Applying Styles to Web Pages using CSS

By

Narasimha Rao T

Microsoft.Net FSD Trainer

Professional Development Trainer

tnrao.trainer@gmail.com

1. Introduction to CSS

What is CSS?

CSS (Cascading Style Sheets) is a stylesheet language used to control the presentation (look and feel) of web pages written in HTML.
It defines how elements such as text, images, and layouts are displayed on screen, paper, or other media.

```
<p style="color: blue;">This text is blue.</p>
```

or using an external stylesheet:

```
p {
  color: blue;
}
```

How CSS Works

- HTML provides **structure**.
- CSS provides **style**.
- The browser combines both to render a styled web page.

Ways to Include CSS

1. **Inline CSS** — inside an HTML tag

```
<h1 style="color:red;">Hello</h1>
```

2. **Internal CSS** — inside a `<style>` tag in `<head>`

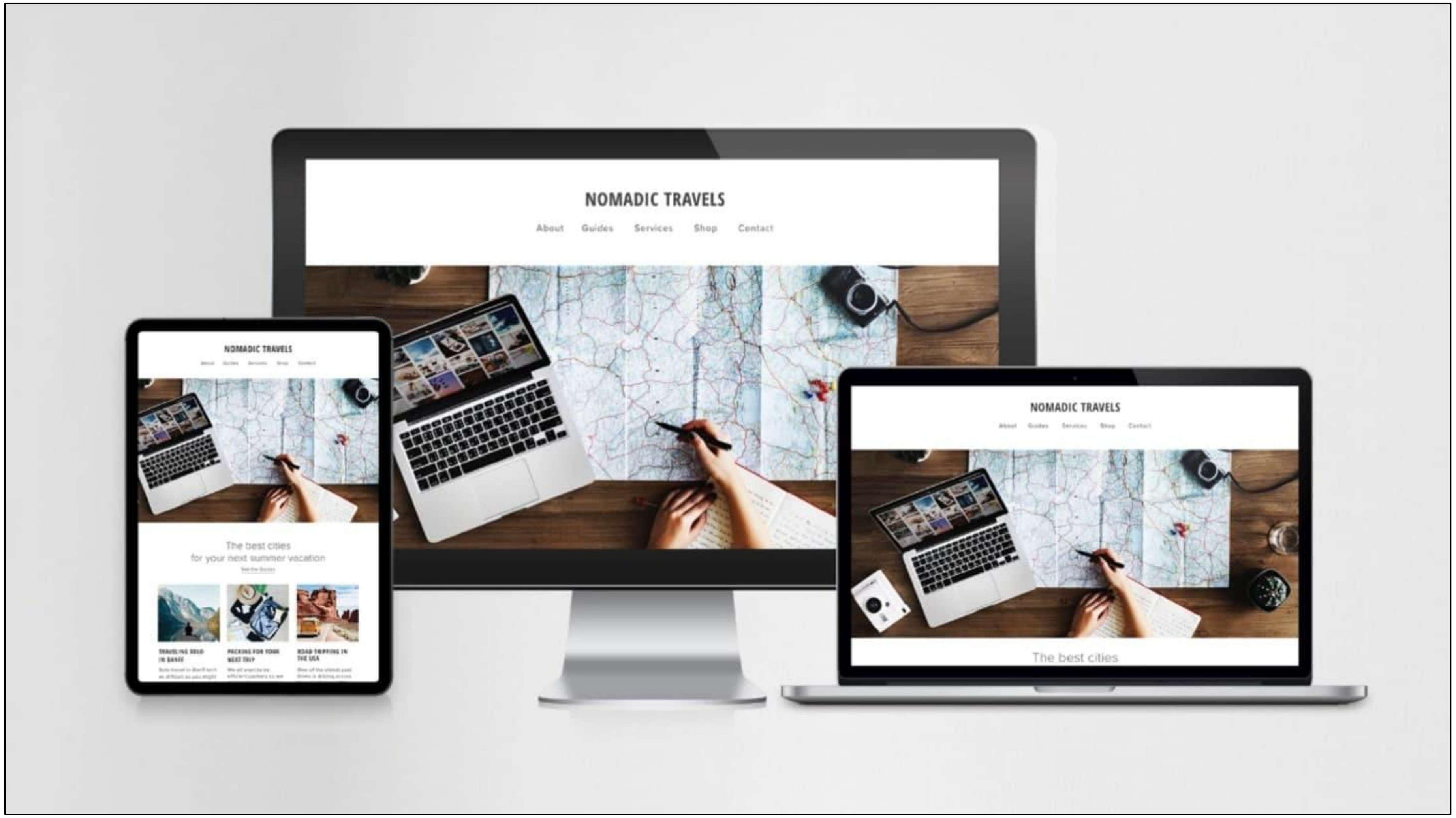
```
<style>
  h1 { color: red; }
</style>
```

3. **External CSS** — linked stylesheet file

```html
<link rel="stylesheet" href="styles.css">
```

2. Why Do We Use CSS?

- **Separation of Concerns:** Keeps design separate from structure (HTML).
- **Reusability:** One stylesheet can style multiple pages.
- **Consistency:** Apply uniform styles across a website.
- **Maintainability:** Easier to update or change designs.
- **Performance:** Reduces redundancy and page size.
- **Responsiveness:** Helps design adaptive layouts for different devices.


