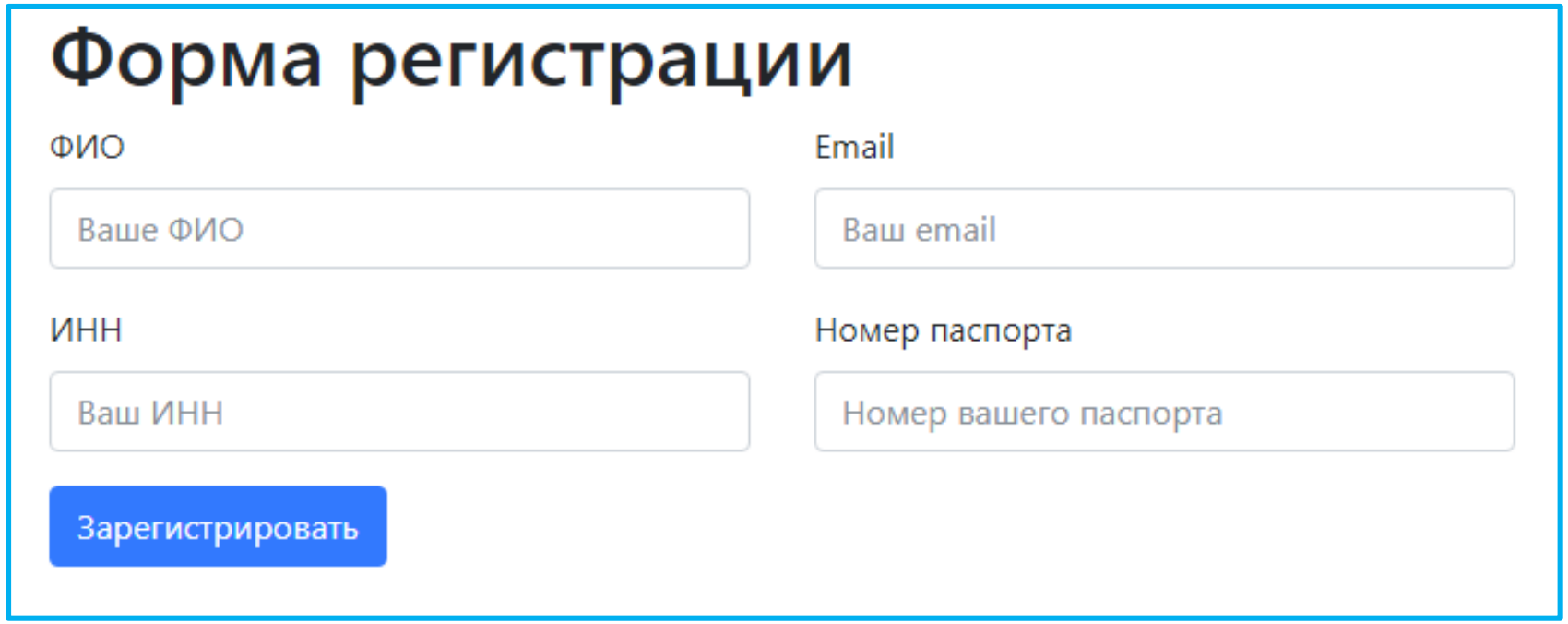
Регулярные выражения, элементы ввода, валидация данных

JavaScript Courses

www.courses.dp.ua

Немного практики #1

Валидация данных



Воспользуемся заготовкой: **./source/ex01.html**

Реализуем вариацию данных в форме

Валидация данных

(проверка на корректность)

Валидация данных сводиться к проверке соответствуют ли введённые данные шаблону или нет. Что считать корректным, а что нет – полностью определяет разработчик.

Проверка вводимых данных (валидация)

Проверка вводимых данных сводиться к:

