Условная отрисовка

- Условия
- Тернарный оператор
- Сложные условия

Условия

С помощью директивs **v-if** можно показывать или скрывать элементы. Как эта директива работает: **параметром она принимает любое свойство объекта** data. Если это свойство имеет значение **true** - то элемент будет показан, а если **false** - то скрыт.

Разберем на примере.

v-if

Директива **v-if** используется для отрисовки блока **по условию**. Блок будет отображаться только в том случае, если выражение директивы возвращает значение, приводимое к **true**.

```
<h1 v-if="isTrue">Мир разумен и справедлив!</h1>
```

example_1. Директива v-if

```
<div id="app">

    <div v-if="isWelcome">

        <h2>Добро пожаловать в мир прогрессивного фреймворка
        Vue!</h2>

        <h3>ну и к нам заходите... ;)</h3>

    </div>

</div>
```

```
<script>
    let vm =  Vue.createApp({
        data() {
            return {
                // логическая переменная
                isWelcome: false
            }
        }
    }).mount("#app");
    // определим логическое значение переменной isWelcome
    vm.isWelcome = confirm("Вы готовы изучать фреймворк Vue?");
</script>
```

v-else

Для указания **блока иначе** директива **v-if** может работать в комплекте с **v-else**:

```
<h1 v-if="isTrue">Мир разумен и справедлив!</h1>
<h1 v-else>Весь Мир бардак ... </h1>
```

Элемент с директивой **v-else** должен следовать сразу за элементом с директивой **v-if** или **v-else-if**, иначе он не будет распознан.

Разберем директиву **v-else** на практическом примере.

example_2. Директива v-else

```
<div id="app">
    <div v-if="isWelcome">
        <h2>Добро пожаловать в мир прогрессивного фреймворка
        Vue!</h2>
        <h3>Делать то, что тебе нравится, — это свобода. Любить
```

```
        то, что ты делаешь, — это счастье</h3>
        P/S. (Frank Tyger)
    </div>
    <div v-else>
        <p>- <b>Вот скажите мне, куда могут нас отправить, если мы не сдадим эти драконовы экзамены?</b> - поинтересовался Чез. - <b>И с каким званием мы тогда покинем эти стены?</b>
        <p>- <b>Со званием - неудачник,</b> - охотно подсказала Алиса.
        <p>Чез на секунду задумался. - <b>Это не звание, это судьба,</b> - наконец нашелся он.</p>
        P/S. (Алекс Кош, «Огненный Факультет»)
    </div>
</div>
<script>
    let app =  Vue.createApp({
        data() {
            return {
                // логическая переменная
                isWelcome: false
            }
        }
    });
    // монтируем экземпляр
    const vm=app.mount("#app");
    // // определим логическое значение переменной isWelcome
    vm.isWelcome = confirm("Вы готовы изучать фреймворк Vue?");
```

```
</script>
```

v-else-if

Вместо нескольких **v-if** можно использовать конструкции **v-else-if**. Как следует из названия, **v-else-if** служит в качестве блока **else if** для директивы **v-if**.

Можно объединять эти директивы в длинные цепочки:

```
<div v-if="type === 'A'">
     A
</div>
<div v-else-if="type === 'B'">
     B
</div>
<div v-else-if="type === 'C'">
     C
</div>
<div v-else>
     Не A / B / C
</div>
```

Как и **v-else**, **v-else-if** должен следовать сразу за элементом с **v-if** или **v-else-if**.

example_3. Директива v-else-if

```
<div id="app">
    <div v-if="isBest==true">
        <h2>Свой человек ;))) !</h2>
        <img src="img.jpg" width="350">
    </div>
```

```
    <div v-else-if="isBest==false">
        <h2>УК РФ Статья 213. Хулиганство</h2>
        <h3>1. Хулиганство, то есть грубое нарушение
        общественного порядка, выражающее явное неуважение к
        обществу, совершенное:</h3>
        <!-- ... -->
    <div v-else>
        <h3>
        Что же ты, зараза, бровь себе подбрила,<br>
        <!-- ... -->
        </h3>
    </div>
</div>
<script>
    let app =  Vue.createApp({
        data() {
            return {
                // логическая переменная
                isBest: undefined
            }
        }
    }).mount("#app");
    // определитесь с вашим отношением к нам
    let response = prompt("pechora-PRO... Мы классные, клевые и
    суперешные!\n(Да - 1 | Нет - 2 | Не знаю - 3)", "1");
    // переведем на язык программы
    switch (response) {
        case "1" : app.isBest = true; break;
```

```
        case "2" : app.isBest = false;
    }
</script>
```

