**Тест 💎**

1. Какими способами можно подключать CSS-стили? Найдите сами еще один способ, не указанный в уроке

The first method is by using the already familiar **style** **attribute** - this is convenient if you need to quickly test something or for a short set of styles. But in a real project it is better not to use this method, because it is very difficult to maintain.

<span style="color:'red'">красным</span>

The second method is by using the **<style> tag**. In this case, CSS properties are described in the document itself and are located in the header of the web page. This allows you to store styles in one place, right on the same page - and this method is much more maintainable, because it makes styles much easier to find and change if necessary.

<head>

<style>

**h1** { color: #333366; }

**img** { width: '100%' }

</style>

</head>

The third method is by using the **<link> tag**, you can connect styles that are in a separate file. This is the most optimal way, because then the same styles can be used on different pages if there is more than one of them.

<head>

<meta charset="utf-8">

<title>Стили</title>

<link rel="stylesheet" href="style.css">

</head>

Also, it is possible to add CSS style in **JavaScript**, when using the **jQuery library**, and there are many different functions for styling HTML elements. For example, the .css () function - sets the CSS style for the element, .hide () - adds the display property to the CSS element: none; (element hiding), etc.

When using these features, CSS properties are added to the style attribute of the tag.

1. Зачем нужен Normalize.css?

\*\* Normalize.css \*\* is a small CSS file that provides better cross-browser compatibility (cross-browser support) for HTML elements in the default styles. The targets of normalize.css are:

- save useful browser settings, not erase them;

- normalize styles for a wide range of HTML elements;

- correct errors and major browser inconsistencies;

- improve usability with subtle improvements;

- explain the code using comments and detailed documentation.

1. Что такое CSS-директивы?

CSS-Directives are constructs that allow you to create instructions in CSS to change the display or behavior of page elements. The directive begins with the **@ sign**, followed by one of the service words. This is the general syntax that all directives follow.

Such directives follow the standard syntax: @ [KEYWORD] (RULE);

For example, @charset

This directive defines the encoding used by the browser. This is useful if your style sheet contains non-ASCII characters (such as UTF-8). Note that the encoding specified in the HTTP header overrides any @charset specified in your stylesheet.

@charset "UTF-8";

1. В чем разница между **margin** и **padding**?

You can create gaps between elements in either way, but if **padding** is the indent from the content to the edge of a block, then **margin** is the distance from one block to another, interblock space. Note that negative values can also be used for margins (for example, -3px), which can be quite useful sometimes.

1. Как в CSS определяются приоритеты? Какое из свойств будет приоритетнее - #link .main или span #login?

Multiple CSS rules can be applied to the same element. In this case, the properties of the rules are combined. Element properties can be combined and explicitly using multiple classes. But when different CSS rules have the same properties with different values, then they conflict. he more specific the CSS selector is, the higher is the precedence of the CSS property declarations inside the CSS rule owning the selector.

In general terms, the more specifically (uniquely) a CSS selector targets an HTML element, the higher is its specificity. That is why, a priority system was created: in the end, whichever style comes from a source with a higher priority is applied. The specificity of a selector is split into 4 groups: a, b, c, d.

If the CSS properties are set directly on element, inside style attribute, then a=1, otherwise a=0.

The value of b is equal to the number of id selectors, which begin with #.

The value of c is equal to the number of class selectors, pseudo-classes and attributes selectors.

The value of d is equal to the number of selectors of the types of elements and pseudo-elements.

When an important rule is used on a style declaration, this declaration overrides any other declarations. Although technically !important has nothing to do with specificity, it interacts directly with it. Using !important, however, is **bad practice** and should be avoided because it makes debugging more difficult by breaking the natural [cascading](https://developer.mozilla.org/en-US/docs/Web/CSS/Cascade) in your stylesheets.

#link .main will have more priority.

1. В чем разница между CSS1 и CSS3?

CSS 1 had font properties and specification for typeface and emphasis. The text attributes were supported such as spacing of letters and line of text.

Alignment of text, positioning and tables were also added. Margin, padding, border and positioning for elements was also implemented. This recommendation was not maintained by W3C.

Before CSS3 the developers used to design images which looked like rounded corners to different structures and background gradients. With CSS3 it is just necessary to write the code like “roundBorder {border-radius:10px}”.