1. *Заполнено ли поле (выбран ли вариант, для полей выбора)?*

2. *Соответствуют ли введённые данные шаблону?*

Регулярные выражения

Регулярные выражения

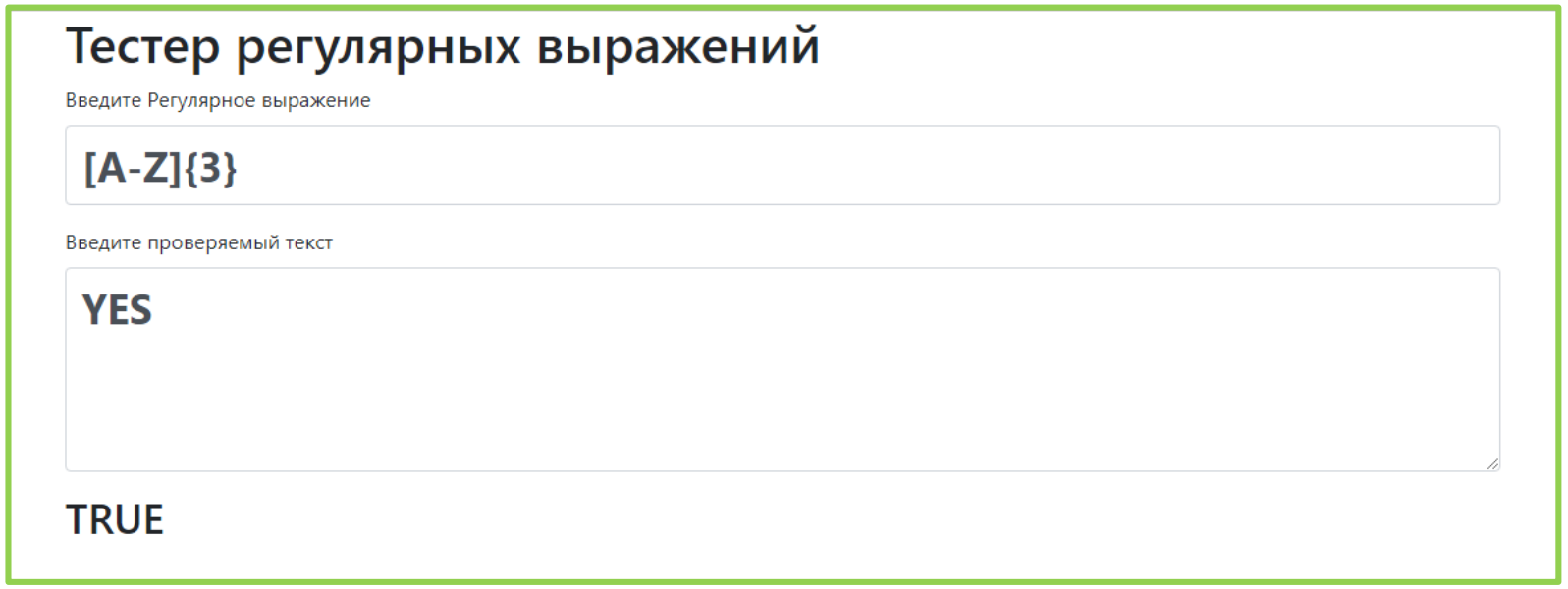
Короткие записи популярных символьных классов

Короткая запись	Полная запись	Описание
\s	[\f\t\n\r]	Пробельный символ
\S	[^\f\t\n\r]	Любой символ, кроме пробельного
\d	[0-9]	Цифра
\D	[^0-9]	Любой символ, кроме цифры
\w	[a-zA-Z0-9]	Латиница или цифра
\W	[^a-zA-Z0-9]	Любой символ, кроме латиницы или цифры
.	[^\n\r]	Любой символ, кроме перевода строки
\b		Граница слова
\B		Не граница слова
\A		Начало строки
\Z		Конец строки

Регулярное выражение – шаблон которым проверяется строка, строка может соответствовать шаблону, а может не соответствовать.

https://ru.wikibooks.org/wiki/Регулярные_выражения
https://uk.wikipedia.org/wiki/Регулярний_вираз

Тестер регулярных выражения



При помощи него мы можем протестировать регулярное выражение перед использованием.

Воспользуйтесь файлом ***./source/regexp.html***

Валидация HTML + Регулярные выражения

```
pattern="^[А-Яа-я]{2,} [А-Яа-я]{2,} [А-Яа-я]{5,}$"

pattern="^\w+@\w+\.\w+$"

pattern="^\s*\d{10}s*$"

pattern="^\s*[А-Яа-я]{2}\d{6}\s*$"
```

*Атрибут **pattern** у элементов ввода позволяет задать шаблон (в виде регулярного выражения) которым браузер будет проверять которые ввёл пользователь. П.С. Более точный шаблон для номера паспорта* **^(((?!Э|Ъ|Ы)[А-ЯІЇЄҐ]){2}[0-9]{6}|[0-9]{9})$** атрибут **pattern** стоит использовать в паре с **required**