Цепочки условных конструкций могут быть настолько длинными, насколько это требуется разработчику приложения.

example_4. Цепочка директив v-else-if

```
<div id="app">
    <div v-if="temperament == 'Флегматик'">
        <h2>Флегматик</h2>
        В своей деятельности они всегда склонны к порядку и любят привычную для них обстановку.
        <!-- ... -->
    </div>
    <div v-else-if="temperament == 'Сангвиник'">
        <h2>Сангвиник</h2>
        Это крайне непоседливые, энергичные, легкие на подъем люди, которые очень любят движение.
        <!-- ... -->
    </div>
    <div v-else-if="temperament == 'Холерик'">
        <h2>Холерик</h2>
        Холерики, по сравнению с сангвиниками, вспыльчивы, непостоянны, агрессивны и импульсивны.
        <!-- ... -->
    </div>
    <div v-else-if="temperament == 'Меланхолик'">
        <h2>Меланхолик</h2>
```

```
        Достаточно ранимые, обидчивые и скрытые личности. Они
        склонны к глубоким переживаниям и грустным мыслям.
        <!-- ... -->
    </div>
    <div v-else>
        <h2>Упс...</h2>
        Вы какой-то бестолковый что ли... Что вы там вводили?
        Сказано было выбрать из следующих вариантов:
        <ul>
            <li>Флегматик</li>
            <li>Сангвиник</li>
            <li>Холерик</li>
            <li>Меланхолик</li>
        </ul>
    </div>
    <!—
    переменная temperament будет вычислена как true или false
    -->
    <img v-if="temperament" src="temperament.jpg">
</div>
<script>
    let app =  Vue.createApp({
        data() {
            return {
                temperament: undefined
            }
        }
    }).mount("#app");
```

```
    // определим значение переменной
    app.temperament = prompt("Кто вы по темпераменту? (Флегматик
    | Сангвиник | Холерик | Меланхолик)");
</script>
```

В примерах выше мы вынуждены несколько элементов располагать в одном **родительском теге** с условной конструкцией. Назначение родителя - объединять элементы. Такой способ написания кода может приводить к захламлению HTML-кода и появлению в нем необязательных элементов.

Решением может стать псевдоэлемент **template**.

v-if на <template>

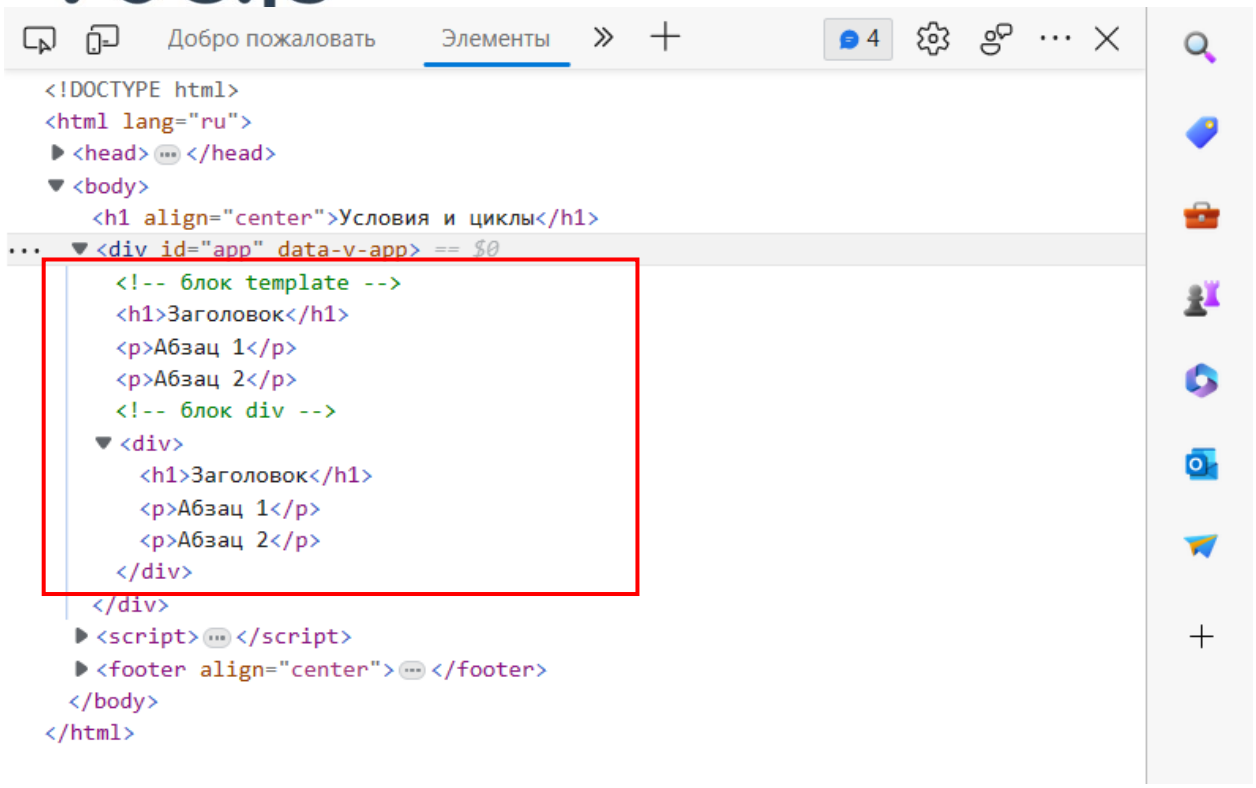
Поскольку **v-if** — директива, она должна быть указана в одном конкретном теге. Но что если мы хотим **управлять отображением сразу нескольких элементов**? Мы оборачиваем группу элементов в блоковый элемент, например, в тег **div**. Получается, что в разметке документа появляется один **дополнительный узел**. Не всегда это удобно.

