CSS Базовые

**Значения по умолчанию:**

- В <body> по умолчанию стоит margin: 8px; - его нужно убирать 🡪 margin: 0 auto;

- height: auto; -- стоит всегда по умолчанию

- Иконки Font-awesome можно перекрашивать меняя color в .CSS.

Если папка стиля лежит в другой папке чем HTML, то при добавлении картинки:

background-image: url("../img/avatar.png") - добавлять ../ смотря на сколько уровней углубиться

**Функции:**

* padding-top / right / bottom / left: - внутренние границы
* margin-top / right / bottom / left: - внешние границы
* margin: 16px auto; - отцентровать текст на экране с любым разрешением
* .clearfix:
* display: inline;
* display: block;
* display: inline-block; - расположить 2 блока рядом. Прижимает всегда иконки к самому низу блока, тк по умолчанию стоит vertical-align: bottom. Отцентровать блок по самому высокому объекту можно, изменив этот параметр на top/middle
* display: flex;

**Визуальная обработка (кнопок, фото, элементов):**

* border:
* border-radius:
* overflow: hidden; - обрезать углы фото для скругления

**Изменить размер фото:**

.img-block>img {

width: 250px;}

**Скролл:**

* overflow: hidden;
* overflow: auto; - появляется скролл в блоке текста
* overflow: scroll; - скролл гориз+вертик будет всегда, даже если текста мало
* overflow-y: hidden; - скролл по вертикали
* overflow-x: auto; - скролл по горизонтали

**Поставить 2 блокa в ряд:**

<div class="block1"></div>

<div class="block2"></div>

.block1{

    width: 200px;

    padding: 5px;

    padding-right: 20px;

    border: solid 1px black;

    float: left; }

.block2{

    width: 400px;

    padding: 5px;

    padding-right: 20px;

    border: solid 1px black;

    float: left;

    position: relative;

    left: 10px; }

**Поставить 2 блокa в ряд:**

<div class="content-wrapper clearfix"> - ставить после создания класса .clearfix::after

<div class="left-block">

First

</div>

<div class="right-block">

Second

</div>

</div>

.content-wrapper{

    width: 800px;

    margin: 0 auto;

}

.clearfix::after{ - писать его в конце

    content: "";

    display: table;

    clear: both;

}

.left-block,

.right-block{

    padding: 20px;

    text-align: center;

    width: 400px;}

**Отцентровать блок:**

Создавая блок текста header/section/footer лучше не указывать width , т.к. эти блочные элементы сами растянутся по ширине всего экрана.

Но если нужно именно 1400px: - они буду отцентрованы

    width: 1400px;

    margin: 0 auto;

**Float**

Элемент помещён вдоль левой или правой стороны его контейнера (родительского элемента), где текст и inline элементы будут обтекать его.

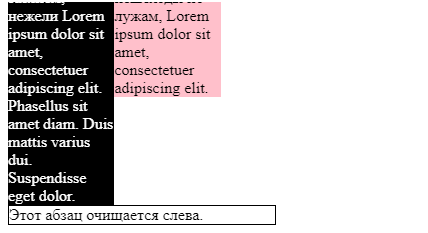
**Сlear**

Свойство clear CSS указывает, может ли элемент быть рядом с плавающими floating элементами, которые предшествуют ему или должны быть перемещены вниз (очищены) под ними.

* clear: none;
* clear: left; - элемент перемещается вниз, чтобы очистить левые поплавки.
* clear: right; - элемент перемещается вниз, чтобы очистить правые поплавки.
* clear: both;
* clear: inline-start; - элемент перемещается вниз для очистки поплавков в начале содержащего его блока
* clear: inline-end; - в конце содержащего его блока

*Пример:*

.wrapper {

 border: 1px solid black;

padding: 10px; }

.left {

border: 1px solid black;

clear: left; }

.black {

float: left;

margin: 0;

 background-color: black;

color: #fff;

width: 20%; }

.red {

float: left;

margin: 0;

background-color: pink;

width: 20%; }

p {

width: 50%; }

**Flex**

display: flex;

