Настройка для любой витрины

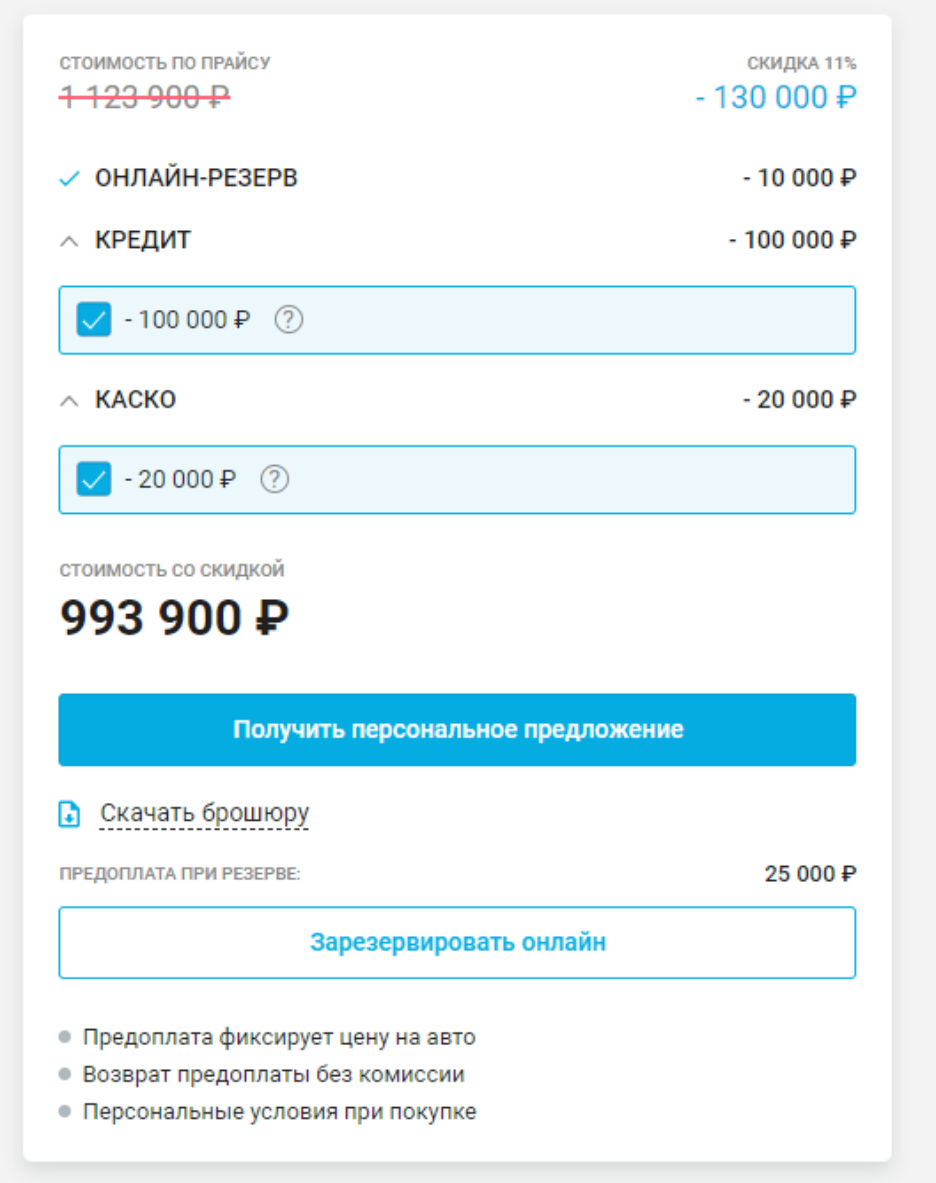
Настройка Smart Contract на любую витрину происходит через `JavaScript` код, и может быть выполнена практически любым специалистом.

