ chọn tất cả các phần tử <p> cùng cấp với phần tử <h3>.  
**Câu hỏi 1 trang 87 Tin học 12**: Trong ví dụ ở Hình 15.7, nếu thay mẫu em ~ strong bằng p > strong thì kết quả sẽ như thế nào?  
**Lời giải:**  
Nếu thay mẫu em ~ strong bằng p > strong, kết quả sẽ không thay đổi. Lí do là trong cấu trúc HTML, phần tử <strong> đang nằm trực tiếp trong phần tử <p>. Do đó, cả hai trường hợp em ~ strong và p > strong đều sẽ chọn phần tử <strong> đó và áp dụng các định dạng CSS tương ứng.  
Mã HTML sau khi sửa:  
  
Kết quả hiển thị:  
  
**Câu hỏi 2 trang 87 Tin học 12**: Trong ví dụ ở Hình 15.7 nếu thay mẫu em ~ strong bằng em + strong thì kết quả sẽ như thế nào?  
**Lời giải:**  
Nếu thay mẫu em ~ strong bằng p > strong, kết quả sẽ không thay đổi. Lí do là trong cấu trúc HTML, phần tử <strong> đang nằm trực tiếp trong phần tử <p>. Do đó, cả hai trường hợp em ~ strong và p > strong đều sẽ chọn phần tử <strong> đó và áp dụng các định dạng CSS tương ứng.  
Mã HTML sau khi sửa:  
  
Kết quả hiển thị:  
  
  
**Câu hỏi 3 trang 87 Tin học 12**: Trong ví dụ ở Hình 15.7, nếu thay mẫu em ~ strong bằng p strong thì kết quả sẽ như thế nào?  
**Lời giải:**  
Kết quả là các phần tử <strong> ngay sau phần tử <em> sẽ có màu chữ xanh.  
Mã HTML sau khi thay đổi:  
  
Kết quả hiển thị:  