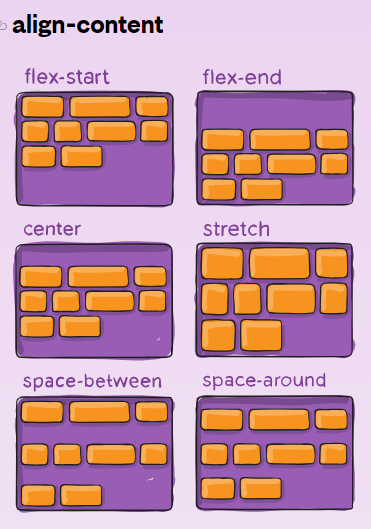
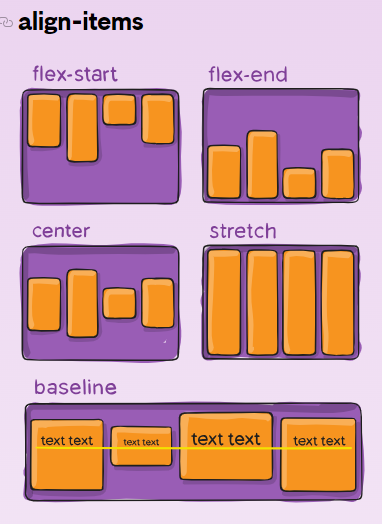
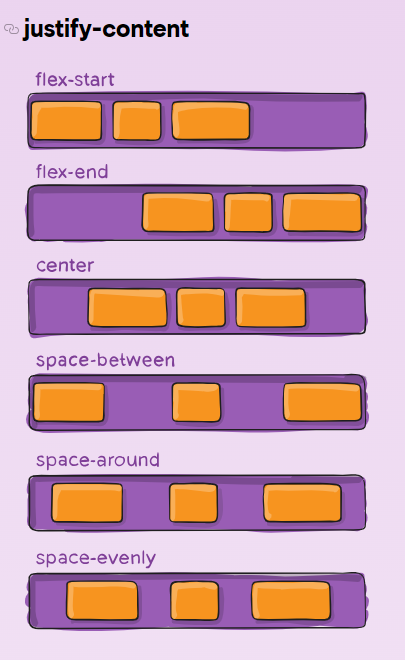
* flex-direction: row; - по умолчанию
* flex-direction: row-reverse; - размещение справа-налево
* flex-direction: column; - блоки занимают 100% ширины, столбик
* flex-direction: column-reverse; - столб, снизу-вверх
* flex-wrap: nowrap; - по умолчанию, запрещает перенос блоков на новый ряд
* flex-wrap: wrap; - блоки берут ширину которую мы выставили сами, переносятся если не вмещаются
* flex-wrap: wrap-reverse; - блоки которые не вмещаются перемещаются не вниз, а наверх
* flex-flow - flex-direction + flex-wrap; -
* justify-content: flex-start; - расположение сверху-слева

горизонталь(row)

* justify-content: flex-end; - расположение сверху-справа
* justify-content: space-between; - располагает 2 блока по двум разным краям, с равным отступом между всеми блоками
* justify-content: space-around; - отступ маленький по краям и крупный внутри блоков
* justify-content: space-evenly; - равный отступ от краев и между блоков
* align-items: stretch; -

вертикаль (column)

* align-items: center; - по центру высоты самого высокого блока
* align-items: flex-start; - сверху
* align-items: flex-end; - снизу
* align-items: baseline; -
* row-gap: 50px;
* column-gap: 50px;
* align-self: center; - особые задачи для определенного объекта

****

|  |  |
| --- | --- |
| **Менять очередность объектов:**  <div class="d-flex">          <div class="item">1</div>          <div class="item">2 </div>          <div class="item">3</div>          <div class="item">4</div>      </div> | .item:nth-child(2){      order: 2; }  .item:nth-child(3){      order: 3;      align-self: center; }  .item:nth-child(4){      order: 1;      align-self: stretch; } |

