

HTML - язык гипертекстовой разметки + CSS - каскадные таблицы стилей



ПЛАН ЗАНЯТИЯ по HTML

1. Как устроен сайт и веб-разработка
2. Что такое **HTML**, теги и их атрибуты, синтаксис
3. Основные элементы html (! + **tab**) страницы: **head, body, script, style, meta**
4. Заголовки, работа с текстом(**h1 - h6, p, span**)
5. Изображения и ссылки, кнопка: **img (src, alt) , a(target, href), button**
6. Семантические элементы html: **header, footer, nav, section, article**
7. Работа со списками (**ul, ol, li, list-style**)
8. Работа с таблицами (**table, tr, td/th, border-collapse, colspan и rowspan**)
9. Accessibility (**a11y**, доступность страницы)
10. Теги стилизации: **, , <small>**
11. Теги **pre, br**
12. Тер **<div>**
13. **Комментарии** в html файле

ПЛАН ЗАНЯТИЯ по CSS. Часть 1

1. Что такое **CSS**
2. Подключение стилей к файлу **html** (`<link rel="stylesheet" src="">`)
3. Варианты работы со стилями в html документе (**inline стили**, `<style>`)
4. Основные селекторы (**tag, class, id, data attribute**) - название классов и id
5. Обнуление дефолтных стилей браузера: *user agent stylesheet*
6. Работы с корневыми элементами страницы: `*`, `<html>`, `<body>`
7. Инструменты разработчика (как открыть и как пользоваться)
8. Приоритет стилей (спецификация селекторов, работа с несколькими классами)
9. Цвета в css: **rgb, hex**
10. Работа со шрифтами, подключение на страницу
11. Высота строки: **line-height, vertical align**

ПЛАН ЗАНЯТИЯ по CSS. Часть 2

1. Блочные и строчные элементы, свойства **display**: (**none, block, inline, inline-block, flex, grid**)
2. **Border** - основные свойства и варианты стилизации
3. **Padding и margin** - основные различия и синтаксис
4. **Box-sizing** (**border-box, content-box, initial**) - основное значение
5. Position: **static, absolute, relative, sticky, fixed**
6. Псевдоклассы: **hover, focus, active, :nth-child, :first-child, :last-child**
7. Псевдоэлементы: **::before, ::after, ::first-line, ::placeholder**
8. **Z-index** - наложение элементов друг на друга
9. Абсолютные и относительные величины в CSS: **%, vh, vw, em, rem**
10. **Object-fit, Box-shadow, visibility, float, box-shadow**
11. Анимация: **Transition и transform**

ПЛАН ЗАНЯТИЯ по CSS. Flexbox

1. Display-flex: основные преимущества
2. Justify-content
3. Align-items
4. Flex-direction
5. Flex-wrap
6. Flex-flow: flex-direction + flex-wrap
7. Align-content
8. Gap: row-gap, column-gap
9. Order
10. Align-self
11. Flex: flex-grow, flex-shrink, flex-basis

ПЛАН ЗАНЯТИЯ. Часть 5. Форма

1. Функция формы на странице
2. Методы get и post (Запросы на сервер)
3. Основные элементы формы: button, input, textarea, select
4. Атрибут input'a формы (его значение и функция): value, name, id, type
5. Radio button и checkbox