В таком случае **можно применить** v-if **к псевдоэлементу** <template>, **который служит невидимой обёрткой и сам в результатах отрисовки не появляется**.

```
<template v-if="isShow">
    <h1>Заголовок</h1>
    <p>Абзац 1</p>
    <p>Абзац 2</p>
</template>
```

В демонстрационном примере **example_5** протестируем и сравним результат оборачивания группы элементов блоком **div** и псевдоэлементом **template**.

Результат работы скрипта продемонстрирован на рисунке. **Псевдоэлемент отсутствует** при отрисовки конечного дерева DOM.



example_5. Псевдоэлемент template

```
<div id="app">

    <!-- блок template -->

    <template v-if="isShow">

        <h1>Заголовок</h1>

        <p>Абзац 1</p>

        <p>Абзац 2</p>

    </template>

    <!-- блок div -->

    <div v-if="isShow">

        <h1>Заголовок</h1>

        <p>Абзац 1</p>

        <p>Абзац 2</p>

    </div>

</div>
```

```
<script>
    let app =  Vue.createApp({
        data() {
        return { isShow: true }
        }
    }).mount("#app");
</script>
```

v-show

Ещё одним вариантом условного отображения является директива **v-show**. Используется очень похоже:

```
<h1 v-show="isOk">Привет!</h1>
```

> **Важно.** Отличия в том, что элемент с **v-show** будет **всегда отрисовываться** и **оставаться в DOM**, а переключаться будет лишь его CSS свойство **display**

Директиву **v-show** нельзя использовать на элементе **<template>** и она не работает с **v-else**.

Обратите внимание в примере **example_6**, изменение состояния переменной **login** привязано к хуку **updated**. Хук вызывается после того, как обновится виртуальный DOM из-за **изменений данных**.

DOM компонента Vue будет обновлён к моменту вызова этого хука, поэтому здесь можно выполнять операции связанные с DOM.

---

example_6++.

```
<div id="app">
    <p>Введите логин для продолжения работы: <input type="text"
    v-model="login"></p>
```

```
    <div v-show="isShow">
        <h3>Личный кабинет пользователя</h3>
        <img src="login.jpg" width="200">
    </div>
    <div v-show="!isShow">
        <h3>Добрый день, для продолжения работы необходима
        регистрация на сайте! Желаем удачи!</h3>
        <img src="no-login.jpg" width="200">
    </div>
</div>
<script>
    let rootApp = {
        data() {
            return {
                login: "",
                isShow: false
            }
        },
        // хук updated
        updated () {
            if (this.login == "admin")
                this.isShow = true;
            else
                this.isShow = false;
        }
    };
    let vm = Vue.createApp(rootApp).mount("#app");
```

```
</script>
```

v-if против v-show

v-if выполняет настоящую условную отрисовку, так как гарантирует, что слушатели событий и дочерние компоненты внутри блока должным образом уничтожаются и воссоздаются при переключениях условия.

v-if также ленивый: если условие ложно на момент первоначальной отрисовки, то он ничего не сделает — условный блок не будет отрисован до тех пор, пока условие не станет истинным.

