Новые возможности html5/css3

Перед тем как мы начнём, давайте вспомним что ни в html ни в css изначально не было возможности добавить интерактивные возможности для контента, даже эффекты наведения добавлялись с помощью языка программирования javascript, но технологии не стоят на месте и html5, css3 сильно шагнули вперед и теперь обладают достаточно серьезным функционалом, который включает в себя даже возможность добавить интерактивн на странице.

Именно эти возможности мы сейчас с вами и рассмотрим

details

Виджет раскрытия обычно представлен на экране с использованием небольшого треугольника, который поворачивается, чтобы показать состояние открытия / закрытия, с меткой рядом с треугольником. Если первый дочерний элемент элемента **<details>** является **<summary>**, содержимое элемента <summary> используется в качестве метки для виджета раскрытия.

Категории контента:

Потоковое содержимое, корневое секционное содержимое, интерактивное содержимое, видимое содержимое.

Контекст, в котором этот элемент может быть использован:

Где ожидается потоковое содержимое.

Пропуск тегов:

Ни один из тегов не может быть пропущен.

Для элемента доступны глобальные атрибуты, а также атрибут **open**, который делает дополнительную информацию видимой при загрузке страницы.

Пример

Конечно теоретической информации может быть очень много, но очень сложно понять как всё устроено, без практики, именно на ней мы и остановимся подробнее

Первым делом давайте рассмотрим самый простой пример

```
<details>Lorem ipsum dolor sit amet</details>
```

В результате мы не видим текст, мы видим треугольник, который указывает как выпадающий список и текст “Сведения” который формируется нашим браузером и сам текст зависит от языка выбранного в вашей системе.

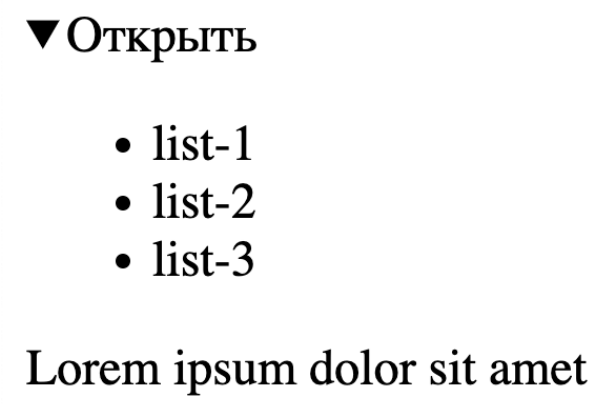
Первое что мы можем исправить, это поменять текст “Сведения” на любой другой контент, для этого используется тег summary

```
<details>
    <summary>Открыть</summary>
    Lorem ipsum dolor sit amet
</details>
```

При добавлении details лучше всегда использовать summary.

Содержимое внутри тега details может быть абсолютно любым, здесь может быть и список и параграф и любые другие теги

```
<details>
    <summary>Открыть</summary>
    <ul>
        <li>list-1</li>
        <li>list-2</li>
        <li>list-3</li>
    </ul>
    <p>Lorem ipsum dolor sit amet</p>
</details>
```

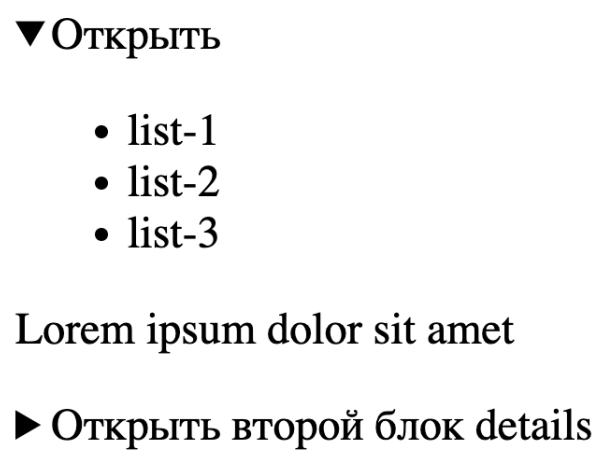


При нажатии на текст **открыть** мы увидим все элементы, которые находятся внутри тега **details**

