Css Function, Css Pseudo, Combinators(+ ~ >), Type Selectors & Child Selectors.

Nội dung bài học:

1. Css function.
2. Css pseudo.
3. Combinators(+ ~ >).
4. Type selectors & child selectors.

1. Css function:

Color Functions:

- **rgb()**: Định nghĩa một màu sử dụng giá trị RGB.
- **rgba()**: Định nghĩa một màu sử dụng giá trị RGB và độ trong suốt.
- **hsl()**: Định nghĩa một màu sử dụng giá trị HSL.
- **hsla()**: Định nghĩa một màu sử dụng giá trị HSL và độ trong suốt.
- **currentColor**: Sử dụng giá trị màu của thuộc tính color của phần tử hiện tại.

Image Functions:

- **url()**: Định nghĩa một URL của hình ảnh hoặc tệp tin khác.

Math Functions:

- **calc()**: Thực hiện phép tính số học trong CSS.
- **min()**: Trả về giá trị nhỏ nhất của các giá trị được cung cấp.
- **max()**: Trả về giá trị lớn nhất của các giá trị được cung cấp.

Other Functions:

- **var()**: Sử dụng biến CSS.
- **attr()**: Trả về giá trị của thuộc tính HTML.
- **translate(), translateX(), translateY(), translateZ()**: Dịch chuyển phần tử.
- **rotate(), rotateX(), rotateY(), rotateZ()**: Xoay phần tử.
- **scale(), scaleX(), scaleY(), scaleZ()**: Thay đổi kích thước của phần tử.
- **skew(), skewX(), skewY()**: Xoay phần tử theo trục.
- **linear-gradient()**: Định nghĩa gradient tuyến tính.

- **radial-gradient()**: Định nghĩa gradient hình tròn.

2. Css Pseudo:

Trong CSS, các pseudo-classes và pseudo-elements cho phép bạn chọn các phần tử dựa trên trạng thái hoặc vị trí của chúng trong DOM. Dưới đây là một số pseudo-classes và pseudo-elements phổ biến:

Pseudo-classes:

- **:hover**: Chọn phần tử khi con trỏ của người dùng đặt lên.
- **:active**: Chọn phần tử khi nó đang được nhấp chuột hoặc đang tương tác.
- **:focus**: Chọn phần tử khi nó đang được tập trung (focus).
- **:visited**: Chọn các liên kết đã được truy cập.
- **:first-child**: Chọn phần tử là con đầu tiên của phần tử cha.
- **:last-child**: Chọn phần tử là con cuối cùng của phần tử cha.
- **:nth-child()**: Chọn phần tử con dựa trên vị trí của nó trong danh sách con.
- **:nth-of-type(an+b, even, odd)**: Chọn phần tử con dựa trên loại của nó và vị trí trong danh sách con.
- **:not()**: Loại bỏ các phần tử không phù hợp với một selector cụ thể.
- **:has()**: Chọn các phần tử dựa trên các phần tử con hoặc phần tử trực tiếp của chúng.

Link tham khảo thêm các pseudo-class khác: https://www.w3schools.com/css/css_pseudo_classes.asp

Pseudo-elements:

- **::before**: Thêm nội dung vào phần tử trước nội dung của phần tử.
- **::after**: Thêm nội dung vào phần tử sau nội dung của phần tử.
- **::first-line**: Chọn dòng đầu tiên trong một phần tử văn bản.
- **::first-letter**: Chọn chữ cái đầu tiên của một phần tử văn bản.
- **::selection**: Chọn phần tử mà người dùng đã chọn.
- **::placeholder**: Chọn phần tử mà đang hiển thị giá trị placeholder.

Link tham khảo thêm các pseudo-element khác: https://www.w3schools.com/css/css_pseudo_elements.asp

3. Combinators(+ ~ >):

Trong CSS, các combinators được sử dụng để kết hợp các selector để chọn các phần tử cụ thể trong cây DOM. Dưới đây là một số combinators phổ biến:

Descendant combinator (Space):

- Ký hiệu: (khoảng trắng)
- Ví dụ: div `p` chọn tất cả các phần tử `<p>` mà là con cháu của phần tử `<div>`.

Child combinator (>):

- Ký hiệu: `>`
- Ví dụ: `ul > li` chọn tất cả các phần tử `<li>` là con trực tiếp của phần tử `<ul>`.

Adjacent sibling combinator (+):

- Ký hiệu: `+`
- Ví dụ: `h2 + p` chọn tất cả các phần tử `<p>` là anh em kế tiếp của phần tử `<h2>`.

General sibling combinator (~):

- Ký hiệu: `~`
- Ví dụ: `h2 ~ p` chọn tất cả các phần tử `<p>` là anh em của phần tử `<h2>`.

```
/* Descendant combinator */
div p {
  color: blue;
}

/* Child combinator */
ul > li {
  list-style-type: none;
}

/* Adjacent sibling combinator */
h2 + p {
  font-weight: bold;
}

/* General sibling combinator */
h2 ~ p {
  margin-top: 10px;
}
```

4. Type selectors & child selectors:

Trong CSS, Type selectors và Child selectors là hai loại selector khác nhau được sử dụng để chọn các phần tử trong cây DOM.