В данной статье рассмотрим настройку на сайт https://avilon.ru/catalog/cars/new/

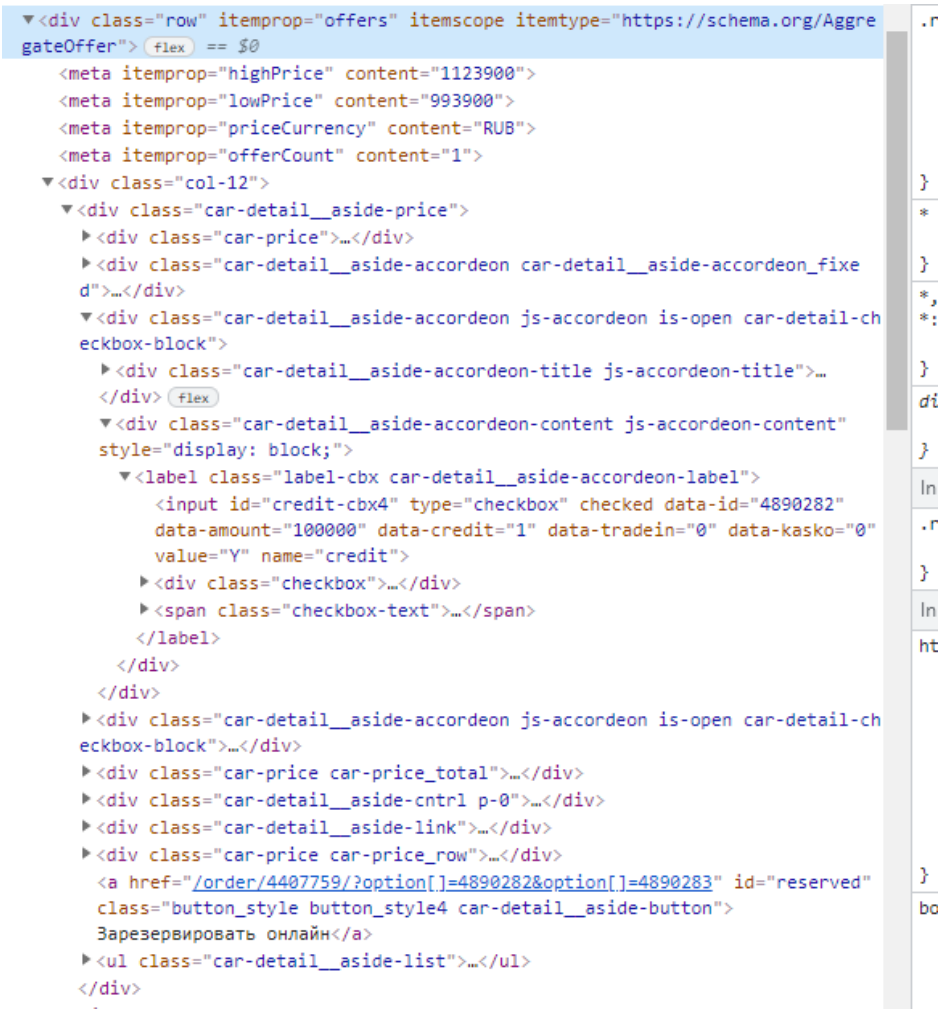
Настройки детальной страницы

Детальная страница: https://avilon.ru/catalog/cars/new/4407759/

Мы будем интегрироваться в правый блок детальной страницы



Нам необходимо проинспектировать код этого блока, для получения цен и выгод



На данной витрине не передаётся VIN номер автомобиля, поэтому мы будем использовать URL для определения идентификатора автомобиля.

WidgetId

Получаем его со вкладки виджетов. Код будет выглядеть так

```
InitSmartContract({
  widget_id: "widgetId"
})
```

Идентификатор автомобиля

Так как у нас нет информации о VIN’е, то будет привязываться к ссылке на автомобиль, для этого определим функцию для получения идентификатора автомобиля.

```
detail: {
```

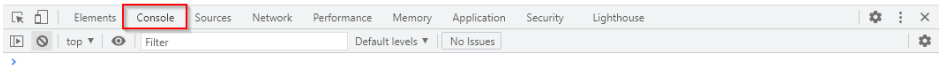
```
  identity: function() {
    return window.location.href.split('#')[0];
  }
}
```

Мы используем переменную `window.location.href`, которая вернёт нам ссылку на страницу, например `https://avilon.ru/catalog/cars/new/4407759/` и обрезаем всё, что находится после символа `#` в строке.