Валидация средствами JavaScript

JavaScript, формы и элементы ввода

Для JavaScript'а элементы ввода это текстовые поля. Получить (или установить) информацию из них можно обратившись к свойству ***value*** *элемента.*

Но нам не обязательно проверять значение, мы можем «спросить» браузер считает ли он значение валидным, обратившись к методу ***.checkValidity()*** *элемента ввода.*

Алгоритм валидации с JavaScript:

1. *Подписываемся на событие отправки формы (**onsubmit**) или на клик (**onclick**) кнопки запускающей отправку данных;*

2. *В обработчике событие проверяем **данные** на соответствие;*

3. *Если данные не корректны сообщаем об это пользователю и **отменяет отправку данных** (Для этого достаточно вызвать метод **.preventDefault()** у объекта с информацией о событии);*

4. *Если данные корректны, запускаем процесс отправки данных вызвав метод **.submit()** для формы.*

Доступ к формам и элементам ввода из JavaScript

```html
<html>
    <head>
    </head>
    <body>
        <form name="mainform">
            <label for="some_input_id">Введите имя: </label>
            <input type="text" id="some_input_id" name="name_input">
            <input type="submit" name="send_btn" value="Отправить">
        </form>
    </body>
</html>
```

```js
var a = document.querySelector("input");
console.log(a.value);

var b = document.getElementsByTagName("input")[0];
console.log(b.value);

var c = document.getElementsByName("name_input")[0];
console.log(c.value);

var d = document.getElementById("some_input_id");
console.log(d.value);
```

Все известные методы обращения к элементу работают!

Валидация данных средствами JavaScript + HTML

```
12      window.onload = function(){
13          var form = document.querySelector("form");
14
15          document.querySelector("button").onclick = function(e){
16              if(!form.checkValidity()){
17
18                  document.querySelectorAll("input").forEach(function(element){
19                      if(!element.checkValidity()){
20                          element.classList.remove("is-valid");
21                          element.classList.add("is-invalid");
22                      }else{
23                          element.classList.remove("is-invalid");
24                          element.classList.add("is-valid");
25                      }
26                  });
27
28                  e.preventDefault();
29              }
30
31          }
32      }
```

Работа с регулярными выражениями в JavaScript

Регулярные выражения в JavaScript

RegExp.test(*string*)

String.match(*RegExp*)

String.search(*RegExp*)

String.replace(*RegExp*, *Function*)

Методы строк и методы объекта RegExp которые нам могут пригодиться.

https://learn.javascript.ru/regexp-methods

Немного практики #2

Регулярные выражения на практике

	SSD 240/256 ГБ, SATA	1132 грн.
Корпус и БП	Middle Tower ATX, 450 Вт	1415 грн.
	Сумма для ПК на платформе Intel	**18253.5 грн.**
	Сумма для ПК на платформе AMD	**16980 грн.**

При первой же возможности мы вернули в базовые игровые конфигурации чипы **Core i3-8100**. Катализатором здесь стал некоторый откат стоимости процессоров Intel на нашем рынке. Нет, они по-прежнему еще стоят очень дорого, но лишь по этой причине отбрасывать возможность сборки ПК на данной платформе – не оправдано. Итак, самые доступные 4-ядерные чипы Coffee Lake сейчас можно купить в Украине за 4811 грн., то есть на 849 грн.–40 дешевле, чем полтора месяца назад. На зарубежных площадках эта модель предлагается за 3679 грн.–140. Если вас не смущают нюансы с гарантийным обслуживанием и даже с учетом стоимости доставки покупка процессора обойдется дешевле, можно рассмотреть такой вариант дополнительной экономии.

Увы, других способов пока нет. Вернее, есть, но они вам вряд ли понравятся. Самый очевидный – покупка процессора семейства **Pentium Gold**. Это изначально будут откровенно слабые чипы для игровой системы. Двух вычислительных ядер уже недостаточно для тяжелых проектов, несмотря на их достаточно высокую эффективность и поддержку технологии многопоточности Hyper-Threading.

