**CSS заметки**

***Селекторы:***

selector {

свойство: значение;

}

p / h1/ etc - типы

#name - id селектор

.name – селектор класса

To select elements with a specific class, use a period character, followed by the name of the class.  
Do **NOT**start a class or id name with a number!!!!

***Селекторы – Потомки:***

см. примеры в WebStorm

***Комментарии***

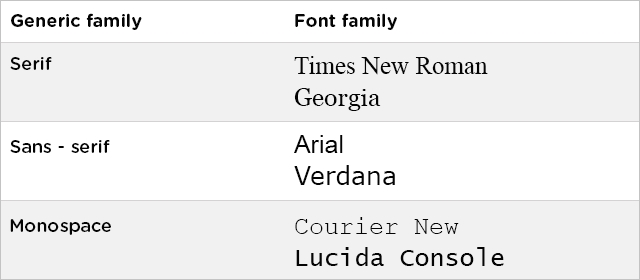
Comments are used to explain your code, and may help you when you edit the source code later. Comments are ignored by browsers.  
  
A CSS comment look like this:

/\* Comment goes here \*/

Comments can also span multiple lines.

***The font-family Property***

The font-family property specifies the font for an element.   
There are two types of font family names:   
- **font family**: a specific font family (like Times New Roman or Arial)  
- **generic family**: a group of font families with a similar look (like Serif or Monospace)  
  
Here is an example of different font styles:



Separate each value with a **comma** to indicate that they are alternatives.   
If the name of a font family is more than one word, it must be in quotation marks:**"Times New Roman"**!!!

***The font-family Property***

The font-family property should hold several font names as a "fallback" system. When specifying a web font in a CSS style, add more than one font name, in order to avoid unexpected behaviors. If the client computer for some reason doesn't have the one you choose, it will try the next one.   
  
It is a good practice to specify a generic font family, to let the browser pick a similar font in the generic family, if no other fonts are available.

body {  
font-family: Arial, "Helvetica Neue", Helvetica, sans-serif;  
}  
If the browser does not support the font **Arial**, it tries the next fonts (**Helvetica Neue**, then **Helvetica**). If the browser doesn't have any of them, it will try the generic **sans-serif**.   
Remember to use quotation marks if the font name consists of more than one word.

***The font-size Property***

The font-size property sets the size of a font. One way to set the size of fonts on the web is to use **keywords.**For example **xx-small**, **small**, **medium**, **large**, **larger**, etc.

Keywords are useful if you do not want the user to be able to increase the size of the font because it will adversely affect your site's appearance.

You can also use numerical values in **pixels** or **ems** to manipulate font size.   
Setting the font size in pixel values (**px**) is a good choice when you need pixel accuracy, and it gives you full control over the text size.   
The **em** size unit is another way to set the font size (**em**is a relative size unit). It allows all major browsers to resize the text. If you haven't set the font size anywhere on the page, then it is the browser default size, which is **16px**.   
  
To calculate the em size, just use the following formula: **em = pixels / 16**  
**For example:**

h1 {  
font-size: 20**px**;  
}

h1 {  
font-size: 1.25**em**;   
}

Both of the examples will produce the same result in the browser, because **20px/16=1.25em**.  
Try different combinations of text size and page zooming in a variety of browsers to ensure that the text remains readable.

***The font-style Property***

The font-style property is typically used to specify italic text.  
  
**The HTML:**

<p class="italic">This is a paragraph in italic style.</p>  
**The CSS:**

p.italic {  
font-style: italic;  
}

The font-style property has three values: **normal**, **italic**, and **oblique**.   
Oblique is very similar to italic, but less supported.   
  
**The HTML:**

<p class="normal">This paragraph is normal.</p>  
<p class="italic">This paragraph is italic.</p>  
<p class="oblique">This paragraph is oblique.</p>  
**The CSS:**

p.normal {font-style: **normal**;}  
p.italic {font-style: **italic**;}  
p.oblique {font-style: **oblique**;}

The HTML **<i>**tag will produce exactly the same result as the **italic font style**.

***The font-weight Property***

The font-weight controls the boldness or thickness of the text. The values can be set as **normal** (default size), **bold**, **bolder**, and **lighter**.  
  