NOMADIC TRAVELS
About
Guides
Services
Shop
Contact
The best cities
for your next summer vacation

3. Selectors & Specificity

What are Selectors?

Selectors target HTML elements you want to style.

Examples:

```css
/* Element Selector */
p { color: gray; }

/* Class Selector */
.intro { font-size: 20px; }

/* ID Selector */
#main-title { text-transform: uppercase; }

/* Attribute Selector */
input[type="text"] { border: 1px solid #ccc; }
```

Specificity

When multiple rules apply to the same element, **specificity** determines which one wins.

Specificity Hierarchy (from lowest to highest):

1. Element selectors (e.g. `p`)
2. Class selectors (e.g. `.btn`)
3. ID selectors (e.g. `#header`)
4. Inline styles (e.g. `style="color:red;"`)
5. `!important` (overrides all others, but use sparingly)

Example:

```css
p { color: black; }         /* Low specificity */
p.intro { color: blue; }    /* Higher specificity */
#main p { color: red; }     /* Even higher */
```

4. Box Model

Every HTML element is a **rectangular box** made up of:

Part	Description
Content	The text or image inside the box
Padding	Space between content and border
Border	Wraps the padding and content
Margin	Space outside the border

Visualization:

```
┌──────────────────────────────┐  ← margin
│  ┌────────────────────────┐  │  ← border
│  │   ┌────────────────┐   │  │  ← padding
│  │   │    content     │   │  │
│  │   └────────────────┘   │  │
│  └────────────────────────┘  │
└──────────────────────────────┘
```

Example:

```css
div {
  width: 200px;
  padding: 10px;
  border: 2px solid black;
  margin: 20px;
}
```

5. Typography & Text Styling

Font Properties

```css
p {
  font-family: 'Roboto', sans-serif;
  font-size: 16px;
  font-weight: 400;
  font-style: italic;
}
```

Text Formatting

```css
h1 {
  text-align: center;
  text-transform: uppercase;
  letter-spacing: 2px;
  word-spacing: 4px;
  line-height: 1.6;
}
```

Web Fonts

Use Google Fonts or custom fonts:

```html
<link href="https://fonts.googleapis.com/css2?family=Open+Sans&display=swap" rel="stylesheet">
```

6. Colors and Backgrounds

Color Values

- Named colors: `red`, `blue`, `green`
- HEX: `#ff0000`
- RGB: `rgb(255, 0, 0)`
- RGBA (with transparency): `rgba(255, 0, 0, 0.5)`

Background Properties

```css
body {
  background-color: #fafafa;
  background-image: url('bg.jpg');
  background-repeat: no-repeat;
  background-position: center;
  background-size: cover;
}
```

7. Display and Positioning

Display Types

- `block` : takes full width (`div` , `p` , `section`)
- `inline` : fits content width (`span` , `a`)
- `inline-block` : inline but allows box model properties
- `none` : hides the element

Positioning

```
.box {
  position: relative;  /* default: static */
  top: 10px;
  left: 20px;
}
```

Position Values:

- `static` : default, not positioned
- `relative` : offset from normal position
- `absolute` : positioned relative to the nearest positioned ancestor
- `fixed` : positioned relative to the viewport
- `sticky` : switches between relative and fixed based on scroll

8. Layout Techniques

Float Layout

```
.left { float: left; width: 50%; }
.right { float: right; width: 50%; }
```

Flexbox

Powerful for one-dimensional layouts.

```
.container {
  display: flex;
  justify-content: space-between;
  align-items: center;
}
```

Grid Layout

Ideal for two-dimensional layouts.

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  gap: 10px;
}
```

Responsive Design

Use **media queries**:

```
@media (max-width: 768px) {
  .container {
    flex-direction: column;
  }
}
```

Summary

Concept	Purpose
Selectors & Specificity	Target elements accurately
Box Model	Understand element spacing
Typography	Enhance text readability
Colors & Backgrounds	Create visual appeal
Display & Positioning	Control how elements appear
Layout Techniques	Build structured, responsive pages

Q & A

Narasimha Rao T

tnrao.trainer@gmail.com