



**ACADEMY**  
OF DIGITAL ARTS  
EGYPT



Adobe



Microsoft

CompTIA.



START  
YOUR TECH JOURNEY  
WITH ADA





# Agenda

- Grid
  - Responsive Design
- 





# Grid Fundamentals

## What is CSS Grid?

- Two-dimensional layout system (rows AND columns)
- Container-based layout (like Flexbox)
- Explicit control over both axes
- When to use Grid vs Flexbox



# Grid Fundamentals

## Grid Terminology

```
/* Grid Container: The parent element */  
.grid-container {  
  display: grid; /* Creates a grid formatting context */  
}  
  
/* Grid Items: Direct children of grid container */  
.grid-item {  
  /* Automatically become grid items */  
}
```

# Grid Fundamentals

## Grid Lines, Tracks, and Areas

Grid Lines (the dividing lines):

	1	2	3	4
1				
2	A	B	C	
3	D	E	F	
4				

Grid Tracks: Rows and columns between lines

Grid Areas: Rectangular regions (can span multiple cells)

# Grid Fundamentals

## Creating Your First Grid

```
.basic-grid {  
  display: grid;  
  /* Creates a grid, but items stack in single column by default */  
}  
  
.three-column-grid {  
  display: grid;  
  grid-template-columns: 200px 200px 200px; /* 3 columns, 200px each */  
  grid-template-rows: 100px 100px;         /* 2 rows, 100px each */  
}
```

# Defining Grid Structure

## Grid Template Columns & Rows

```
.grid-examples {  
  display: grid;  
  
  /* Fixed sizes */  
  grid-template-columns: 200px 300px 200px;  
  
  /* Flexible sizes with fractions (fr) */  
  grid-template-columns: 1fr 2fr 1fr; /* 1:2:1 ratio */  
  
  /* Mixed units */  
  grid-template-columns: 200px 1fr 100px; /* Fixed sides, flexible middle */  
  
  /* Percentage */  
  grid-template-columns: 25% 50% 25%;  
  
  /* Minimum and maximum sizes */  
  grid-template-columns: minmax(200px, 1fr) minmax(300px, 2fr) minmax(100px, 1fr);  
}
```



# Defining Grid Structure

## The Repeat Function

```
.repeat-examples {  
  display: grid;  
  
  /* Basic repeat */  
  grid-template-columns: repeat(4, 1fr); /* Same as: 1fr 1fr 1fr 1fr */  
  
  /* Repeat with different patterns */  
  grid-template-columns: repeat(3, 200px 100px); /* 200px 100px 200px 100px 200px 100px */  
  
  /* Auto-fit: Fits items in available space */  
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));  
  
  /* Auto-fill: Creates empty columns if space available */  
  grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));  
}
```

# Defining Grid Structure

## Understanding Auto-fit vs Auto-fill

```
/* Container width: 1000px */  
  
.auto-fit-demo {  
  /* If 3 items fit: each gets ~333px (1000px ÷ 3) */  
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));  
}  
  
.auto-fill-demo {  
  /* If 3 items fit: items get 250px, remaining space stays empty */  
  grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));  
}
```

# Defining Grid Structure

## Grid Gaps (Gutters)

```
.grid-with-gaps {  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
  grid-template-rows: repeat(2, 200px);  
  
  /* Modern syntax */  
  gap: 20px;           /* Same gap for rows and columns */  
  row-gap: 10px;       /* Vertical gap */  
  column-gap: 20px;    /* Horizontal gap */  
  
  /* Legacy syntax (still supported) */  
  grid-gap: 20px;  
  grid-row-gap: 10px;  
  grid-column-gap: 20px;  
}
```