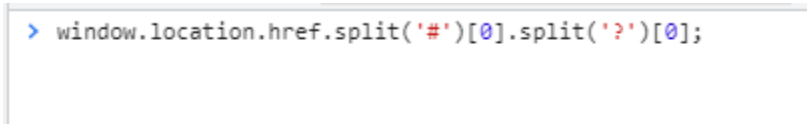
Если у нас используется ЧПУ (Как в примере выше), то правильным будет так же вырезать всё, что находится после символа `?`, т.к. параметры автомобиля зашиты в ссылку, тогда код будет выглядеть таким образом

```
detail: {
  identity: function() {
    return window.location.href.split('#')[0].split('?')[0];
  }
}
```

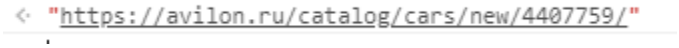
Проверить, что мы получили правильную ссылку очень просто. Так можно проверить любой код. Откроем консоль браузера (F12 для Google Chrome) и перейдём на вкладку Console



Вставим наш код в окно ввода



И нажмём клавишу `Enter` для выполнения, чтобы увидеть результат.



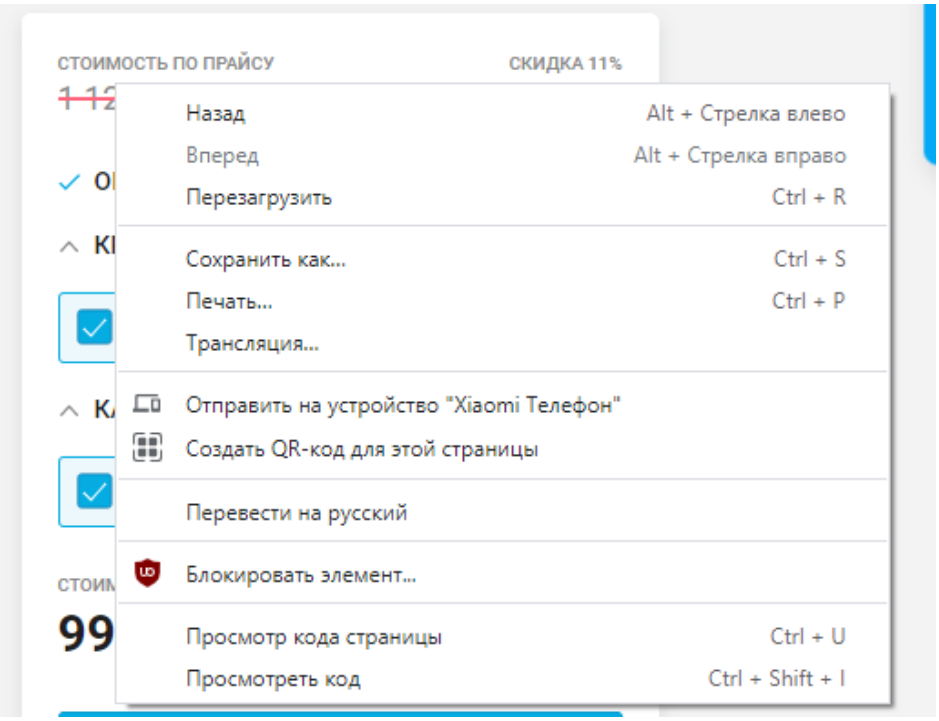
Мы получили строку со ссылкой на автомобиль. Можем попробовать выполнить указанный код, на странице с utm меткой, для этого перейдём на страницу `https://avilon.ru/catalog/cars/new/4407759/?utm_source=test` и выполним код в консоли

```
> window.location.href.split('#')[0].split('?')[0];
"https://avilon.ru/catalog/cars/new/4407759/"
>
```

Мы увидим, что параметры, не влияющие на идентификатор автомобиля, успешно удалились.

Цена автомобиля

Нам необходимо получить максимальную цену за автомобиль. Для этого найдём максимальную цену на странице и откроем её исходный код



Мы увидим примерно следующий код

Естественно, он может и будет разным в зависимости от платформы. Попробуем получить данную цену, для этого используем функцию `document.querySelector()`, которая позволит получить нам DOM элемент.