**Before, After** – псевдо-элементы, размещают элементы перед или после текста.

|  |  |
| --- | --- |
| <h1>Hello CSS</h1> | h1::before{      content: "\*";      color: orange;      font-size: 40px; }  h1::after{      content: "!!!";      color: green;      font-size: 30px; } |
|  | h1::before{      content: "";      display: inline-block;      width: 100px; - можно редактировать для изменения гориз/вертик полосы      height: 10px; - можно редактировать для изменения гориз/вертик полосы      background-color: orange;      margin-right: 30px;      margin-bottom: 10px;  Скопировать те же настройки в  h1::after{} |

**Hover** – при наведении курсора, фон станет черным

|  |  |
| --- | --- |
|  | h1:hover{      background-color: black;      color: white; } |
| <h1>Hello CSS <span>Inside</span> </h1> | Ко всем предыдущим настройкам добавляем:  h1:hover span{      color:red; } |

**Сделать кнопку при наведении курсора другого цвета:**

.main button:hover{

    color: yellow;

    background-color: blue; }

**Position**

* position: static; - по умолчанию
* position: absolute; - прификсовать блок на стр, что бы при скролле он там и оставался
* position: fixed; - нужно задавать значения (left100px,bottom200px / top,right). Занимает ширину согласно ширине контента, ее нужно доп. прописывать
* position: relative;
* position: sticky; - приклеивает блок во время движения
* z-index: 0-999; - для нумерации приоритетности блоков

Absolute – накладывается поверх всех блоков и текста. Лучше всего для создания блоков и анимации.  
Обозначить блоки, которые он не должен перекрывать - z-index:в тэг блока.  
Начинают отсчет от *верхнего левого угла*, поэтому нужно указывать координаты (top/left)

Relative – занимает 100% страницы, если его двигать по ней, он расширяет саму страницу. Резервирует свое место на стр после его сдвига.

**Сделать двигающуюся анимацию:**

<div class="pos-r">

    Relative

</div>

<div class="pos-r second">

  Relative 2

<div class="abs"> - Абсолют ставим во внутрь Рилейтив 2

    Absolute

  </div>

</div>

.pos-r{

    background-color: lawngreen;

    text-align: center;

    padding: 30px;

    position: relative;

    height: 400px;

    z-index: 3; }

.pos-r.second{

    overflow: hidden; - навести на Рилейтив блок, и Абсолют выедет при наведении курсора

    top: -50px;

    left: 200px;

    background-color: orange;

    z-index: 2; }

.pos-r.second:hover .abs{ - создать анимацию перемещения Абс. из Рил. 1 в Рил. 2

    top: 50px;

    left: 100px;

    transition: all 5s; }

**Зафиксировать маленький блок на сайте:**

<body>

<a href="#top">Top page</a>

</body>

a{

    display: inline-block;

    padding: 20px;

    color: white;

    background-color: black;

    border-radius: 10px;

    position: fixed;

    right: 10px;

    bottom: 100px; }

**Background image**

* background-size: contain; - растягивает картинку по высоте

**Изменить размер бэкграунда**

* background-size: cover; - растягивает картинку по ширине, высота вместится сколько получится
* background-size: 200px; / 30% - писать значение какое нам нужно
* background-attachment: fixed; - не двигается, фиксирован

**Фиксация бэкграунда**

* background-attachment: scroll; - скроллится
* background-attachment: local;

|  |  |
| --- | --- |
| * background-repeat: no-repeat; * background-repeat: repeat; / repeat repeat * background-repeat: repeat-x; / repeat no-repeat * background-repeat: repeat-y; / no-repeat repeat * background-repeat: space; / space space * background-repeat: round; / round round | В двойных синтаксисах первый валью отвечает за горизонталь, второй за вертикаль |

* background-position: top / bottom / left / right / center;
* background-position: 25% 75%; / bottom 10px right 20px;
* background-position-x: left / right / center; / px; / em; / %; - по горизонтали
* background-position-y: top / bottom / center; / px; / em; / %; - по вертикали

background-attachment: fixed; - (Это значение несовместимо с [background-clip: text](https://developer.mozilla.org/ru/docs/Web/CSS/background-clip#text))

background-attachment: local; - Фон фиксируется относительно содержимого элемента. Если элемент имеет механизм прокрутки, фон прокручивается с содержимым элемента.