Воспользуемся заготовкой: **./source/ex02.html**

Реализуем замену всех долларовых цен на гривневые (с пересчётом)

Регулярные выражения на практике

```
19      window.onload = function(){
20
21          var ps = document.querySelectorAll("p, td");
22
23          ps.forEach(item =>{
24
25              item.innerHTML = item.innerHTML.replace(/\$[0-9]*/g, function(price){
26                      price = parseInt(price.replace('$', ''));
27                      price = price * 28.3 + " грн.";
28                  return price;
29              });
30
31          });
32
33      }
```

Воспользуемся заготовкой: **./source/ex02.html**

Реализуем замену всех долларовых цен на гривневые (с пересчётом)

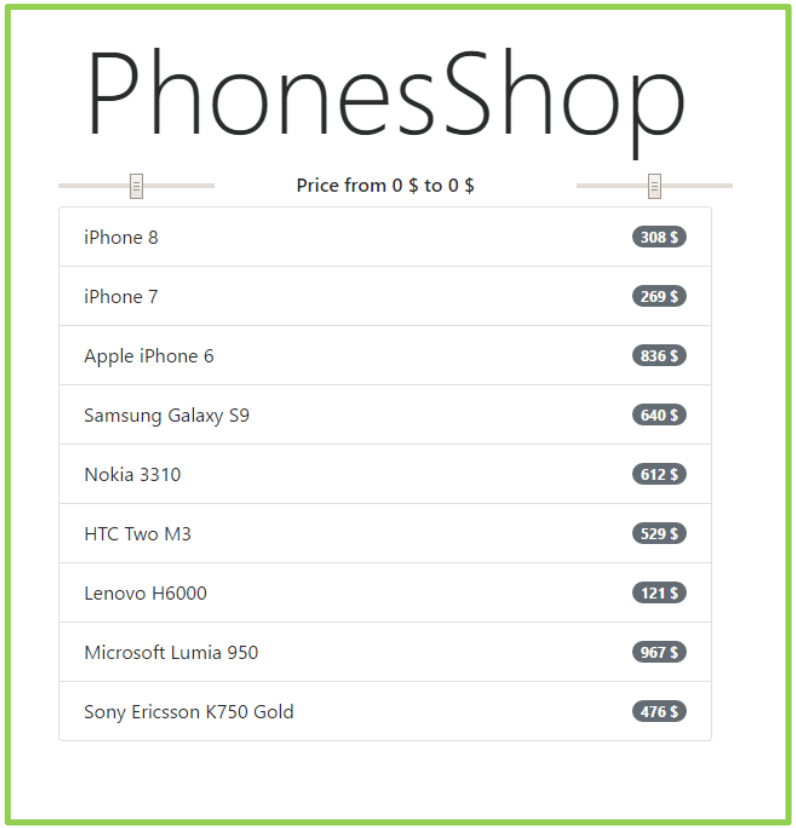
События элементов ввода **onInput** и **onChange**

Событие элементов ввода: onInput и onChange

*События **onInput** и **onChange** отвечают за моменты когда ввод данных в элементу осуществляется, т.е. в процессе ввода. Но есть один ньюанс, **onInput** срабатывает при каждом изменении данных в элементе ввода, а **onChange** только когда пользователь закончит ввод данных в элемент.*

Немного практики #3

Динамическая фильтрация данных



Необходимо сделать установку диапазона цен при помощи слайдеров, при этом в динамике скрывать те товары которые не входят в выбранный диапазон.