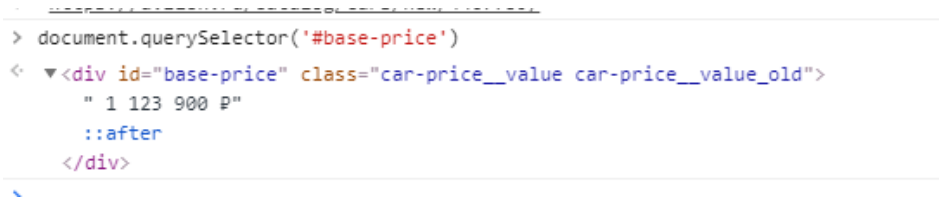
Используйте # для ID и . для классов. Т.е. в приведённом выше примере, если мы решим получать элемент по ID = base-price, нам нужно составить такой запрос

document.querySelector('#base-price')

А если по классам, то

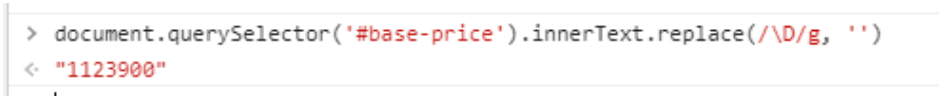
document.querySelector('.car-price__value')

Допустим, мы будем обращаться по ID, выполним код и проверим что мы получили



Теперь нам нужно извлечь стоимость, для этого используем свойство `.innerText` у данного DOM элемента.

Теперь нам осталось вырезать из этого текста всё кроме цифр, используем регулярное выражение для этого `.replace(/\D/g, '')`, который заменит все не цифры на пустые строчки.



И приведём данное значение к целому числу с помощью функции `parseInt`

Теперь, когда у нас есть цена мы можем использовать её при передаче параметров в функцию настройки, например так

```
InitSmartContract({
  detail: {
    price: function() {
      return parseInt(document.querySelector('#base-
price').innerText.replace(/\D/g, '')) || 0;
```

```
    }
  }
})
```

Мы так же используем конструкцию `|| 0` в конце, чтобы передать значение 0 в цену, если нам не удалось расшифровать её.

Как правило, коды детальных страниц совпадают, и кода выше будет достаточно, но периодически случается так, что на некоторых витринах автомобили со скидками и без скидок имеют разный код. Например, на сайте выше есть 2 представления.

Что делать если цена указана по-разному?

Например, цена на нашем сайте для автомобиля https://avilon.ru/catalog/cars/new/4630404/ указана совершенно по-другому

```
<meta itemprop="highPrice" content="1127575.1">
<meta itemprop="lowPrice" content="1127580">
<meta itemprop="priceCurrency" content="RUB">
<meta itemprop="offerCount" content="1">
▼<div class="col-12">
  ▼<div class="car-detail__aside-price">
    ▼<div class="car-price car-price_total">
      ▼<div class="car-price__row">flex
        ▼<div class="car-price__col">
            <div id="final-price-title" class="car-price__title">Стоимость</div>
          ▼<div class="car-detail__aside-price-full car-price__value car-price_
          _value_total" data-dis="1127575.1" data-baseprice="1127575.1" data-
          price="1127580">
              <span>1 127 575</span> == $0
              " ₽ "
            </div>
```

Она обёрнута в дополнительный span, и элемента с ID = `base-price` нет на странице вообще.

В таком случае, необходимо найти что-то общее, или писать более сложный код. Например, я вижу, что в коде на той и другой странице есть элемент

```
<meta itemprop="highPrice" content="1127575.1"> ==
```

Который общий для всех страниц. Сделаем его обработку
`document.querySelector('[itemprop="highPrice"]').content`

```
> document.querySelector('[itemprop="highPrice"]').content
< "1123900"
```

Как видно выше, данный код может содержать точку, обрежем его

И приведём к целому числу

Тогда, итоговый код, который будет работать для всех страниц, будет выглядеть вот так:

```
InitSmartContract({
  detail: {
    price: function() {
      return
parseInt(document.querySelector('[itemprop="highPrice"]').content.split('.')[
0]) || 0;
    }
  }
})
```

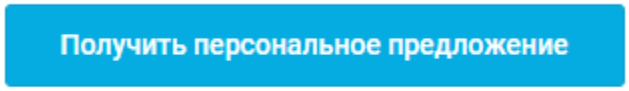
Если же потребуется обработать несколько различных элементов цен, то обычно потребуется сделать код, примерно такого содержания.

```
const prices = []; // Инициализируем массив цен

const element1 = document.querySelector('selector-1'); // Ищем элемент 1
const element2 = document.querySelector('selector-2'); // Ищем элемент 2

const parsePrice = price => parseInt(price.split('.')[0].replace(/\D/g, ''))
|| 0; // Определяем функцию, которая нам по значению извлечёт нормально цену

if( element1 ) prices.push(parsePrice(element1.innerText)); // Добавляем цену
из 1 элемента
if( element2 ) prices.push(parsePrice(element2.innerText)); // Добавляем цену
из 2 элемента

if( prices.length === 0 ) return 0; // Если цен не найдено, возвращаем 0

return Math.max(...prices); // Если цены найдены, возвращаем максимальную
```