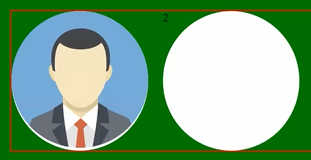
|  |  |
| --- | --- |
| **Размещение текста поверх картинки:**  <div class="simple-bg">  <p>  Lorem...  </p>  <img src="img/avatar.png" alt="">      </div> | .simple-bg{  Position: relative;  Width: 600px;  P{  Position: absolute; |

**Добавить картинку поверх фона картинки:**

background-image: url("../img/avatar.png"),url("../img/1.webp"); - через запятую

**Добавить круглую картинку:**

.item{

    background-color: white; - фон (круг)

    border-radius: 100%; - сделали форму круга

    /\* height: 100px; \*/ - высота мешает % ширины отображаться верно

    width: 31.1%; - % ширины относительно осн. блока

    margin-top: 25%;

    margin-left: 25%;

    padding: 10%; }

.item img{

    width: 100%; - занимает весь родительский элемент (круг)

**При наведении курсора меняется цвет фона:**

.simple-bg:hover{

    background-color: yellow;

**Отцентровать бэкграунд:**

background-repeat: no-repeat;

background-position: top *200*px left *200*px; / bottom / right;

background-position: bottom center / top center;

background-position: center;

Шрифты5

**Подключить fonts в html:** (на выбор)

ИЛИ / Ссылка в <body> .HTML

- Скопировать ссылки с «*Use on the web*» 🡪 ПОСЛЕ <title>Document</title> 🡪 вставляем (3 ссылки)

- Скопировать ссылки с «*CSS rules*» 🡪 .CSS 🡪 вставить в body{*ссылка*}

ИЛИ / Импорт в CSS

ссылку внутри <styles> вставляем в Style.css *(@import url('https://...)*

**Подключить шрифты к файлу:**

Если большинство шрифта в font-weight: 400 / font-size:16px, то это значение пишем в body{}. Другие значения прописываем в стиле элемента.

*Пример:*

body{

    font-family: 'Roboto', sans-serif;

    font-weight: 400;

    font-size: 16px; }

h1{

    font-weight: 700; }

span{

    font-weight: 100; }

Далее создаем второй шрифт:

.secondary-font{

    font-family: 'Roboto Slab', serif;

    font-weight: 500; }

Подключаем второй шрифт:

<h2 class="secondary-font"> </h2>

**6 Reset, Normalize**

<https://meyerweb.com/eric/tools/css/reset/>

Reset CSS – набор стилей, которые задает правила опред. элементам. Марджин/паддинг/бординг убирает на 0, удаляет стилизацию цитат, в таблицах добавляет некоторые стили и тд.

**Добавить Reset CSS в свою библиотеку:**

1. Копируем с сайта 🡪 создаем папку *.reset* в том же дереве в VSC 🡪 вставляем
2. В <body>.HTML 🡪 <link rel="stylesheet" href="style/reset.css">

ИЛИ  
  
1. Normalize СSS – не убирает все к 0, а формирует все в один стиль

**Font-Awesome**

**Подключить font-awesome:**

<https://cdnjs.com/libraries/font-awesome>

https://fontawesome.com/search?o=r&m=free



Копируем ссылку 🡪 вставить в .HTML <body> после шрифтов 🡪 заходим в font-awesome 🡪 клик на иконку 🡪 копируем ее тэг 🡪 вставляем вместо экспор. картинки

Иконки можно перекрашивать меняя color в .CSS.