Воспользуемся заготовкой: **./source/ex03.html**

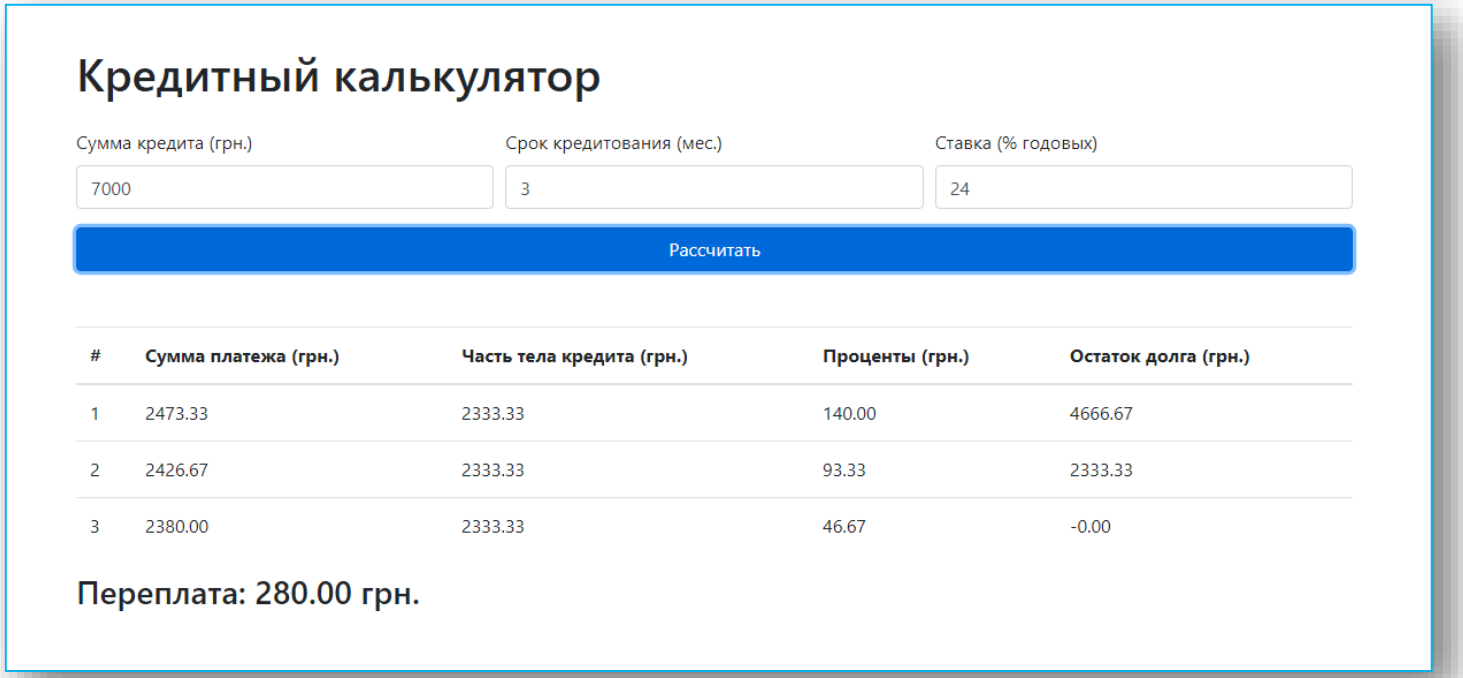
Динамическая фильтрация данных

```
<script>

    function filter(){
        var min = min_price_range.value;
        var max = max_price_range.value;

        min_price_label.innerHTML = min;
        max_price_label.innerHTML = max;

        var prices = document.querySelectorAll("ul > li > span");

        prices.forEach(function(element){
            var price = parseInt(element.innerHTML);
            if(price < min || price > max){
                element.parentNode.style.display = "none";
            }else{
                element.parentNode.style.display = "flex";
            }
        });
    }

    var max_price;
    var min_price;

    window.onload = function(){
        var prices = document.querySelectorAll("ul > li > span");

        max_price = min_price = parseInt(prices[0].innerHTML);

        prices.forEach(function(element){
            var price = parseInt(element.innerHTML);
            max_price = price > max_price ? price : max_price;
            min_price = price < min_price ? price : min_price;
        });

        max_price_range.min = min_price_range.min =  min_price;
        max_price_range.max = min_price_range.max =  max_price;

        min_price_label.innerHTML = min_price_range.value = min_price;
        max_price_label.innerHTML = max_price_range.value = max_price

        min_price_range.oninput = filter;
        max_price_range.oninput = filter;
    }

</script>
```

Необходимо сделать установку диапазона цен при помощи слайдеров, при этом в динамике скрывать те товары которые не входят в выбранный диапазон.

Немного практики #4

Кредитный калькулятор 3.0



Воспользуемся заготовкой: **./source/ex04.html**

Реализуем кредитный калькулятор, который будет получать данные для расчёта из элементов ввода, и выводить план погашения кредита в разметку. Используем алгоритм классической схемы погашения.

