МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

Е.Г. Селина

ОРГАНИЗАЦИЯ ИНТЕРАКТИВНОГО ВЗАИМОДЕЙСТВИЯ В HTML-ДОКУМЕНТАХ

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ В УНИВЕРСИТЕТЕ ИТМО
по направлениям подготовки (специальностям) 19.03.02 Продукты питания из растительного сырья,19.03.03 Продукты питания животного происхождения, 14.03.01 Ядерная энергетика и теплофизика
в качестве учебного пособия для реализации основных профессиональных образовательных программ высшего образования бакалавриата

Санкт-Петербург

2018

Е.Г. Селина Организация интерактивного взаимодействия в HTML-документах. – СПб: Университет ИТМО, 2018. – 35 с.

Рецензенты: Муромцев Д.И. , кандидат технических наук, доцент
Симоненко З.Г. , кандидат технических наук, старший преподаватель.

Пособие содержит основы работы с полями форм и программирования на языке JavaScript. Рассмотрены примеры использования интерактивного взаимодействия.

Учебно-методическое пособие предназначено для бакалавров направлений 19.03.02 Продукты питания из растительного сырья, 19.03.03 Продукты питания животного происхождения, 14.03.01 Ядерная энергетика и теплофизика.

Университет ИТМО – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

Введение

JavaScript используется для создания интерактивных веб-страниц. Интерактивные страницы могут взаимодействовать с пользователем, например, выводить сообщения, изменять содержимое после определенных действий и т.д. Объектно-ориентированный язык разработкивстраиваемых приложений JavaScript разработан компанией Netscape Communications Corporation в сотрудничестве с Sun Microsystems. Как и у любого язык программирования, основная задача Javascript создавать последовательность действий, которые будут приводить к определенному результату.

Создание интерактивного взаимодействия разбивается на 2 этапа. На первом этапе создаётся бланк, в который пользователь сможет внести свою информацию; на втором – программа, которая принимает эту информацию, обрабатывает её и, если нужно, подготавливает и пересылает пользователю ответ. “Эта программа называется термином *скрипт*. С помощью скриптов Вы сможете создавать *интерактивные веб-страницы*.

JavaScript был создан в 1995 году и представляет собой инструмент, дающий возможности программирования. JavaScript обладает простым синтаксисом и его достаточно легко изучить.

JavaScript встраивается прямо в веб-страницы и выполняется браузером во время загрузки. Все современные браузеры имеют поддержку JavaScript.

JavaScript дает возможности:

- Динамически изменять содержимое веб-страниц;
- Привязывать к документам или его компонентам обработчики событий (те функции, которые выполнятся только после того, как совершатся определенные распознаваемые действия);
- Выполнять действия через заданные промежутки времени;
- Управлять поведением браузера: открывать новые окна, загружать указанные документы и т.д.;
- Создавать и считывать cookies;
- Определять, какой браузер использует пользователь, каково разрешение экрана, какие страницы, которые посещал пользователь и т.д.;
- Проверять данные форм перед отправкой их на сервер и др.

1. Создание HTML форм

Бланк называется термином форма. Формы и обеспечивают получение информации от пользователя и передачу их программе, которая обра-

батывает эту информацию. В настоящее время большинство сайтов содержит такие элементы интерфейса, как поле ввода текста, кнопки, переключатели и флажки.

1.1. Создание формы

Все средства, которые нужны для её создания, размещаются в HTML-документе в зоне <BODY>…</BODY> внутри контейнера <FORM>…</FORM>.

Открывающий тег <FORM> называется *заголовком формы*. В нём с помощью атрибутов уточняется, как называется обрабатывающая программа, где она расположена, каков тип собранных данных и как их следует пересылать обрабатывающей программе. Подробное описание этих атрибутов приведено в Приложении 1.

Теги, создающие поля для ввода данных, воспроизводят все средства пользовательского интерфейса, привычные для пользователя Windows. Основные из них имеют следующий вид (значения атрибутов выделять кавычками необязательно):

– <INPUT TYPE="тип элемента управления" NAME="имя" VALUE="значение по умолчанию" SIZE="число ">.

– <TEXTAREA ROWS="число" COLS="число" NAME="имя" VALUE="значение по умолчанию"> … </TEXTAREA>

– <SELECT NAME="имя" SIZE="число" MULTIPLE>
<OPTION SELECTED> …
<OPTION> …
…
<OPTION> …
</SELECT>

Смысл ключевых слов:

– INPUT – тип элемента, который создаётся в этом теге, определяется атрибутом TYPE. Обозначения основных элементов управления и виды полей, которые они создают, приведены в табл. 1.

– TEXTAREA – поля для ввода длинных текстов, оформленные в несколько строк.

– SELECT – создаёт поле с выпадающим списком выбора. Каждый элемент списка указывается после тега <OPTION> (вместо многоточия).

Смысл атрибутов:

– TYPE="тип элемента управления". Обозначения разных элементов управления и виды полей, которые они создают, приведены в табл. 1.

– NAME="имя". Имя используется в обрабатывающей программе как идентификатор введённых значений. Имя должно быть составлено только из латинских букв и цифр и начинаться с буквы. Имя – необязательный атрибут. Если оно не указано, браузер присваивает данным, введённым в этот элемент, порядковый номер (нумерация начинается с числа "0"), и в дальнейшем ссылка на эти данные будет иметь менее понятный вид.

– VALUE="значение по умолчанию". Это значение выдаётся в поле ввода при открытии формы, и его часто используют как комментарий, подсказывающий, что следует вводить в это поле. Для большинства тегов это необязательный параметр. Для элементов типа CHECKBOX и RADIO этот атрибут обязателен. Имя в атрибуте NAME принимает значение, указанное в атрибуте VALUE выбранных вариантов.

– ROWS="число" и COLS="число". Эти атрибуты в контейнере <TEXTAREA> … </TEXTAREA> задают размеры окна для ввода текстов заранее неизвестной длины. ROWS определяет число строк, COLS – число символов, которое размещается на одной строке поля. Если текст не укладывается в отведённые размеры, в окне появляется полоса прокрутки.

– SELECTED – указывает, какой пункт выпадающего списка в контейнере <SELECT> … </SELECT> будет выбран по умолчанию.

– MULTIPLE – указывает, что в выпадающем списке можно выбрать несколько пунктов одновременно. Если в теге не указан этот параметр, то разрешается выбрать только один пункт.

– SIZE="число" – сколько строк должно быть одновременно видно в выпадающем списке. Если в списке собрано большее количество строк, появляется полоса прокрутки.

Таблица 1.