Для сравнения, v-show **намного проще — элемент** всегда отрисовывается**, вне зависимости от** исходного состояния **с переключением на основе CSS**.

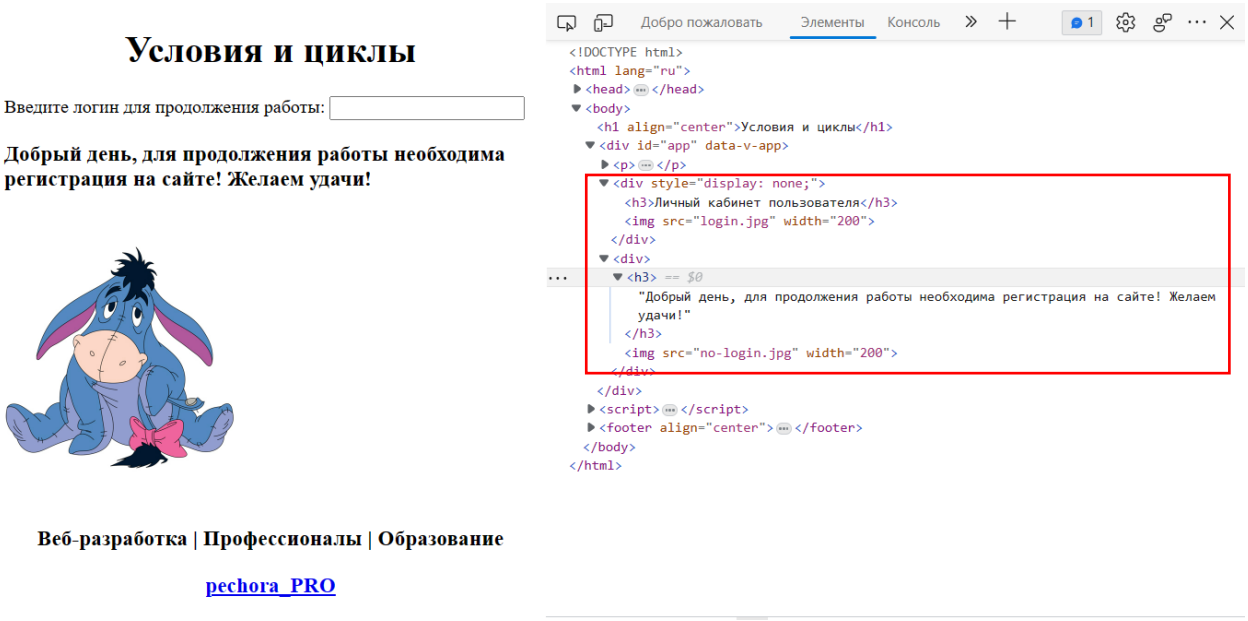
В целом, у **v-if** выше затраты на переключение, в то время как **v-show** имеет больше затрат на первичную отрисовку. Так что рекомендуется использовать **v-show**, если переключения будут частыми, и предпочтительнее применение **v-if**, если условие может и не измениться во время исполнения.