Генератор кнопки

Последний штрих, для минимальной настройки виджета — это указание кода, который будет отвечать за отображение кнопки.

Мы возьмём данную кнопку за основу



Посмотрим её код

И опишем функцию, которая генерирует новую кнопку

```
InitSmartContract({
  detail: {
    generator: function(text) {
      // В параметре text будет находиться текст кнопки

      const original = document.querySelector('.car-detail__aside-cntrl'); // Получаем текущую кнопку (Мы получили её враппер, что бы всё скопировалось красиво)
      const clone = original.cloneNode(true); // Клонируем эту кнопку, что бы заменить в неё всё лишнее

      const cloneLink = clone.querySelector('a'); // Находим элемент A, который отвечает закнопку и удаляем лишнее

      cloneLink.classList.remove('js-open-personalOffer'); // Удаляем класс, отвечающий за открытие модального окна

      const dataset = cloneLink.dataset; // Получаем data атрибуты
      for (let key in dataset) { // Идём по всем data атрибутам
```

```
        cloneLink.removeAttribute("data-" + key.split(/(?=[A-Z])/).join("-
").toLowerCase()); // Удаляем все дата атрибуты, что бы у нас не открывалось
модельное окно
      }

      cloneLink.innerText = text; // Устанавливаем текст кнопки на полученный
из Smart Contract

      original.before(clone); // Вставляем новую кнопку до старой
      original.remove(); // Удаляем оригинальную кнопку (Этого можно не
делать)

      return [ cloneLink ]; // Возвращаем элемент кнопки, что бы при нажатии
открылся Smart Contract
    }
  }
})
```

В целом, после того как мы указали 3 параметра выше, код УЖЕ можно установить на витрину. Общий код будет иметь следующий формат:

```
InitSmartContract({
  widget_id: "widgetId",
  detail: {
    identity: function() {
      return window.location.href.split('#')[0].split('?')[0];
    },
    price: function() {
      return
parseInt(document.querySelector('[itemprop="highPrice"]').content.split('.')[
0]) || 0;
    },
    generator: function(text) {
      // В параметре text будет находиться текст кнопки

      const original = document.querySelector('.car-detail__aside-cntrl'); //
Получаем текущую кнопку (Мы получили её враппер, что бы всё скопировалось
красиво)
      const clone = original.cloneNode(true); // Клонируем эту кнопку, что бы
заменить в неё всё лишнее

      const cloneLink = clone.querySelector('a'); // Находим элемент A,
который отвечает закнопку и удаляем лишнее

      cloneLink.classList.remove('js-open-personalOffer'); // Удаляем класс,
отвечающий за открытие модального окна

      const dataset = cloneLink.dataset; // Получаем data атрибуты
      for (let key in dataset) { // Идём по всем data атрибутам
        cloneLink.removeAttribute("data-" + key.split(/(?=[A-Z])/).join("-
").toLowerCase()); // Удаляем все дата атрибуты, что бы у нас не открывалось
модельное окно
      }

      cloneLink.innerText = text; // Устанавливаем текст кнопки на полученный
из Smart Contract
```