Атрибут open

Как мы успели заметить, элемент details всегда закрыт, но что если нам нужно чтобы он был изначально открыт, для таких ситуаций используется атрибут open. На этом особенности атрибута не заканчиваются, в итоге вы можете поменять значение summary или

```
<details open>
   <summary>Открыть</summary>
   <ul>
       <li>list-1</li>
       <li>list-2</li>
       <li>list-3</li>
   </ul>
   <p>Lorem ipsum dolor sit amet</p>
</details>
<details close>
   <summary>Открыть второй блок details</summary>
   <ul>
       <li>list-1</li>
       <li>list-2</li>
       <li>list-3</li>
   </ul>
   <p>Lorem ipsum dolor sit amet</p>
</details>
```



В результате работы программы мы можем заметить что первый элемент **details** открыт, так как мы ему задали атрибут **open**, а второй закрыт, так как у него выставлен атрибут **close**, мы можем не выставлять атрибут close он идёт у элемента details по определению

Стилизация details

Очень частой задачей является поменять внешний вид треугольника или вообще поменять стили для тега summary при открытии блока. Давайте рассмотрим как мы можем стилизовать элемент details

Следуя более новой спецификации, Firefox отображает элемент summary как **display: list-item** и маркер можно стилизовать так же, как и элементы списка.

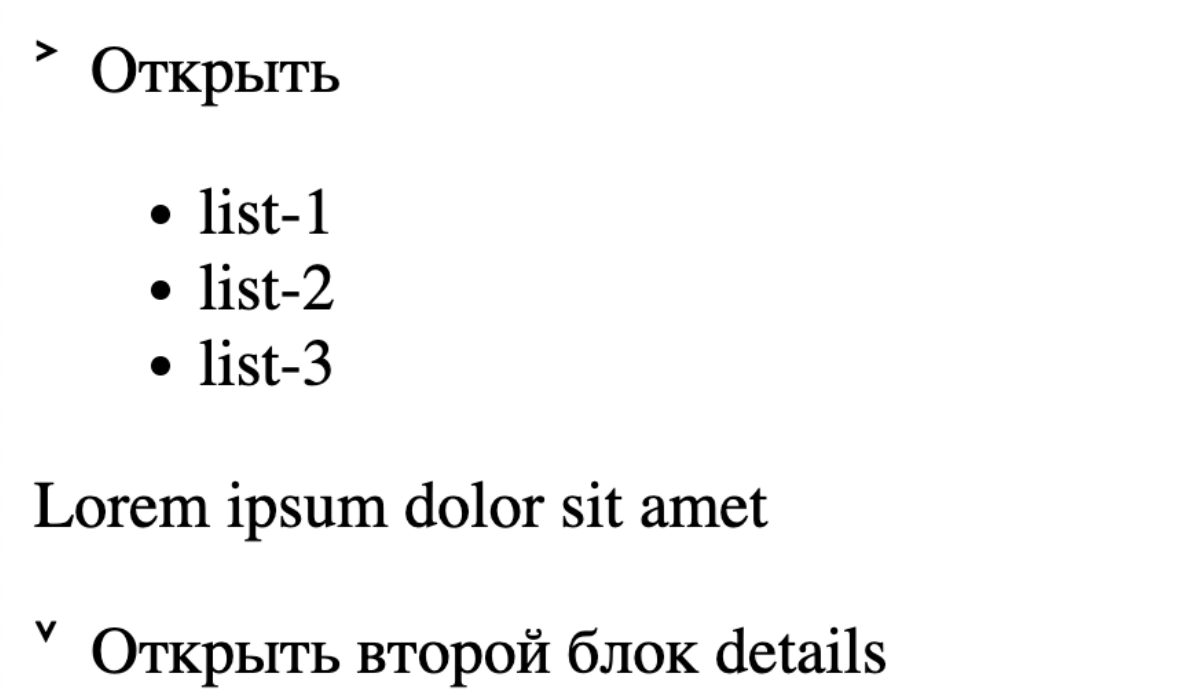
Следуя более старой спецификации, в Chrome and Safari существует пользовательский псевдо-элемент **::-webkit-details-marker**, с помощью которого можно стилизовать маркер.