CSS3 is the most recent and currently used. It has XHTML specification. CSS3 has its major focus on modularisation and separation of concerns.

Different modules now go through different stages of the recommendation process. CSS3 has support for almost all recent web browsers.

It has even included new selectors along with new combinator and new pseudo-elements.

CSS3 has several new CSS properties. It supports animation which is not a part of earlier recommendations. Earlier it was written in JavaScript and JQuery.

There were various properties added such as transforms, gradients, animation and transition for animation effect in the website.

Recent add-ons are like border-radius, box-shadow, flex-box and CSS grid.

CSS3 is backward compatible with CSS1. They load faster and the time required to build a page is even less.

1. Что такое псевдоклассы? А псевдоэлементы?

Pseudo-classes define the dynamic state of elements, which is changed by user actions, as well as the position in the document tree. An example of such a state is a text link that changes its color when the mouse cursor is hovering over it. When using pseudo-classes, the browser does not overload the current document, so using pseudo-classes you can get different dynamic effects on the page.

The syntax for using pseudo-classes is as follows.

selector: pseudo-class {...}

The selector is indicated first, to which the pseudo-class is appended, followed by a colon, followed by the name of the pseudo-class. It is allowed to apply pseudo-classes to identifier or class names (`a.menu:hover {color: green}`), as well as to context selectors (`.menu A: hover {background: # ffcc00}`). If the pseudo-class is specified without a selector in front (`: hover`), then it will be applied to all elements of the document.

Examples of pseudo-classes:

- those determining the state of the elements (link, hover)

- those related to the tree of elements (root, first child)

- those used to work with forms (valid, checked)

1. Изучите статью про "плохие" теги <https://msiter.ru/tutorials/html-srednego-urovnya/plokhie-tegi> и пришлите список тегов, которые нежелательно использовать

<u>, <center>, <layer>, <blink> или <marquee>, <font>, **<b>, <i>, <big>, <small>, <hr>**

1. Как можно подключать шрифты локально?

The @ font-face rule allows you to define font settings, as well as download a specific font to the user's computer. Inside the @ font-face construction, there can be a set of properties to change the font parameters (font-family, font-size, font-style, etc.), as well as a link to the font file. The link is written in the form src: url (URI), where URI is the relative or absolute path to the file.

**<style>**

@**font**-face {

font-family: Pompadur; /\* Имя шрифта \*/

src: **url(**fonts/pompadur.ttf**)**; /\* Путь к файлу со шрифтом \*/

}

**P** {

font-family: Pompadur;

}

**</style>**

1. Почему не стоит использовать сокращенную запись без необходимости? И если все же использовать, как это делать правильно?

The use of these properties reduces the amount of code and increases its readability, but on the other hand, sometimes adds confusion. When we see the following code:

background-color: pink;

background: url ('mycat.png');

we should know that the second variant will prevail, so a picture with a cat, because when using a short notation, the properties that were not specified are reset to their default values, that is, in fact, the browser will see this:

background: transparent url ('mycat.png');

If resetting the background color was not part of your plans, the background image should be set using the `background-image` property:

background-color: pink;

background-image: url ('mycat.png');

To avoid the problems, you should, firstly, group properties by meaning, this will allow you to find errors faster, and secondly, if you need to override the values of previously set properties, do not use the shorthand notation.

1. Разберитесь самостоятельно, как сделать анимацию через CSS

Keyframes are used to specify values for animation properties at various points in the animation. Keyframes define the behavior of one animation loop; animation can be repeated zero or more times.

Keyframes are specified using the **@keyframes rule**, defined as follows:

@keyframes animation name {list of rules}

When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.

To get an animation to work, you must bind the animation to an element.

The following example binds the "example" animation to the <div> element. The animation will last for 4 seconds, and it will gradually change the background-color of the <div> element from "red" to "yellow":

/\* 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-duration property defines how long an animation should take to complete. If the animation-duration property is not specified, no animation will occur, because the default value is 0s (0 seconds).

In the example above we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).

The animation-delay property specifies a delay for the start of an animation.

The animation-iteration-count property specifies the number of times an animation should run.

**Link to the Figma template, which I made:** <https://www.figma.com/file/aIEteGtMB9yunMumDtPUMZ/Landing-Page-Concept-For-Data-Processing-Company-(Copy)?node-id=1%3A2>