HTML - Hyper Text Markup Language

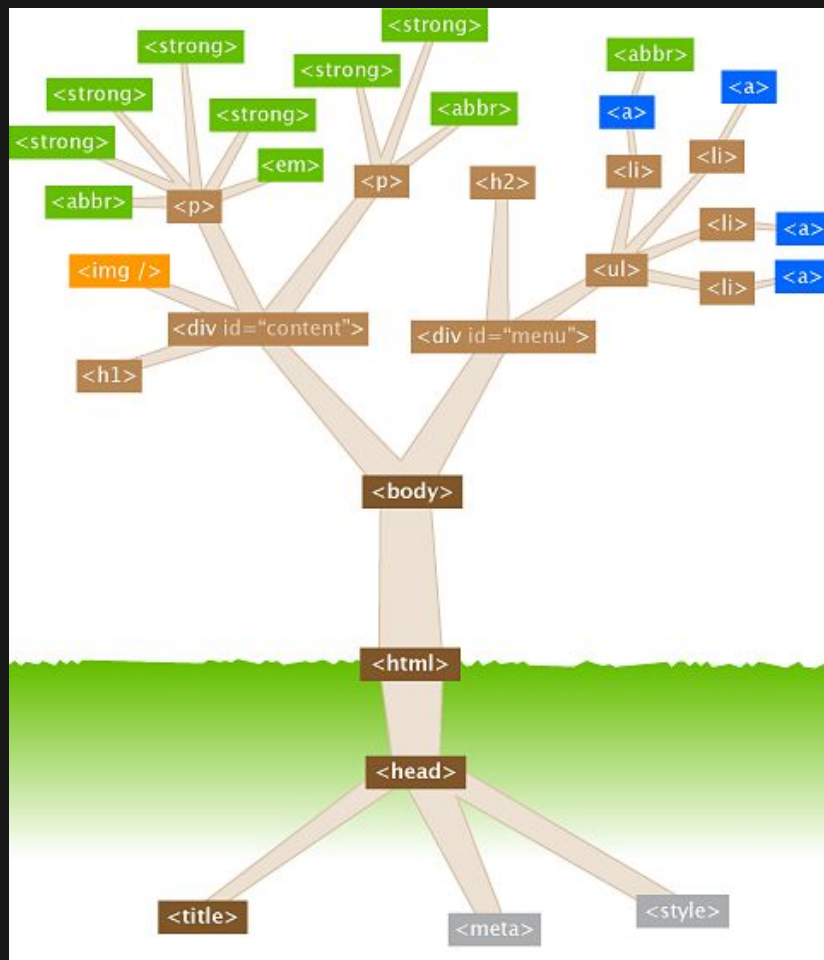
Это язык разметки, который мы используем для визуального и смыслового структурирования нашего web контента, например, определяем параграфы, заголовки, таблицы данных или вставляем изображения и видео на страницу.

