**CSS Grid**

### **1. Temel Kavramlar**

CSS Grid, web sayfalarındaki düzenleri oluşturmak için kullanılan güçlü bir düzen sistemidir. Grid düzeni, bir ızgara yapısı oluşturarak öğeleri bu ızgara üzerinde konumlandırmanıza olanak tanır.

* **Grid Container:** Grid düzeninin uygulandığı ana konteynerdir.
* **Grid Item:** Grid konteynerinin içindeki öğelerdir.
* **Grid Line:** Izgaranın yatay ve dikey çizgileridir, öğeler bu çizgiler arasında yerleştirilir.

### **1.1. Grid Düzeni ile Diğer Düzen Yöntemlerinin İlişkisi**

* **Flexbox:** Flexbox, tek boyutlu düzenler için idealken, Grid iki boyutlu düzenler sağlar. Flexbox ile Grid birlikte kullanılabilir.
* **Float:** Eskiden düzenlemelerde kullanılan float, Grid ile daha modern ve esnek çözümler sunar.
* **Positioning:** Grid, position özelliği ile birlikte kullanılarak daha hassas yerleşim sağlar.

### **1.2. Grid Template Areas**

* **Tanım:** Grid template areas, ızgara düzenindeki alanların adlandırılmasına olanak tanır.
* **Kullanım:** grid-template-areas özelliği ile alanları adlandırarak düzeni tanımlayabilirsiniz.

.container {  
 display: grid;  
 grid-template-areas:   
 "header header"  
 "sidebar content"  
 "footer footer";  
}

### **1.2.1. Grid Layout Kullanarak Çizgi Tabanlı Yerleştirme**

* **Tanım:** Çizgi tabanlı yerleştirme, öğelerin grid çizgileri arasında yerleştirilmesidir.
* **Kullanım:** grid-column ve grid-row özellikleri kullanılarak yapılır.

.item {  
 grid-column: 1 / 3;  
 grid-row: 1 / 2;  
}

### **1.2.2. Grid Layout Kullanarak Adlandırılmış Grid Çizgileri**

* **Tanım:** Adlandırılmış grid çizgileri, çizgileri anlamlı isimlerle adlandırarak daha anlaşılır bir düzen sağlar.
* **Kullanım:** grid-template-columns ve grid-template-rows ile birlikte adlandırılır.

.container {  
 display: grid;  
 grid-template-columns: [start] 1fr [content] 1fr [end];  
}

### **1.2.3. Grid Düzeninde Otomatik Yerleşim**

* **Tanım:** Otomatik yerleşim, öğelerin otomatik olarak boş alanlara yerleştirilmesidir.
* **Kullanım:** grid-auto-rows, grid-auto-columns ve grid-auto-flow özellikleri kullanılarak yapılır.

.container {  
 display: grid;  
 grid-auto-flow: dense;  
}

### **1.3. Box Alignments**

* **Tanım:** Box alignment, grid öğelerinin ızgara hücreleri içinde hizalanmasını sağlar.
* **Kullanım:** align-items, justify-items, align-content ve justify-content özellikleri kullanılır.

.container {  
 display: grid;  
 align-items: center;  
 justify-items: start;  
}

### **1.4. Grid, Mantıksal Değerler ve Yazı Modları**

* **Tanım:** Mantıksal değerler, yazı yönüne bağlı olarak grid düzeninin uyumlu hale gelmesini sağlar.
* **Kullanım:** writing-mode, inline-size ve block-size gibi özellikler ile birlikte kullanılır.

.container {  
 writing-mode: vertical-rl;  
 grid-template-columns: block-size(100px) inline-size(1fr);  
}

### **1.5. Grid Düzeni ve Erişilebilirlik**

* **Tanım:** Grid düzeninin erişilebilirliği, ekran okuyucular ve diğer yardımcı teknolojiler için düzenin anlaşılır olmasını sağlar.
* **Kullanım:** aria etiketleri ve rol özellikleri ile grid düzeninin erişilebilirliğini artırabilirsiniz.

<div role="grid" aria-labelledby="grid-label">  
 <div role="row">  
 <div role="gridcell">Öğe 1</div>  
 <div role="gridcell">Öğe 2</div>  
 </div>  
</div>

### **1.6. Yaygın Düzenlerin Gerçekleştirilmesi**

* **Tanım:** Yaygın düzenler, grid ile kolayca oluşturulabilir.
* **Örnekler:** Üç sütunlu düzenler, temel ızgara düzenleri ve kart düzenleri grid ile uygulanabilir.

.container {  
 display: grid;  
 grid-template-columns: repeat(3, 1fr);}

**1.7. Subgrid**

* **Tanım:** Subgrid, bir grid öğesinin kendi alt düzenini tanımlamasına olanak tanır.
* **Kullanım:** display: subgrid ile birlikte kullanılır.

.subgrid-container {  
 display: grid;  
 grid-template-columns: repeat(3, 1fr);  
}  
  
.subgrid-item {  
 display: subgrid;  
 grid-column: 1 / 3;  
}

### **1.8. Masonry Düzeni**

* **Tanım:** Masonry, öğelerin düzensiz ama düzenli bir şekilde yerleştirildiği bir düzen türüdür.
* **Kullanım:** CSS Grid ile tam olarak uygulanamasa da, bazı JavaScript kütüphaneleri ve CSS özellikleri kullanılarak yapılabilir.

.masonry {  
 column-count: 3;  
 column-gap: 1em;  
}  
  
.masonry-item {  
 break-inside: avoid;  
 margin-bottom: 1em;  
}

**2.Örnek**

<div class="wrapper">

<div class="one">One</div>

<div class="two">Two</div>

<div class="three">Three</div>

<div class="four">Four</div>

<div class="five">Five</div>

<div class="six">Six</div>

</div>

.wrapper {

display: grid;

grid-template-columns: repeat(3, 1fr);

gap: 10px;

grid-auto-rows: minmax(100px, auto);

}

.one {

grid-column: 1 / 3;

grid-row: 1;

}

.two {

grid-column: 2 / 4;

grid-row: 1 / 3;

}

.three {

grid-column: 1;

grid-row: 2 / 5;

}

.four {

grid-column: 3;

grid-row: 3;

}

.five {

grid-column: 2;

grid-row: 4;

}

.six {

grid-column: 3;

grid-row: 4;

}

