CSS

Animesh Bhadra

Semester - 1

M.Tech - Adv Information Technology

CSS and Document

Cascading Style Sheets (CSS) are a powerful way to affect the presentation of a document or a collection of documents.

The Web's Fall from Grace

As a result of pressures to make presentation in HTML in those pre-css days, markup elements like and started to creep into the language. Suddenly, a structural language started to become presentational.

CSS to the Rescue

Of course, the problem of polluting HTML with presentational markup was not lost on the World Wide Web Consortium (W3C), which began searching for a quick solution. In 1995, the consortium started publicizing a work-in-progress called CSS. By 1996, it had become a full Recommendation, with the same weight as HTML itself.

Benefits of CSS:

- Rich Styling
- Ease of Use
- Using Your Styles on Multiple Pages
- Cascading
- Compact File Size

Applying CSS

There are three ways to apply CSS to HTML: In-line, internal, and external.

1. In-line

2. Internal
3. External

- **In-Line**:-
  In-line styles are plonked straight into the HTML tags using the style attribute. They look something like this,

```
<p style="color: red">text</p>
```

- **Internal**:- Embedded, or internal, styles are used for the whole page. Inside the head element, the style tags surround all of the styles for the page.

```
<!DOCTYPE html>
<html>
<head>
<title>CSS Example</title>
<style>

    p {
        color: red;
    }

    a {
        color: blue;
    }

</style>
...
```

- **External**:-
  External styles are used for the whole, multiple-page website. There is a separate CSS file, which will simply look something like:

```
p {
    color: red;
}

a {
    color: blue;
}
```

Selectors

Whereas HTML has **tags**, CSS has **selectors**. Selectors are the names given to styles in internal and external style sheets. CSS has its own terminology to describe the CSS language.

```
strong {
  color: red;
}
```

In CSS terminology, this entire line is a rule. This rule starts with strong, which is a **Tag selector**. It selects which elements in the DOM the rule applies to.

- The part inside the curly braces is the **declaration**.
- The keyword color is a property, and red is a value.
- The semicolon after the property-value pair separates it from other property-value pairs in the same declaration.

Class selectors

Use the `class` attribute in an element to assign the element to a named class. It is up to you what name you choose for the class. Multiple elements in a document can have the same class value.

In your stylesheet, type a full stop (period) before the class name when you use it in a selector.

ID selectors

Use the `id` attribute in an element to assign an ID to the element. It is up to you what name you choose for the ID. The ID name must be unique in the document.

In your stylesheet, type a number sign (hash) before the ID when you use it in a selector.

```
<p class="key" id="principal">

.key {
  color: green;
}
```

```
#principal {
  font-weight: bolder;
}
```

Pseudo-classes selectors

A CSS `pseudo-class` is a keyword added to selectors that specifies a special state of the element to be selected. For example :hover will apply a style when the user hovers over the element specified by the selector.

Pseudo-classes, together with pseudo-elements, let you apply a style to an element not only in relation to the content of the document tree, but also in relation to external factors like the history of the navigator (:visited, for example), the status of its content (like :checked on some form elements), or the position of the mouse (like :hover which lets you know if the mouse is over an element or not).

```
selector:pseudo-class {
  property: value;
}
```

Descendant Selector

```
A E{
    color:red;
}
```

Any E element that is a descendant of an A element (that is: a child, or a child of a child, etc.)

Child Selector

```
A > E{
    color:red;
}
```

Any E element that is a child of an A element

Sibling Selector

```
A + E{
    color:red;
}
```

Any E element that is the next sibling of a B element (that is: the next child of the same parent)

You can combine these to express complex relationships.

Universal Selector

You can also use the symbol * (asterisk) to mean "any element".

Padding Borders and Margin

When your browser displays an element, the element takes up space. There are four parts to the space that it takes up.

In the middle, there is the space that the element needs to display its content.
Around that, there is padding.
Around that, there is a border.
Around that, there is a margin that separates the element from other elements.

The padding, border and margin can have different sizes on the top, right, bottom and left of the element. Any or all of these sizes can be zero.

Coloring

The padding is always the same color as the element's background. So when you set the background color, you see the color applied to the element itself and its padding. The margin is always transparent.

Borders

You can use borders to decorate elements with lines or boxes.
To specify the same border all around an element, use the `border` property. Specify the width (usually in pixels for display on a screen), the style, and the color.
You can also set the style to `none` or `hidden` to explicitly remove the border, or set the color to `transparent` to make the border invisible without changing the layout.

Floating and Positioning

You can use CSS to specify various visual effects that change the layout of your document.
When you design a layout to look similar in many browsers, your stylesheet interacts with the browser's default stylesheet and layout engine in ways that can be complex.

Float

The `float` property forces an element to the left or right. This is a simple way to control its position and size.

The rest of the document's content normally flows around the floated element. You can control this by using the `clear` property on other elements to make them stay clear of floats.

Positioning

You can specify an element's position in four ways by specifying the position property and one of the following values.

relative : The element's position is shifted relative to its normal position. Use this to shift an element by a specified amount. You can sometimes use the element's margin to achieve the same effect.

fixed : The element's position is fixed. Specify the element's position relative to the document window. Even if the rest of the document scrolls, the element remains fixed.

** absolute** : The element's position is fixed relative to a parent element. Only a parent that is itself positioned with `relative`, `fixed` or `absolute` will do. You can make any parent element suitable by specifying `position: relative`; for it without specifying any shift.

static : This is the default for every single page element. Different elements don't have different default values for positioning, they all start out as static. Static doesn't mean much, it just means that the element will flow into the page as it normally would. The only reason you would ever set an element to position: static is to forcefully-remove some positioning that got applied to an element outside of your control. This is fairly rare, as positioning doesn't cascade.

Along with these values of the `position` property (except for `static`), specify one or more of the properties: `top`, `right`, `bottom`, `left`, `width`, `height` to identify where you want the element to appear, and perhaps also its size.