

**CSS**



CSS allows you to create rules that specify how the content of an element should appear. For example, you can specify that the background of the page is cream, all paragraphs should appear in gray using the Arial typeface, or that all level one headings should be in a blue, italic.

# How to insert styles in project

```
<styles></styles>
```

```
<link href="css/example.css" rel="stylesheet" />
```

```
style=""
```

CSS works by associating rules with HTML elements. These rules govern how the content of specified elements should be displayed. A CSS rule contains two parts: a selector and a declaration.

SELECTOR



p {

font-family: Arial;}



DECLARATION

CSS declarations sit inside curly brackets and each is made up of two parts: a **property** and a **value**, separated by a colon. You can specify several properties in one declaration, each separated by a semi-colon.

```
h1, h2, h3 {  
    font-family: Arial;  
    color: yellow;}  
└──┬──┘ └──┬──┘  
    PROPERTY  VALUE
```

# CSS Selectors

Universal Selector	Applies to all elements in the document	<b>* {}</b> Targets all elements on the page
Type Selector	Matches element names	<b>h1, h2, h3 {}</b> Targets the <h1>, <h2> and <h3> elements
Class Selector	Matches an element whose class attribute has a value that matches the one specified after the period (or full stop) symbol	<b>.note {}</b> Targets any element whose class attribute has a value of note <b>p.note {}</b> Targets only <p> elements whose class attribute has a value of note

ID Selector	Matches an element whose id attribute has a value that matches the one specified after the pound or hash symbol	<b>#introduction {}</b> Targets the element whose id attribute has a value of introduction
Child Selector	Matches an element that is a direct child of another	<b>li&gt;a {}</b> Targets any <a> elements that are children of an <li> element (but not other <a> elements in the page)
Descendant Selector	Matches an element that is a descendent of another specified element (not just a direct child of that element)	<b>p a {}</b> Targets any <a> elements that sit inside a <p> element, even if there are other elements nested between them
Adjacent Sibling Selector	Matches an element that is the next sibling of another	<b>h1+p {}</b> Targets the first <p> element after any <h1> element (but not other <p> elements)

# How CSS Rules Cascade

```
* { font-family: Arial, Verdana, sans-serif;}
```

```
h1 { font-family: "Courier New", monospace;}
```

```
i { color: green;}
```

```
i { color: red;}
```

```
b { color: pink;}
```

```
p b { color: blue !important;}
```

```
p b { color: violet;}
```

```
p#intro { font-size: 100%;}
```

```
p { font-size: 75%;}
```



**!important**

# Colors

rgb values

hex codes

color names

# Colors: RGB, RGBA

## rgb values

These express colors in terms of how much red, green and blue are used to make it up. For example:

```
color: rgb(100,100,90);
```

```
color: rgb(100,100,90, .9);
```

# Colors: HEX

These are six-digit codes that represent the amount of red, green and blue in a color, preceded by a pound or hash #sign. For example:

color: #CC0000;

# Color names

There are 147 predefined color names that are recognized by browsers. For example: DarkCyan

color: darkcyan;

# Background color

```
body { background-color: rgb(200,200,200);}
```

```
h1 { background-color: DarkCyan;}
```

```
h2 { background-color: #ee3e80;}
```

```
p { background-color: white;}
```

## RESULT

### Marine Biology

#### The Composition of Seawater

Almost anything can be found in seawater. This includes dissolved materials from Earth's crust as well as materials released from organisms. The most important components of seawater that influence life forms are salinity, temperature, dissolved gases (mostly oxygen and carbon dioxide), nutrients, and pH. These elements vary in their composition as well as in their influence on marine life.

## RGB VALUES

Values for red, green, and blue are expressed as numbers between 0 and 255.



`rgb(102,205,170)`

This color is made up of the following values:

102 red

205 green

170 blue

## HEX CODES

Hex values represent values for red, green, and blue in hexadecimal code.



`#66cdaa`

The value of the red, 102, is expressed as 66 in hexadecimal code. The 205 of the green is expressed as cd and the 170 of the blue equates to aa.

## COLOR NAMES

Colors are represented by predefined names. However, they are very limited in number.



`MediumAquaMarine`

There are 147 color names supported by browsers (this color is MediumAquaMarine). Most consider this to be a limited color palette, and it is hard to remember the name for each of the colors so (apart from white and black) they are not commonly used.

# CSS 3: Opacity

```
p.one {
```

```
    background-color: rgb(0,0,0);
```

```
    opacity: 0.5;}
```

```
p.two {
```

```
    background-color: rgb(0,0,0);
```

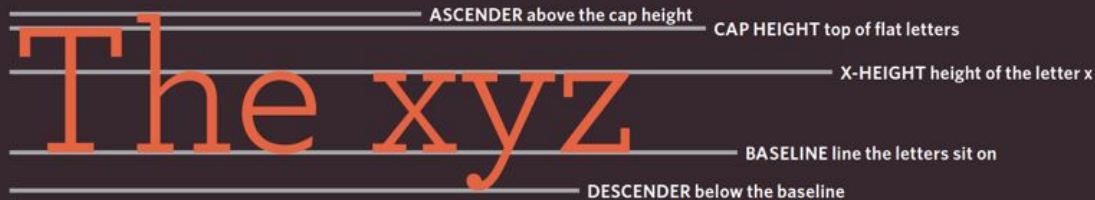
```
    background-color: rgba(0,0,0,0.5);
```

RESULT





TEXT



## WEIGHT

Light

Medium

**Bold**

**Black**

The font weight not only adds emphasis but can also affect the amount of white space and contrast on a page.

## STYLE

Normal

*Italic*

*Oblique*

Italic fonts have a cursive aspect to some of the lettering. Oblique font styles take the normal style and put it on an angle.

## STRETCH

Condensed

Regular

**Extended**

In condensed (or narrow) versions of the font, letters are thinner and closer together. In expanded versions they are thicker and further apart.

## SERIF

Serif fonts have extra details on the ends of the main strokes of the letters. These details are known as serifs.



In print, serif fonts were traditionally used for long passages of text because they were considered easier to read.

## SANS-SERIF

Sans-serif fonts have straight ends to letters, and therefore have a much cleaner design.



Screens have a lower resolution than print. So, if the text is small, sans-serif fonts can be clearer to read.

## MONOSPACE

Every letter in a monospace (or fixed-width) font is the same width. (Non-monospace fonts have different widths.)



Monospace fonts are commonly used for code because they align nicely, making the text easier to follow.

When choosing a typeface, it is important to understand that a browser will usually only display it if it's installed on that user's computer.

## SERIF

Serif fonts have extra details on the end of the main strokes of the letters.

### EXAMPLES:

Georgia

Times

Times New Roman

## SANS-SERIF

Sans-serif fonts have straight ends to letters and therefore have a much cleaner design.

### EXAMPLES:

Arial

Verdana

Helvetica

## PIXELS

## PERCENTAGES

## EMS

### TWELVE PIXEL SCALE

h1	24px
h2	18px
h3	14px
body	12px

=

h1	200%
h2	150%
h3	117%
body	75%

=

h1	1.5em
h2	1.3em
h3	1.17em
body	100%
p	0.75em

### SIXTEEN PIXEL SCALE

h1	32px
h2	24px
h3	18px
body	16px

=

h1	200%
h2	150%
h3	133%
body	100%

=

h1	2em
h2	1.5em
h3	1.125em
body	100%
p	1em

## BROWSER

## FORMAT

	eot	woff	ttf / otf	svg
Chrome (all)				●
Chrome 6+		●	●	●
Firefox 3.5			●	
Firefox 3.6+		●	●	
IE 5 - 8	●			
IE 9+	●	●	●	
Opera 10+			●	●
Safari 3.1+			●	●
iOS <4.2				●
iOS 4.2+			●	●

# Font

font-family: Arial, ...

font-size: 0 - ..