HTML структура файла

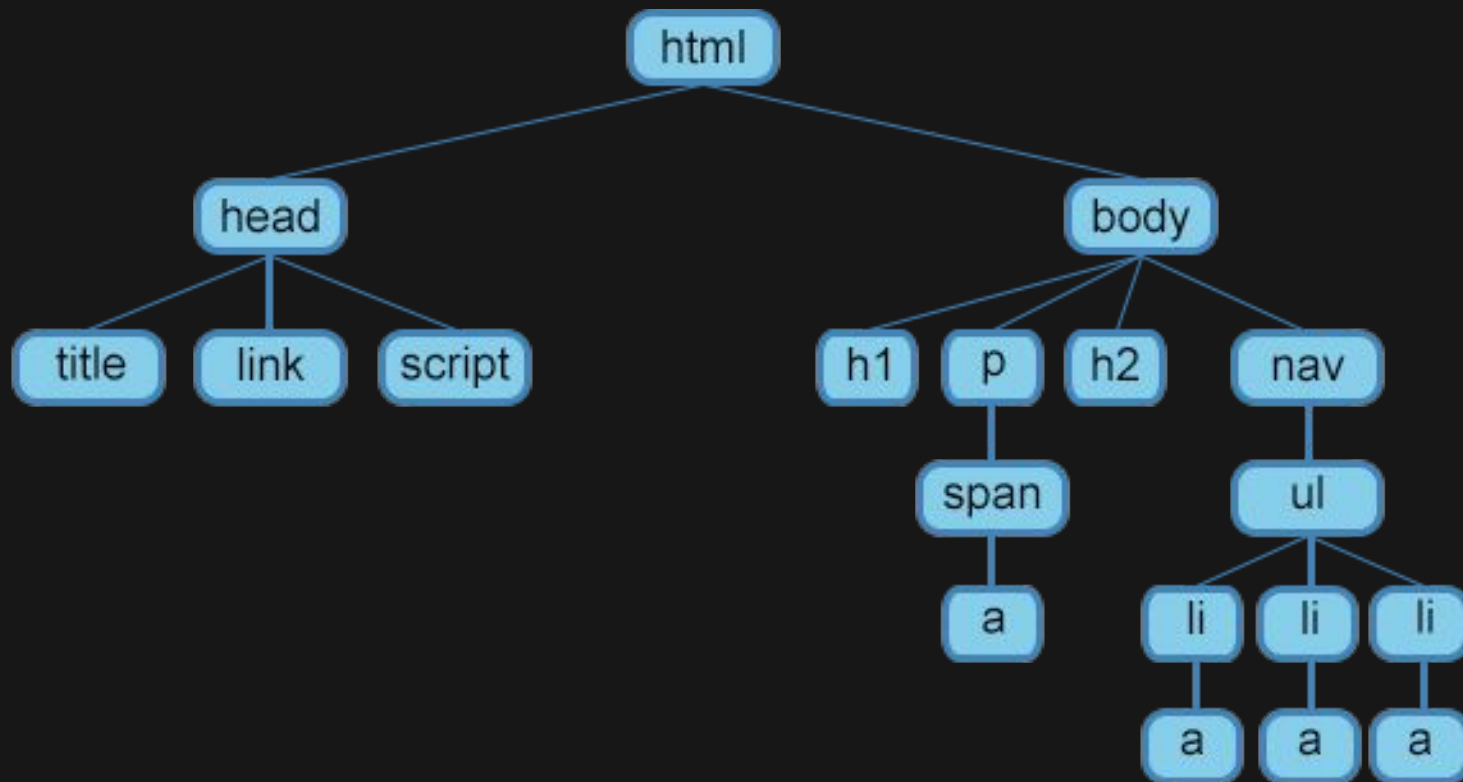


- HTML-документ организован в виде дерева элементов и текстовых узлов. Модель визуального форматирования CSS представляет собой алгоритм, который обрабатывает HTML-документ и выводит его на экран устройства.
- Каждый блок в дереве представляет соответствующий элемент или псевдоэлемент, а текст (буквы, цифры, пробелы), находящийся между открывающим и закрывающим тегами, представляет содержимое текстовых узлов.

Структура HTML документа



Структура HTML документа



Как CSS интерпретирует документ



- Чтобы создать дерево блоков, CSS сначала использует каскадирование и наследование, позволяющие назначить вычисленное значение для каждого CSS-свойства каждому элементу и текстовому узлу в исходном дереве.
- Затем для каждого элемента CSS генерирует ноль или более блоков в соответствии со значением свойства `display` этого элемента. Как правило, элемент генерирует один основной блок, который представляет самого себя и содержит свое содержимое.

Как CSS интерпретирует документ



Положение блоков на странице определяется следующими факторами:

- размером элемента (с учётом того, заданы они явно или нет);
- типом элемента (строчный или блочный);
- схемой позиционирования (нормальный поток, позиционированные или плавающие элементы);
- отношениями между элементами в DOM (родительский — дочерний элемент);
- внутренними размерами содержащихся изображений;
- внешней информацией (например, размеры окна браузера).

Блочные и строчные элементы в html

С помощью блочных элементов можно создавать структуру веб-страницы, строчные элементы используются для форматирования текстовых фрагментов.

Блочные элементы

— элементы высшего уровня, которые форматируются визуально как блоки, располагаясь на странице в окне браузера вертикально. Значения свойства `display`, такие как `block`, `list-item` и `table` делают элементы блочными. Блочные элементы генерируют основной блок, который содержит только блок элемента.

Наиболее популярные элементы:

`<article>`, `<aside>`, `<div>`, `<footer>`, `<form>`,
`<h1>-<h6>`, `<header>```, `<nav>````<p>`, `<pre>``<section>``<table>`,
``

Встроенные (строчные) элементы

- генерируют внутристрочные контейнеры. Они не формируют новые блоки контента. Значения свойства `display`, такие как `inline` и `inline-table` делают элементы строчными.
- могут содержать только данные и другие строчные элементы. Исключение составляет элемент `<a>`, который может обрачивать целые абзацы, списки, таблицы, заголовки и целые разделы при условии, что они не содержат другие интерактивные элементы — другие ссылки и кнопки.

Наиболее популярные теги:

`<a>`, `<code>`, ``, `<i>`, `<iframe>`, ``, `<label>`, ``, ``,
`<sub>`, `<sup>`,



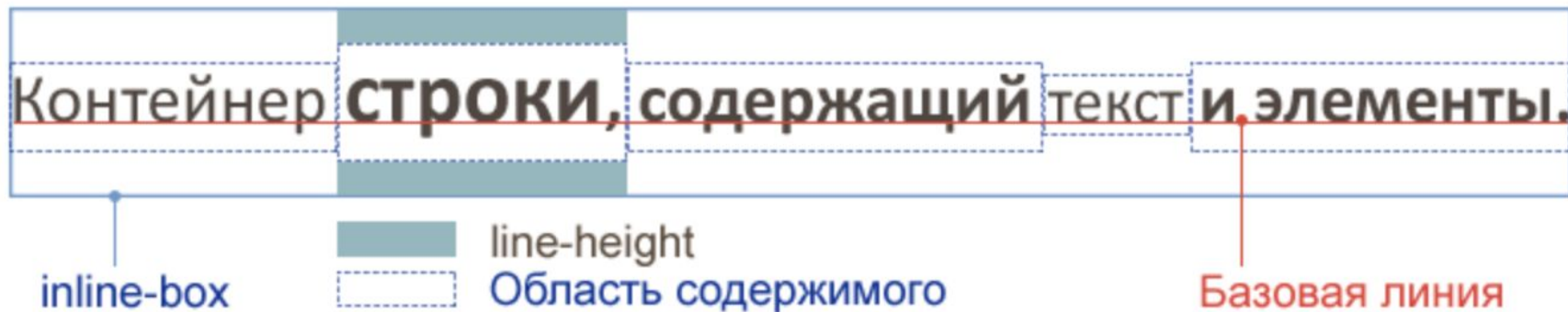
Строчно-блочные элементы

Существует еще одна группа элементов, которые браузер обрабатывает как строчно-блочные `{display: inline-block;}`. Такие элементы являются встроенным, но для них можно задавать поля, отступы, ширину и высоту.

Наиболее популярные элементы:

`<audio>`, `<button>`, `<canvas>`, `<input>`, `<progress>`, `<select>`, `<textarea>`, `<video>`.

Line-height



Полезные ссылки:

- <https://htmlbook.ru/>
- <https://learn.javascript.ru/>
- <https://flatuicolors.com/>
- https://www.w3schools.com/css/css_border.asp
- <https://developer.mozilla.org/en-US/docs/Web/CSS/border>
- <https://css-tricks.com/different-ways-to-get-css-gradient-shadows/>
- <https://2023.stateofcss.com/en-US>



ПЛАН СЕГОДНЯШНЕГО ЗАНЯТИЯ 04.09.2023

1. Относительные и абсолютные величины (vw, vh, %, em/rem),
продолжение
2. Z-index
3. Visibility: hidden vs display: none
4. Object-fit
5. Box-shadow
6. Animation: **transition + transform**
7. Псевдоэлементы ::before ::after
8. **Form**
9. Float



Относительные величины:

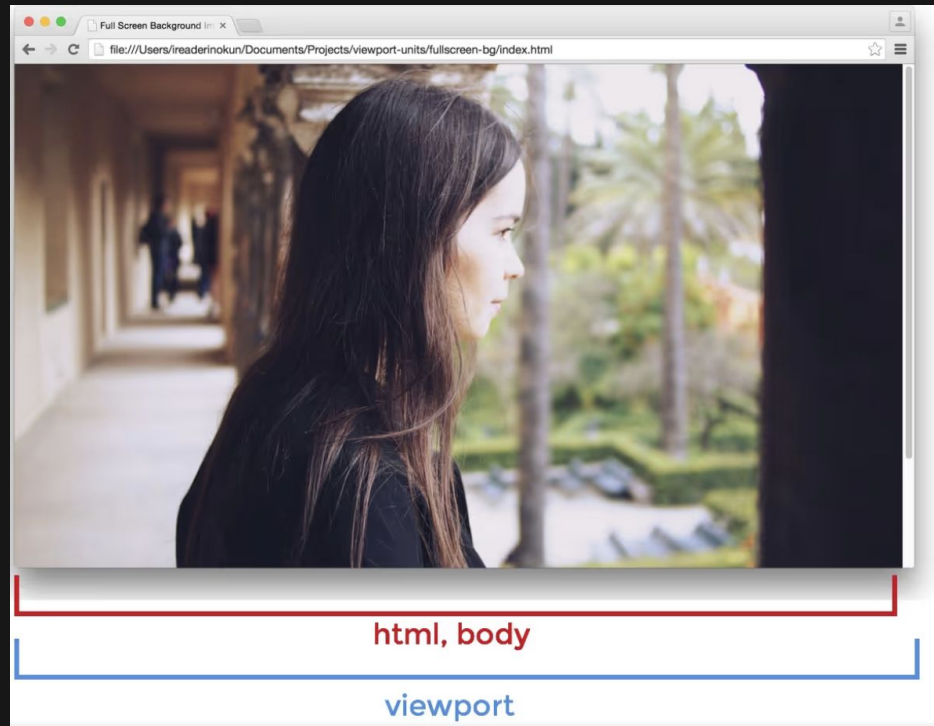
vh или %, vmin, vmax

- Относительные единицы измерения не имеют объективного измерения.
- Вместо этого их фактический размер определяется размером родительского элемента. Это означает, что их размер может быть изменен путем изменения размера этого зависимого элемента.
- В большинстве браузеров дефолтное значение размера шрифта для html и body тегов - 100%.
- Можно сделать следующее сравнение - **$100\% = 1em = 1rem = 16px = 12pt$**
- **vmin** - 1/100th от viewport's наименьшего измерения (height or width)
- **vmax** Relative 1/100th от viewport's наибольшего измерения (height or width)