Для кроссбраузерной стилизации маркера, чтобы скрыть дефолтный и добавить кастомный, необходимо для Firefox установить элементу summary {display: block;}, а для Chrome и Safari's установить ::-webkit-details-marker {display: none;}. После этого дефолтный маркер будет скрыт, после чего можно установить свой маркер любым доступным способом стилизации. В примере ниже с помощью псевдоэлементов маркер возвращается обратно.

Из примера выше, возьмем html и добавим к нему css

```
summary {
   display: block;
}

summary::-webkit-details-marker {
```

```
    display: none;
}

summary::before {
    content: '\02C5';
    padding-right: 0.5em;
}

details[open]>summary::before {
    content: '\02C3';
}
```



Как мы можем заметить, стандартные стрелки удалось заменить на любой другой символ, в данной ситуациии на острые стрелки

Важные моменты

1. Мы использовали псевдоэлемент **before** чтобы добавить контент в html, вы можете использовать стрелку after если вы хотите разместить ее справ от текста

2. Вы можете выбрать любой символ в таблице юникодов https://unicode-table.com/ru/02C3/ просто заменив на нужный нам css
3. В примере используете конструкция `details[open]` что означает как только блок открыт, применить данные стили

Бургер меню на чистом html5/css3

Одним из самых частых элементов на странице является меню сайта и если для десктопной версии чаще всего есть место для элементов меню, то в мобильной версии чаще всего его скрывают и делают бургер меню, чтобы элементы не занимали так много пространства, но возникает вопрос, а можем ли мы сделать такое бургер меню, используя только html/css. Да, можем, поэтому давайте разбираться как реализовать данную задачу.

Самой большой проблемой является добавление кликов. Мы знаем что html это контент сайта, а css стили для данной страницы, но вот клик по элементу это уже будет интератив на странице и за эту часть отвечает javascript, но есть один очень интересный элемент, это `<input type="checkbox">` чекбокс.

У элемента input type=”checkbox” есть атрибут **checked** благодаря которому мы можем менять содержимое или говоря простым языком может реализовать клик по чекбоксу.

Давайте создадим аналог элемента **details** только будем использовать **checkbox**

Работа с checkbox

```
<input id="check" type="checkbox">
<label for="check">Открыть текст</label>
<p class="text">Lorem ipsum dolor sit amet consectetur adipisicing elit.</p>
```

Что мы видим в нашей заготовке, первое это элемент **checkbox** который связан со словом “Открыть текст” с помощью тега **label** и атрибута **for**

☐ Открыть текст

Lorem ipsum dolor sit amet consectetur adipisicing elit.

Самое главное это скрыть текст внутри, для этого используем стандартные стили

```
.text {
```

```
    display: none;
}
```

Далее начинается самая интересная часть, нам необходимо сделать так, чтобы когда был выбран чекбокс, параграфу необходимо поменять значение display: none, на display: block;

```
.text {
    display: none;
}


#check:checked~.text {
    display: block;
}
```

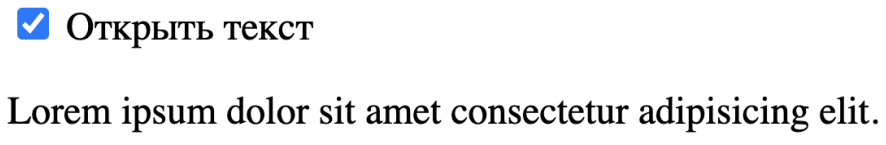
Давайте подробно разберем строку `#check:checked~.text`

