شیوه نامه های آبشاری

CASCADING STYLE SHEETS

M.Madadyar

http://www.Madadyar.ir

Advantages of Using CSS

- Separation of presentation & content
- Less time needed for site maintenance & revisions
- Better accessibility
- Better semantics
- Improved graphic design possibilities
- Faster page load times

Structure and presentation combined

```
<h1 align="center">
  <font size="7"
        color="#800080">
    <strong>A Purple Heading</strong>
  </font>
</h1>

<h1 align="center">
  <font size="7"
        color="#800080">
    <strong>Another Purple Heading</strong>
  </font>
</h1>
```

The CSS Version

```
<h1>A Purple Heading</h1>

<h1> Another Purple Heading</h1>
```

the associated CSS stylesheet

```
h1 {
        font-family: "Tahoma";
        font-size: 14px;
        font-weight: bold;
        color: #800080;
        text-align: center;
}
```

CSS Structure

```
selector
{
property: value; property: value;...
}
```

- **Selector**
  - on a simple level HTML element you wish to define
- **Property**
  - attribute you wish to change
- **Value**
  - value the property takes

Internal Style Sheets (1/3)

- **inline**
  - override other author styles.
  - only affect an individual element.

```
<h1 style="color: purple;">Purple inline</h1>
```

- **embed**
  - applies only to the document.
  - appears in head section.

```
<style type="text/css"> css codes </style>
```

Internal Style Sheets (2/3)

- Inline Example

```
<html>
  ...

  <body>
   ...
   <p style="font-family:Arial,sans-serif; ">
   some text
   </p>
<p>salam</p>
  </body>
</html>
```

Internal Style Sheets (3/3)

- Embedded Example

```
<html>
  <head>
     <style type="text/css">
        p {font-family: Arial, sans-serif;}
     </style>
  </head>

  <body>
    ...
    <p>some text</p>
<p>salam</p>
  </body>
</html>
```

External Style Sheets (1/5)

- **linked**
  - style sheet is external to the document
  - placed inside the head

```
<head>
<link href="file.css" rel="stylesheet" type="text/css" />
</head>
```

- **imported**
  - style sheet is external to the document
  - placed inside the head

```
<style type="text/css">
@import url(mystyle.css);
…..
</style>
```

External Style Sheets

- Separate text file (.css)
  - e.g. styles.css

```
p {
  font-family: Arial, Sans-serif;
  }
```

External Style Sheets (3/5)

- Linked styles Example

```
<html>
    <head>
        ...
<link href="styles.css"
                rel="stylesheet" type="text/css" />
    </head>
    <body>
        ...
        <p>some text</p>
    </body>
</html>
```

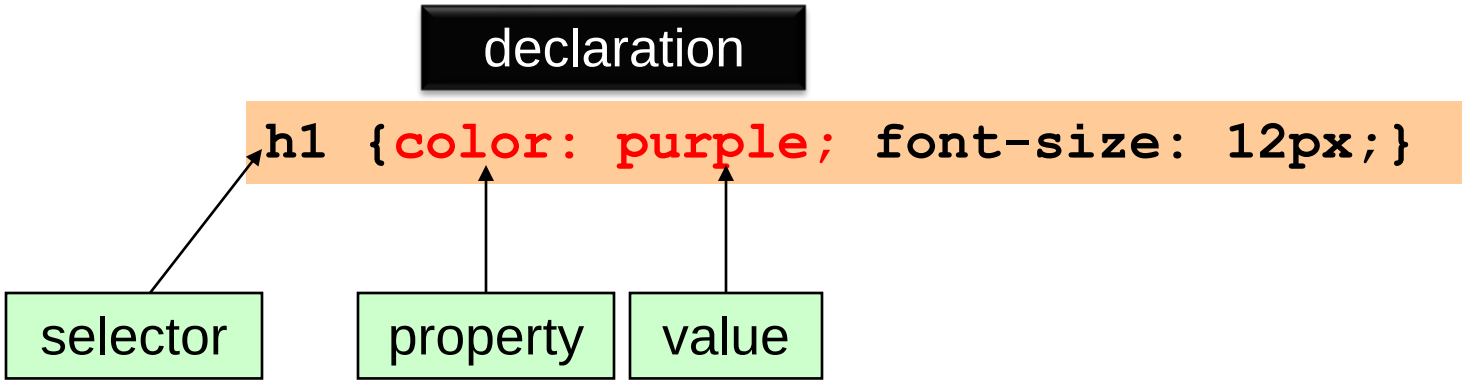
CSS rules

- CSS allows rules to be created
  - can apply to all elements of a particular type
  - e.g.

```
h1 {color: purple;}
```

