

# CSS and Responsive Web Design

# Pseudo classes in CSS

- Defined a special state of an element
  - Examples:
    - Mouse over
    - Visited/unvisited link
    - Element on focus
- : before class name
  - Examples:
    - :root
    - :hover

# Example

## HTML

```
<div>  
  <a href="...">test</a>  
</div>
```

## CSS

```
a:link {  
  color: blue;  
}  
a:hover {  
  color: red;  
}
```

# Recommended reading

- W3School's Pseudo Classes
  - [https://www.w3schools.com/css/css\\_pseudo\\_classes.asp](https://www.w3schools.com/css/css_pseudo_classes.asp)

# Pseudo elements in CSS

- Style specific parts of an element
  - Examples:
    - First letter
    - First line
- Syntax
  - `selector::pseudo-element`
  - Example:
    - `div::first-line`

# Example

## HTML

```
<div>  
  first line<br>  
  second line<br>  
</div>
```

## CSS

```
div::first-line {  
  color: red;  
}
```

# Recommended reading

- W3School's CSS Pseudo-elements
  - [https://www.w3schools.com/css/css\\_pseudo\\_elements.asp](https://www.w3schools.com/css/css_pseudo_elements.asp)

# Variables in CSS

- Also known as CSS Custom Properties
- Variable's name begins with 2 dashes (--)
  - Example:
    - `--color: red; /* variable name is --color and its value is red */`
- CSS's variables also have scope
  - Global or local
  - Variables declare in `:root` or `html` selector become global var.
  - Variables declare in other selectors become local var.
- Value of the CSS's variable can be referenced by `var()`
  - Example:
    - `color: var(--color);`



# Recommended reading

- Everything you need to know about CSS Variables by Emmanuel Ohans
  - <https://www.freecodecamp.org/news/everything-you-need-to-know-about-css-variables-c74d922ea855/>

# Content Lay out

- Flex
  - Efficient way to lay out, align and distribute space
  - Can be applied even when content size is unknown or dynamic
  - Intended for 1-dimensional layouts
- Grid
  - 2-dimensional layout system

# Flex

- Composed of 2 parts
  - The container
  - The items (which reside in the container)
- The size of the items can be automatically altered to best filled the container
- Directional-agnostic (free from any directional constraints)

# Example

## HTML

```
<div class="container">  
  <div class="item">  
    item1  
  </div>  
  <div class="item">  
    item2  
  </div>  
</div>
```

## CSS

```
.container {  
  display: flex;  
}  
.item {  
  border-style: dotted;  
}
```

# Recommended reading

- A Complete Guide to Flexbox by Chris Coyier
  - <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

# Grid

- Same as flex, composed of 2 parts
  - Container and items

# Example

## HTML

```
<div class="container">  
  <div class="item1">  
    item01  
  </div>  
  <div class="item2">  
    item02  
  </div>  
  <div class="item3">  
    item03  
  </div>  
  <div class="item4">  
    item04  
  </div>  
</div>
```

## CSS

```
.container {  
  display: grid;  
}  
.item1 {  
  grid-column-start: 1;  
  grid-row-start: 1;  
}  
.item2 {  
  grid-column-start: 2;  
  grid-row-start: 1;  
}
```

# Recommended reading

- A Complete Guide to Grid by Chris House
  - <https://css-tricks.com/snippets/css/complete-guide-grid/>



# CSS position revisited

- position property define how to position the element in the document flow
- Default value is static – normal position in document flow (so the top/bottom/left/right properties will be no effect on the element)
- Other possible values of position
  - Relative
  - Absolute
  - Fixed
  - Sticky
  - Inherit
- Recommended reading: <https://css-tricks.com/almanac/properties/p/position/>

# Position of parent and child element

- If position of parent element is not specific
  - child with absolute position will be positioned related to grand parent or great grand parent with position absolute or relative
  - No ancestor with absolute or relative position, child will be positioned related to display area

# Responsive Web Design (RWD)

# Introduction

- Viewport = viewable area of a browser
- RWD displays web content relates to viewport
  - Utilizes CSS media queries to target breakpoints
  - Breakpoints or CSS breakpoints are the widths that the page (layout) switches to a different view that is better suited to that viewport.
  - The main concept is to "shrink to fit"



Image Source: <https://learn.onemonth.com/responsive-vs-adaptive-vs-fluid-design/>

# Other than RWD there are...

- AWD (Adaptive Web Design)
  - Detects screen size (viewport) and then uses the most suitable static style.
  - Also uses CSS media queries
  - At least 6 layout sets to cover a common viewport (for now)
- FWD (Fluid Web Design)
  - Use percentage for widths
- Fixed Design
  - Design based on fixed pixel widths

# Related

- Relative unit
  - %
  - em = length relative to font size of the element
  - rem = length relative to font size of the root element
- CSS units and relative units
  - [https://www.w3schools.com/cssref/css\\_units.asp](https://www.w3schools.com/cssref/css_units.asp)

# Image

- Display image which width related to viewport
  - max-width: 100%

# CSS media queries

```
@media screen and (min-width: nnnpx) {  
    style1  
}
```

```
@media screen and (min-width: ooopx) {  
    style2  
}
```



# CSS media queries in link tags

- `<link rel="style sheet" type="text/css" media="screen" href="s.css">`
- `<link rel="style sheet" type="text/css" media="screen and (orientation: portrait)" href="p-s.css">`

# Media queries with @import

- Conditionally load style sheets into the existing style sheet
- `@import url("small.css") screen and (max-width:320px);`

# Media Queries

- Basic Media Queries by W3School
  - [https://www.w3schools.com/css/css3\\_mediaqueries.asp](https://www.w3schools.com/css/css3_mediaqueries.asp)
- CSS Variables and Media queries section in Everything you need to know about CSS variable by Emmanuel Ohans
  - <https://www.freecodecamp.org/news/everything-you-need-to-know-about-css-variables-c74d922ea855/>

# Media queries for high-resolution device

- @media (min-resolution: 2dppx)
  - 2dppx = 2 dots per pixel
  - 1dppx = 96dpi
  - 192dpi
  - 1 pixel = 1/96 inch
  - 1 point = 1/72 inch