Let's go through each topic one by one with a detailed explanation and a code example for each.

16. CSS Variables:

CSS Variables, also known as custom properties, allow you to define reusable values in your stylesheets. They are defined using the `--` prefix and can be used to store colors, sizes, and other property values.

Code Example:

```css
/* Define CSS variables */
:root {
  --main-color: #3498db;
  --font-size: 16px;
}

/* Use CSS variables in your styles */
body {
  background-color: var(--main-color);
  font-size: var(--font-size);
}

h1 {
  color: var(--main-color);
}
```

17. Custom Properties:

Custom properties in CSS, commonly referred to as CSS variables, provide a way to define reusable values for your stylesheets. They can be changed dynamically using JavaScript.

Code Example:

```css
/* Define CSS variables */
:root {
  --main-color: #3498db;
}

/* Use CSS variables in your styles */
body {
  background-color: var(--main-color);
}

/* Change CSS variables dynamically with JavaScript */
document.documentElement.style.setProperty('--main-color', '#e74c3c');
```

18. Responsive Typography:

Responsive typography involves adjusting font sizes and styles based on the device or screen size. Media queries can be used to target specific screen sizes and apply different styles accordingly.

Code Example:

```css
/* Default font size */
body {
  font-size: 16px;
}

/* Responsive typography using media queries */
@media screen and (min-width: 768px) {
  body {
    font-size: 18px;
  }
}

@media screen and (min-width: 1024px) {
  body {
    font-size: 20px;
  }
}
```

19. Transparency and Opacity:

Transparency and opacity in CSS allow you to control the visibility of elements. The `opacity` property can be used to make an entire element transparent, while `rgba` color values can be used for transparency in specific elements.

Code Example:

```css
```

```
/* Using opacity for entire element transparency */
.element1 {
  opacity: 0.5;
}

/* Using rgba for color transparency */
.element2 {
  background-color: rgba(52, 152, 219, 0.7);
}
```

20. Filters:

CSS filters allow you to apply visual effects to elements, such as blur, brightness, contrast, and more. They are applied using the `filter` property.

Code Example:

```css
/* Applying a blur filter */
.element {
  filter: blur(5px);
}

/* Applying multiple filters */
.image {
  filter: grayscale(50%) contrast(150%) brightness(120%);
}
```

These examples provide a basic understanding of each topic. Feel free to customize the values and styles based on your specific requirements.