Тип поля	**Значение атрибута TYPE**	**Вид на экране**
Текстовое поле	TEXT	(текст пользователя)
Поле пароля	PASSWORD	**********
Флажок	CHECKBOX	☐ ☐ ☑ ☐ ☑
Переключатель	RADIO	○ ○ ◉ ○
Кнопка, заказывающее действие пользователя	BUTTON	(значение атрибута value)

Кнопка для отправки данных для обработки	SUBMIT	(значение атрибута value)
Кнопка для очистки формы от данных пользователя	RESET	(значение атрибута value)

Пример 1 Форма с разными типами элементов управления.

```
<html>
<head>
<title> Пример формы для анкеты </title>
</head>
<body>
<form >
<h2> Пожалуйста, примите участие в нашем опросе! </h2>

Ваше имя: <input type=text name="nic" size=30> <br>
Ваш возраст: <input type=text name="age" size=5> <br>

<h4> Нравится ли Вам наш сайт? </h4>
<input type=radio name=otnosh value=da> Да <br>
<input type=radio name=otnosh value=net> Нет <br>
<input type=radio name=otnosh value=so> Так себе <br>
<h4> К какой группе посетителей Вы себя относите? </h4>
<input type=radio name=group value=sl> Случайные <br>
<input type=radio name=group value=pos> Постоянные <br>

<h4> Каким из сервисов Вы пользуетесь чаще всего? </h4>
<input type=checkbox name=serv value=www> WWW <br>
<input type=checkbox name=serv value=email> E-mail <br>
<input type=checkbox name=serv value=ftp> FTP <br>

<h4> Каким браузером Вы пользуетесь? </h4>
  <select   name=brow > <br>
    <option select> Internet Explorer <br>
    <option > Netscape Navigator <br>
    <option > Opera <br>
    <option > Google <br>
  </select> <br>

<h4> Какую информацию Вы хотели бы видеть на нашем сайте? </h4>
```

<textarea name=inf rows=4 cols=50> Впишите сюда свои пожелания </textarea>

<pre> <input type=submit value="Отправить"> <input type=reset value="Очистить форму"> </pre>
</form >
</body>
</html>

1.2. Атрибуты открывающего тега формы <FORM>

ACTION="URL скрипта, обрабатывающего данные из формы". Задаёт адрес, по которому направляются данные из формы после щелчка на кнопке SUBMIT. Обычно программы скриптов хранятся в каталоге с зарезервированным именем *cgi-bin*. Пусть скрипт хранится в таком каталоге на сервере www.mycompany.ru и имеет имя *bd.exe*, тогда указание на адрес скрипта имеет следующий вид:

ACTION="http://www.mycompany.ru/cgi-bin/bd.exe".

Если скрипт хранится на том же компьютере, что и форма, которую он обрабатывает, то в URL указывают только название скрипта и путь к нему (лучше относительный).

METOD="способ передачи данных из формы в обрабатывающую программу". Все введённые в форму данные кодируются в браузере. Полученная цепочка символов либо добавляется после символа "?" в конце URL, указанного в атрибуте ACTION, либо для неё создаётся специальный файл, который передаётся по указанному URL как самостоятельный объект. Первый способ называется *метод GET*, второй – *метод POST*. Метод GET можно применять, если совокупная длина URL и данных не превышает 1Кб, метод POST – для любого количества информации, собранной в форме.

Способ кодировки данных таков:

- латинские буквы остаются неизменными;
- пробелы заменяются знаком плюс;
- знаки препинания, русские буквы и другие проблемные символы заменяются шестнадцатеричными кодами со знаком процента перед ними;
- данные разбиваются на пары "имя=значение" и перечисляются через символ "&".

2. Скрипты

Программа, которая обрабатывает данные, введённые в форму, называется термином *скрипт*. В зависимости от целей создания скрипты могут располагаться и обрабатывать информацию либо на компьютере, адрес которого указан в заголовке формы, либо в самом HTML-документе. Например, при заполнении анкеты для регистрации в почтовой программе Вы вводите логин и пароль. Скрипт, который проверяет, правильно ли Вы составили пароль, может находиться в коде самой анкеты, а скрипт, проверяющий логин, должен находиться на почтовом сервере, так как он проверяет, не зарегистрирован ли уже такой логин другими пользователями.

Скрипты, расположенные вне HTML-документа, называются *серверными скриптами* и могут писаться на любом языке программирования, например, Java, Perl, C и других. Адрес такого скрипта указывается в заголовке формы в атрибуте ACTION. Скрипты, встроенные в HTML-документ, называют *клиентскими*. Для такого скрипта адрес и, следовательно, ACTION, указывать не надо.

Клиентские скрипты обычно пишут на языке JavaScript или его модификациях. Например, Microsoft разработала для браузера Explorer две модификации этого языка: JScript и VBScript. Они содержат одни и те же операторы, которые нужны для написания скриптов, но различаются техникой их написания: Jscript использует технику базового языка JavaScript, VBScript – технику языка VBA, который используется для написания программ MS Office. Мы занимаемся только клиентскими скриптами.

2.1. Добавление скриптов на веб-страницу

Клиентский скрипт помещается в контейнер

<SCRIPT LANGUAGE=язык скрипта> текст программы </SCRIPT>

Этот контейнер можно расположить в любом месте HTML-документа, но обычно его размещают в заголовке <HEAD> … </HEAD>.

С помощью команд JavaScript Вы можете осуществлять взаимодействие с браузером. Например, команда document.write('Доброе утро!'); сообщает браузеру, что хотите вывести на страницу текст "'Доброе утро!".

Точка с запятой ставится в конце каждой команды. В случае если Вы не поставили точку с запятой, браузер автоматически подставит ее, если следующая команда будет написана на новой строке. В отличие от HTML JavaScript чувствителен к регистру букв, то есть различает маленькие и большие буквы.

Пример 2:

```
<html>
<body>
<script type="text/javascript">
document.write("Данный текст выведен на
страницу с помощью JavaScript."+ "<br>");
document.write("Тег br нужен для перехода на
новую строку.");
</script>
</body>
</html>
```

Объяснение примера 2:

- Атрибут **type** тэга <script> говорит браузеру о том, какой скриптовый язык мы встраиваем. Например, если мы встраиваем скрипт, написанный на JavaScript, атрибут **type** должен иметь значение **text/javascript**;
- **document.write()** - команда JavaScript, которая позволяет выводить произвольное содержимое на страницу;
- Разместив document.write() между тэгами <script> и </script> мы указываем браузеру обрабатывать ее как команду JavaScript, поэтому после загрузки страницы браузер выведет: *"Данный текст выведен на страницу с помощью JavaScript."*

Задание. Выведите следующие строчки текста на страницу с помощью JavaScript:

1. Ваши фамилия, имя, отчество.
2. Дата Вашего рождения и название населенного пункта, в котором Вы родились.
3. Номер группы, в которой Вы учитесь.