Type Selectors:

Type selectors (còn được gọi là element selectors) là cách chọn các phần tử dựa trên tên thẻ HTML của chúng.

Ví dụ:

```
/* Chọn tất cả các phần tử <p> */
p {
  color: blue;
}

/* Chọn tất cả các phần tử <h1> */
h1 {
  font-size: 24px;
}
```

Child Selectors:

Child selectors cho phép bạn chọn các phần tử con trực tiếp của một phần tử cụ thể. Điều này được thực hiện bằng cách sử dụng dấu >.

Ví dụ:

```
/* Chọn tất cả các phần tử <li> là con trực tiếp của <ul> */
ul > li {
  list-style-type: none;
}

/* Chọn tất cả các phần tử <span> là con trực tiếp của <div> */
div > span {
  font-weight: bold;
}
```

Sự khác biệt:

- Type selectors chọn tất cả các phần tử cùng loại trên trang, trong khi child selectors chọn các phần tử con trực tiếp của một phần tử cụ thể.
- Type selectors sử dụng tên thẻ HTML để chọn, trong khi child selectors sử dụng dấu > để chỉ định mối quan hệ giữa phần tử cha và phần tử con.

Khi sử dụng kết hợp cả hai, bạn có thể tùy chỉnh giao diện của trang web của mình một cách linh hoạt và mạnh mẽ.

Ví dụ 1: Tạo hiệu ứng trên các phần tử con khi rê chuột vào phần tử cha:

```
<div class="parent">
  <div class="child">Child 1</div>
  <div class="child">Child 2</div>
```

```
  <div class="child">Child 3</div>
</div>
```

```
/* Khi con trỏ rê chuột vào phần tử cha, thay đổi màu nền của các phần tử con */
.parent:hover > .child {
  background-color: lightblue;
}
```

Trong ví dụ này, khi bạn rê chuột vào phần tử cha `<div class="parent">`, màu nền của các phần tử con `<div class="child">` sẽ thay đổi thành lightblue.

Ví dụ 2: custom input type checkbox & type radio:

```
<div style="display: flex; flex-direction: row;" class="contrainer">
  <div class="checkbox">
    <input type="checkbox" name="" id="checkbox_input" class="checkbox_input">
    <label for="checkbox_input" class="checkbox_label">I gree </label>
  </div>
  <div class="radio">
    <input type="radio" name="" id="radio_input" class="radio_input">
    <label for="radio_input" class="radio_label">I gree </label>
  </div>
</div>
```

```
html {
  font-size: 62.5%;
  background-color: #f3faff;
}
* {
  box-sizing: border-box;
}
body {
  /* rem là đơn vị phụ thuộc vào thuộc tính font size của html */
  font-size: 1.6rem;
  font-family: "Roboto", sans-serif;
  font-weight: normal;
  padding: 2rem;
}
.checkbox {
  padding: 1rem;
}
.checkbox_input {
  display: none;
}
.checkbox_label {
  position: relative;
  padding-left: 3rem;
```

```
  cursor: pointer;
}
.checkbox_label::before {
  content: "";
  width: 2rem;
  height: 2rem;
  background-color: #eee;
  border: 2px solid #b7c1cb;
  position: absolute;
  top: 50%;
  left: 0;
  transform: translateY(-50%);
  border-radius: 3px;
}
.checkbox_label:hover::before {
  background-color: #ccc;
  transition: 0.25s ease;
}
.checkbox_label::after {
  content: "";
  position: absolute;
  top: 3px;
  left: 6px;
  width: 10px;
  height: 5px;
  transform: rotate(-45deg);
  border-left: 3px solid white;
  border-bottom: 3px solid white;
  opacity: 0;
  visibility: hidden;
}
.checkbox_input:checked + .checkbox_label::after {
  opacity: 1;
  visibility: visible;
}
.checkbox_input:checked + .checkbox_label::before {
  background-color: #fc556f;
  border: 2px solid #fc556f;
}
.checkbox_input:checked + .checkbox_label:hover:before {
  background-color: #b93a4d;
  border: 2px solid #b93a4d;
}

/* radio */

.radio {
  padding: 1rem;
}
.radio_input {
  display: none;
}
.radio_input:checked + .radio_label:before {
  background-color: #fc556f;
```

```
    box-shadow: 0 0 0 4px #f3faff, 0 0 0 6px #fc556f;
  }
  .radio_input:checked + .radio_label:hover:before {
    background-color: #b93a4d;
  }
  .radio_label {
    position: relative;
    padding-left: 3rem;
    cursor: pointer;
  }
  .radio_label::before {
    content: "";
    width: 1.5rem;
    height: 1.5rem;
    background-color: #eee;
    box-shadow: 0 0 0 4px #eee, 0 0 0 6px #b7c1cb;
    border-radius: 3rem;
    position: absolute;
    top: 50%;
    left: 0;
    transform: translateY(-50%);
    transition: 0.25s ease;
  }
  .radio_label:hover::before {
    background-color: #ccc;
  }
```