    - makes all <h1> text purple

- the general form of a rule is



Element Selectors

- elements of a document are the most basic selectors
  - they may be grouped:

```
h1, h2, p {color: purple}
```

    make headers 1 and 2 and paragraphs purple
  - sometimes more than keyword may be used for a properties value

```
p {font: bold "Courier New";}
```

    sets the paragraph font
    the space between values is a separator
- the universal selector matches all elements

```
* {color: purple;}
```

  assign the color purple to every element in the document
- Body of page:

```
body {direction: rtl;}
```

Class selectors

- often it is desirable to assign styles without specifying an element.
- How to create a class in css:

.className {style... }

```
.pp2 {color: purple;}
```

```
<h1 class="pp2"> a purple one </h1>

<span class="pp2"> some purple </span>
```

Specific class selectors

- class selectors can made be specific to an element

```
.sama {color: purple;}
p.sama {font-style: italic}
```

```
<h1>A normal heading</h1>
<h1 class="sama"> and a purple one</h1>
<h1> or a heading with <p class="sama"> some
purple</p> color and italic</h1>
```

Combining class selectors

- they can also be combined both in the CSS and in the markup

```
.red {color: red;}
.alert {font-weight: bold}
```

```
<h1 class="red">A red heading</h1>
<h1 class="alert">An alert</h1>
<h1 class="red alert">Red alert!!!!</h1>
```

ID selectors

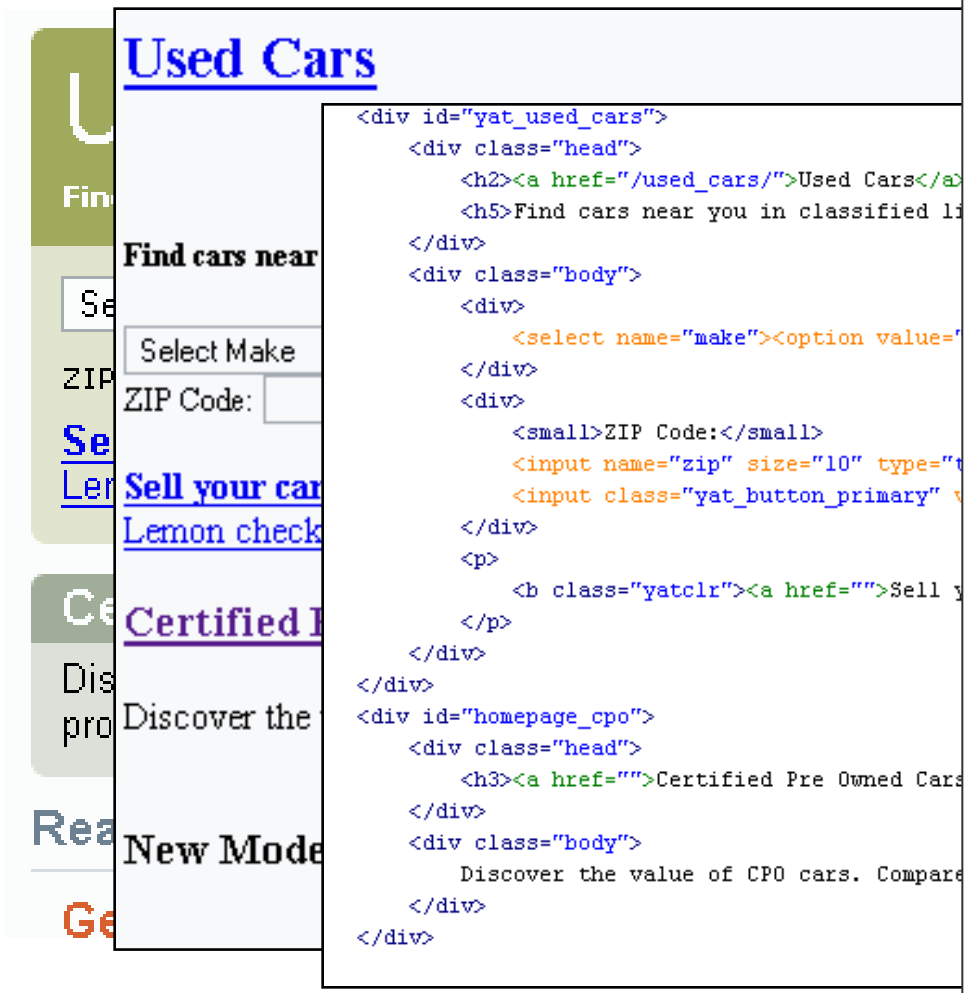
- similar to class selectors but
  - preceded by a **#**
  - refer to values in **id** attributes

