CSS

CSS, stands for Cascading Style Sheets. It is used to style websites written in .html files.

Comment

```css
/* this is a comment */
```

Styling HTML using CSS selectors

```css
p{
  color: blue;
}
.className{
  color: red;
}
#idName{
  color: green;
}
h1{
  color: aquamarine;
}
/*If an element is in some other element*/
#idName p{
  color: blue;
}
.className h1{
  color: red;
}
/*If an element is specifically in some other element*/
#idName > p{
   color: blue;
}
.className > h1{
  color: red;
}
```

Text Edit

```css
p{
  /*changes Text color form current Text color*/
  color: blue;
  /*changes Text color form current Text color in hex code*/
  color: #ffffff;
  /*changes font size in pixels*/
  font-size: 20px;
  /*changes font size in em*/
  font-size: 20em;
  /*changes font size in rem*/
  font-size: 20rem;
  /*underlining text*/
  text-decoration: underline;
  text-decoration: line-through;
  /*changing the letter spacing in pixels*/
  letter-spacing: 2px;
  /*changing the font style to italic, bold...*/
  font-style: italic; font-style: bold;
  /*changing the line height in pixels*/
  line-height: 16px;
  /*changing the text to all uppercase, lowercase, capitalized..
  text-transform: uppercase; text-transform: lowercase;
  text-transform: capitalized;
  /*add word spacing*/
  word-spacing: 10px;
}
```

Box Model

The CSS box model describes the rectangular boxes that are generated for elements in the document tree and consists of: content, padding, border and margin.

```css
.box{
  width: 200px; /* content width */
  padding: 10px; /* inside space */
  border: 2px solid #333; /* border */
  margin: 20px; /* outside space */
  box-sizing: border-box; /* include padding and border in width
}
```

Layout, display

Use display to control element layout: block, inline, inline-block, none.

```css
.hidden{ display: none; }
```

Flexbox

Flexbox makes it easy to align items in one dimension (row or column).

```css
.flex-container{
  display: flex; /* establishes flex context */
  flex-direction: row; /* row or column */
  justify-content: center; /* main-axis alignment */
  align-items: center; /* cross-axis alignment */
  gap: 10px; /* space between items */
}
.flex-item{
  flex: 1; /* grow/shrink */
  min-width: 100px;
}
```

CSS Grid

Grid provides a two-dimensional layout system for rows and columns.

```
.grid-container{
  display: grid;
  grid-template-columns: 1fr 2fr 1fr; /* column sizes */
  grid-gap: 10px; /* space between cells */
}
.grid-item{
  padding: 10px;
  border: 1px solid #ccc;
}
```

Backgrounds and Borders

```
body{
  background-color: #fafafa;
  background-image: linear-gradient(45deg, #fff, #eee);
  background-size: cover;  background-repeat: no-repeat;
}
.rounded{
  border-radius: 8px;
/* rounded corners */
  border: 2px dashed #0077cc;
}
```

Transitions and Animations

Transitions allow smooth changes between property values. Keyframe animations define more complex sequences.

```css
.button{
  background: #0077cc;
  color: #fff;
  padding: 8px 12px;
  transition: background 0.3s ease, transform 0.2s ease;
}
.button:hover{
  background: #005fa3;
  transform: translateY(-2px);
}

@keyframes fadeIn{
  from{ opacity: 0; }
  to{ opacity: 1; }
}
.fade{
  animation: fadeIn 0.6s ease both;
}
```

Responsive Design and Media Queries

Use media queries to apply styles for different viewport sizes.

```css
/* Mobile first */
.container{ width: 100%; }

@media (min-width: 768px){
  .container{ width: 750px; }
}

@media (min-width: 1024px){
  .container{ width: 960px; }
}
```

CSS Variables, Custom Properties

Define reusable values with custom properties.

```css
:root{
  --primary: #0077cc;
  --accent: #ffcc00;
  --radius: 6px;
}
.card{
  background: var(--primary);
  border-radius: var(--radius);
  color: white;
}
```

Specificity and Cascade

Specificity determines which rule wins. Order matters (later rules override earlier ones when specificity is equal).

```css
/* specificity: element < class < id */
p{
  color: black;
}
.important p{
  color: red;
}
#main p{
  color: green;
}
```

Pseudo-classes and Pseudo-elements

Use pseudo-classes like :hover, :focus and pseudo-elements like ::before, ::after.

```
a:hover{ text-decoration: underline; }
input:focus{ outline: 2px solid #0077cc; }
.item::before{
  content: "• ";
  color: var(--accent);
}
```