Кредитный калькулятор 3.0

```
function calc(){
    results.innerHTML = ""; //Clear output table

    var sum     = sum_input.value;
    var rate    = rate_input.value;
    var term    = term_input.value;

    var body_payment    = sum / term;
        body_payment    = Math.floor(body_payment * 100)/100;
    var month_rate      = (rate / 12) / 100;

    var overpay_sum = 0;

    for(var i = 1; i <= term; i++){

        var percents_payment = sum * month_rate;
            percents_payment =  Math.floor(percents_payment * 100)/100;

        var month_payment = body_payment + percents_payment;
            month_payment = Math.floor(month_payment * 100)/100;

        overpay_sum = overpay_sum + percents_payment;
        sum = sum - body_payment;
        sum = Math.floor(sum * 100 / 100);

        var new_row =   `<tr>
                            <td>${i}</td>
                            <td>${month_payment}</td>
                            <td>${body_payment}</td>
                            <td>${percents_payment}</td>
                            <td>${sum}</td>
                        </tr>`;

        results.innerHTML = results.innerHTML + new_row;
    }
    overpay_sum = Math.floor(overpay_sum*100)/100;
    overpay.innerHTML = overpay_sum;
}
```

Домашнее задание #H.1

#H.1 Поиск текстовых фрагментов

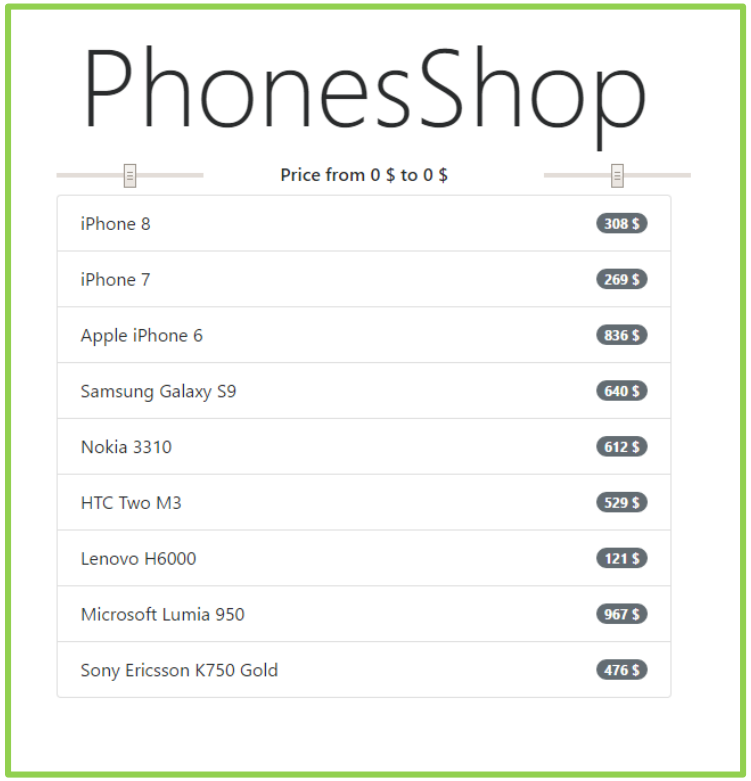
```
<script>

    var data = "Pellentesque ex eros, +380665433321 porttitor eu bibendum ac,
    aliquam tincidunt urna. Mauris tristique lobortis orci, nec varius magna
    convallis interdum. Etiam pharetra tempor ex, vel eleifend (067) 678 44 21 odio
    lacinia (0562) 35-30-38 eget. Morbi maximus libero vitae aliquet facilisis.
    Vivamus vitae quam nisi. Quisque quis venenatis lacus. Sed ac lorem
    (050)567-45-33 nec leo pharetra dapibus sed eu +38067432112 ex. In hac
    habitasse platea dictumst. In dignissim suscipit rutrum. Ut luctus sapien in
    risus auctor, ac placerat 067-678-44-21  quam malesuada. Pellentesque (056)
    7783322 bibendum justo tempus purus convallis, a viverra nunc ullamcorper.
    Nulla eget lectus gravida, porta eros vitae, semper erat +39-926-1234567.
    Aenean volutpat vehicula dui ut pharetra.";

    /*Выбрать из строки номера телефонов и вывести их в полном международном
    формате, если код страны не указан, считать что Украина.

    Например:
    +380675556677;
    +380503335588;
    ...
    */
```

Воспользуйтесь заготовкой: **./homework/hw_h1.html**

Задача: выбрать все телефоны из строки и вывести их список в консоль в международном формате.

Домашнее задание #H.2

#H.2 Динамическая фильтрация данных



В нашем примере у пользователя есть возможность при помощи левого ползунка выбрать цену большую чем в этот момент выбрано в правом ползунке. Исправьте это.

Воспользуйтесь заготовкой: **./homework/hw_h2.html**