1. `#check:checked` Когда элемент input выбран
2. ~ этот знак означает обращение ко всем элементам на одном уровне, в данной ситуации это **label** и **p**
3. .text это класс у параграфа, так как всегда лучше обращаться по классам

В итоге получаем вот такой результат

Когда чекбокс не выбран

Когда чекбокс выбран



Заготовка для бургер меню

Что мы можем сделать дальше? Первым делом нам не хватает стилистики для внешнего вида, давайте ее добавим.

```
<input id="check" type="checkbox">
<label for="check"><img width="24" src="bars.svg" alt="bars"></label>
<nav class="nav">
   <a class="nav__link" href="#">link-1</a>
   <a class="nav__link" href="#">link-2</a>
   <a class="nav__link" href="#">link-3</a>
</nav>
```

Добавили картинку бургера, вместо текста “Открыть” и наполнение теперь это элемент навигации, содержит в себе 3 ссылки.

```
.nav {
   background-color: #234567;
}

.nav__link {
   color: white;
   font-family: sans-serif;
   text-decoration: none;
}
```

Внешний вид не принципиален, тут вы можете выставлять любые значения

Скрываем чекбокс

Дальше нам нужно скрыть чекбокс. это можно сделать разным способом, но один из самых надежных это

```
#check {
   position: absolute;
   left: -999999px;
   visibility: hidden;
}
```

Данный способ скроет блок и разместит его далеко за экран, почему бы просто не использовать display: none; мы можем потерять самый главный функционал, это работу атрибута checked, поэтому лучше данный элемент просто спрятать.

Добавляем элемент закрывающий меню

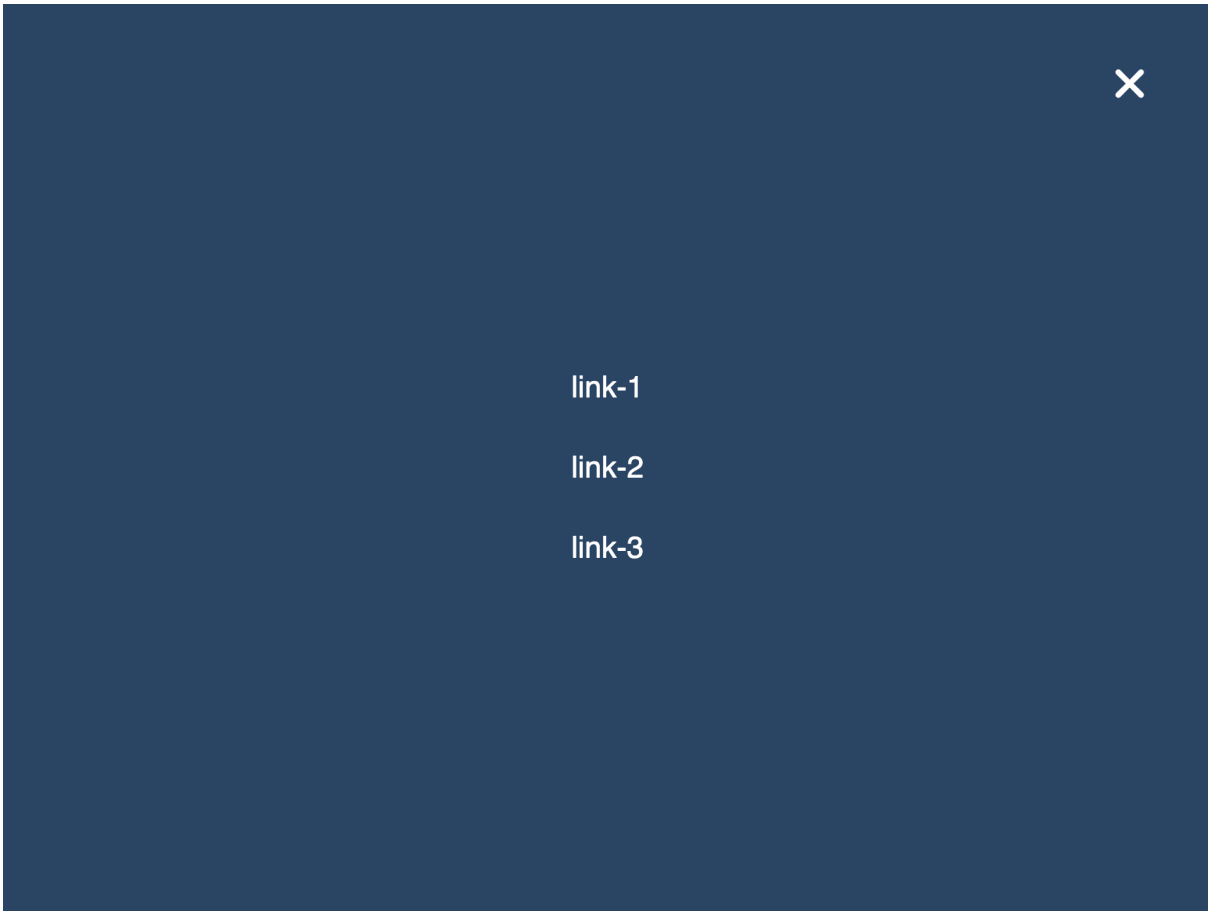
Осталось добавить крестик, который будет закрывать меню

```
<nav class="nav">
   <a class="nav__link" href="#">link-1</a>
   <a class="nav__link" href="#">link-2</a>
   <a class="nav__link" href="#">link-3</a>
   <label class="close" for="check">
       <svg width="24" height="24" xmlns="http://www.w3.org/2000/svg"
viewBox="0 0 320 512">
           <path
               d="M310.6 150.6c12.5-12.5 12.5-32.8
0-45.3s-32.8-12.5-45.3 0L160 210.7 54.6 105.4c-12.5-12.5-32.8-12.5-45.3
0s-12.5 32.8 0 45.3L114.7 256 9.4 361.4c-12.5 12.5-12.5 32.8 0
45.3s32.8 12.5 45.3 0L160 301.3 265.4 406.6c12.5 12.5 32.8 12.5 45.3
0s12.5-32.8 0-45.3L205.3 256 310.6 150.6z"
               fill="white" />
       </svg>
   </label>
</nav>
```