2.2. События

Скрипт обычно запускается как реакция на некоторое действие (***метод***) с документом или его компонентами. Распознаваемые действия называются *событиями*. Например, для элементов формы вид события и способ реакции на него задаются как дополнительный атрибут тега, создающего этот элемент. Имя атрибута, определяющего событие, начинается с приставки on, после которой идёт английское название события. Основные события, связанные с мышью, приведены в табл. 2.

Таблица 2

Атрибут события	Содержание действия	Объекты, в которых распознаётся событие
onClick	Щелчок на элементе или гиперссылке	Button, radio, checkbox, submit, reset, link
onDblClick	Двойной щелчок на элементе или гиперссылке	Button, radio, checkbox, submit, reset, link
onMouseUp	Отпускается кнопка мыши	Document, button, link
onMouseDown	Утапливается кнопка мыши	Document, button, link
onMouseOut	Выход курсора из области изображения или связи (гиперссылки)	Area, link
onMouseOver	Перемещение курсора над связью (гиперссылкой)	Link

Кроме этих событий в языке JavaScript определено ещё много других событий, которые связаны действиями над элементами форм и других объектов, входящих в HTML-документ.

Если реакцию на событие можно описать одним оператором JavaScript, то для него можно не оформлять скрипт, а указать этот оператор прямо как значение атрибута события. Если реакция задаётся несколькими операторами, то её оформляют в виде скрипта и в атрибуте ссылаются на него.

Пример 3. Определить размер премии по введённому доходу (реакция – один оператор).

```
<html>
<head>
<title> форма с вычислением премии (без скриптов) </title>
</head>
<body>
<form name=form1>
Вычисление размера премии <br>
Введите размер дохода и нажмите кнопку "Вычислить" <br>
Доход: <input type=text name=doh size=10 > <br>
<input type=button value="Вычислить" onclick="form1.rez.value=0.25*form1.doh.value"> <br>
```

```
Премия: <input type=text name=rez size=10> <br>
<input type=reset value="Очистить форму">
</form >
</body>
</html>
```

Рассмотрим подробнее выражение

```
onclick="form1.rez.value=0.25*form1.doh.value"
```

2.3. Объекты JavaScript

JavaScript относится к классу объектно-ориентированных языков. Это значит, что его операторы манипулируют не данными, введёнными с клавиатуры или рассчитанными в процессе работы, а *объектами*. В роли объекта в каждом языке программирования может выступать, например, программа, создающая на экране нужный пользователю образ, или файл, содержащий нужные сведения. В нашем случае в роли объекта выступает кнопка "Вычислить". Применяя к объекту оператор, мы формулируем только задание, что и с чем мы хотим сделать. Правая часть оператора "=" – это что надо сделать, левая – с чем.

Каждый объект описывается какими-то характеристиками. Они называются термином *свойства*. Объекты, характеризующиеся одинаковым наборов свойств, образуют *классы*. Объекты могут делиться на более мелкие классы, то есть могут иметь структуру. Каждый более мелкий объект называется *потомком*, а исходный объект, из которого он выделен, называется *предком*. Потомки характеризуются и теми свойствами, которые имели предки, и рядом новых. Это свойство обозначается термином *наследование*. Группа потомков, имеющих одинаковый набор характеристик (но не их значений!), отличающих их от предка, называется *подклассом* потомка.

В нашем примере использован один элемент класса Forms. В форме использовано четыре элемента: текстовое поле, кнопка, текстовое поле, кнопка. Все они вне зависимости от типа образуют подкласс elements и нумеруются по порядку. В операторах надо указывать через точку полный путь к объекту и то свойство, с которым надо работать. Стандартный способ указания на объект – Название класса[порядковый номер элемента данного класса в документе]. Нумерация начинается с "0". Если объекту присвоено имя, то его можно использовать вместо стандартного обозначения. Записи, приведённые ниже, эквивалентны, но использование имён делает обращение к объекту более понятным:

```
form1.rez.value
```

```
forms[0].elements[2].value
```

Таким образом, совокупность объектов, с которыми манипулирует язык, образует чёткую иерархическую структуру. Пример: в JavaScript иерархия объектов одной ветви такая: window (окно браузера)→ document (открытый в браузере документ)→ form (бланк, созданный в документе)→ (элемент управления). То, что слева от стрелки – предок того, что справа, то, что справа – потомок того, что слева.

В операторах, манипулирующих с объектами, следует при обращении к объекту перечислять через точку всех его предков в порядке старшинства. Если же вы уже находитесь внутри какого-либо объекта, то можно указывать только ту часть цепочки, которая доведёт Вас до нужного объекта от этого места. В приведённом примере мы начали путь только с формы, так как в окне браузера открыт только один документ.

Пример 4. Определить сумму баллов и среднюю оценку за сессию у студента.

```
<html>
<head>
<title> Средняя оценка и сумма баллов (со скриптом) </title>
<script language=Javascript>
function ball(){
 var a1=Number(form1.nam1.value)
 var a2=Number(form1.nam2.value)
 var a3=Number(form1.nam3.value)
 var a4=Number(form1.nam4.value)
 form1.sumb.value=(a1+a2+a3+a4)
 form1.midlb.value=(a1+a2+a3+a4)/4
}
</script>
</head>
<body>
<h2> Средняя оценка по результатам экзаменов </h2>
<form name="form1">
<h4> Введите оценки и нажмите кнопку "Вычислить" </h4>
математика: <input type=text name=nam1 size=2 > <br>
физика: <input type=text name="nam2" size=2 > <br>
информатика: <input type=text name="nam3" size=2 > <br>
Макроэкономика: <input type=text name="nam4" size=2 > <br>

<input type=button value=Вычислить onClick=ball()> <br>

Средняя оценка: <input type=text name=midlb size=4> <br>
сумма баллов: <input type=text name=sumb size=4> <br>
```

```
<input type=reset value="Очистить форму">
</form >
</body>
</html>

<script language=Javascript>
function ball(){
 var a1=Number(form1.nam1.value)
 var a2=Number(form1.nam2.value)
 var a3=Number(form1.nam3.value)
 var a4=Number(form1.nam4.value)
 form1.sumb.value=(a1+a2+a3+a4)
 form1.midlb.value=(a1+a2+a3+a4)/4
}
</script>
```

Разберём структуру скрипта.