Для придания Hover иконки:

*.social-block* a:hover{

    color: rgb(187, 6, 181); }

**Размеры страницы 8**

**Приавильное отображение размеров на странице:**

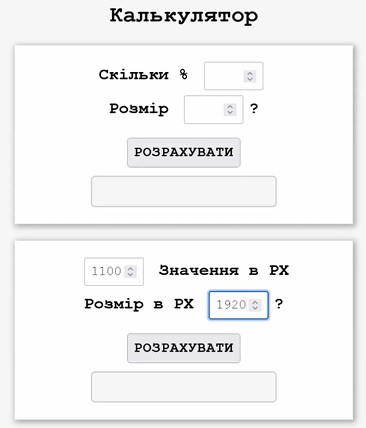
max-width: 1000px; - контент будет сжиматься правильно при расширении страницы

    если написать width: 1000px - это неверно

    min-width: 1000px; - контент будет сжиматься правильно при сжатии страницы до этого миним. значения

Создать правильный **header**

**Ширина блока**

В % страница на разных устройствах будет выглядеть правильно.

% пропорционально уменьшают объекты на странице по мере уменьшения экрана.

**Ширина экрана**

header{

    /\* width: 1920px; - ширину не указывать

    /\* width: 100%; - по умолчанию, более правильный вариант

    background-color: green;

    padding: 20px 0; }

.wrapper{

    /\* width: 1100px; \*/

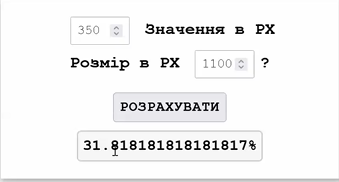
    width: 57.2%;

    /\* max-width: 1100px; \*/

    margin: 0 auto; - рекомендуется

Рассчитать размер внутреннего блока во врапере в %

**Ширина вн.блока**

****

**Ширина осн.блока**

VH, VP

Сделать блок на всю высоту экрана: (без скролла)

.vh{

    background-color: orange;

    display: flex;

    justify-content: center;

    align-items: center;

    min-height: 100vh; - высота просчитывается самостоятельно для каждого браузера. Если есть текст, то располагает его правильно по высоте блока

width: 57vw;

    margin: 0 auto;

|  |  |
| --- | --- |
| 2 блока на всю высоту экрана, скроллятся:  <section class="vh">       <h1>Hello</h1>  </section>  <section class="vh">  <h1>Hello2</h1>  </section> | .vh:nth-child(2){      background-color: gray; } |

**Sprite, Parallax 7**

|  |  |
| --- | --- |
| Создать иконку Спрайт  <button class="fb">  </button> | .fb{      width: 84px; - уменьшаем ширину и длину до размера иконки в спрайте (инспектор)      height: 84px;      border: none;      display: block;      margin-bottom: 50px;      background-image: url('../img/sprite-icon-29.jpg');      background-position: top 0 left 0; - меняя координаты в инспекторе, можно выбирать нужную иконку в выделенном квадрате }  .fb:hover{      background-position: top -84px left 0; - выставили координаты, что при наведении курсора иконка будет меняться на другую } |

|  |  |
| --- | --- |
| **Создать Параллакс страницу:**  <div class="parralax">          <header>              Header content          </header>          <section>              Lorem          </section>          <section>              Lorem          </section>  </div> | .parralax{      padding-top: 250px;      background-image: url('bg.jpeg');      background-attachment: fixed;      background-position: center;      background-repeat: no-repeat;      background-size: cover;      min-height: 100%; }  header, section{      padding: 50px;      background-color: gray;      margin-bottom: 200px; } |

**Java Script 1**

- Для инспектора в браузере исп. Console.