font-weight: normal | bold | 100-900

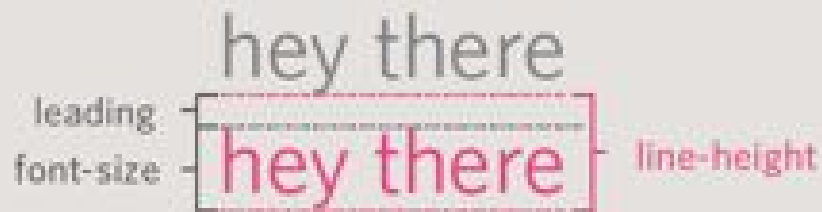
font-style: normal | italic | oblique

text-transform: uppercase | lowercase | capitalize

text-decoration: none | underline | overline | line-through

# line-height

```
p {  
    line-height: 1.4em;  
}
```





# letter-spacing, word-spacing

```
h1, h2 {
```

```
    text-transform: uppercase;
```

```
    letter-spacing: 0.2em;}
```

```
.credits {
```

```
    font-weight: bold;
```

```
    word-spacing: 1em;}
```

# text-align

left

right

center

justify

# vertical-align

baseline

sub

super

top

text-top

middle

bottom

text-bottom

# text-indent

text-indent: -9999px;

text-indent: 20px;

# text-shadow

`text-shadow: 1px 1px 0px #000000;`

1. value = The X-coordinate
2. value = The Y-coordinate
3. value = The blur radius
4. value = The color of the shadow

# :first-letter, :first-line

```
p.intro:first-letter {
```

```
    font-size: 200%;}
```

```
p.intro:first-line {
```

```
    font-weight: bold;}
```

# Styles for links

:link

:visited

:hover

:active

:focus

# Attribute Selectors

SELECTOR	MEANING	EXAMPLE
EXISTENCE	[ <b>]</b> Matches a specific attribute (whatever its value)	p[class] Targets any <p> element with an attribute called class
EQUALITY	[ <b>=</b> ] Matches a specific attribute with a specific value	p[class="dog"] Targets any <p> element with an attribute called class whose value is dog
SPACE	[ <b>~</b> ] Matches a specific attribute whose value appears in a space-separated list of words	p[class~="dog"] Targets any <p> element with an attribute called class whose value is a list of space-separated words, one of which is dog
PREFIX	[ <b>^</b> ] Matches a specific attribute whose value begins with a specific string	p[attr^="d"] Targets any <p> element with an attribute whose value begins with the letter "d"
SUBSTRING	[ <b>*</b> ] Matches a specific attribute whose value contains a specific substring	p[attr*="d"] Targets any <p> element with an attribute whose value contains the letters "do"
SUFFIX	[ <b>\$</b> ] Matches a specific attribute whose value ends with a specific string	p[attr\$="d"] Targets any <p> element with an attribute whose value ends with the letter "g"



BOXES

# Width & height

width, height

min-width, max-width

min-height, max-height

Every box has three available properties that can be adjusted to control its appearance:

1

### BORDER

Every box has a border (even if it is not visible or is specified to be 0 pixels wide). The border separates the edge of one box from another.

2

### MARGIN

Margins sit outside the edge of the border. You can set the width of a margin to create a gap between the borders of two adjacent boxes.

3

### PADDING

Padding is the space between the border of a box and any content contained within it. Adding padding can increase the readability of its contents.

If you specify a width for a box, then the borders, margin, and padding are added to its width and height.



# border-style

border-style:

none|hidden|dotted|dashed|solid|double|groove|ridge|inset|outset|initial|inherit;

# **border-color**

border-top-color

border-right-color

border-bottom-color

border-left-color

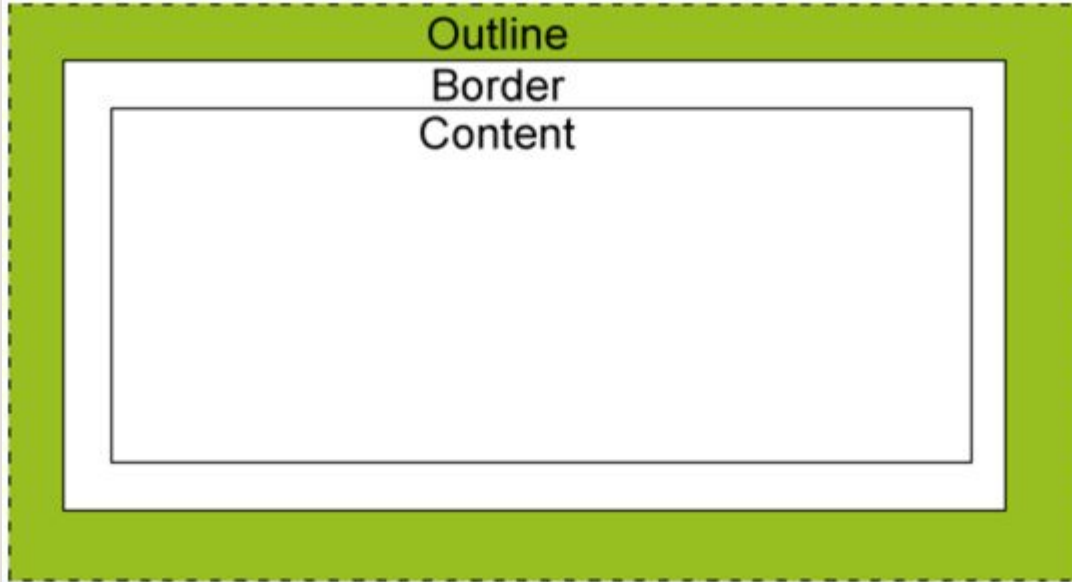
# border short form

border: 3px dotted #0088dd;

Value	Description	Play it
none	Default value. Specifies no border	<a href="#">Play it »</a>
hidden	The same as "none", except in border conflict resolution for table elements	<a href="#">Play it »</a>
dotted	Specifies a dotted border	<a href="#">Play it »</a>
dashed	Specifies a dashed border	<a href="#">Play it »</a>
solid	Specifies a solid border	<a href="#">Play it »</a>
double	Specifies a double border	<a href="#">Play it »</a>
groove	Specifies a 3D grooved border. The effect depends on the border-color value	<a href="#">Play it »</a>
ridge	Specifies a 3D ridged border. The effect depends on the border-color value	<a href="#">Play it »</a>
inset	Specifies a 3D inset border. The effect depends on the border-color value	<a href="#">Play it »</a>
outset	Specifies a 3D outset border. The effect depends on the border-color value	<a href="#">Play it »</a>
initial	Sets this property to its default value. <a href="#">Read about <i>initial</i></a>	<a href="#">Play it »</a>
inherit	Inherits this property from its parent element. <a href="#">Read about <i>inherit</i></a>	

# Outline

An outline is a line that is drawn around elements (outside the borders) to make the element "stand out". The outline properties specify the style, color, and width of an outline.





# padding

padding-top

padding-right

padding-bottom

padding-left

# margin

margin-top

margin-right

margin-bottom

margin-left

# Размеры

<b>pt</b>	<p>это довольно таки старинная единица измерения, которая уже не один десяток лет используется для печати, в наборных машинках и во всякого рода программах для набора текста.</p> <p>1px = 0.75pt</p>
<b>%</b>	<p>Проценты (%) — это уникальная единица измерения. Эта относительная единица работает так же, как и слышится. Т.е, если, например, у родительского элемента установлен размер шрифта 24px, то выставив у дочернего элемента размер шрифта в 50%, последний будет меньше первого ровно в два раза, и будет составлять 12px.</p>
<b>em</b>	<p>Один <b>em</b> равен значению свойства font-size заданного шрифта.</p> <p><a href="http://topfunky.com/baseline-rhythm-calculator/">http://topfunky.com/baseline-rhythm-calculator/</a></p>
<b>rem</b>	<p>Новая единица измерения, введённая спецификацией. Она означает примерно "Корневой em" (root em). Если <b>em</b> — это единица, которая пляшет относительно шрифта родительского элемента, то <b>rem</b> — это единица измерения, которая пляшет относительно корневого элемента, т.е, как вы уже догадались — <b>html</b>. Это означает, что мы можем определить единый размер шрифта в &lt;html&gt;, и отталкиваться уже от него, причём при любой вложенности.</p> <pre>html { font-size: 62.5%; }</pre> <pre>body { font-size: 1.4rem; } /* =14px */</pre> <pre>h1 { font-size: 2.4rem; } /* =24px */</pre>

# Размеры

## **vh vw**

С помощью **vw** и **vh** мы можем вычислить размер элемента относительно области просмотра. Один **vw** равен 1/100 ширины всего экрана, а один **vh** соответственно 1/100 высоты. Для того, чтобы элемент занимал, например, всю ширину окна браузера, его ширине следует выставить 100vw.