#IDName {style… }

```
#special {color: purple; background: black}
```

```
<h1 id= "special"> and a special one</h1>
```

Rich Meaning Within Each Module




```
<div id="yat_used_cars">
    <div class="head">
        <h2><a href="/used_cars/">Used Cars</a>
        <h5>Find cars near you in classified li
    </div>
    <div class="body">
        <div>
            <select name="make"><option value="
        </div>
        <div>
            <small>ZIP Code:</small>
            <input name="zip" size="10" type="t
            <input class="yat_button_primary" v
        </div>
        <p>
            <b class="yatclr"><a href="">Sell y
        </p>
    </div>
</div>
<div id="homepage_cpo">
    <div class="head">
        <h3><a href="">Certified Pre Owned Cars
    </div>
    <div class="body">
        Discover the value of CPO cars. Compare
    </div>
</div>
```

```
#homepage_cpo {
    margin-top: 10px;
}
#homepage_cpo .head {
    color: #fff;
    background-color: #A0AD99;
    padding: 0px 10px 4px 10px;
}
#homepage_cpo .head H3 {
    color: #fff;
    margin:0;
}
#homepage_cpo .head H3 A {
    color: #fff;
    text-decoration: none;
}
#homepage_cpo .body {
    padding: 5px 10px 7px 10px;
    background-color: #DBE0D9;
}
```

Advanced CSS

- Pseudo-classes
- Pseudo-elements
- CSS Attribute Selectors
- CSS Counters
- CSS Units
- CSS Web Fonts
- CSS 2D & 3D Transforms
- CSS Transitions
- CSS Animations
- CSS Media Queries
- CSS Functions

PSEUDO-CLASSES

- A pseudo-class is used to define a special state of an element.
- For example, it can be used to:
  - Style an element when a user mouses over it
  - Style visited and unvisited links differently
  - Style an element when it gets focus

```
selector:pseudo-class {
    property:value;
}
```

PSEUDO-CLASSES

Selector	Example	Example description
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the <input> element that has focus
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects <input> elements with a value within a specified range
:invalid	input:invalid	Selects all <input> elements with an invalid value
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"

PSEUDO-CLASSES

Example

```
/* unvisited link */
a:link {
  color: #FF0000;
}

/* visited link */
a:visited {
  color: #00FF00;
}

/* mouse over link */
a:hover {
  color: #FF00FF;
}

/* selected link */
a:active {
  color: #0000FF;
}
```

PSEUDO-ELEMENTS

- A CSS pseudo-element is used to style specified parts of an element.
- For example, it can be used to:
  - Style the first letter, or line, of an element
  - Insert content before, or after, the content of an element

```
selector::pseudo-element {
    property:value;
}
```

PSEUDO-ELEMENTS

All CSS Pseudo Elements

Selector	Example	Example description
::after	p::after	Insert something after the content of each <p> element
::before	p::before	Insert something before the content of each <p> element
::first-letter	p::first-letter	Selects the first letter of each <p> element
::first-line	p::first-line	Selects the first line of each <p> element
::selection	p::selection	Selects the portion of an element that is selected by a user

Example

```
h1::before {
  content: url(smiley.gif);
}
```

```
p::first-letter {
  color: #ff0000;
  font-size: xx-large;
}

p::first-line {
  color: #0000ff;
  font-variant: small-caps;
}
```

CSS ATTRIBUTE SELECTORS

- **Style HTML Elements With Specific Attributes**
- It is possible to style HTML elements that have specific attributes or attribute values