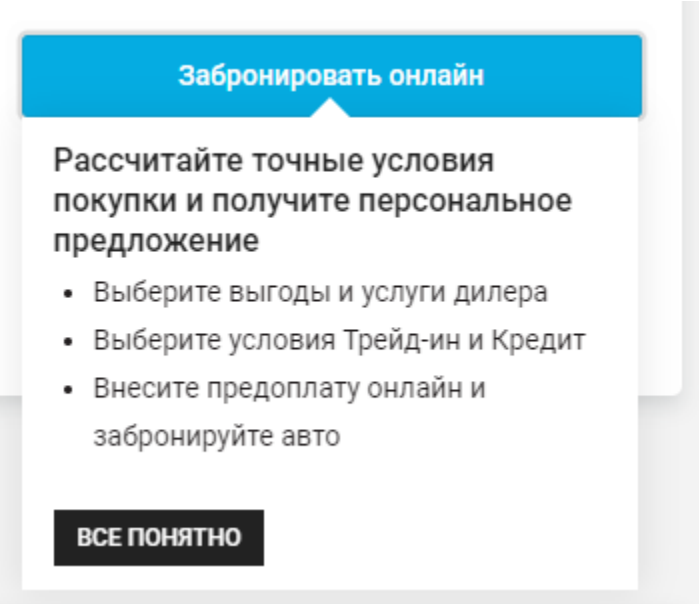
```
      original.before(clone); // Вставляем новую кнопку до старой
      original.remove(); // Удаляем оригинальную кнопку (Этого можно не
делать)

      return [ cloneLink ]; // Возвращаем элемент кнопки, что бы при нажатии
открылся Smart Contract
    }
  }
})
```

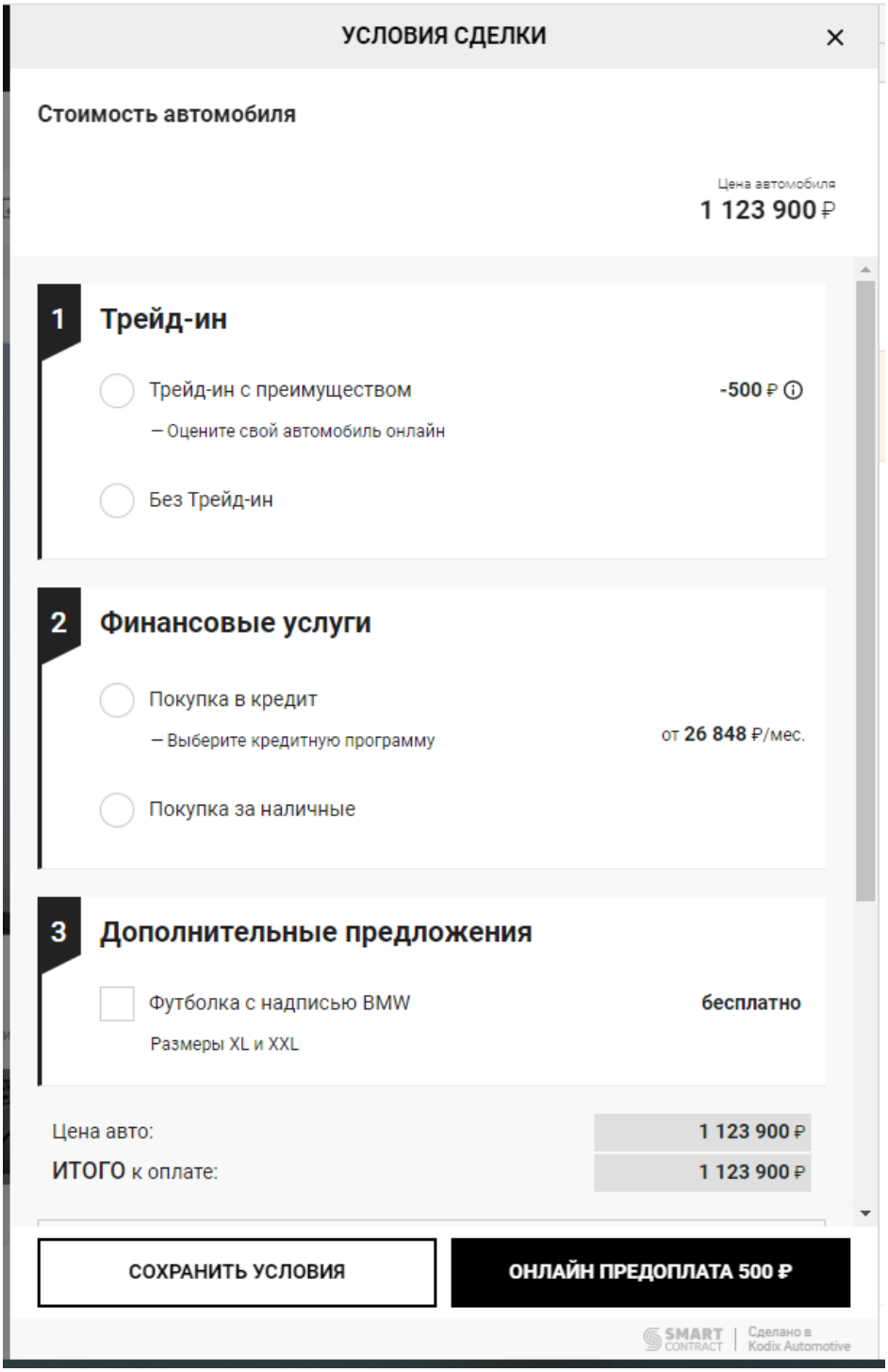
Не забываем обернуть указанный выше код в загрузку скрипта Smart Contract

```
var s = document.createElement('script');
s.src = "https://script.smart-contract.digital/bundle.js?" +
Math.random(10000);
s.onload = function(){
  InitSmartContract({
    widget_id: "widgetId",
    detail: {
      identity: function() {
        return window.location.href.split('#')[0].split('?')[0];
      },
      price: function() {
        return
parseInt(document.querySelector('[itemprop="highPrice"]').content.split('.')[
0]) || 0;
      },
      generator: function(text) {
        // В параметре text будет находиться текст кнопки

        const original = document.querySelector('.car-detail__aside-cntrl');
// Получаем текущую кнопку (Мы получили её враппер, что бы всё скопировалось
красиво)
        const clone = original.cloneNode(true); // Клонируем эту кнопку, что
бы заменить в неё всё лишнее

        const cloneLink = clone.querySelector('a'); // Находим элемент A,
который отвечает закнопку и удаляем лишнее

        cloneLink.classList.remove('js-open-personalOffer'); // Удаляем
класс, отвечающий за открытие модального окна

        const dataset = cloneLink.dataset; // Получаем data атрибуты
        for (let key in dataset) { // Идём по всем data атрибутам
          cloneLink.removeAttribute("data-" + key.split(/(?=[A-Z])/).join("-
").toLowerCase()); // Удаляем все дата атрибуты, что бы у нас не открывалось
модельное окно
        }

        cloneLink.innerText = text; // Устанавливаем текст кнопки на
полученный из Smart Contract

        original.before(clone); // Вставляем новую кнопку до старой
        original.remove(); // Удаляем оригинальную кнопку (Этого можно не
делать)
```

```
        return [ cloneLink ]; // Возвращаем элемент кнопки, что бы при
нажатии открылся Smart Contract
      }
    }
  })
}
document.head.append(s);
```