2.4. Функции

При создании программы обычно весь алгоритм разделяют на логически независимые куски и каждую оформляют как отдельный блок, называемый подпрограммой. В зависимости от функций выделенного блока и способа его оформления различают разные варианты подпрограмм (функции, процедуры, модули и т.п.). В нашем скрипте используется подпрограмма типа функция (это основной элемент языка JavaScript). В простейшем виде описание функции таково:

Function имя() {тело функции}

Тело функции – это программа того фрагмента, который выделен из общего алгоритма. Тело функции выделяется фигурными скобками. В теле функции нельзя размещать описание другой функции.

Скобки после имени оставляют пустыми, если всё, что нужно для расчётов, задано в теле, и в момент запуска функции на работу никаких дополнительных сведений для расчётов не требуется. У нас названия всех элементов, с которыми работает программа, указаны в явном виде так, как они введены в форме, поэтому у функции ball() скобки пустые. Если же для одна и та же функция запускается из разных мест формы, и при этом на каждом месте она должна использовать разные данные, то в скобках через запятую перечисляют имена переменных, обозначающих в теле функции данные, которые при новом обращении к ней будут другими. Значения этих переменных в момент запуска функции на работу из того или иного места указывают в скобках после имени в том же порядке, в котором они перечислены в описании. Переменные, которые указываются в скобках

при описании функции, называются *формальными параметрами*, значения, который указываются для них в тех же скобках при вызове функции на работу, называются *фактическими параметрами*. В нашем скрипте использована функция с параметрами из стандартной библиотеки JavaScript

Number(a.value)

Number – название функции, которая делает преобразует цифровые символы, введённые в текстовое поле, в число.

a.value – формальный параметр. Он условно обозначает тот текст, который следует перекодировать в число.

form1.nam1.value, form1.nam2.value и т.д. – фактические параметры, которые надо перекодировать для дальнейших расчётов.

Оператор *var* служит для объявления названий, которые используются как переменные. В нём же можно задать их начальные значения, если они известны. Тип переменной определяется в момент присвоения этой переменной значения. Если тип значения изменился в процессе выполнения, то меняется и тип переменной (например, одна и та же переменная может служить и для записи числового результата, и для вывода пояснительного текста, если получить результат не удалось). У нас переменные a1, a2, a3, a4 должны фигурировать в программе как числовые значения оценок, но введены они в текстовые поля. Поэтому используется стандартная функция *Number*, которая делает перекодировку строковых значений в числовые. В скобках функции указаны *фактические параметры*, то есть какие конкретно значения должна обрабатывать эта функция по своему алгоритму при каждом новом обращении к ней.

Далее идут два оператора присваивания, в которых справа указываются выражения, которые надо вычислить, а слева – объекты и их свойства, в которых надо хранить полученные значения выражений.

Программа скрипта состоит из одной функции, поэтому скрипт закрывается.

В теле формы скрипт запускается в ответ на щелчок по кнопке вычислить. Так как точные названия всех объектов и их свойств уже указаны в скрипте, формальных имён, которые надо заменять на фактические, нет, и скобки при вызове опять пустые:

<input type=button value=Вычислить onClick=ball()>

Пример 5. ***Самостоятельная работа.*** Составить скрипт, который содержит две функции: sumball() для расчёта суммы баллов и middlball() – для среднего балла. В форме предусмотреть отдельные кнопки для запуска каждой функции, чтобы пользователь мог заказать либо один нужный ему результат, либо оба.

<html>
<head>

```
<title> Средняя оценка и сумму баллов (со скриптом) </title>
<script language=Javascript>
function sumball(){
 var a1=Number(form1.nam1.value)
 var a2=Number(form1.nam2.value)
 var a3=Number(form1.nam3.value)
 var a4=Number(form1.nam4.value)
 form1.sumb.value=(a1+a2+a3+a4)
 form1.midlb.value=(a1+a2+a3+a4)/4
}
function middlball(){
 var a1=Number(form1.nam1.value)
 var a2=Number(form1.nam2.value)
 var a3=Number(form1.nam3.value)
 var a4=Number(form1.nam4.value)
form1.midlb.value=(a1+a2+a3+a4)/4
}
</script>
</head>
<body>
<h2> Средняя оценка по результатам экзаменов </h2>
<form name="form1">
<h4> Введите оценки и нажмите кнопку "Вычислить" </h4>
математика: <input type=text name=nam1 size=2 > <br>
физика: <input type=text name="nam2" size=2 > <br>
информатика: <input type=text name="nam3" size=2 > <br>
Макроэкономика: <input type=text name="nam4" size=2 > <br>

<input type=button value="Вычислить сумму" on-Click=sumball()> <br> <br>
<input type=button value="Вычислить среднее" on-Click=middlball()> <br> <br>

Средняя оценка: <input type=text name=midlb size=4> <br>
сумма баллов: <input type=text name=sumb size=4> <br>
<input type=reset value="Очистить форму">
</form >
```

```
</body>
</html>
```

Пример 6. ***Самостоятельная работа.*** Составить скрипт, который по нажатию кнопки "Вычислить" выдаёт суммарный доход за квартал, налог, который следует заплатить с этого дохода, и сумму, которая останется на руках после этого.

```
<html>
<head>
<title> Доход за квартал (со скриптом)</title>
<script language=JavaScript>
function doh(){
 var a1=Number(form1.nam1.value)
 var a2=Number(form1.nam2.value)
 var a3=Number(form1.nam3.value)
 form1.sumd.value=(a1+a2+a3)
 form1.nal.value=(a1+a2+a3)*0.13
 form1.nar.value=(a1+a2+a3)*(1-0.13)
}
</script>
</head>
<body>
<h1>Расчёт доходов за квартал </h1><br>
<form name="form1">
Введите доходы по месяцам и нажмите кнопку "Вычислить"
<br><br>
1-й месяц: <input type=text name=nam1 size=8 > <br>
2-й месяц: <input type=text name="nam2" size=8 > <br>
3-й месяц: <input type=text name="nam3" size=8 > <br>
<input type=button value=Вычислить onClick=doh()> <br>
Общий доход: <input type=text name=sumd size=8> <br>
Налог: <input type=text name=nal size=8> <br>
На руки: <input type=text name=nar size=8> <br>
<input type=reset value="Очистить форму">
</form >
</body>
</html>
```

2.5. Выражения

Выражения строятся из констант (*литералов*), переменных и знаков операций. К моменту вычисления все операнды в выражении должны

иметь значения. В зависимости от типа результата выражения делятся на числовые, строковые (цепочка символов) и логические (*булевы*).