Относительные величины:

vw или %

- При работе с шириной больше подходит единица измерения % браузеры рассчитывают размер области просмотра (viewport) как окна браузера, которое включает в себя пространство для полосы прокрутки (scrollbar).
- viewport > html > body
- Следовательно, если задать элементу значение 100vw, элемент будет выходить за пределы html-элементов и body-элементов



Относительные величины:

vh или %



- Когда мы хотим сделать элемент на всю высоту экрана удобнее использовать vh , чем проценты
- Поскольку размер элемента, определенный в процентах, определяется его родительским элементом, мы можем заставить элемент заполнять всю высоту экрана только в том случае, если родительский элемент также заполняет всю высоту экрана. Обычно это означает, что мы должны расположить элемент как фиксированный

Относительные величины:

em, rem, %



- Единицы `vh` и `vw` идеально подходят для адаптивного дизайна, поскольку они полностью независимы от базового размера шрифта.
- Хотя это кажется отличной единицей для адаптивного дизайна, у него есть свои недостатки. Они не дают достаточного контроля над размером шрифта, который почти всегда получается слишком большим или слишком маленьким.

Альтернативой может быть:

- `em` - относительная единица родительского размера шрифта (Relative to the parent element's font size)
- `rem` - относительная единица по отношению к корневому элементу. (Relative (root em) Relative to the html font size)
- Когда вы используете единицы измерения `em` для элемента, вы должны учитывать размер шрифта всех родительских элементов. Как вы можете видеть, это может довольно быстро усложниться и запутаться.
- Решением этой проблемы является `rem`. `rem` вычисляется только на основе размера шрифта `html`, а не родительского элемента

Относительные величины: em, rem, %

```
html {  
  font-size: 100% /* =16px */  
}  
body {  
  font-size: 2em; /* =32px */  
}  
p {  
  font-size: 1em; /* =32px */  
  /* font-size: 0.5em; =16px */  
}
```


Z-index



- Z-index используется для стабилизации порядка элементов, которые перекрываются.
- Указывает на уровень элемента в стеке. (дефолтное значение по умолчанию: auto)

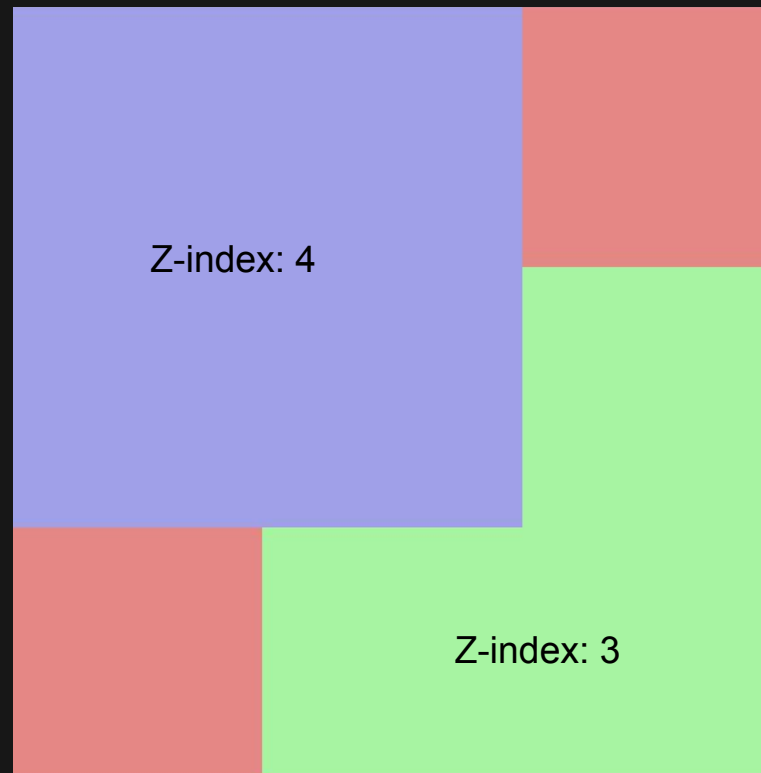
Используется число, пример: `z-index: 1;`

- Регулирует вертикальный порядок перекрытия элементов(в глубину)
- А сам z-index определяет, какой элемент будет располагаться выше остальных.
- Если у элемента значение z-index: -1, он будет находится под всеми остальными элементами

ВАЖНО:

- Работает только с элементами у которых задана position! absolute, relative, fixed или sticky и с элементами, которые являются потомками display: flex
- Если два компонента перекрывают друг друга без z-index, то элемент расположенный ниже в html документе будет показан сверху

```
.first-container {  
  position: relative;  
  width: 300px;  
  height: 300px;  
  background: ■ rgb(244, 129, 129);  
}  
.first-child {  
  z-index: 4;  
  position: absolute;  
  width: 200px;  
  height: 200px;  
  background: ■ rgb(160, 160, 238);  
}  
.second-child {  
  z-index: 3;  
  position: absolute;  
  bottom: 0;  
  right: 0;  
  width: 200px;  
  height: 200px;  
  background: ■ rgb(141, 247, 154);  
}
```



Важно, что z-index у фиолетового контейнера больше

Visibility: hidden vs display: none

`visibility: hidden:`

- Элемент остается в макете, занимая место, как если бы он был виден. Однако он становится полностью невидимым, включая его содержимое, и не взаимодействует с пользователем (например, вы не можете нажать на него или взаимодействовать с ним)
- Занимает! место в разметке, что может повлиять на общую верстку и окружающие элементы
- Обычно используется, когда надо скрыть и отобразить элемент без изменения макета. Например, для всплывающих подсказок или выпадающих меню, которые появляются и исчезают, не затрагивая близлежащее содержимое.

`display: none:`

- Полностью удаляется из документа, не оставляет места, не влияет на разметку
- При изменении элемента, окружающие элементы будут занимать места
- Минус: элемент не воспринимается парсером, полностью

Object-fit

свойство CSS object-fit используется для указания того, как следует изменять размер `` или `<video>`, чтобы они соответствовали своему контейнеру.

- `fill` - это значение по умолчанию. Размер изображения изменяется таким образом, чтобы оно соответствовало заданному размеру. При необходимости изображение будет растянуто или сжато, чтобы соответствовать
- `contain` - Изображение сохраняет свое соотношение сторон, но изменяется размер, чтобы соответствовать заданному размеру
- `cover` - Изображение сохраняет свое соотношение сторон и заполняет заданный размер. Изображение будет обрезано так, чтобы оно не соответствовало размеру
- `none` - не изменяется, вставляется просто кусок картинки
- `scale-down` - изображение уменьшено до наименьшей версии `none` или содержит от `none` или `contain`

Float

The float property is used for positioning and formatting content e.g. let an image float left to the text in a container.

- left
 - right -
 - none - default
-
- When we use the float property, and we want the next element below (not on right or left), we will have to use the clear property.
 - The clear property specifies what should happen with the element that is next to a floating element.

Pseudo-elements. Псевдоэлементы

::before, ::after, ::first-line, ::placeholder

1. The ::before pseudo-element для того чтобы вставить какой-то контент перед элементом

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

2. The ::selection pseudo-element matches the portion of an element that is selected by a user

3. ::first-line для первой линии

```
p::first-line {  
  color:  #ff0000;  
  font-variant: small-caps;  
}
```

Transition

Основные свойства анимации:

1. transition
2. transition-delay
3. transition-duration
4. transition-property
5. transition-timing-function

```
div {  
  transition: width 2s, height 4s;  
}
```

Чтобы создать анимацию надо указать:

- css свойство на котором будет проходить анимация
- длительность эффекта

Transition. Тип анимации

The transition-timing-function property specifies the speed curve of the transition effect.

The transition-timing-function property can have the following values:

- ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
- linear - specifies a transition effect with the same speed from start to end
- ease-in - specifies a transition effect with a slow start
- ease-out - specifies a transition effect with a slow end
- ease-in-out - specifies a transition effect with a slow start and end
- cubic-bezier(n,n,n,n) - lets you define your own values in a cubic-bezier function

The transition-delay property specifies a delay (in seconds) for the transition effect.

The following example has a 1 second delay before starting:

Transition. Shorthand

```
div {  
  transition-property: width;  
  transition-duration: 2s;  
  transition-timing-function: linear;  
  transition-delay: 1s;  
}
```

```
div {  
  transition: width 2s linear 1s;  
}
```

Transform

1. CSS transforms allow you to move, rotate, scale, and skew elements

With the CSS transform property you can use the following 2D transformation methods:

- `translate()` moves an element from its current position (according to the parameters given for the X-axis and the Y-axis). transform: `translate(50px, 100px);`
- `rotate()` rotates an element clockwise or counter-clockwise according to a given degree transform: `rotate(20deg);`
transform: `rotate(-20deg);`
- `scale()` increases or decreases the size of an element (according to the parameters given for the width and height). transform: `scale(2, 3);` `scaleY()`-height `scaleX()` -width
- `skewX()` method skews an element along the X-axis by the given angle. transform: `skewX(20deg);`
- `skewY()` same for Y axis
- `skew()` for X and Y If the second parameter is not specified, it has a zero value. So, the following example skews the `<div>` element 20 degrees along the X-axis:
- `matrix()` The `matrix()` method combines all the 2D transform methods into one.
- The `matrix()` method takes six parameters, containing mathematic functions, which allows you to rotate, scale, move (translate), and skew elements.
- The parameters are as follows: `matrix(scaleX(), skewY(), skewX(), scaleY(), translateX(), translateY())` transform: `matrix(1, -0.3, 0, 1, 0, 0);`

Form. Работа с формами

1. **Используется для сбора данных пользователя**


2. An HTML form is used to collect user input. The user input is most often sent to a server for processing.

3. Элементы формы представляют собой различные типы входных элементов, такие как текстовые поля, флажки, переключатели, кнопки отправки и многое другое

4. **name - уникальное** Notice that each input field must have a `name` attribute to be submitted. If the `name` attribute is omitted, the value of the input field will not be sent at all.

5. **value, type, id, required**

6. **Результат отправки:**

 127.0.0.1:5501/?name=Jean+&group=Doe

```
<form>
  <label for="name">Name</label>
  <input type="text" id="name" name="name" required />
  <input type="text" name="group" required />
  <input type="submit" value="submit" />
</form>
```

Form. Input

Элемент `<input>` является наиболее важным элементом формы.

`<input>`элемент может отображаться несколькими способами в зависимости от атрибута **Type** .

```
<form>
  <input type="text" />
  <input type="radio" />
  <input type="submit" />
  <input type="checkbox" />
  <input type="button" />
</form>
```

Form. Label

1. Notice the use of the `<label>` element in the example above.
2. The `<label>` tag defines a label for many form elements.
3. The `<label>` element is useful for screen-reader users, because the screen-reader will read out loud the label when the user focuses on the input element.
4. The `<label>` element also helps users who have difficulty clicking on very small regions (such as radio buttons or checkboxes) - because when the user clicks the text within the `<label>` element, it toggles the radio button/checkbox.
5. The `for` attribute of the `<label>` tag should be equal to the `id` attribute of the `<input>` element to bind them together.

```
<label for="name">Name</label>
<input type="text" id="name" name="name" required />
```

Form. Checkbox

```
<form>
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike" />
  <label for="vehicle1"> I have a bike</label><br />
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car" />
  <label for="vehicle2"> I have a car</label><br />
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat" />
  <label for="vehicle3"> I have a boat</label>
</form>
```

UI:

- ☐ I have a bike
- ☐ I have a car
- ☐ I have a boat

The HTML `<form>` Elements

The HTML `<form>` element can contain one or more of the following form elements:

- `<input>`
- `<label>`
- `<select>`
- `<textarea>`
- `<button>`
- `<fieldset>`
- `<legend>`
- `<datalist>`
- `<output>`
- `<option>`
- `<optgroup>`

POSITION:

- absolute выхватывается из общего потока элементов другие элементы его не видят, они занимают пустое место
- relative выхватывается из своей позиции/ по отношению к своему месту в общем порядке другие элементы стоят на своих местах
- sticky = fixed + relative



FrontEnd / BackEnd

FrontEnd отвечает за визуализацию и отображение и функционирование сайта на стороне клиента/пользователя, работаю с отображаемыми компонентами

BackEnd - внутренние процессы, базы данных, обработка данных на серверах

Функционируют через API - определенные правила, которые позволяют взаимодействовать фронтенду и бекенду в частности(а также между сайтам и различными приложениями)

Устройство сайта

Цель браузера - прочитать HTML документ и отобразить его корректно. Браузер не отображает теги, но интерпретирует их и отображает согласно определенным правилам. Внутри браузера есть механизм который переводит написанный код в непосредственно в сайт в изображения

Из чего состоит сайт

HTML - Hyper Text Markup Language

структура(скелет) сайта. это язык разметки, который мы используем для визуального и смыслового структурирования нашего web контента

CSS - Cascading Style Sheets

визуальная составляющая (цветовая палитра, дизайн элементов на странице, шрифты, анимация)

JavaScript

позволяет вам создать динамически обновляемый контент, управляет мультимедиа, анимирует изображения

HTML

HTML — это язык **гипертекстовой** разметки текста. Он нужен, чтобы размещать на веб-странице элементы: текст, картинки, таблицы и видео.