- console.log() для выведения инфы в консоль браузера

**Пробел:** let spaceMark = **" "**;

${} - указать переменную к которой обращаться

! – конвертирует значение. True превращает в false

НЕ использовать:

- цифры вначале значения. В конце допустимо, но нежелательно

- тире

- верхний регистр

- зарезервированные значения https://www.w3schools.com/js/js\_reserved.asp

**Создание Java документа:**

В <head> под <title>Document</title> 🡪 подключаем файл со скриптом <script src="main.js"></script>

Есть вариант подключения, но используется редко:



**Варианты письма в JS**

userName / UserName / user\_name / user-name / username - *Tom*

USERNAME - запись в верхнем регистре JS распознает как const

**Изменить значение переменной:**

userName = *'Bob'*;

|  |  |
| --- | --- |
| let newValue = **userName** + **numberValue**;  console.log('newValue =', newValue); | Стринг + номер дает значение стринг |

**Переменные :**

var salt = 'salt';

*var* – название переменной

*salt* – имя перем.

*=* - в языке программирования означает «присвоить»

*'salt'* – значение переменной

let

console.log

console.log(object); - () выводит информацию в консоль

console.log('Hello console'); - (``) стринг для написания просто текста

**Диалоговое окно при открытии страницы:**

* alert(''); - Блокирует обработку другого JS, пока не нажать ок
* prompt(''); - выводит диалоговое окно с input и button, сохраняет ответ пользователя

|  |  |
| --- | --- |
| **Let + Prompt** **Диалоговое окно с input и button , сохраняет ответ и использует в дальнейшем:**  let firstPrompt = prompt('*Ваше перше число*');  let secondPrompt = prompt('*Ваше друге число*');  console.log(**+**firstPrompt**+** **+**secondPrompt); | **+** перед firstValue даст результат в цифровом значении |

**Диалоговое окно с input и button , сохраняет ответ и использует в дальнейшем:**

* Let dayInfo + Prompt

let dayInfo = prompt('*Як ваш день?*');

let dayInfo = prompt('*Як ваш день?*', '*запропонований текст в input*);

console.log('DayInfo from prompt', dayInfo);

**Для написания цифр:**

* console.log(numberInfinity = Infinity); - Infinity
* console.log(numberValue) = 100; - для письма цифр
* console.log(numberNan =); - NaN
* console.log('NumbeFloat', numberFloat = 0.33;) - для письма цифр с десятыми
* console.log('Max value:', Number.MAX\_VALUE);
* console.log('Min value:', Number.MIN\_VALUE);
* console.log('Max value:', Number.POSITIVE\_INFINITY); - расширит значение бесконечности
* console.log('Min value:', Number.NEGATIVE\_INFINITY); - негативное значение бесконечности

**Отображение целых цифр с десятыми**

* **parseInt / parseFloat**

console.log**(parseInt**(firstPrompt)**)**; - округляет числа до первого цельного

console.log**(parseFloat**(firstPrompt)**)**; - с плавающей запятой между числами

console.log**(Number**(firstPrompt)**)**; - автоматически системой расчитывает как отображать число

let firstPrompt = prompt('*Напишіть перше число*');

let secondPrompt = Number(prompt('*Напишіть друге число*'));

let firstPromptNumber = parseFloat(firstPrompt);

let summOperation = firstPromptNumber+secondPrompt;

console.log(summOperation);

**Изменить значение переменной:**

let firstValue = 1;

Вариант 1:

firstValue = firstValue +3; // 1+3

Вариант 2:

firstValue +=3; firstValue \*=5;

firstValue -=4; firstValue %=2;

|  |  |
| --- | --- |
| **Математические исчесления: Расчет суммы от 2х значений:**  let firstValue = '10';  let secondValue = 10;  let summVariables = **firstValue + secondValue**;  console.log('Result', summVariables);   * console.log(firstValue**+**secondValue); - соединит в одно число * console.log(firstValue**-**secondValue); * console.log(firstValue**\***secondValue); * console.log(firstValue**/**secondValue); * console.log(firstValue**%**secondValue); - для получения остатка от деления (5%2=1) | **+** перед firstValue даст результат в цифровом значении |