Обозначения для выполнения арифметических действий, применяемые в языке JavaScript, приведены в табл. 3.

Таблица 3

Операция	**Обозначение**	**Пример**	**Операция**	**Обозначение**	**Пример**
Сложение	+	a + 5	Умножение	*	a * b
Вычитание	–	a – 5	Деление	/	a / b
Увеличение на единицу (*инкремент*)	+ +	a = a + 1 → a + +	Остаток от деления	%	a % b
Уменьшение на единицу (*декремент*)	– –	a = a – 1 → a – –	Возведение в степень	**	

В текстовых выражениях для сцепления отдельных фрагментов используется знак плюс, постоянные операнды, заданные явно, выделяются двойными кавычками, переменные – указываются именем. Пример. Пусть переменная *a* содержит имя человека. Тогда выражения для обращения к нему имеет вид: "Уважаемый" + *a*

Логические выражения (*операции сравнения*) – это сравнение значений двух операндов. Типы сравнения и их обозначения приведены в табл. 4. Результат логического выражения обозначается одним из двух слов: при правильном соотношении – *true*, при неправильном – *false*. В операторах эти слова заменяются числовыми величинами: *true*→ 1, *false* → 0.

Таблица 4

Тип сравнения	**Обозначение**	**Пример**
Больше	>	a > 5
Больше или равно	>=	a >= 5
Меньше	<	a < 5
Меньше или равно	<=	a <= 5
Не равно	!=	a != b

Равно (пробел между знаками равно не нужен!)	==	a == b

Часто требуется выполнять комплексную проверку сразу нескольких простых условий. Это делается с помощью логических операций. Их смысл и обозначения приведены в табл. 5.

Таблица 5

Операция	**Обозначение**	**Пример**
Отрицание (*инверсия*)	!	!a – если a= *true*, то !a= *false* и наоборот
Логическое умножение (*конъюнкция*, операция "*и*")	&&	A && b = *true* только тогда, когда оба операнда имеют значение *true* , в противном случае *false*
Логическое сложение (*дизъюнкция*, операция "*или*")	||	a || b = *true*, когда хотя бы один операнд имеет значение *true*, в противном случае *false*

Пример. Выражение ((a2>160)&&((a3>50)&&(a3<120)) соответствует следующей цепочке неравенств: a2>160 и одновременно 50<a3<120. Выражение ((a1<2)||(a1>5)||(a2<2)||(a2>5)) будет иметь значение *true* если выполняется хотя бы одно из этих соотношений: a1<2, a1>5, a2<2, a2>5.

Числовые операнды в выражениях сравниваются по правилам арифметики, Текстовые строки – по ASCII-кодам начиная с левого конца строк. В ASCII-кодировке сначала идёт пробел, затем цифры, затем латинские прописные буквы, латинские строчные, кириллица прописная, кириллица строчная и, наконец, специальные символы для оформления таблиц. Примеры выражений отношения:

"abcd" = = "abc" → *false*; "abcd" = = "abcd"→ *true*; "abcd" > " bcd"→ *false*; "abc" < "abcd"→ *true*; "11b" < "10b" → *false*.

2.6. Операторы для создания вариантных сценариев

Кроме оператора арифметического присваивания в программах часто используются операторы условного перехода и циклов.

Операторы условного перехода применяют в тех случаях, когда действия, которые должен выполнять браузер, зависят от результатов проверки некоторых условий. Если организуется выбор одного из двух вариантов, удобно воспользоваться оператором *if*. Общий вид этого оператора:

if (условное выражение)

{действия, которые надо выполнить при правильном условии (*true*)}

else

{ действия, которые надо выполнить при неверном условии (*false*)}

Допускается сокращённая форма этого оператора:

if (условное выражение)

{действия, которые надо выполнить при правильном условии (*true*)}

Пример 7. Создать форму, которая определяет, годен или нет пользователь к военной службе. Она запрашивает возраст пользователя, рост и вес. Если ему от 18 до 28 лет, рост больше 160 см, а вес от 50 до 120 кг, то выдаётся ответ "Годен к призыву", в противном случае – "Не годен к призыву " или "Вы не подлежите призыву ".

```
<html>
<head>
<title> Подлежите ли Вы призыву </title>

<script language=Javascript>
function osm()
{
 var a1=Number(form1.age.value)
 var a2=Number(form1.rost.value)
 var a3=Number(form1.ves.value)
 if((a1<18)||(a1>28))
   {form1.rez.value="Вы не подлежите призыву"}
 else
                                  {if((a2>160)&&((a3>50)&&(a3<120)))
                                    { form1.rez.value="Годен к призыву"}
    else{ form1.rez.value="Не годен к призыву"}
   }
}
 </script>
</head>
<body>
<h1> Подлежите ли Вы призыву?  </h1><br>
<form name=form1>
Введите свои данные и нажмите кнопку "Результат" <br><br>
Возраст: <input type=text name=age size=8 > <br>
Рост: <input type=text name=rost size=8 > <br>
Вес: <input type=text name=ves size=8 > <br>
<input type=button value= onClick=osm()> <br> <br>
```

```
Результат: <input type=text name=rez size=30 > <br> <br>
<input type=reset value="Очистить форму">
</form >
</body>
</html>
```

Зелёным цветом отмечен оператор, который выполняется для людей, возраст которых находится за пределами диапазона от 18 до 28 лет (непризывной возраст), жёлтым – для потенциальных призывников, так как для этой категории условие в первом операторе *if* имеет значение false. Эта ветвь содержит дополнительный оператор *if*, который проверяет, подходит ли человек для призыва по росто-весовым показателям.