<http://css-live.ru/articles/novye-i-starye-edinicy-izmereniya-kratkij-obzor.html>

# Overflow

The overflow property specifies what happens if content overflows an element's box.

`overflow: visible|hidden|scroll|auto|initial|inherit;`

Value	Description	Play it
visible	The overflow is not clipped. It renders outside the element's box. This is default	<a href="#">Play it »</a>
hidden	The overflow is clipped, and the rest of the content will be invisible	<a href="#">Play it »</a>
scroll	The overflow is clipped, but a scroll-bar is added to see the rest of the content	<a href="#">Play it »</a>
auto	If overflow is clipped, a scroll-bar should be added to see the rest of the content	<a href="#">Play it »</a>
initial	Sets this property to its default value. <a href="#">Read about <i>initial</i></a>	<a href="#">Play it »</a>
inherit	Inherits this property from its parent element. <a href="#">Read about <i>inherit</i></a>	

# display

inline-block | inline | block | none

inline-block problem:

<https://css-tricks.com/fighting-the-space-between-inline-block-elements/>

Value	Description	Play it
inline	Default value. Displays an element as an inline element (like <span>)	<a href="#">Play it »</a>
block	Displays an element as a block element (like <p>)	<a href="#">Play it »</a>
flex	Displays an element as an block-level flex container. New in CSS3	
inline-block	Displays an element as an inline-level block container. The inside of this block is formatted as block-level box, and the element itself is formatted as an inline-level box	
inline-flex	Displays an element as an inline-level flex container. New in CSS3	
inline-table	The element is displayed as an inline-level table	
list-item	Let the element behave like a <li> element	<a href="#">Play it »</a>
run-in	Displays an element as either block or inline, depending on context	
table	Let the element behave like a <table> element	
table-caption	Let the element behave like a <caption> element	
table-column-group	Let the element behave like a <colgroup> element	
table-header-group	Let the element behave like a <thead> element	
table-footer-group	Let the element behave like a <tfoot> element	

table-row-group	Let the element behave like a <tbody> element	
table-cell	Let the element behave like a <td> element	
table-column	Let the element behave like a <col> element	
table-row	Let the element behave like a <tr> element	
none	The element will not be displayed at all (has no effect on layout)	<a href="#">Play it »</a>
initial	Sets this property to its default value. <a href="#">Read about <i>initial</i></a>	<a href="#">Play it »</a>
inherit	Inherits this property from its parent element. <a href="#">Read about <i>inherit</i></a>	

**Note:** The values "inline-table", "table", "table-caption", "table-cell", "table-column", "table-column-group", "table-row", and "table-row-group" are not supported in IE7 and earlier. IE8 requires a !DOCTYPE. IE9 supports the values.

**Note:** The values "flex" and "inline-flex" requires a prefix to work in Safari. For "flex" use "display: -webkit-flex", for "inline-flex" use "display: -webkit-inline-flex;".



# visibility

visibility: visible|hidden|collapse|initial|inherit;

Value	Description	Play it
visible	Default value. The element is visible	<a href="#">Play it »</a>
hidden	The element is invisible (but still takes up space)	<a href="#">Play it »</a>
collapse	<p>Only for table elements. collapse removes a row or column, but it does not affect the table layout. The space taken up by the row or column will be available for other content.</p> <p>If collapse is used on other elements, it renders as "hidden"</p>	<a href="#">Play it »</a>
initial	Sets this property to its default value. <a href="#">Read about initial</a>	<a href="#">Play it »</a>
inherit	Inherits this property from its parent element. <a href="#">Read about inherit</a>	

# box-shadow

Horizontal offset

Vertical offset

Blur distance

Spread of shadow

`box-shadow: 0 0 10px #777777;`



# border-radius

border-top-right-radius

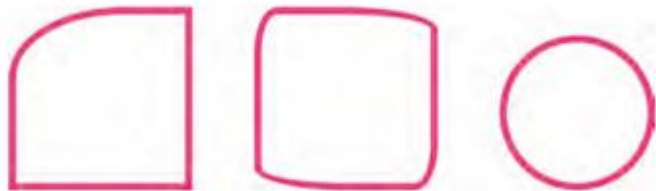
border-bottom-right-radius

border-bottom-left-radius

border-top-left-radius

border-radius: 10px;

RESULT



# Centering boxes

<https://css-tricks.com/centering-css-complete-guide/>

# LIST & TABLES

# list-style-type

ol {

list-style-type: lower-roman;

}

## UNORDERED LISTS

For an unordered list you can use the following values:

- none
- disc
- circle
- square

## ORDERED LISTS

For an ordered (numbered) list you can use the following values:

decimal

1 2 3

decimal-leading-zero

01 02 03

lower-alpha

a b c

upper-alpha

A B C

lower-roman

i. ii. iii.

upper-roman

I II III

# list-style-image

```
ul {  
    list-style-image: url("images/star.png");  
}
```

## RESULT

### Index of Translated Poems

#### Arthur Rimbaud

- ✧ Ophelia
- ✧ To Music
- ✧ A Dream for Winter
- ✧ Vowels
- ✧ The Drunken Boat

# list-style-position

## outside

The marker sits to the left of the block of text. (This is the default behaviour if this property is not used.)

## inside

The marker sits inside the box of text (which is indented). In the example shown, the width of the list has been limited to 150 pixels. This ensures that the text wraps onto a new line so you can see how the value of inside sits the bullet inside the first line of text.

### RESULT

- That idol, black eyes and yellow mop, without parents or court ...
- Gracious son of Pan! Around your forehead crowned with flowerets ...
- When the world is reduced to a single dark wood for our four ...

### RESULT

- Once, if my memory serves me well, my life was a banquet ...
- Hadn't I once a youth that was lovely, heroic, fabulous ...
- Autumn already! - But why regret the everlasting sun if we are



# list-style

```
ul {  
    list-style: inside circle;  
}
```

# Tables

**width** to set the width of the table

**padding** to set the space between the border of each table cell and its content

**text-transform** to convert the content of the table headers to uppercase

**letter-spacing, font-size** to add additional styling to the content of the table headers

**border-top, border-bottom** to set borders above and below the table headers

**text-align** to align the writing to the left of some table cells and to the right of the others

**background-color** to change the background color of the alternating table rows

**:hover** to highlight a table row when a user's mouse goes over it

# border-spacing, border-collapse

## **collapse**

Borders are collapsed into a single border where possible.

(border-spacing will be ignored and cells pushed together, and empty-cells properties will be ignored.)

## **separate**

Borders are detached from each other. (border-spacing and empty-cells will be obeyed.)

<http://htmlbook.ru/css/border-spacing>

<http://htmlbook.ru/css/border-collapse>

# Simple table styling

```
table {  
    border-collapse: collapse;  
    border-spacing: 0;  
    width: 100%;  
}  
  
th { text-align: left; }  
  
td {vertical-align: top; }
```

# :nth-child()

:nth-child(even)

:nth-child(odd)

Значение	Номера элементов	Описание
1	1	Первый элемент, является синонимом псевдокласса <b>:first-child</b> .
5	5	Пятый элемент.
2n	2, 4, 6, 8, 10	Все четные элементы, аналог значения <b>even</b> .
2n+1	1, 3, 5, 7, 9	Все нечетные элементы, аналог значения <b>odd</b> .
3n+2	2, 5, 8, 11, 14	—
-n+3	3, 2, 1	—
5n-2	3, 8, 13, 18, 23	—
even	2, 4, 6, 8, 10	Все четные элементы.
odd	1, 3, 5, 7, 9	Все нечетные элементы.

nth-cl

# Examples

<http://css.yoksel.ru/nth-child/>

LAYOUT

# Position

static

relative

absolute

fixed

z-index (<http://bitsofco.de/2015/how-z-index-works>)



# Float

left

right

clear: left | right | both

# box-sizing

The box-sizing property is used to tell the browser what the sizing properties (width and height) should include.

```
box-sizing: content-box|border-box|initial|inherit;
```

Value	Description
content-box	Default. The width and height properties (and min/max properties) includes only the content. Border, padding, or margin are not included
border-box	The width and height properties (and min/max properties) includes content, padding and border, but not the margin
initial	Sets this property to its default value. <a href="#">Read about <i>initial</i></a>
inherit	Inherits this property from its parent element. <a href="#">Read about <i>inherit</i></a>

# How to make a simple grid

<http://j4n.co/blog/Creating-your-own-css-grid-system>

<https://css-tricks.com/dont-overthink-it-grids/>

# Formula

$$\text{target} \div \text{context} = \text{result}$$

IMAGES

# Background

background-image

background-color

background-repeat

background-attachment

background-position

background-size

# Position

background-position: center top;

background-position: 10px 10px;

background-position: 2% 5%;

<http://prgssr.ru/development/5-osobennostej-pozicionirovaniya-v-css.html>



left top



left center



left bottom



center top



center center



center bottom



right top



right center



right bottom

# Repeat & Attachment

## repeat

The background image is repeated both horizontally and vertically (the default way it is shown if the background-repeat property isn't used).