Selector	Example	Example description
[*attribute*]	[target]	Selects all elements with a target attribute
[*attribute=value*]	[target=_blank]	Selects all elements with target="_blank"
[*attribute~=value*]	[title~=flower]	Selects all elements with a title attribute containing the word "flower"
[*attribute|=value*]	[lang|=en]	Selects all elements with a lang attribute value starting with "en"
[*attribute^=value*]	a[href^="https"]	Selects every <a> element whose href attribute value begins with "https"
[*attribute$=value*]	a[href$=".pdf"]	Selects every <a> element whose href attribute value ends with ".pdf"
[*attribute*=value*]	a[href*="w3schools"]	Selects every <a> element whose href attribute value contains the substring "w3schools"

CSS ATTRIBUTE SELECTORS

Example

```
a[target="_blank"] {
  background-color: yellow;
}
```

```
[class^="top"] {
  background: yellow;
}
```

```
input[type="text"] {
  width: 150px;
  display: block;
  margin-bottom: 10px;
  background-color: yellow;
}

input[type="button"] {
  width: 120px;
  margin-left: 35px;
  display: block;
}
```

CSS ATTRIBUTE SELECTORS

```
<!DOCTYPE html>
<html>
<head>
<style>
[title~=flower] {
  border: 5px solid yellow;
}
</style>
</head>
<body>

<p>All images with the title attribute containing the word "flower" get a yellow border.</p>

<img src="klematis.jpg" title="klematis flower" width="150" height="113">
<img src="img_flwr.gif" title="flower" width="224" height="162">
<img src="img_tree.gif" title="tree" width="200" height="358">

</body>
</html>
```

All images with the title attribute containing the word "flower" get a yellow border.

CSS COUNTERS

- CSS counters are "variables" maintained by CSS whose values can be incremented by CSS rules (to track how many times they are used).
  - Counters let you adjust the appearance of content based on its placement in the document.
- To work with CSS counters we will use the following properties:
  - counter-reset - Creates or resets a counter
  - counter-increment - Increments a counter value
  - content - Inserts generated content
  - counter() or counters() function - Adds the value of a counter to an element
- To use a CSS counter, it must first be created with counter-reset.

CSS COUNTERS

Example

```css
body {
  counter-reset: section;
}

h2::before {
  counter-increment: section;
  content: "Section " counter(section) ": ";
}
```

```html
<h1>Using CSS Counters:</h1>
<h2>HTML Tutorial</h2>
<h2>CSS Tutorial</h2>
<h2>JavaScript Tutorial</h2>
```

Using CSS Counters:

Section 1: HTML Tutorial

Section 2: CSS Tutorial

Section 3: JavaScript Tutorial

CSS UNITS

- CSS has several different units for expressing a length.
  - Many CSS properties take "length" values, such as width, margin, padding, font-size, etc.
  - Length is a number followed by a length unit, such as 10px, 2em, etc.

Unit	Description
cm	centimeters
mm	millimeters
in	inches (1in = 96px = 2.54cm)
px *	pixels (1px = 1/96th of 1in)
pt	points (1pt = 1/72 of 1in)
pc	picas (1pc = 12 pt)

*** Pixels (px) are relative to the viewing device. For low-dpi devices, 1px is one device pixel (dot) of the display.**

CSS UNITS

Relative Lengths

Unit	Description
em	Relative to the font-size of the element (2em means 2 times the size of the current font)
ex	Relative to the x-height of the current font (rarely used)
ch	Relative to width of the "0" (zero)
rem	Relative to font-size of the root element
vw	Relative to 1% of the width of the viewport*
vh	Relative to 1% of the height of the viewport*
vmin	Relative to 1% of viewport's* smaller dimension
vmax	Relative to 1% of viewport's* larger dimension
%	Relative to the parent element

*** Viewport = the browser window size. If the viewport is 50cm wide, 1vw = 0.5cm.**

CSS WEB FONTS

- **The CSS @font-face Rule**
  - Web fonts allow Web designers to use fonts that are not installed on the user's computer.
- **TrueType Fonts (TTF)**
- **OpenType Fonts (OTF)**
- ***The Web Open Font Format (WOFF)***
- **The Web Open Font Format (WOFF 2.0)**
- **SVG Fonts/Shapes**
- **Embedded OpenType Fonts (EOT)**