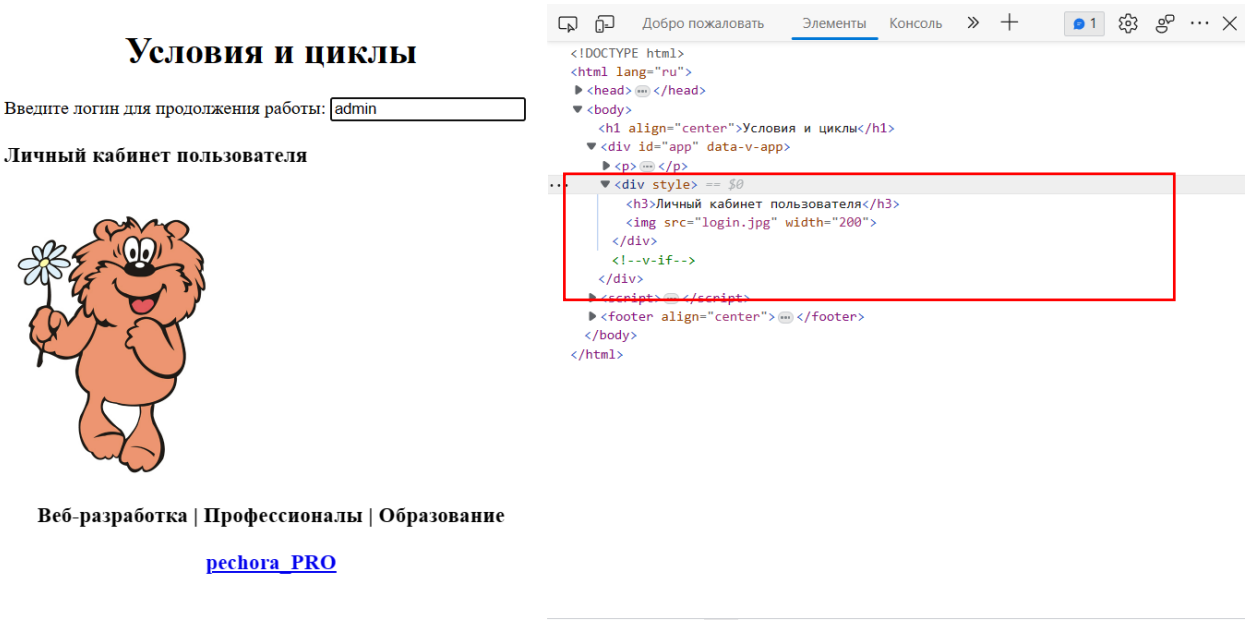
Изменим немного код примера example_6, проанализируйте внимательно разметку отрисованного шаблона Vue на блоке с директивой **v-if** и блоке с директивой **v-show**.

example_7. Сравнение v-if и v-show

```
<div id="app">
    <p>Введите логин для продолжения работы: <input type="text"
    v-model="login"></p>
    <div v-show="isShow">
        <h3>Личный кабинет пользователя</h3>
        <img src="login.jpg" width="200">
    </div>
    <div v-if="!isShow">
        <h3>Добрый день, для продолжения работы необходима
```

```
        регистрация на сайте! Желаем удачи!</h3>

        <img src="no-login.jpg" width="200">

    </div>

</div>

<script>

    // ...

</script>
```





v-if вместе с v-for

При одновременном использовании **v-if** и **v-for** на одном элементе, **v-if** будет исполняться первым.

> **Важно.** Совместное использование **v-if** и **v-for** не рекомендуется

Подробнее в следующем уроке.

Тернарный оператор

Как любой язык программирования **Vue** реализует синтаксис тернарного оператора.

```
{{ isShow ? 'Скрыть' : 'Показать' }}
```

В данном примере в зависимости от переменной **isShow** будет отображаться текст 'Скрыть' или 'Показать'.

Тернарный оператор удобно использовать, например, в контейнере **button** следующим образом:

```
<button v-on:click="toggleElem">
    {{ isShow ? 'Скрыть' : 'Показать' }}
</button>
```

В этом примере, если свойство **isShow** имеет значение **true**, то следующее действие будет сокрытие элемента и мы покажем текст 'Скрыть', а если свойство имеет значение **false**, то следующее действие будет показ элемента и мы покажем текст 'Показать'.

Применим возможности тернарного оператора в коде следующего примера.

example_8. Тернарный оператор

```
<div id="app">
```

```
    <button v-on:click="toggleElem">
        {{ isShow ? 'Не интересует' : 'О вреде курения' }}
    </button>
    <template v-if="isShow">
        <h2>Бросай курить!</h2>
        <img src="no-smok.jpg" width="300">
    </template>
</div>
<script>
    let app =  Vue.createApp({
        data() {
            return { isShow: false }
        },
        methods: {
            toggleElem: function() {
                this.isShow = !this.isShow;
            }
        }
    }).mount("#app");
</script>
```

Сложные условия

В директиве **v-if** можно делать не только свойства со значениями **true** или **false**, но и передавать более сложные условия.