## repeat-x

The image is repeated horizontally only (as shown in the first example on the left).

## repeat-y

The image is repeated vertically only. no-repeat The image is only shown once.

The **background-attachment** property specifies whether a background image should stay in one position or move as the user scrolls up and down the page. It can have one of two values:

## fixed

The background image stays in the same position on the page.

## scroll

The background image moves up and down as the user scrolls up and down the page.



# Background short form

```
background: #ffffff url("images/img.gif") no-repeat top right;}
```

# Background Size:

background-size: auto | *length* | cover | contain | initial | inherit; z

Value	Description	Play it
auto	Default value. The background-image contains its width and height	<a href="#">Play it »</a>
<i>length</i>	Sets the width and height of the background image. The first value sets the width, the second value sets the height. If only one value is given, the second is set to "auto"	<a href="#">Play it »</a>
<i>percentage</i>	Sets the width and height of the background image in percent of the parent element. The first value sets the width, the second value sets the height. If only one value is given, the second is set to "auto"	<a href="#">Play it »</a>
cover	Scale the background image to be as large as possible so that the background area is completely covered by the background image. Some parts of the background image may not be in view within the background positioning area	<a href="#">Play it »</a>
contain	Scale the image to the largest size such that both its width and its height can fit inside the content area	<a href="#">Play it »</a>
initial	Sets this property to its default value. <a href="#">Read about <i>initial</i></a>	<a href="#">Play it »</a>
inherit	Inherits this property from its parent element. <a href="#">Read about <i>inherit</i></a>	

# Multiple background

<http://caniuse.com/#feat=multibackgrounds> > IE9

```
background: url("media/fishing.svg") top right 10px no-repeat,  
            url("media/mermaid.svg") bottom left repeat-x,  
            url("media/fish.svg") 30px 90px no-repeat,  
            url("media/sea.png") repeat-x;
```

# Gradients

<http://www.colorzilla.com/gradient-editor/>

[http://www.w3schools.com/css/css3\\_gradients.asp](http://www.w3schools.com/css/css3_gradients.asp)

<https://css-tricks.com/css3-gradients/>

`background: linear-gradient(angle, color-stop1, color-stop2);`

## Example:

```
background: linear-gradient(white, green);
```

# Sprites

FORMS

# Form elements

labels,

fieldset

input,

textarea,

select

file upload,

checkbox, radio

legend,

buttons

errors, invalid states

# Use correct types



type="email"



type="url"



# Placeholder styles

```
::-webkit-input-placeholder {color:#c0392b;} /* Chrome, Safari */  
::-moz-placeholder          {color:#c0392b;} /* Firefox 19+ */  
:-moz-placeholder           {color:#c0392b;} /* Firefox 18- */  
:-ms-input-placeholder      {color:#c0392b;} /* IE */
```

# Usefull CSS selectors

`:active`

`:focus`

`:hover`

`:enabled`

`:disabled`

`:checked`

`:indeterminate`

`:default`

`:valid`

`:invalid`

`:in-range`

`:out-of-range`

`:required`

`:optional`

`:read-only`

`:read-write`

# Example

## RENDERED HTML

Input Label

Input Label

Input Label

Input Label

Select Box

Husker

Choose Your Favorite

☐ Red ☐ Blue

Check These Out

☐ Checkbox 1 ☐ Checkbox 2

Textarea Label

## RENDERED HTML

Hex Value

Go

Label

Label

Go

Go

## RENDERED HTML

Error

Invalid entry

Another Error

Invalid entry

Message...

Invalid entry

# Input styles

- make sure your input has 100% width
- provide special class for input styles (example: `class="form-item"`)

# Custom Checkbox

<http://www.hongkiat.com/blog/css3-checkbox-radio/>

# Custom Upload

<https://css-tricks.com/snippets/css/custom-file-input-styling-webkitblink/>

# Custom select

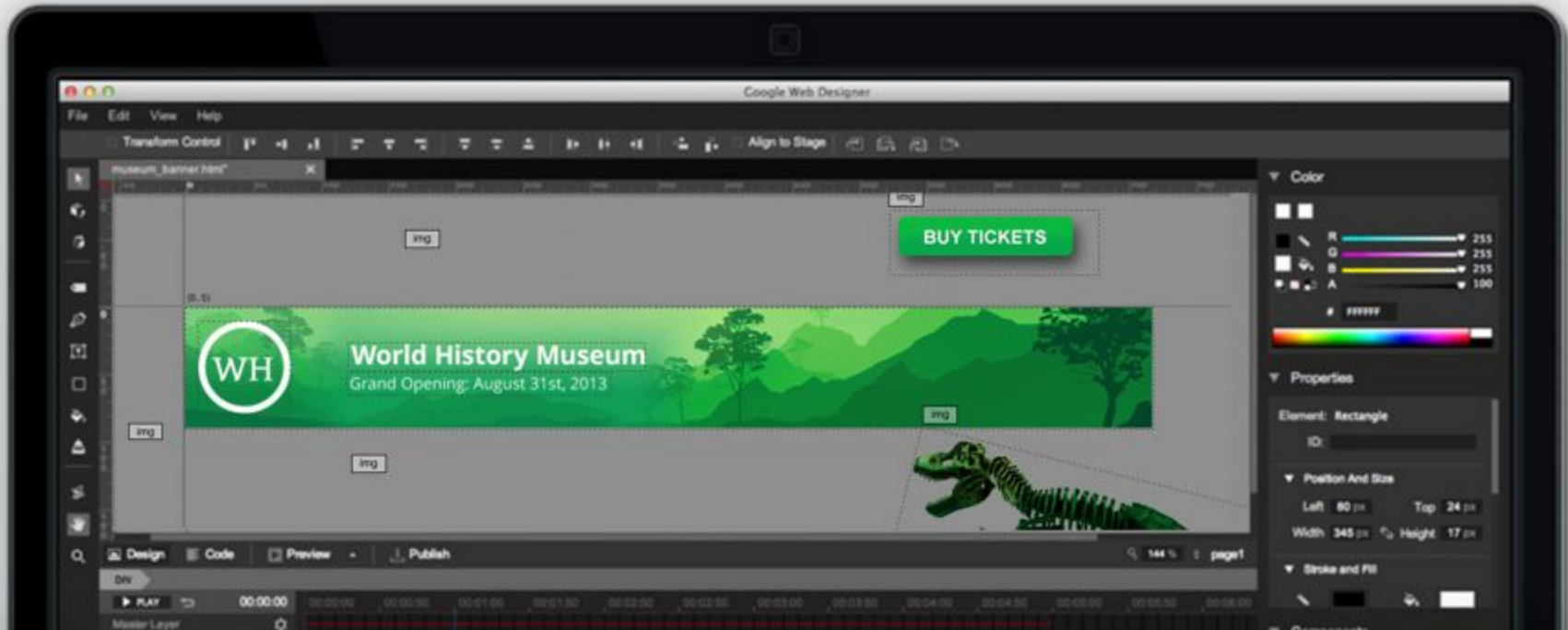
<http://codepen.io/bepfh/pen/ogNBYW>

ANIMATION



# Google

<https://www.google.com/webdesigner/>



FLEXBOX

# What is flexbox ?

The Flexbox Layout (Flexible Box) module (currently a W3C Last Call Working Draft) aims at providing a more efficient way to lay out, align and distribute space among items in a container, even when their size is unknown and/or dynamic (thus the word "flex").