Гипертекст - текст с управляющими элементами языка разметки гипертекста. Текст в котором находятся ссылки на другие документы или переходы внутри исходного документа.

Тэг - элемент разметки языка. Теги бывают закрывающиеся и открытые.

Элемент - то, что находится внутри тега

Атрибут - дополнительные значения, которые настраивают элементы или регулируют их поведение различным способом, чтобы соответствовать критериям пользователей.

***Программу на языке программирования можно писать в любом текстовом редакторе, но есть более удобные программы - редакторы коды - которые упрощают разработку.**

Структура HTML документа

<!DOCTYPE html> - объявляет тип документа и объясняет браузеру, в какой версии языка разметки он сверстан

<head> - Содержит машиночитаемую информацию (metadata) о документе, например его заголовок, скрипты и страницы стилей. В метатеггах хранится служебная информация

Примеры тегов в head : <title> <style> <link> <script>

<body> - все что отображается на странице

Типы тегов

- форматирование
- верстка таблиц
- верстка списков
- форматирование ссылок/гиперссылок
- вставка изображений

Атрибуты тегов

- Все HTML теги могут иметь атрибуты
- Атрибуты предоставляют дополнительную информацию о элементе
- Атрибуты всегда! ставятся в начале тега
- Атрибуты чаще всего идут парой ключ/значение name="value"

Примеры:

```

```

```
<p style="color: blue"> Параграф для редактирования </p>
```

Теги для работы с текстом

Заголовки

`<h1> </h1>`

...

`<h6> </h6>`

Для больших абзацев

`<p></p>`

Inline/Block Элементы

` <div>`

Теги для форматирования исходного текста

- `` - Bold text
- `` - Important text
- `<i>` - Italic text
- `` - Emphasized text
- `<mark>` - Marked text
- `<small>` - Smaller text
- `` - Deleted text
- `<ins>` - Inserted text
- `<sub>` - Subscript text
- `<sup>` - Superscript text

Комментарии

Синтаксис:

```
<!-- Write your comments here -->
```

Для чего нужны:

- Коммуникация с другими разработчиками
- Понимание кода
- Дебаг Debug

Ссылки и навигация

```
<a href="https://www.w3schools.com">This is a link</a>
```

Гиперссылка, которая позволяет перейти в другой документ, на другой сайт.

Самый важный атрибут href который указывает куда переходит ссылка

Не посещенная ссылка - синяя, подчеркнутая,

Посещенная ссылка - фиолетовая.

Атрибут `target` позволяет выбрать, как открывать ссылку:

- `_self` - дефолтный, открывает в той же вкладке
- `_blank` - открывает в новой вкладке
- `_top` - открывает на весь экран

Работа с изображениями

```

```

Работа со списками

Два основных вида списков в html - буллеты и пронумерованные

` ` - неперенумерованный.(Unordered)

` ` - пронумерованный (Ordered)

`<ol start="50">` - нумерация начинается с 50

Таблицы

```
<table>
```

```
<tr>
```

```
<th>Company</th>
```

```
<th>Contact</th>
```

```
<th>Country</th>
```

```
</tr>
```

```
<tr>
```

```
<td>Alfreds Futterkiste</td>
```

```
<td>Maria Anders</td>
```

```
<td>Germany</td>
```

```
</tr>
```

```
</table>
```

Selectors: Classes и id

class - для нескольких элементов

```
<div class="many"> </div>
```

```
<div class="many"> </div>
```

```
<div class="many"> </div>
```

id - уникальный для одного

```
<p id="unique"> <p>
```

Семантические верстка и доступность

Семантическая верстка помогает определить смысловое предназначение каждого блока и логическую структуру документа.

Для незрячих или частично незрячих всё сложнее. Основным инструментом для просмотра сайтов не браузер, который отрисовывает страницу, а скринридер, который читает текст со страницы вслух.

Основные элементы семантической верстки:

- `<article>`
- `<aside>`
- `<details>`
- `<footer>`
- `<header>`
- `<main>`
- `<mark>`
- `<nav>`
- `<section>`
- `<summary>`
- `<time>`

CSS верстка проекта

Верстка в документе html:

```
<style>

.city {

    background-color: tomato;

    color: white;

    border: 2px solid black;

}

</style>
```

Инлайн верстка:

```
<p style="background-color: tomato"> Томатный супчик </p>
```


JavaScript - добавляем интерактив в проект

```
<script>
```

```
document.getElementById("demo").innerHTML = "Hello JavaScript!";
```

```
</script>
```