И на детальной странице всё успешно работает.





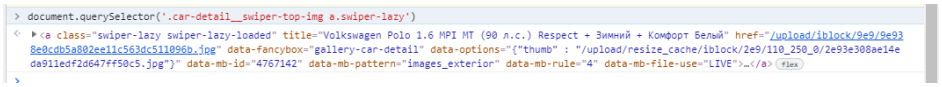
Дальнейшая настройка необходима для того, чтобы Smart Contract был более полным и сохранял в себя больше информации, по которой он сможет произвести более точные расчёты.

Получаем фотографию автомобиля

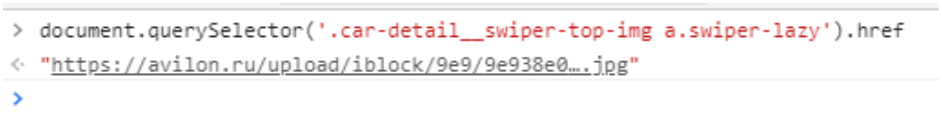
Воспользуемся известным нам методом `document.querySelector` и с его помощью получим первую фотографию на автомобиль

Находим необходимые элементы на странице

И формируем запрос для их получения `document.querySelector('.car-detail__swiper-top-img a.swiper-lazy')`, получаем из него значение ссылки




```
document.querySelector('.car-detail__swiper-top-img a.swiper-lazy').href
```



И вставляем в инициализацию виджета

```
InitSmartContract({
  detail: {
    picture: function() {
      return document.querySelector('.car-detail__swiper-top-img a.swiper-lazy').href;
    }
  }
})
```

Проверяем, что в наш контракт загрузилось изображение

УСЛОВИЯ СДЕЛКИ

Стоимость автомобиля

Цена автомобиля
1 123 900 ₽