The main idea behind the flex layout is to give the container the ability to alter its items' width/height (and order) to best fill the available space (mostly to accommodate to all kind of display devices and screen sizes). A flex container expands items to fill available free space, or shrinks them to prevent overflow.

From : <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

# Main attributes

`display: flex; /* or inline-flex */`

`order: <integer>;`

`flex-direction: row | row-reverse | column | column-reverse;`

`flex-grow: <number>; /* default 0 */`

`flex-wrap: nowrap | wrap | wrap-reverse;`

`flex-shrink: <number>; /* default 1 */`

`flex-flow: <'flex-direction'> || <'flex-wrap'>`

`flex-basis: <length> | auto; /* default auto */`

`flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]`

# Main flex attributes

`justify-content: flex-start | flex-end | center | space-between | space-around;`

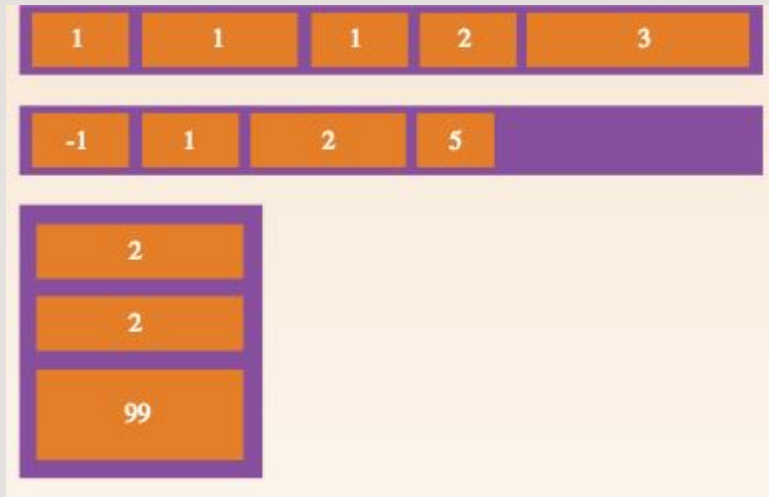
`align-self: auto | flex-start | flex-end | center | baseline | stretch;`

`align-items: flex-start | flex-end | center | baseline | stretch;`

`align-content: flex-start | flex-end | center | space-between | space-around | stretch;`

# Order

By default, flex items are laid out in the source order. However, the `order` property controls the order in which they appear in the flex container.



# Flex-direction

This establishes the main-axis, thus defining the direction flex items are placed in the flex container. Flexbox is (aside from optional wrapping) a single-direction layout concept. Think of flex items as primarily laying out either in horizontal rows or vertical columns.



**row** (default): left to right in ltr; right to left in rtl

**row-reverse**: right to left in ltr; left to right in rtl

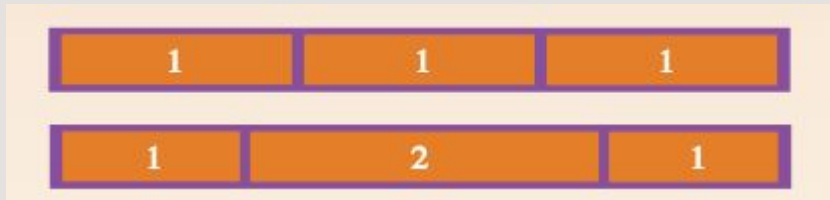
**column**: same as row but top to bottom

**column-reverse**: same as row-reverse but bottom to top

# Flex-grow

This defines the ability for a flex item to grow if necessary. It accepts a unitless value that serves as a proportion. It dictates what amount of the available space inside the flex container the item should take up.

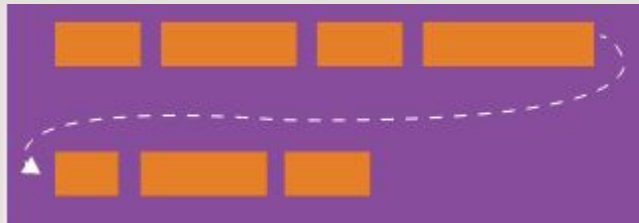
If all items have flex-grow set to 1, the remaining space in the container will be distributed equally to all children. If one of the children a value of 2, the remaining space would take up twice as much space as the others (or it will try to, at least).





# Flex-wrap

By default, flex items will all try to fit onto one line. You can change that and allow the items to wrap as needed with this property. Direction also plays a role here, determining the direction new lines are stacked in.



**nowrap** (default): single-line / left to right in ltr; right to left in rtl

**wrap**: multi-line / left to right in ltr; right to left in rtl

**wrap-reverse**: multi-line / right to left in ltr; left to right in rtl

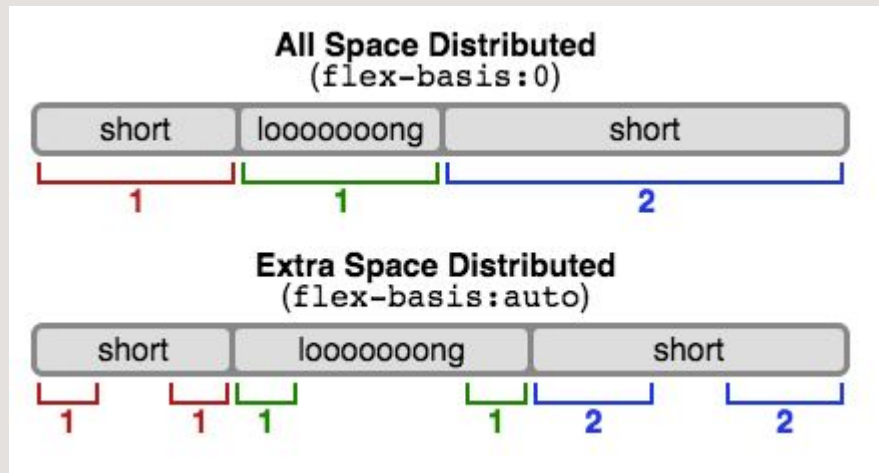
# flex-shrink

This defines the ability for a flex item to shrink if necessary.

# flex-basis

This defines the default size of an element before the remaining space is distributed. It can be a length (e.g. 20%, 5rem, etc.) or a keyword. The auto keyword means "look at my width or height property" (which was temporarily done by the main-size keyword until deprecated). The content keyword means "size it based on the item's content" - this keyword isn't well supported yet, so it's hard to test and harder to know what its brethren max-content, min-content, and fit-content do.

If set to 0, the extra space around content isn't factored in. If set to auto, the extra space is distributed based on its flex-grow value.



# justify content

This defines the alignment along the main axis. It helps distribute extra free space left over when either all the flex items on a line are inflexible, or are flexible but have reached their maximum size. It also exerts some control over the alignment of items when they overflow the line.

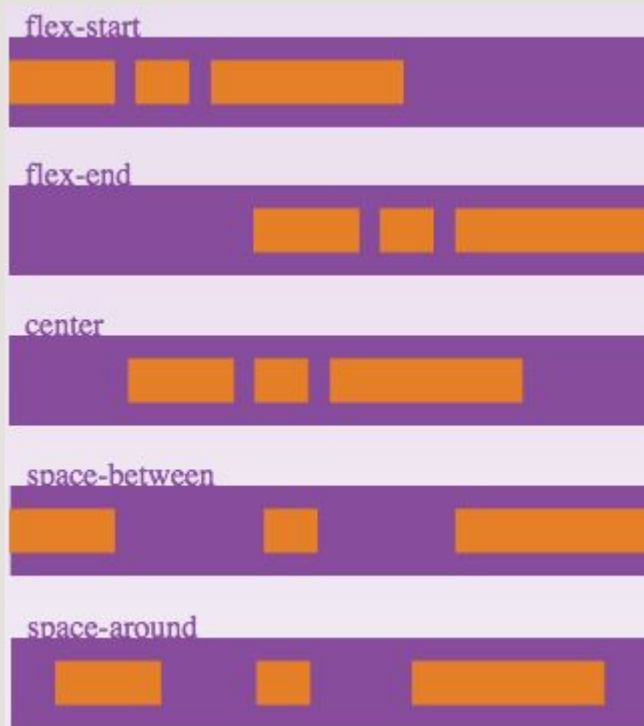
**flex-start (default):** items are packed toward the start line

**flex-end:** items are packed toward to end line

**center:** items are centered along the line

**space-between:** items are evenly distributed in the line; first item is on the start line, last item on the end line

**space-around:** items are evenly distributed in the line with equal space around them. Note that visually the spaces aren't equal, since all the items have equal space on both sides. The first item will have one unit of space against the container edge, but two units of space between the next item because that next item has its own spacing that applies.