**The HTML:**

<p class="light">This is a font with a "lighter" weight.</p>  
<p class="bold">This is a font with a "bold" weight.</p>  
<p class="bolder">This is a font with a "bolder" weight.</p>  
**The CSS:**

p.light { font-weight: **lighter**;}  
p.bold { font-weight: **bold**;}  
p.bolder {font-weight: **bolder**;}

You can also define the font weight with a number from **100** (thin) to **900** (thick), according to how thick you want the text to be.   
400 is the same as normal, and 700 is the same as bold.   
  
**The HTML:**

<p class="light">This is a font with a "lighter" weight.</p>  
<p class="thick">This is a font with a "bold" weight.</p>  
<p class="thicker">This is a font with a "700" weight.</p>  
**The CSS:**

p.light {font-weight: **lighter**;}  
p.thick {font-weight: **bold**;}  
p.thicker {font-weight: **700**;}

The HTML **<strong>** tag also makes the text **bold**.

***The font-variant Property***

The CSS font-variant property allows you to convert your font to all small caps. The values can be set as **normal**, **small-caps**, and **inherit**.   
  
**The HTML:**

<p class="normal">Paragraph font variant set to normal.</p>  
<p class="small">Paragraph font variant set to small-caps.</p>  
**The CSS:**

p.normal {font-variant: **normal**;}  
p.small {font-variant: **small-caps**;}

Not every font supports CSS font-variant, so be sure to test before you publish.

***The color Property***

The CSS **color** property specifies the color of the text.  
One method of specifying the color of the text is using a **color name**: like red, green, blue, etc.   
Here's an example of changing the color of your font.  
  
**The HTML:**

<p class="example">The text inside the paragraph is green.</p>  
The text outside the paragraph is black (by default).   
**The CSS:**

p.example {**color**: green;}

Another way of defining colors is using **hexadecimal values** and **RGB**.   
Hexadecimal form is a pound sign (**#**) followed by at most, **6 hex values** (0-F).  
RGB defines the individual values for **Red**, **Green**, and **Blue**.  
  
In the example below, we use hexadecimal value to set the heading color to blue, and RGB form to make the paragraph red.   
  
**The HTML:**<h1>This is a heading</h1>  
<p class="example">This is a paragraph</p>  
**The CSS:**

h1 {color: **#0000FF;**}  
p.example {color: **rgb(255,0,0);**}

***The text-align Property***

The text-align property specifies the horizontal alignment of text in an element. By default, text on your website is aligned to the left. However, at times you may require a different alignment.   
  
text-align property values are as follows: **left**, **right**, **center**, and **justify**.   
  
**The HTML:**

<p class="left">This paragraph is aligned to <strong>left.</strong></p>  
<p class="right">This paragraph is aligned to <strong>right.</strong></p>  
<p class="center">This paragraph is aligned to <strong>center.</strong></p>  
**The CSS:**

p.left {text-align: **left**;}  
p.right {text-align: **right**;}  
p.center {text-align: **center**;}

When text-align is set to "**justify**", each line is stretched so that every line has equal width, and the left and right margins are straight (as in magazines and newspapers).

***The vertical-align Property***

The vertical-align property sets an element's vertical alignment. Commonly used values are: **top**, **middle**, and **bottom**.  
  
The example below shows how to vertically align the text between the table.   
  
**The HTML:**

<table border="1" cellpadding="2" cellspacing="0" style="height: 150px;">  
<tr>  
<td class="top">Top</td>  
<td class="middle">Middle</td>  
<td class="bottom">Bottom</td>  
</tr>  
</table>  
**The CSS:**

td.top {**vertical-align: top**;}  
td.middle {**vertical-align: middle**;}  
td.bottom {**vertical-align: bottom**;}

The vertical-align property also takes the following values: **baseline**, **sub**, **super**, **%** and **px**(or pt, cm).   
The example below shows the difference between them.   
  