# Grid Item Placement

## Grid Line Positioning

```
.positioned-items {  
  display: grid;  
  grid-template-columns: repeat(4, 1fr);  
  grid-template-rows: repeat(3, 100px);  
  gap: 10px;  
}  
  
.item-1 {  
  /* Place using grid lines (start from 1) */  
  grid-column-start: 1;  
  grid-column-end: 3;      /* Spans columns 1-2 */  
  grid-row-start: 1;  
  grid-row-end: 2;  
}  
  
.item-2 {  
  /* Shorthand syntax */  
  grid-column: 3 / 5;      /* Columns 3-4 */  
  grid-row: 1 / 2;        /* Row 1 */  
}
```

```
.item-3 {  
  /* Span notation */  
  grid-column: span 2;     /* Spans 2 columns from auto-placement */  
  grid-row: span 2;       /* Spans 2 rows */  
}  
  
.item-4 {  
  /* Negative line numbers (count from end) */  
  grid-column: 1 / -1;    /* Full width */  
  grid-row: 3 / 4;  
}
```

# Grid Item Placement

## Grid Area Shorthand

```
.area-placement {  
  /* grid-area: row-start / column-start / row-end / column-end */  
  grid-area: 1 / 1 / 3 / 3; /* Top-left 2x2 square */  
  grid-area: 2 / 3 / 4 / 5; /* Bottom-right 2x2 square */  
}
```



# Grid Item Placement

## Named Grid Lines

```
.named-lines-grid {  
  display: grid;  
  grid-template-columns:  
    [sidebar-start] 250px  
    [sidebar-end main-start] 1fr  
    [main-end];  
  grid-template-rows:  
    [header-start] 80px  
    [header-end content-start] 1fr  
    [content-end footer-start] 60px  
    [footer-end];  
}  
  
.header {  
  grid-column: sidebar-start / main-end;  
  grid-row: header-start / header-end;  
}
```

```
.sidebar {  
  grid-column: sidebar-start / sidebar-end;  
  grid-row: content-start / content-end;  
}  
  
.main-content {  
  grid-column: main-start / main-end;  
  grid-row: content-start / content-end;  
}
```

# Grid Template Areas

## Defining Layout with Areas

```
.area-layout {  
  display: grid;  
  grid-template-columns: 200px 1fr 200px;  
  grid-template-rows: 80px 1fr 60px;  
  grid-template-areas:  
    "header header header"  
    "sidebar content ads"  
    "footer footer footer";  
  gap: 1rem;  
  min-height: 100vh;  
}  
  
/* Assign areas to items */  
.header { grid-area: header; }  
.sidebar { grid-area: sidebar; }  
.content { grid-area: content; }  
.ads { grid-area: ads; }  
.footer { grid-area: footer; }
```

# Grid Template Areas

## Empty Areas and Complex Shapes

```
.complex-areas {  
  display: grid;  
  grid-template-columns: repeat(4, 1fr);  
  grid-template-rows: repeat(4, 100px);  
  grid-template-areas:  
    "logo    nav    nav    search"  
    "hero    hero    sidebar sidebar"  
    "content content sidebar sidebar"  
    "footer  footer  footer  footer";  
}  
  
/* Use "." for empty areas */  
.l-shaped-layout {  
  grid-template-areas:  
    "header header ."  
    "nav    content content"  
    "nav    content content";  
}
```

# Grid Template Areas

## Responsive Area Reconfiguration

```
.responsive-areas {  
  display: grid;  
  gap: 1rem;  
  
  /* Mobile: stacked layout */  
  grid-template-areas:  
    "header"  
    "nav"  
    "content"  
    "sidebar"  
    "footer";  
}  
  
@media (min-width: 768px) {  
  .responsive-areas {  
    grid-template-columns: 200px 1fr 200px;  
    grid-template-areas:  
      "header header header"  
      "nav content sidebar"  
      "footer footer footer";  
  }  
}
```