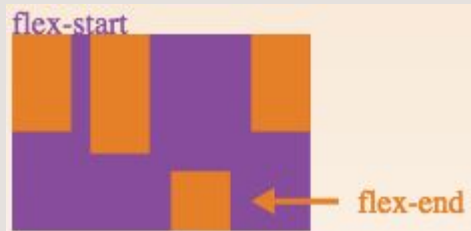


# align-self

This allows the default alignment (or the one specified by align-items) to be overridden for individual flex items.

Please see the align-items explanation to understand the available values.

Note that float, clear and vertical-align have no effect on a flex item.



# align-items

This defines the default behaviour for how flex items are laid out along the cross axis on the current line. Think of it as the justify-content version for the cross-axis (perpendicular to the main-axis).

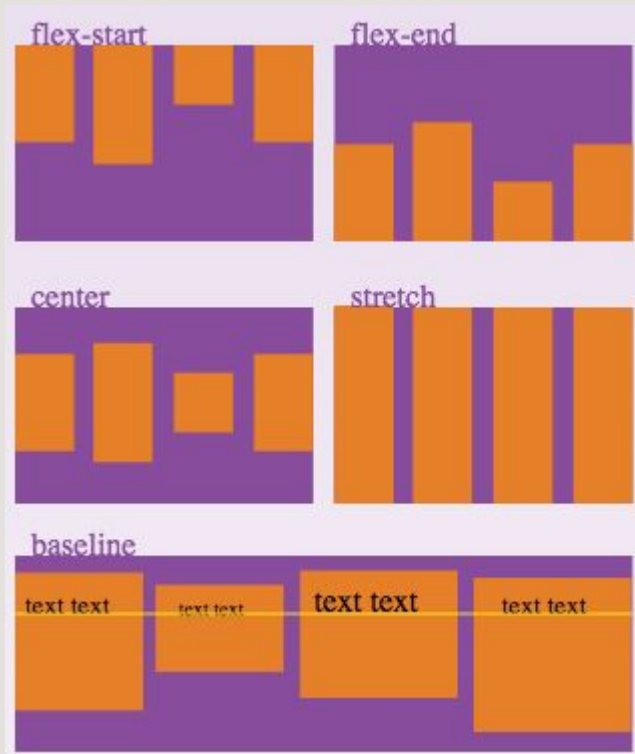
**flex-start:** cross-start margin edge of the items is placed on the cross-start line

**flex-end:** cross-end margin edge of the items is placed on the cross-end line

**center:** items are centered in the cross-axis

**baseline:** items are aligned such as their baselines align

**stretch (default):** stretch to fill the container (still respect min-width/max-width)



# align-content

This aligns a flex container's lines within when there is extra space in the cross-axis, similar to how justify-content aligns individual items within the main-axis.

**Note:** this property has no effect when there is only one line of flex items.

**flex-start:** lines packed to the start of the container

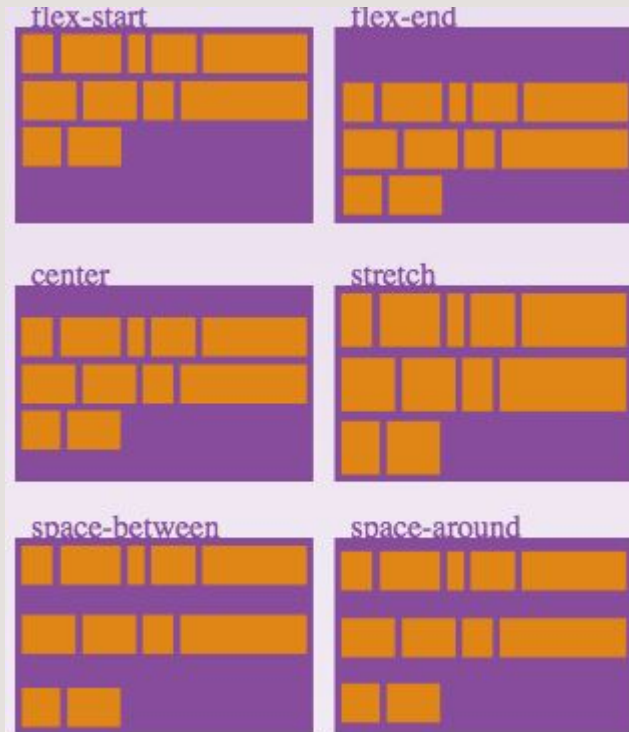
**flex-end:** lines packed to the end of the container

**center:** lines packed to the center of the container

**space-between:** lines evenly distributed; the first line is at the start of the container while the last one is at the end

**space-around:** lines evenly distributed with equal space around each line

**stretch (default):** lines stretch to take up the remaining space



# flex prefixes

```
display: -webkit-box;
```

```
display: -moz-box;
```

```
display: -ms-flexbox;
```

```
display: -webkit-flex;
```



# flex support

IE	Edge <sup>*</sup>	Firefox	Chrome	Safari	Opera
				Chrome 43 Supported Browser usage Global: <b>0.74%</b>	
8			43		
9		40	44		
<sup>2</sup> 10	12	41	45	8	32
11	13	42	46	9	33
	14	43	47		34
		44	48		35
		45	49		

# Resources

<http://www.smashingmagazine.com/2015/11/flexbox-interfaces-tracks-case-study/>

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

<http://webdesign.tutsplus.com/courses/css-flexbox-essentials>

<http://flexbox.io/>

<http://flexboxgrid.com/>

<http://philipwalton.github.io/solved-by-flexbox/>

RESPONSIVE

# Steps to make your site

- Text in em, rem
- responsive grid
- images
- IE9 support
- media queries

# Meta

```
<meta
```

```
  name="viewport"
```

```
  content="width=device-width,
```

```
  initial-scale=1.0">
```

# IE 9

```
<!--[if lt IE 9]>  
  <script src="http://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-mediaqueries.js">  
  
</script>  
<![endif]-->
```

# Media queries

media screen and (max-width: 600px)

<http://nmsdvid.com/snippets/>

<http://cssmediaqueries.com/>

<http://mattkersley.com/responsive/>

# Images

max-width: 100%;



# Source

```
<picture>
```

```
  <source media="(min-width: 45em)" srcset="large.jpg">
```

```
  <source media="(min-width: 32em)" srcset="med.jpg">
```

```
  
```

```
</picture>
```

# Text

All text in rem

# Grid in %

target ÷ context = result

# FRAMEWORKS



Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.

Download Bootstrap

Currently v3.3.5

## Designed for everyone, everywhere.

Bootstrap makes front-end web development faster and easier. It's made for folks of all skill levels, devices of all shapes, and projects of all sizes.



Menu

Tweet 3,509

Star 21,348

English

2.1.6

# Semantic UI

User Interface is the language of the web

Get Started

What's New in 2.1



# Foundation

The most advanced responsive front-end framework in the world.

Download Foundation 5

★ 21.4k stargazers    @ZURBfoundation



## Responsive design gets a whole lot faster

A [Framework](#) for any device, medium, and accessibility. Foundation is a family of responsive front-end frameworks that make it easy to design beautiful responsive websites, apps and emails that look amazing on any device. Foundation is semantic, readable, flexible, and completely customizable. We're constantly adding new resources and [code snippets](#), including these handy [HTML templates](#) to help get you started!

# A dead simple, responsive boilerplate.

DOWNLOAD



Light as a feather at ~400 lines & built with mobile in mind.



Styles designed to be a starting point, not a UI framework.



Quick to start with zero compiling or installing necessary.