Плюс немного стилей

```
* {
   margin: 0;
   padding: 0;
}

body {
   position: relative;
}

.nav {
   display: none;
```

```
    position: absolute;
    width: 100%;
    height: 100vh;
    background-color: #456789;
    top: 0;
    justify-content: center;
    align-items: center;
    gap: 24px;
    flex-direction: column;
}

#check {
    position: absolute;
    left: -999999px;
    visibility: hidden;
}

#check:checked~.nav {
    display: flex;
}

.nav {
    background-color: #234567;
}

.nav__link {
    color: white;
    font-family: sans-serif;
    text-decoration: none;
}

.close {
    position: absolute;
    top: 32px;
    right: 32px;
}
```



Осталось добавить плавность появления элементов меню

```
* {
    margin: 0;
    padding: 0;
}

body {
    position: relative;
}

.nav {
    left: -100%;
    position: absolute;
    width: 100%;
```

```
    height: 100vh;
    background-color: #456789;
    top: 0;
    justify-content: center;
    align-items: center;
    gap: 24px;
    flex-direction: column;
    transition: left 0.5s;
    display: flex;
}

#check {
    position: absolute;
    left: -999999px;
    visibility: hidden;
}

#check:checked~.nav {
    left: 0;
}

.nav {
    background-color: #234567;
}

.nav__link {
    color: white;
    font-family: sans-serif;
    text-decoration: none;
```

```
}


.close {
   position: absolute;
   top: 32px;
   right: 32px;
}
```

Заключение

В данном примере мы можем заметить что мы смогли повторить функционал элемента **details** с использованием только html/css заменили событие click из javascript и смогли реализовать такой отличный пример адаптивного гамбургер меню