Пример 8. Составить скрипт, который при наведении мыши на одну картинку будет увеличивать картинки, при наведении на другую – уменьшать их.

```
<html>
<head>
<title> эффект приближения изображения </title>
<script >
 var w=200
 var h=200
  //увеличение изображения
function grpict()
 {
       if (w<300)
   {w=w+1; pict1.width=w;pict2.width=w
    h=h+1; pict1.height=h;pict2.height=h
       }
  setTimeout("grpict()",10)
 }
  //уменьшение изображения
function lspict()
 {
       if (w>50)
   {w=w-1; pict1.width=w;pict2.width=w
    h=h-1; pict1.height=h;pict2.height=h
   }
  setTimeout("lspict()",10)
 }
</script>
</head>
<body>
<center>
```

<h2> При наведении курсора мыши на первый рисунок изображения начинают приближаться к нам, на второй - уеньшаться </h2>

<img src="верблюд.jpg" name=pict1 width=200 height=200 onMouseOver="grpict()">

<img src="заяц на траве.jpg" name=pict2 width=200 height=200 onMouseOver="lspict()">

</body>

</html>

В тех случаях, когда при решении задачи требуется выбрать один вариант из нескольких возможных, удобно воспользоваться оператором *switch*. Общий вид этого оператора:

*switch(*выражение*)*
{ *case* L1: S1; *break*;
case L2: S2; *break*;
……
case Ln: Sn; *break*;
default: S
}

Здесь L1, L2, …, Ln – значения выражения, на которые оператор *switch* должен реагировать действиями S1, S2, …, Sn соответственно. S – действия, которые следует выполнять, если выражение не равно ни одному из перечисленных значений. Если таких действий нет, допускается сокращённая форма этого оператора:

*switch(*выражение*)*
{ *case* L1: S1; *break*;
case L2: S2; *break*;
……
case Ln: Sn; *break*;
}

Можно опустить ключевое слово *break*. В этом случае будет выполняться не только выбранная ветвь, но и следующие за ней до тех пор, пока не встретится *break* или *default*.

Пример 9. Составить форму, в которой выдаётся стоимость подписки в зависимости от длины подписного периода.

<html>
<head>
<title> Вычисление стоимости животного</title>
<script language=JavaScript>

```
function def()
{
var n=form1.anim.value
var s
switch (n)
        {case "рыбка": s="60 р.";break;
         case "киса": s="168 р.";break;
         case "собака": s="312 р.";break;
         case "удав": s="$540 ";break;
         default: s="Это животное отсутствует"
        }
form1.st.value=s
}
</script>
</head>
<body>
<form name=form1>
<h2> Цены на животных </h2>
Введите название животного: <input type=text name="anim" size=10 >
<br>
<input type=button value=Определить onClick=def()> <br>
Цена: <input type=text name=st size=35> <br>
<input type=reset value="Очистить форму">
</form >
</body>
</html>
```

2.7. Обработка данных из элементов управления переключатель, флажок, меню выбора

Эти элементы управления отличаются от элемента типа *text* тем, что одно и то же имя присваивается группе элементов. Эти элементы рассматриваются как массив, и к ним в программе обращаются, добавляя к имени группы индекс. Индекс заключается в квадратные скобки. Нумерация элементов начинается с нуля. Ниже приведены примеры программной обработки элементов этих типов.

Пример 10. В форме с помощью элемента управления *переключатель* выбирается животное для покупки и выдаётся его стоимость.

```
<html>
<head>
<title> форма с переключателем   </title>
<script>
function chek()
```

```
{
 var s
 var t=form1.anim.value
 if(t=="") {s="Вы не выбрали животное"}
 else
 {if(anim[0].checked){s="рыбки – 150 р."}
  else
    {if (anim[1].checked) {s="кошка – 500 р."}
     else { s="собака – 2500 р."}
    }
  }
   form1.rez.value=s
}
</script>
</head>
<body>
<form name=form1 >
<h2> Сделайте выбор животного: </h2>
   < input type=radio name=anim > рыбки <br>
   < input type=radio name=anim > кошка <br>
   < input type=radio name=anim > собака <br> <br>
<input type=button value="Рассчитать стоимость" onClick="chek()">
<br><br>
Стоимость покупки:<input type=text name=rez><br><br>
<input type=reset value="Очистить форму">
</form >
</body>
</html>
```

Пример 11. В форме с помощью элемента управления *флажок* выбираются животные для покупки, и выдаётся их общая стоимость. Так как элемент *флажок* допускает одновременный выбор нескольких значений, отбор выбранных вариантов удобно делать с помощью оператора цикла:

for(способ изменения счётчика циклов) {тело цикла}

В примере 10 переменная *i* играет роль счётчика циклов и должна изменяться от 0 до 2 с шагом 1. Телом цикла является оператор *if*. При каждом прохождении тела цикла определяется, выбран ли очередной элемент из группы флажка *anim*, и если он выбран, стоимость этого элемента (значение атрибута *value*) добавляется к стоимости покупки (переменная *s*). Окончательная стоимость выдаётся в текстовое поле *rez*.

```
<html>
```

```
<head>
  <title> форма с флажком   </title>
<script>
function chek()
{
 var s=0
var i
 for (i=0;i<=2;i++)
    { if(form1.anim[i].checked){s=s+Number(form1.anim[i].value)}}
 form1.rez.value=s
}
</script>
</head>
<body>
  <form name=form1 >
  <h2> Сделайте выбор: </h2>
  <input type=checkbox name=anim value=150> рыбки <br>
  <input type=checkbox name=anim  value=500> кошка <br>
  <input type=checkbox name=anim  value=2500> собака <br> <br>
  <input type=button value="Рассчитать стоимость" onClick="chek()">
<br><br>
  Стоимость покупки:<input type=text name=rez><br><br>
  <input type=reset value="Очистить форму">
 </form >
</body>
</html>
```

Пример 12. В форме с помощью элемента управления *выпадающее меню* выбираются животные для покупки и выдаётся их стоимость при одиночном и множественном выборе.

```
<html>
<head>
<title> форма с меню</title>
<script>
//функция для одиночного выбора – эквивалент переключателя
function chek1()
{
 var s
  if(form1.anim1[0].selected){s="рыбки – 150 р."}
  else
    {if (form1.anim1[1].selected) {s="кошка – 500 р."}
     else
```

```
        {if(form1.anim1[2].selected) {s="собака – 2500 р."}
         else
           {s="Вы не выбрали животное"}
        }
      }
      form1.rez1.value=s
    }
    //функция для множественного выбора – эквивалент флажка
    function chek2()
    {
     var s=0
     for (var i=0;i<=2;i++)
       if(form2.anim2[i].selected){s=s+Number(form2.anim2[i].value)}
     form2.rez2.value=s
    }

    </script>
    </head>
    <body>
    <form name=form1 >
    <h2> Сделайте выбор: </h2>
      <select name=anim1> <br>
        <option > рыбки <br>
        <option > кошка <br>
        <option  selected> собака <br>
      </select> <br> <br>
    <input type=Button value="Рассчитать стоимость" onClick="chek1()">
<br><br>
    Стоимость покупки:<input type=text name=rez1><br><br>
    <input type=reset value="Очистить форму">
    </form >

    <form name=form2 size=3 multiple >
    <h2> Сделайте выбор нескольких животных: </h2>
      <select name=anim2  size=3 multiple> <br>
        <option value=150> рыбки <br>
        <option value=500> кошка <br>
        <option value=2500 selected> собака <br>
      </select> <br> <br>
    <input type=Button value="Рассчитать стоимость" onClick="chek2()">
<br><br>
    Стоимость покупки:<input type=text name=rez2><br><br>
    <input type=reset value="Очистить форму">
```

```
</form >
</body>
</html>
```

2.8. Стандартные функции для обеспечения диалога пользователя с браузером

Приведённые ниже функции выдают на экран окна разных видов, которые облегчают программирование диалога и выбор варианта развития сценария.