---

INTRO

CODE

EXAMPLES

MORE

---

## IS SKELETON FOR YOU?

You should use Skeleton if you're embarking on a smaller project or just don't feel like you need all the utility of larger frameworks. Skeleton only styles a handful of standard HTML elements and includes a grid, but that's often more than enough to get started. In fact, [this site is built on Skeleton and has ~200 lines of custom CSS](#) (half of which is the docking navigation).



DOWNLOAD COMPONENTS UPDATES

## Kube CSS Framework

Introducing new Kube, evolution of a CSS framework  
for professional designers and developers.

Download Kube

Version 5.0 from October 11, 2015, zip-archive

[Old Kube Documentation](#)



### Horizontal rhythm

Innovative 8px Baseline Horizontal Rhythm makes building designs, layouts and creative collaboration easier than ever. Blocks and elements naturally snap into perfect spots.



### Beautiful typography

Crisp, sharp and precisely crafted for absolutely best visual harmony, Kube's typography is purely functional and very utilitarian at the same time.



### Flexbox Grid

Innovative and extremely flexible grid offers clear and simple rules, visual balance and solid structure for any web page, desktop or mobile.



Initializr is an HTML5 templates generator to help you getting started with a new project based on HTML5 Boilerplate. It generates for you a clean customizable template with just what you need to start!

**H5BP**  
**5.0**

### Bootstrap 3.3.1

jQuery  
1.11.2

## 1 - Pre-configuration

## Classic H5BP

Docs Demo

## Responsive

Docs Demo

## Bootstrap

[Docs](#) [Demo](#)

## About Initializr

Initializr is here to kick-start the development of your new projects. It generates templates based on HTML5 Boilerplate by allowing you to choose which parts you want or don't want from it. A responsive template has also been added to start from a basic design instead of a blank page.

## International guides

Initializr functioning is pretty intuitive but it can help to read guides about it in your own language. Here are some which will help you using Initializr and understanding HTML5 Boilerplate, HTML5shiv or Modernizr, in



## Github Repositories


- The builder itself



# Responsive Grid System

## Spectacularly Easy Responsive Design

The Responsive Grid System isn't a framework. It's not a boilerplate either. It's a quick, easy & flexible way to create a responsive web site.

 Tweet 1,295

[</> Generate Code](#)

<http://www.responsivegridsystem.com/>

## Why Use It?

### Any Number of Columns

Don't be forced into having a fixed number of columns across a whole page. You can have whatever you want, wherever you need it.

### Scales to Any Width

Because it uses percentages, your fluid columns will fit into any width. The margins (gutters) use percentages too.

### It's Smart

There's no need to hack in any offsets or marginless final columns. It's the last time you need to use `.last` and the end of `.end`.

### Put the Content First

Instead of fitting your content to your grid, you can make your grid suit your content. Doesn't that feel good?

### It Fits In with You

It plugs into your existing HTML and CSS, it will be your friend in no time.

### Simple Breakpoints

Mobile versions of the grid are already baked in, or you can cook up your own.

### It's Easy

Use it on as simple or as complex a project as you wish. You'll be done in minutes.

### No Maths Required

As long as you can count up to the number of columns you need you'll be fine.



**inuitcss**

# 960

## GRID SYSTEM

[Download](#) - CSS, sketch paper, and templates for: Acorn, Fireworks, Flash, InDesign, GIMP, Inkscape, Illustrator, OmniGraffle, Photoshop, QuarkXPress, Visio, Exp Design. Repository at [GitHub](#).

Big ol' DOWNLOAD button :)



INTERVIEW ABOUT 960.gs

VIEW SLIDES ABOUT THE 960 GRID SYSTEM

ADAPT.JS - ADAPTIVE CSS

CUSTOM CSS GENERATOR

GRID OVERLAY BOOKMARK

### Essence

The 960 Grid System is an effort to streamline web development workflow by providing commonly used dimensions, based on a width of 960 pixels. There are two variants: 12 and 16 columns, which can be used separately or in tandem. [Read more.](#)

### Dimensions

The 12-column grid is divided into portions that are 60 pixels wide. The 16-column grid consists of 40 pixel increments. Each column has 10 pixels of margin on the left and right, which create 20 pixel wide gutters between columns. [View demo.](#)

### Purpose

The premise of the system is ideally suited to rapid prototyping, but it would work equally well when integrated into a production environment. There are printable sketch sheets, design layouts, and a CSS file that have identical measurements.

### More Columns

For those more comfortable designing on a 24-column grid, an alternative version is also included. It consists of columns 30 pixels wide, with 10 pixel gutters, and a 5 pixel buffer on each side of the container. This keeps text from touching browser chrome — helpful for devices like the iPhone, where a lower-case "i" or "l" might be easily missed. [View demo.](#)

### Source Order

By utilizing the *push\_XX* and *pull\_XX* classes, elements can be rearranged, independent of the order in which they appear in the markup. This allows you to keep more pertinent info higher in the HTML, without sacrificing precision in your page layout. For instance, view the source code of this page to see how the *H1* tag has been re-positioned.

<http://960.gs/>



# The web's most popular front-end template

HTML5 Boilerplate helps you build **fast**, **robust**, and **adaptable** web apps or sites. Kick-start your project with the combined knowledge and effort of 100s of developers, all in one little package.

[Download v5.2.0](#)

[Get a custom build](#)

[See the CHANGELOG](#)

## Save time. Create with confidence.

### ★ Analytics, icons, and more

A lean, mobile-friendly HTML template; optimized Google Analytics snippet; placeholder touch-device icon; and docs

### ★ Normalize.css and helpers

Includes [Normalize.css](#) — a modern, HTML5-ready alternative to CSS resets — and further base styles, helpers, media queries, and print