CSS WEB FONTS

Example

```
@font-face {
  font-family: myFirstFont;
  src: url(sansation_light.woff);
}

div {
  font-family: myFirstFont;
}
```

```
@font-face {
  font-family: myFirstFont;
  src: url(sansation_bold.woff);
  font-weight: bold;
}
```

CSS 2D & 3D TRANSFORMS

- CSS transforms allow you to move, rotate, scale, and skew elements.
- CSS also supports 3D transformations.
- With the CSS transform property you can use the following 3D transformation methods:
  - rotateX()
  - rotateY()
  - rotateZ()

```
#myDiv {
  -webkit-transform: rotateY(130deg); /* Safari prior 9.0 */
  transform: rotateY(130deg); /* Standard syntax */
}
```

CSS TRANSITIONS

- CSS transitions allows you to change property values smoothly, over a given duration.
  - transition
  - transition-delay
  - transition-duration
  - transition-property
  - transition-timing-function (linear, ease-in , …)

```
div {
  width: 100px;
  height: 100px;
  background: red;
  -webkit-transition: width 2s; /* For Safari 3.1 to 6.0 */
  transition: width 2s;
}

div:hover {
  width: 300px;
}
```

CSS ANIMATIONS

- CSS allows animation of HTML elements without using JavaScript or Flash!
  - @keyframes
  - animation-name
  - animation-duration
  - animation-delay
  - animation-iteration-count
  - animation-direction
  - animation-timing-function
  - animation-fill-mode
  - animation

CSS ANIMATIONS

Example

```
/* The animation code */
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

```
/* The animation code */
@keyframes example {
  0%   {background-color: red;}
  25%  {background-color: yellow;}
  50%  {background-color: blue;}
  100% {background-color: green;}
}
```

```
/* Safari 4.0 - 8.0 */
@-webkit-keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}
```

CSS MEDIA QUERIES

- The @media rule, introduced in CSS2, made it possible to define different style rules for different media types.
  - You could have one set of style rules for computer screens, one for printers, one for handheld devices, one for television-type devices, and so on.
- Media queries in CSS3 extended the CSS2 media types idea:
  - Instead of looking for a type of device, they look at the capability of the device.
- Media queries can be used to check many things, such as:
  - width and height of the viewport
  - width and height of the device
  - orientation (is the tablet/phone in landscape or portrait mode?)
  - resolution

The viewport is the user's visible area of a web page.

CSS MEDIA QUERIES

CSS Syntax

```
@media not|only mediatype and (mediafeature and|or|not mediafeature)
{
  CSS-Code;
}
```

```
body {
  background-color: lightblue;
}

@media screen and (min-width: 400px) {
  body {
    background-color: lightgreen;
  }
}

@media screen and (min-width: 800px) {
  body {
    background-color: lavender;
  }
}
```

CSS FUNCTIONS

- CSS functions are used as a value for various CSS properties.

Function	Description
attr()	Returns the value of an attribute of the selected element
calc()	Allows you to perform calculations to determine CSS property values
cubic-bezier()	Defines a Cubic Bezier curve
hsl()	Defines colors using the Hue-Saturation-Lightness model (HSL)
hsla()	Defines colors using the Hue-Saturation-Lightness-Alpha model (HSLA)
linear-gradient()	Sets a linear gradient as the background image. Define at least two colors (top to bottom)
radial-gradient()	Sets a radial gradient as the background image. Define at least two colors (center to edges)

CSS FUNCTIONS

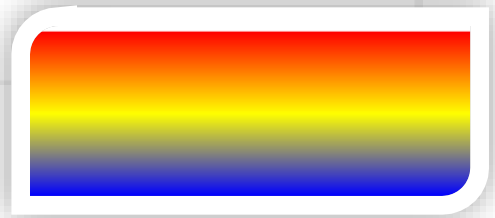
Example

Use calc() to calculate the width of a element:

```
#div1 {
  position: absolute;
  left: 50px;
  width: calc(100% - 100px);
  border: 1px solid black;
  background-color: yellow;
  padding: 5px;
  text-align: center;
}
```

This linear gradient starts at the top. It starts red, transitioning to yellow, then to blue:

```
#grad {
  background-image: linear-gradient(red, yellow, blue);
}
```



http://www.w3schools.com/CSS

http://www.w3.org/Style/CSS/

http://jigsaw.w3.org/css-validator/

Useful Websites for CSS