— alert(("текст сообщения") – вывод окна с сообщением и кнопкой <OK>;

– confirm("текст сообщения") – вывод окна с сообщением и кнопками <OK> и <Отмена>. Обычно текст сообщения подбирается так, что при согласии с ним нажимают <OK>, тогда вырабатывается признак *true*, если с текстом сообщения не согласны, то выбирают кнопку <Отмена> и вырабатывается признак *false*. Это используют в условных операторах для того, чтобы пользователь мог выбрать разные варианты действий по своему желанию;

– prompt("текст сообщения","текстовое поле с подсказкой") – в текстовое поле пользователь может ввести свои данные.

Пример 13. В скрипте примера 3 создать запрос имени пользователя, приветствие и реплики, оценивающие средний балл студента.

```
<script language=Javascript>
function ball()
{
 var a1=Number(form1.nam1.value)
 var a2=Number(form1.nam2.value)
 var a3=Number(form1.nam3.value)
 var a4=Number(form1.nam4.value)
 form1.sumb.value=(a1+a2+a3+a4)
 form1.midlb.value=(a1+a2+a3+a4)/4
 if(confirm("Оценить Ваши успехи?")){
 a=4
 if(form1.midlb.value<a)
   {alert("Вы плохо сдали сессию")}
 else
   {alert("Всё хорошо. Вы заслужили отдых")}
 }
 else{alert("Судите себя сами")}
```

```
}
var a
a=prompt("Как Вас зовут?","Введите сюда своё имя ")
alert("Привет, "+a+". Введите свои оценки и нажмите кнопку Вычислить")
</script>
```

Пример 14. Составить скрипт, который при наведении мыши на одну картинку будет увеличивать картинки, при наведении на другую – уменьшать их.

```
<html>
<head>
<title> эффект приближения изображения </title>
<script >
 var w=200
 var h=200
  //увеличение изображения
function grpict()
 {
       if (w<300)
   {w=w+1; pict1.width=w;pict2.width=w
    h=h+1; pict1.height=h;pict2.height=h
       }
  setTimeout("grpict()",10)
 }
  //уменьшение изображения
function lspict()
 {
       if (w>50)
   {w=w-1; pict1.width=w;pict2.width=w
    h=h-1; pict1.height=h;pict2.height=h
   }
  setTimeout("lspict()",10)
 }
</script>
</head>
<body>
<center>
<h2> При наведении курсора мыши на первый рисунок изображения начинают приближаться к нам, на второй - уеньшаться </h2>
<img src="верблюд.jpg" name=pict1 width=200 height=200 onMouseOver="grpict()">
<img src="заяц на траве.jpg" name=pict2 width=200 height=200 onMouseOver="lspict()">
```

```
</center>
</body>
</html>
```

Пример 15. ***Самостоятельная работа***. Для примера 6 создать скрипт, который определяет, годен или нет пользователь к военной службе. Запрашивается возраст пользователя, и, если ему от 18 до 28 лет, дополнительно – вес и рост. Если рост больше 160 см, а вес от 50 до 120 кг, то выдаётся ответ "Годен", в противном случае – "не годен". Все реплики выдавать через диалоговые окна.

```
<html>
<head>
<title> Подлежите ли Вы призыву </title>

<script language=Javascript>
function osm()
{
 var a1=Number(form1.age.value)
 var a2=Number(form1.rost.value)
 var a3=Number(form1.ves.value)
 if((a1<18)||(a1>28))
   {alert("Вы не подлежите призыву")}
 else
   {if((a2>160)&&((a3>50)&&(a3<120))){alert("Годен к призыву")}
    else{alert("Не годен к призыву")}
   }
}
 </script>
</head>
<body>
<h1> Подлежите ли Вы призыву?  </h1><br>
<form name="form1">
Введите свои данные и нажмите кнопку "Результат" <br><br>
Возраст: <input type=text name=age size=8 > <br>
Рост: <input type=text name=rost size=8 > <br>
Вес: <input type=text name=ves size=8 > <br>
<input type=button value=Результат onClick=osm()> <br> <br>
<input type=reset value="Очистить форму">
</form >
</body>
</html>
```

3.Задания для самостоятельной работы

Задание 1.1. Фирма торгует оконными витражами. Создать документ с формой , в которую клиент вводит размеры треугольного витража и сценарием, вычисляющим площадь витража.

Задание 1.2. Фирма торгует оконными витражами. Создать документ с формой , в которую клиент вводит размеры круглого витража и сценарием, вычисляющим площадь витража.

Задание 1.3. Фирма торгует оконными витражами. Создать документ с формой , в которую клиент вводит размеры прямоугольного витража и сценарием ,вычисляющим площадь витража.

Задание 1.4. Фирма торгует продуктами питания. . Создать документ с формой , в которую клиент вводит кол-во продуктов ,и сценарием вычисляющим стоимость заказа. Цена отражена в документе.

Задание 1.5. Банк информирует клиента о состоянии счета через 5 месяцев. Создать документ с формой , в которую клиент вводит сумму вклада и сценарием, вычисляющим прибыль клиента. Процент задан в документе.

Задание 1.6. Турфирма продает путевки. Создать документ с формой , в которую клиент вводит кол-во дней пребывания на курорте и сценарием, вычисляющим стоимость путевки. Цена за день задана в документе.

Задание 1.7. Фирма продает строительные блоки. Создать документ с формой, в которую клиент вводит размеры перегородки и сценарием , вычисляющим кол-во блоков.

Задание 1.8. Фирма предоставляет в аренду автомобили. Создать документ с формой , в которую клиент вводит кол-во часов и сценарием, вычисляющим стоимость заказа. Цена часа задана в документе.

Задание 1.9. Фирма продает керамическую плитку.. Создать документ с формой , в которую клиент вводит размеры стены и сценарием , вычисляющим кол-во плиток. Размер одной плитки задан в документе.

Задание 1.10. Фирма продает стеклопакеты. Создать документ с формой , в которую клиент вводит кол-во стекол , замков , ручек и сценарием , вычисляющим стоимость заказа. Цены заданы в документе.