**Bolean**

let trueValue = true;

let falseValue = false;

console.log(numberInfinity); - Infinity

console.log(numberValue); - 100

console.log(numberNan); - NaN

console.log('NumbeFloat', numberFloat); - 0.33

console.log('Max value:', Number.MAX\_VALUE);

console.log('Min value:', Number.MIN\_VALUE);

console.log('Max value:', Number.POSITIVE\_INFINITY); - расширит значение бесконечности

console.log('Min value:', Number.NEGATIVE\_INFINITY); - негативное значение бесконечности

Урок 2

**Математические исчесления: Расчет цены:**

let productPrice = 10;

let productAmount = 10;

let productName = 'Milk';

“ ” дает пробел

let totalProductInfo = productPrice+” ”+ productName;

console.log(totalProductInfo); - будет 100 в консоли

${} - указывает переменную к которой обращаемся

**Сокращенный способ: Template Literal (string)**

let newString = `Product Info: ${productName} ${(productPrice\*productAmount\*5)/100}`;

Внутри TL можно рассчитывать формулы/проценты

console.log(newString);

**Альтернативный вариант: Template Literal (string)**

document.write(**`**<h1>${productName}</h1><h2>Product price ${productPrice}</h2>**`**);

**Операторы сравнения:**

**== Равно**

console.log(`Compare 5 and 5 ${5==5}`); *-true*

console.log(`Compare 6 and 5 ${6==5}`); *-false*

console.log(`Compare '5' and 5 ${'5'==5}`); *-true*

console.log(`Compare null and undefined ${null==undefined}`); *-true*

console.log(`Compare 0 and false ${0==false}`); *-true*

**=== Суровое сравнение**

*{'5'===5}* не воспринимает стринг и номер как одно и то же

*{0==false}* – 0 и Bolean элемент не могут сраниваться

**!= Не равно**

console.log(`Compare 5 and 5 ${5!=5}`); *-false*

console.log(`Compare 6 and 5 ${6!=5}`); *-true*

**< > Больше / Меньше**

console.log(`5>5 ${5>5}`); *-false*

Не могут быть равны в данном операторе

**=> Больше / <= Меньше равно**

console.log(`5<=5 ${5<=5}`); *-true*

Значения более/менее и равно

**&& Логичное И**

Чаще дает *false.* Все значения должны быть *true,* что бы получить *true*. Если есть хоть один *false* , то *true* не выйдет

true && true = true

false && true = false

true && false = false

false && false = false

**|| Или**

Дает чаще *true*. Все значения должны быть *false,* что бы получить *false*.

true || false = true

false || true = true

true || true = true

false || false = false

**If, Else**

Помогает проверить правильность кода. Cоздает блок кода, который будет выполнен при выполнии условия кода

let thisYear = 2024;

Из-за 6>8 (false) не работает, тк если есть false, суровое равно не срабатывает

**if**(thisYear **===** 2024 **&&** 6>8){

    alert('Welcome');

}

Если условие === не окажется true, то alert не появится на экране.

**Сокращенное письмо If, Else:**

maxValue = firstNumber**>**secondNumber **?** firstNumber **:** secondNumber;

**?** заменяет If , **:** заменяетElse

**If, Else If, Else**

let userNumber = Number(prompt('Choose number'));

**if**(userNumber == 0 || 19){

Если выбрать 0 или 19 – сработает только 1 код

console.log(`User chose ${userNumber}`);

}

**else if**(userNumber>10){

Если выбрать 11 – сработает только 2 код

console.log(`User chose ${userNumber}`);

}

**else**{

console.log(`User chose ${userNumber} or something else`);

}

Если выбрать 5 – сработает только 3 код

*Пример:*

let checkUserAge = Number(prompt('Type user age'));

let controlAge = checkUserAge>18 **?** 'Yes' **:** 'No';

console.log(controlAge);

if(controlAge == 'Yes'){

    document.write(**`**<h1>Hello</h1>**`**);

}

else{

    document.write(**`**<h1>Goodbye!</h1>**`**);

}