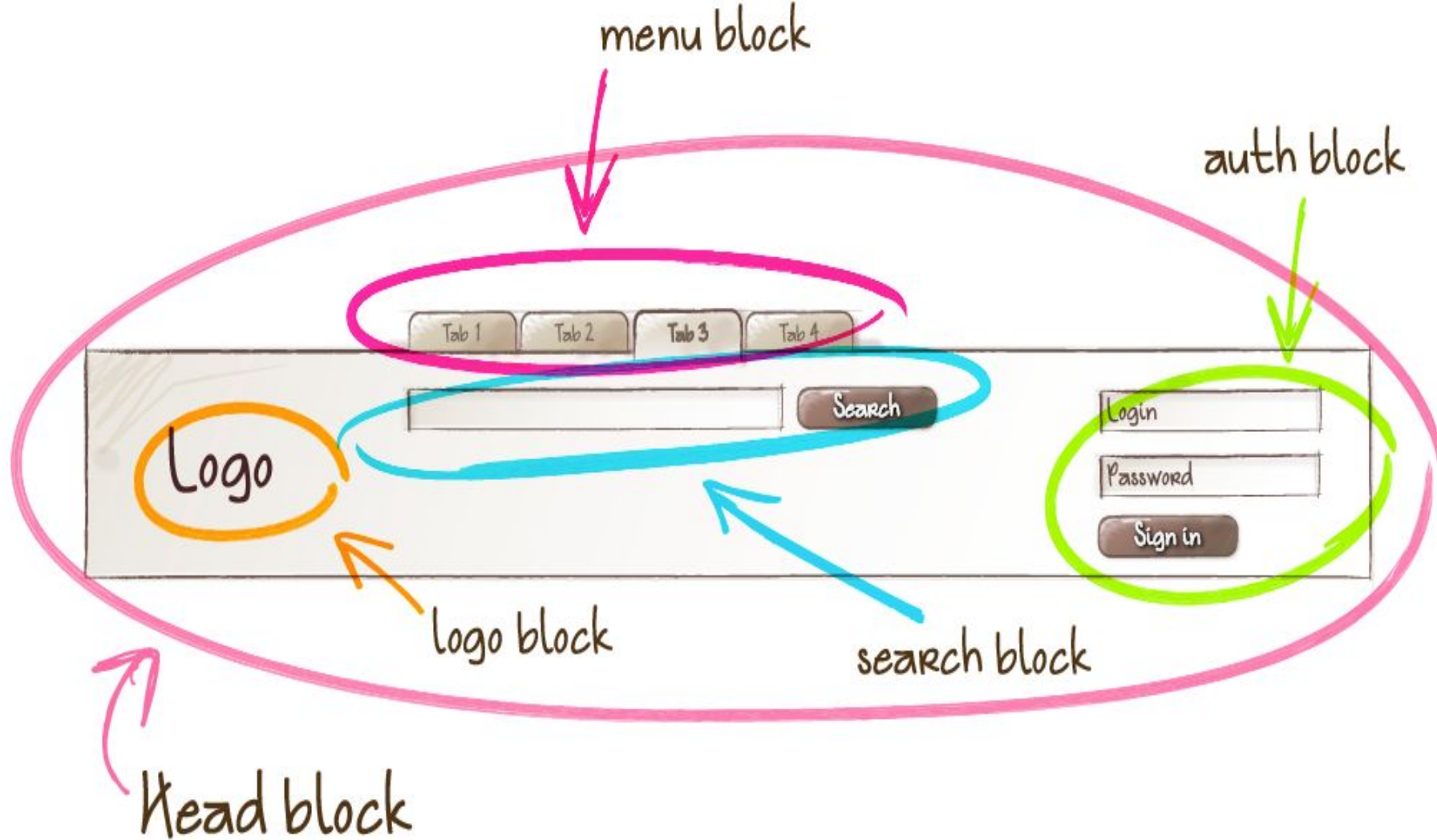
# Методологии верстки

<https://ru.bem.info/>

- Принцип именования классов
- Все блоки независимы друг от друга
- Код можно использовать повторно
- Все изменения точечны касаются только одного блока







# [AMCSS]

*Attribute Modules* for CSS

<https://amcss.github.io/>

# Принципы

кастомные атрибуты

- блоки
- модификаторы
- пространства имен

# Example:

```
<div class="u-posAbsoluteCenter" am-position="absolute center">  
  <div class="u-textTruncate u-textCenter" am-text="truncate center">  
    Very centered text.  
  </div>  
</div>
```

# Object-Oriented CSS

<http://oocss.org/>

Переиспользуемый код

повторяющиеся стили

отделить структуру от оформления



# Atomic CSS

[Get Started](#)

CSS FOR COMPONENT-BASED FRAMEWORKS

<http://acss.io/>

# Принципы

- Один класс одно свойство
- разделение стилей
- для каждого повторного использования свойства должен быть сформирован отдельный класс

# Пример

```
// config object
```

```
'custom': {  
  'brandColor': '#0280ae',  
  'columnWidth': '20px'  
}
```

```
<div class="Pos(a) Bgc(brandColor) W(columnWidth) H(90px)"></div>
```

```
<div class="C(brandColor) BdB Bdc(brandColor) Mstart(columnWidth) P(10px)">
```

```
  Lorem ipsum
```

```
</div>
```



## Пример 2

```
<div>
  <div class="Bgc(#0280ae.5) H(90px) IBox W(50%) foo_W(100%)"></div><!--
--><div class="Bgc(#0280ae) H(90px) IBox W(50%) foo_W(100%)"></div>
</div>
<hr>
<div class="foo">
  <div class="Bgc(#0280ae.5) H(90px) IBox W(50%) foo_W(100%)"></div><!--
--><div class="Bgc(#0280ae) H(90px) IBox W(50%) foo_W(100%)"></div>
</div>
```

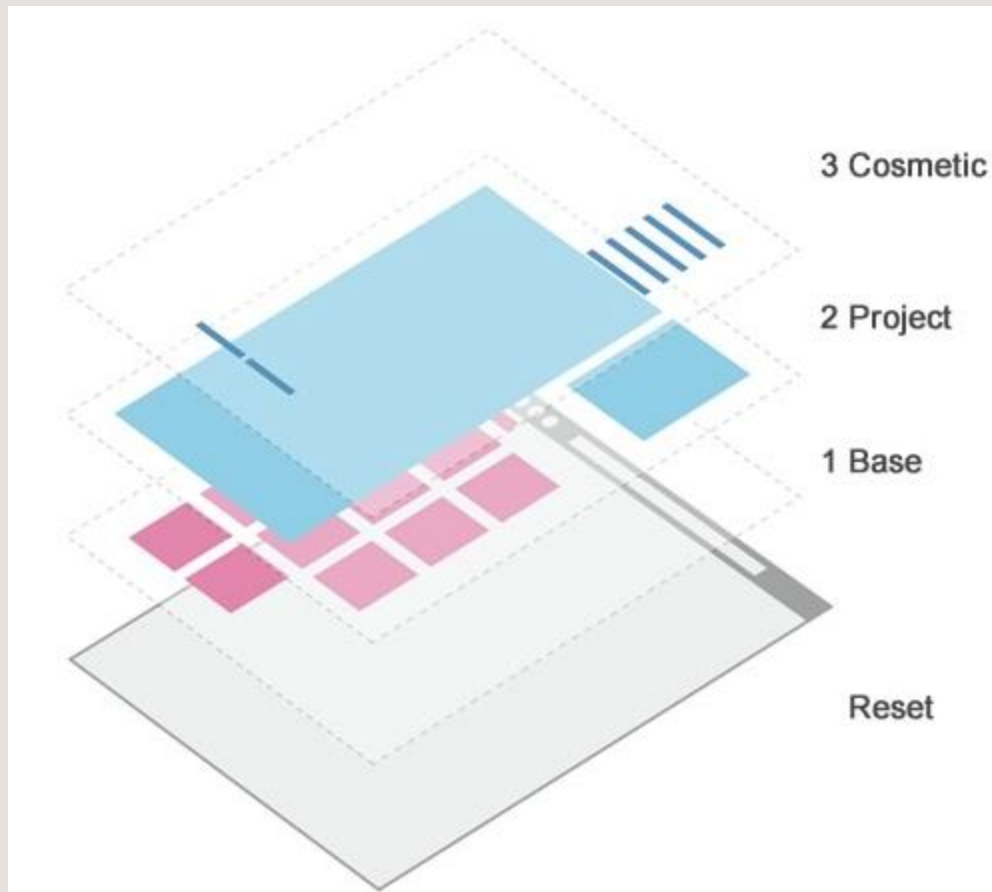
# OPOR

<http://nano.sapegin.ru/all/opor-methodology>

# MCSS

<https://operatino.github.io/MCSS/>

Идея в разделении стилей на слои





# Scalable and Modular Architecture for CSS

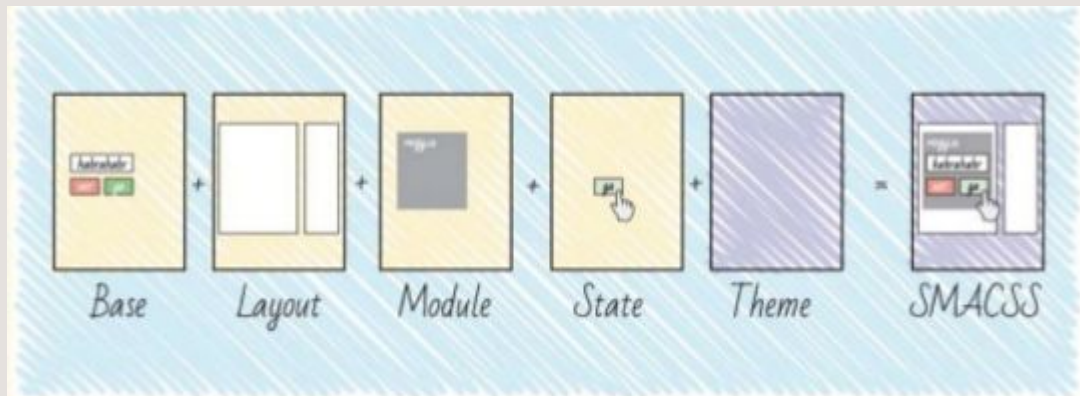
A flexible guide to developing sites small and large.

<https://smacss.com/>

Разделение всех стилей на 5 категорий

1. Base
2. Layout
3. Module
4. State
5. Theme

# 5 категорий стилей



# DoCSSa 2.0 {dok~sa}

Sass based CSS architecture and methodology

<http://docssa.info/>

- Clear and proven folder structure
- Separation of concerns between HTML and CSS
- Scalable and maintainable stylesheets
- Easy to setup and flexible