Задание 1.11. Фирма торгует оконными витражами. Создать документ с формой , в которую клиент вводит размеры витража-паралелограмма и сценарием, вычисляющим площадь витража.

Задание 1.12. Фирма торгует оконными витражами. Создать документ с формой, в которую клиент вводит размеры витража-трапеции и сценарием, вычисляющим площадь витража.

Задание 1.13. Фирма торгует бассейнами. Создать документ с формой , в которую клиент вводит размеры круглого бассейна и сценарием, вычисляющим объем бассейна.

Задание 1.14. Фирма торгует бассейнами. Создать документ с формой , в которую клиент вводит размеры квадратного бассейна и сценарием, вычисляющим объем бассейна.

Задание 1.15. Сайт предоставляет информацию о курсах валют. Создать документ с формой, в которую посетитель вводит сумму в рублях и сценарием, вычисляющим сумму в долларах по курсу.

Задание 1.16. Сайт компьютерной фирмы предлагает ввести в поле формы кол-во символов текста. Создать документ с такой формой и сценарием, вычисляющим объем этого текста в Кбайтах.

Задание 1.17. Сайт медицинской фирмы предлагает ввести в поле формы рост посетителя в см. Создать документ с такой формой и сценарием, вычисляющим идеальный вес. (Самая простая формула рост-100).

Задание 2.1. Написать функцию JavaScript , которая выдает сообщение «ОК» при щелчке мыши на рисунках документа (2 рисунка).

Задание 2.2. Написать функцию JavaScript , которая меняет цвет рамки таблицы при наведении курсора мыши.

Задание 2.3. Написать функцию JavaScript , которая меняет размер букв при выходе курсора мыши из абзаца.

Задание 2.4. Написать функцию JavaScript , которая меняет способ начертания текста абзаца на курсив при щелчке мыши.

Задание 2.5. Написать функцию JavaScript , которая меняет все буквы абзаца на заглавные при наведении курсора мыши на абзац.

Задание 2.6. Написать функцию JavaScript , которая меняет текст в таблице на жирный при наведении курсора мыши.

Задание 2.7. Написать функцию JavaScript , которая меняет цвет фона заголовка при выходе курсора мыши из зоны заголовка.

Задание 2.8. Написать функцию JavaScript , которая меняет курсор на песочные часы при наведении курсора мыши на таблицу.

Задание 2.9. Написать функцию JavaScript , которая меняет размер шрифта ссылки при наведении курсора мыши.

Задание 2.10. Написать функцию JavaScript , которая меняет цвет шрифта ссылки при наведении курсора мыши.

Задание 2.11. Написать функцию JavaScript , которая меняет цвет шрифта заголовка при выходе курсора мыши из зоны заголовка.

Задание 2.12. Написать функцию JavaScript , которая меняет фон абзаца при выходе из него курсора мыши.

Задание 2.13. Написать функцию JavaScript , которая меняет цвет шрифта абзаца при выходе из него курсора мыши.

Задание 2.14. Написать функцию JavaScript , которая меняет размер шрифта абзаца при выходе из него курсора мыши.

Задание 2.15. Написать функцию JavaScript , которая меняет начертание текста заголовка на полужирное при наведении курсора мыши.

Задание 2.16. Написать функцию JavaScript , которая меняет способ начертания шрифта внутри таблицы на курсив при наведении курсора мыши.

Список литературы

1. Дэвид Флэнаган. JavaScript. Подробное руководство. - СПб.: Символ-Плюс, 2012. - 1080 с.
2. Алекс Маккоу. Веб-приложения на javascript. - СПб.: Питер, 2012. - 288 с.
3. Илья Кантор. Современный учебник javascript в 3 книгах. - СПб.: Самиздат, 2017. - 415+353+263 с.
4. Дуглас Крокфорд. JavaScript. Сильные стороны. - СПб.: Питер, 2013. - 176 с.
5. Хавербеке М. Выразительный Javascript , 2-е издание - СПб.: Самиздат, 2015. - 425 с.

Содержание

УНИВЕРСИТЕТ ИТМО

Миссия университета – открывать возможности для гармоничного развития конкурентоспособной личности и вдохновлять на решение глобальных задач.

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

Факультет программной инженерии и компьютерной техники специализируется на подготовке специалистов по разработке компьютерных систем и новейших технологий программирования. Вектор развития факультета определяется прогнозом потребностей компьютерной и программной индустрии через 10-20 лет. Мы готовим специалистов грядущей постинформационной эпохи – эпохи виртуальной реальности, киберфизических систем и интернета вещей.

На факультете умеют и учат создавать компьютеры. Здесь старейшая в России научная и инженерная школа проектирования ЭВМ. И в настоящее время все ведущие сотрудники кафедры – действующие ученые, инженеры и руководители в отрасли компьютерных технологий.

Студентов научат устройству и разработке вычислительных систем от микропроцессоров до компьютеров и смартфонов, от контроллеров устройств интернета вещей до систем управления роботами, от компьютерных кластеров до центров обработки данных.

Здесь создают новые технологии программирования. В основе научных исследований и преподавания лежит перспективная концепция «программной инженерии» – промышленной разработки и поддержки программных систем в рамках единой формализованной системы методологий и технологий.

Студенты приобретают знания и опыт в программировании информационных систем будущего: «облачных» вычислений, искусственного интеллекта, основанного на онтологиях и базах знаний, систем поиска и обработки больших данных (data mining, big data) и интернета вещей.

Выпускники кафедры могут использовать информационные и мультимедиа-технологии в творческой деятельности, в области театрального искусства, киноискусства и телевидения, а также в исследовательской, проектной и практической деятельности, в области средств массовой информации, рекламы, бизнес-коммуникаций, медиа-индустрии.

На факультете осуществляется мультидисциплинарная подготовка методам обработки информации и технологиям программирования. Выпускник кафедры – программист будущего, умеющий применять новейшие нейротехнологии для разработки приложений в области машинного обучения, систем искусственного интеллекта, обработки несистематизированных данных окружающей реальности.

Наши выпускники получают конкурентоспособное образование, способны быстро адаптироваться под новые условия, готовы совершенствовать свою квалификацию и ожидаемы ведущими отечественными и зарубежными предприятиями и фирмами: РЖД, Яндекс, Promt, Luxoft, Газпром, Сбербанк, Intel, Microsoft, Oracle, IBM, AMD, Deutsche Bank – далеко не полный перечень организаций, куда каждый год трудоустраиваются все новые и новые выпускники факультета, и где ценят наше качество образования.

Селина Елена Георгиевна

Организация интерактивного взаимодействия в HTML-документах

Учебно-методическое пособие

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49