Пусть, к примеру, в свойстве **num** может быть одно из чисел 1, 2 или 3:

```
let app =  Vue.createApp({
    data() {
```

```
        return {
            // текущее значение num = 3
            num: 3
        }
    }
}).mount("#app");
```

Создадим 3 абзаца, из которых будет показан только один в зависимости от значения свойства **num**:

```
<div id="app">
    <p v-if="num == 1">Один</p>
    <p v-if="num == 2">Два</p>
    <p v-if="num == 3">Три</p>
</div>
```

Можно составлять условия с применением **логических операторов**, пусть в переменной **age** – хранится возраст:

```
<div id="app">
    <p v-if="age >= 1 && age <= 10">Мир разумен и справедлив</p>
    <p v-if="age >= 11 && age <= 20">Упс...</p>
</div>
```

Разберем практику применения сложных условий в следующем практическом примере.

example_9. Логические операторы в директиве v-if

```
<div id="app">
    <h2>Введите ваш возраст и мы определим сферу ваших
    интересов</h2>
    (укажите возраст в диапазоне 3-16)
    <input type="text" v-model="age"><br>
```

```
    <p v-if="age >= 3 && age < 5"><img src="3+.jpg"></p>
    <p v-if="age >= 5 && age < 7"><img src="5+.jpg"></p>
    <p v-if="age >= 7 && age < 9"><img src="7+.jpg"></p>
    <p v-if="age >= 9 && age < 11"><img src="9+.jpg"></p>
    <p v-if="age >= 11 && age < 13"><img src="11+.jpg"></p>
    <p v-if="age >= 13 && age < 16"><img src="censored.jpg"></p>
</div>
<script>
    let app =  Vue.createApp({
        data() {
            return { age: undefined }
        }
    }).mount("#app");
</script>
```

В конце урока два практических взгляда на реализацию одного функционала. Займемся вопросами литературной осведомленности старшеклассников ;)

example_10. Скрытие блока используя переменную в секции data

```
<div id="app">
    <h3>"Глупый пингвин робко прячет ... в утесах"(М.
    Горький)</h3>
    <ol>
        <li>Постную грудинку</li>
        <li>Заначку</li>
        <li>Тело жирное</li>
        <li>Акваланг, оружие и документы</li>
    </ol>
    Показать ответ: <input type="checkbox" v-model="quest">
```

```
    <div v-if="quest" style="margin-top:10px; padding:10px;
    border:1px solid grey">
        <b>Тело жирное</b>
    </div>
</div>
<script>
    let app =  Vue.createApp({
        data() {
            return {
                quest: false
            }
        }
    }).mount("#app");
</script>
```

При большом количестве вопросов придется организовывать более сложную структуру данных секции data, поэтому иногда удобнее будет использовать CSS свойство **display**.

example_11. Скрытие блока используя свойство display

```
<div id="app">
    <h3>"Глупый пингвин робко прячет ... в утесах"(М.
    Горький)</h3>
    <ol>
        <li>Постную грудинку</li>
        <li>Заначку</li>
        <li>Тело жирное</li>
        <li>Акваланг, оружие и документы</li>
    </ol>
```

```
    Показать ответ: <input type="checkbox" v-
    on:click="fnVisibility">
    <div style="display:none; margin-top:10px; padding:10px;
    border:1px solid grey">
        <b>Тело жирное</b>
    </div>
</div>
<script>
    let app =  Vue.createApp({
        methods: {
            fnVisibility : function() {
                el = event.target.nextElementSibling;
                if (el.style.display == "none")
                    el.style.display = "block"
                else el.style.display = "none"
            }
        }
    }).mount("#app");
</script>
```

На этом все. В конце урока, как обычно, задачи для самостоятельного решения. В одном из заданий допишем полный вариант анкеты.

Задача 1

Напишите сценарий, в котором при вводе в текстовые поля **Логин** / **Пароль** данных администратора будет отображаться блок приветствия.

Задача 2

Создайте блок, в котором будет отображаться текст '**Показать телефон**'. Напишите сценарий, где при помощи **тернарного оператора** клик по блоку будет приводить к показу номера телефона.

Задача 3

В директории раздаточного материала вам предложен полный текст анкеты для старшеклассников. Используя в качестве подсказки код примера **example_11** выведите анкету в браузер**.** Создание собственного варианта скрипта приветствуется.

P.S.

Для отработки и закрепления учебного курса **донам** группы предоставляется следующий раздаточный материал.

К каждому уроку курса**:**

- Файлы **демонстрационного кода** (example);
- **Задачи** с решениями в контексте рассматриваемых вопросов урока (task)**.**

К каждой теме курса:

- **Практические** работы.