# Responsive Design Mastery

## Mobile-First Approach

```
/* Mobile First: Start with mobile styles (no media query needed) */  
.responsive-container {  
  padding: 1rem;  
  max-width: 100%;  
}  
  
.responsive-grid {  
  display: grid;  
  grid-template-columns: 1fr; /* Single column on mobile */  
  gap: 1rem;  
}
```

```
/* Then enhance for larger screens */  
@media (min-width: 768px) {  
  .responsive-container {  
    padding: 2rem;  
    max-width: 1200px;  
    margin: 0 auto;  
  }  
  
  .responsive-grid {  
    grid-template-columns: repeat(2, 1fr); /* Two columns on tablet */  
    gap: 2rem;  
  }  
}  
  
@media (min-width: 1024px) {  
  .responsive-grid {  
    grid-template-columns: repeat(3, 1fr); /* Three columns on desktop */  
  }  
}
```



# Responsive Design Mastery

## Common Breakpoints

```
/* Common breakpoint system */  
/* Mobile: 0-767px (no media query needed) */  
  
@media (min-width: 768px) { /* Tablet */  
    /* Tablet styles */  
}  
  
@media (min-width: 1024px) { /* Desktop */  
    /* Desktop styles */  
}  
  
@media (min-width: 1200px) { /* Large Desktop */  
    /* Large desktop styles */  
}
```

# Responsive Design Mastery

## Content-Based Breakpoints

```
/* Instead of device-based, use content-based breakpoints */
.card-grid {
  display: grid;
  grid-template-columns: 1fr;
  gap: 1rem;
}

/* When content needs more space */
@media (min-width: 500px) {
  .card-grid {
    grid-template-columns: repeat(2, 1fr);
  }
}

/* When content can benefit from 3 columns */
@media (min-width: 800px) {
  .card-grid {
    grid-template-columns: repeat(3, 1fr);
  }
}
```

# Flexible Units and Modern CSS Functions

## Responsive Units

```
.responsive-typography {  
  /* Viewport units */  
  font-size: 4vw;      /* 4% of viewport width */  
  line-height: 6vh;    /* 6% of viewport height */  
  
  /* Container-relative units */  
  padding: 5vmin;      /* 5% of smaller viewport dimension */  
  margin: 3vmax;       /* 3% of larger viewport dimension */  
  
  /* Relative units */  
  font-size: 1.2rem;    /* Relative to root font size */  
  padding: 2em;         /* Relative to element's font size */  
  
  /* Percentage */  
  width: 90%;           /* 90% of parent width */  
  max-width: 1200px;    /* But never exceed 1200px */  
}
```

# Flexible Units and Modern CSS Functions

## Modern CSS Functions

```
.modern-responsive {  
  /* Clamp: min, preferred, max */  
  font-size: clamp(1rem, 4vw, 2.5rem);  
  padding: clamp(1rem, 5vw, 3rem);  
  gap: clamp(1rem, 3vw, 2rem);  
  
  /* Min/Max for flexible sizing */  
  width: min(90%, 1200px);           /* Smaller of 90% or 1200px */  
  height: max(200px, 50vh);         /* Larger of 200px or 50vh */  
  
  /* Calc for precise calculations */  
  width: calc(100% - 2rem);          /* Full width minus padding */  
  margin-left: calc(50% - 50vw);     /* Negative margin technique */  
  
  /* Grid with modern functions */  
  grid-template-columns: repeat(auto-fit, minmax(clamp(250px, 30%, 400px), 1fr));  
}
```

# Flexible Units and Modern CSS Functions

## Intrinsic Web Design

```
.intrinsic-layout {  
  display: grid;  
  
  /* Grid adapts to content and container */  
  grid-template-columns: repeat(auto-fit, minmax(min(300px, 100%), 1fr));  
  
  /* Flexible gaps */  
  gap: clamp(1rem, 3vw, 2rem);  
  
  /* Container padding */  
  padding: clamp(1rem, 5vw, 3rem);  
  
  /* Max width with centering */  
  max-width: min(1200px, 90%);  
  margin-inline: auto;  
}
```



THANK YOU

ADAEGY