**The HTML:**

<p>This is an <span class="baseline">inline text</span> example.</p>  
<p>This is a <span class="sub">sub line text</span> example.</p>  
<p> This is a <span class="super">super line text</span> example.</p>  
<p> This is a <span class="pixel">pixel</span> example.</p>  
**The CSS:**

span.baseline {vertical-align: **baseline;**}  
span.sub {vertical-align: **sub;**}  
span.super {vertical-align: **super;**}  
span.pixel {vertical-align: **-10px;**}

Instead of **px** values, you can use **pt** (points), **cm**(centimeters) and **%** (percentage) values.

Vertical align property does not act the same way for all elements.  
For example, some additional CSS styling is needed for div elements.   
  
**The HTML:**

<div class="main">  
 <div class="paragraph">  
 This text is aligned to middle without using vertical-align property.  
 </div>  
</div>  
**The CSS:**

.main {

height: 150px; width: 400px;  
background-color: LightSkyBlue;  
display: inline-table;  
}  
.paragraph {  
display: table-cell;  
vertical-align: middle;  
}

**display: inline-table;**and**display: table-cell;** styling rules are applied to make the vertical-align property work with divs.

***The text-decoration Property***

The text-decoration property specifies how the text will be decorated.   
  
Commonly used values are:  
**none** - The default value, this defines a normal text  
**inherit** - Inherits this property from its parent element  
**overline** - Draws a horizontal line above the text  
**underline** - Draws a horizontal line below the text  
**line-through** - draws a horizontal line through the text (substitutes the HTML <s> tag)  
  
The example below demonstrates the difference between each value.   
  
**The HTML:**

<p class="none">This is default style of the text (none).</p>  
<p class="inherit">This text inherits the decoration of the parent.</p>  
<p class="overline">This is overlined text.</p>  
<p class="underline">This is underlined text.</p>  
<p class="line-through">This is lined-through text.</p>  
**The CSS:**

p.none {text-decoration: **none**;}  
p.inherit {text-decoration: **inherit**;}  
p.overline {text-decoration: **overline**;}  
p.underline {text-decoration: **underline**;}  
p.line-through {text-decoration: **line-through**;}

You can combine the **underline**, **overline**, or **line-through** values in a space-separated list to add multiple decoration lines.

Another value of text-decoration property is **blink** which makes the text blink.  
  
CSS syntax looks like this:

text-decoration: **blink;**

This value is valid but is deprecated and most browsers ignore it.

***The text-indent Property***

The text-indent property specifies how much horizontal space should be left before the beginning of the first line of the text. Property values are: **length** (px, pt, cm, em, etc.), **%** and **inherit**.   
  
**The HTML:**

<p>This is an example of <strong>text-indent </strong> property.   
First line of our text is indented to the right in 60px.   
Besides pixels you can also use other measurement units, like pt, cm, em, etc. </p>  
**The CSS:**

p {  
**text-indent**: 60px;  
}

Negative values are allowed. The first line will be indented to the left if the value is negative.

***The text-shadow Property***

The text-shadow property adds shadow to text. It takes four values: the first value defines the distance of the shadow in the **x (horizontal) direction**, the second value sets the distance in the **y (vertical) direction**, the third value defines the **blur** of the shadow, and the fourth value sets the **color**.   
  
**The HTML:**

<h1>Text-shadow example</h1>  
**The CSS:**

h1 {color: blue; font-size: 30pt; **text-shadow**: 5px 2px 4px grey;}

In the example above, we created a shadow using the following parameters:  
**5px** – the X-coordinate  
**2px** – the Y-coordinate   
**4px** – the blur radius  
**grey** – the color of the shadow

To add more than one shadow to the text, add a comma-separated list of shadows.

***text-shadow with Blur Effect***

When working with shadows, you can use any CSS-supported color format.  
For the x and y offsets, various types of units can be used (like **px, cm, mm, in, pc, pt,** etc).   
Negative values are also supported.

The example below creates a blue drop-shadow, two pixels higher than the main text, one pixel to the left of it, and with a 0.5em blur:   
  
**The HTML:**

<h1>Text-shadow with blur effect</h1>  
**The CSS:**

h1 {font-size: 20pt; text-shadow: rgba(0,0,255,1) -1px -2px 0.5em;}

Internet Explorer 9 and earlier do not support the text-shadow property.

***The text-transform Property***

The text-transform CSS property specifies how to capitalize an element's text. For example, it can be used to make text appear with each word capitalized.  
  
**The HTML:**

<p class="capitalize">The value capitalize transforms the first character in each word to uppercase; all other characters remain unaffected.</p>  
**The CSS:**

p.capitalize {text-transform: capitalize;}

***text-transform Values***

Using text-transform property you can make text appear in all-uppercase or all-lowercase. Here is an example:  
  
**The HTML:**

<p class="uppercase">This value transforms all characters to uppercase.</p>  
<p class="lowercase">This value transforms all characters to lowercase.</p>  
**The CSS:**

p.uppercase {text-transform: **uppercase;**}  
p.lowercase {text-transform: **lowercase;**}

The value **none** will produce no capitalization effect at all.

***The letter-spacing Property***

The letter-spacing property specifies the **space between characters** in a text. The values can be set as:  
- **normal** defines the default style with no extra space between characters  
- **length** defines an extra space between characters using measurement units like px, pt, cm, mm, etc.;  
- **inherit** inherits the property from its parent element;  
  
**The HTML:**

<p class="normal">This paragraph has no additional letter-spacing applied.</p>  
<p class="positive ">This paragraph is letter-spaced at 4px.</p>  
**The CSS:**

p.normal { **letter-spacing: none;** }  
p.positive { **letter-spacing: 4px;** }

For defining an extra space between characters, negative values are also permitted.   
Here is an example demonstrating the difference between **positive** and **negative** values:   
  
**The HTML:**

<p class="positive">This paragraph is letter-spaced at 4px.</p>  
<p class="negative">This paragraph is letter-spaced at -1.5px</p>  
**The CSS:**

p.positive { letter-spacing: 4px; }  
p.negative { letter-spacing: -1.5px; }

Always test your result, to ensure the text is readable.

***The word-spacing Property***

The word-spacing property specifies the **space between words** in a text. Just like the letter-spacing property, you can set the word-spacing values as **normal**, **length**, and **inherit**.   
  
**The HTML:**

<p class="normal">This paragraph has no additional word-spacing applied.</p>  
<p class="px">This paragraph is word-spaced at 30px.</p>  
**The CSS:**

p.normal { **word-spacing: normal**;}  
p.px { **word-spacing: 30px**;}

When a weird spacing is used, and it is necessary to keep the selected paragraph with normal word spacing, the **normal** option is usually used.

To define an extra space between words, you can use positive measurement values like**px, pt, pc, cm, mm, inches, em,**and**ex**.  
Negative values are also permitted. Here is an example to show the difference.   
  
**The HTML:**

<p class="positive">This paragraph is word-spaced at 20px.</p>  
<p class="negative">This paragraph is word-spaced at -5px.</p>  
**The CSS:**

p.positive { **word-spacing: 20px**;}  
p.negative { **word-spacing: -5px**;}

***The white-space Property***

The white-space property specifies how white-space inside an element is handled. The values can be set as **normal**, **inherit**, **nowrap**, etc.   
The nowrap value suppresses all line breaks in the element.   
  
**The HTML:**

<p>This paragraph has multiple spaces and a line break, but

it will be ignored, as we used the nowrap value. </p>  
**The CSS:**

p {white-space: nowrap;}

The text will continue on the same line until a **<br />** tag is encountered.

***The white-space Values***

The white-space property also supports other values:  
**pre** - text will only wrap on line breaks and white space  
**pre-line** - text will wrap where there is a break in code, but extra white space is still ignored  
**pre-wrap** - text will wrap when necessary, and on line breaks  
  
Here is an example in which all three values are used:  
  
**The HTML:**

<p class="pre"> In the markup we have multiple spaces and a line

break. </p>  
<p class="preline"> In the markup we have multiple spaces   
and a line break, but in the result multiple spaces are ignored. </p>  
<p class="prewrap"> In the markup we have multiple   
spaces and a line break.</p>  
**The CSS:**

p.pre {**white-space: pre;**}  
p.preline {**white-space: pre-line;**}  
p.prewrap {**white-space: pre-wrap;**}

**Pre-wrap** value behaves as the **pre** value, except that it adds extra line breaks to prevent the text breaking out of the element's box.