ГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

ОСНОВЫ ИНТЕРНЕТ-ТЕХНОЛОГИЙ

Методические рекомендации к практическим занятиям для студентов направления подготовки 12.03.04 «Биотехнические системы и технологии» дневной формы обучения

Могилев 2018

УДК 004.738.5
ББК 32.973.202
О 75

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Автоматизированные системы управления» «4» сентября 2018 г., протокол № 2

Составитель ст. преподаватель В. М. Прудников

Рецензент А. П. Прудников

В методических рекомендациях к практическим занятиям по дисциплине «Основы Интернет-технологий» приведены задания и список литературы для подготовки.

Учебно-методическое издание

ОСНОВЫ ИНТЕРНЕТ-ТЕХНОЛОГИЙ

Ответственный за выпуск	А. И. Якимов
Технический редактор	А. Т. Червинская
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 16 экз. Заказ №

Издатель и полиграфическое исполнение:
Государственное учреждение высшего профессионального образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 24.01.2014.
Пр. Мира, 43, 212000, Могилев.

Содержание

Введение

Целью преподавания дисциплины «Основы Интернет-технологий» является изучение современного программного обеспечения, а также приобретение специальных знаний, умений и навыков, необходимых инженеру в процессе подготовки данных, составление отчетов и научных публикаций по результатам проведенных работ, участие во внедрении результатов в медико-биологическую практику.

Методические рекомендации имеют целью помочь студентам в подготовке и выполнении практических работ по дисциплине.

Практическое занятие № 1. Работа в Internet

Цель занятия:
– повторить и изучить термины и понятия;
– изучить интерфейс и настройки браузеров;
– изучить возможности глобальной сети Internet.

Порядок выполнения

1 Ознакомиться со списком литературы по дисциплине.

2 Изучить и выписать в конспект состав меню браузеров MS Internet Explorer и Mozilla Firefox (либо другого).

3 Найти в Internet или в литературе различные варианты определений (минимум из двух источников) следующих терминов и понятий:

– Internet;
– адрес IP v.4;
– маска IP-адреса;
– DNS (Domain Name System);
– DNS (Domain Name Service);
– WWW (World Wide Web);
– сайт;
– HTML (Hypertext Markup Language);
– CSS (Cascading Style Sheets).

4 Оформить отчет с найденными определениями (со ссылками на источники) в виде документа MS Word с гиперссылками на них по документу. При оформлении отчета использовать возможности MS Word по форматированию текстового документа.

Вопросы для контроля

1 Структура Internet.
2 Состав служб (сервисов) Internet.
3 Состав FQDN (Fully Qualified Domain Name).
4 Структура DNS (Domain Name System).
5 Назначение HTML.
6 Как в браузере:

– настроить параметры соединения с Internet;
– просмотреть HTML-код страницы;
– изменить кодировку для интерпретации страницы;
– установить домашнюю страницу;
– просмотреть список загрузок;
– задать стили пользователя (форматирование пользователя)?

Практическое занятие № 2. Введение в HTML

Цель занятия:
– освоить синтаксис HTML;
– изучить текстовое оформление Web-страниц;
– научиться вставлять изображения в Web-страницы.

Порядок выполнения

1 Ознакомиться с основами HTML.
2 Выполнить задание, полученное у преподавателя.
3 Оформить отчет в виде html-файла.

Основные теоретические положения

Web-документы используют язык HTML (Hypertext Markup Language – язык разметки гипертекста).

HTML-документ – это файл, содержащий обыкновенный текст, структурно размеченный (разметкой). Такой файл может быть подготовлен в произвольном текстовом редакторе (существуют, однако, специальные программы-конверторы и HTML-редакторы).

HTML-документ состоит из содержимого и команд, задающих структуру (разметка).

Каждая управляющая конструкция HTML-документа (тег) должна заключаться в угловые скобки: <тег>. Чаще всего в документе встречаются парные теги (открывающий и соответствующий ему закрывающий), т. к. браузеру необходимо знать область действия тега.

Открывающий и закрывающий теги называются одинаково и отличаются друг от друга только символом «наклонная черта» или «слэш» – «/», который ставится сразу после открывающей угловой скобки закрывающего тега. Закрытие парных тегов выполняется так, чтобы соблюдались правила вложения.

```
<b><i>На этот текст воздействуют два тега</i></b>
```

Кроме того, тег может включать атрибут (атрибуты), дающий дополнительную информацию браузеру, т. е. уточняющий действие тега.

Атрибуты представляют собой дополнительные ключевые слова, отделяемые от ключевого слова, определяющего тег, и от других атрибутов пробелами и размещаемые до завершающего тег символа «>». Значение атрибута отделяется от ключевого слова атрибута символом «=» (знак равенства) и заключается в кавычки.

```
<h1 align="left">
```

Вопросы для контроля

1 Что называют разметкой?
2 Приведите примеры парных и непарных тегов.
3 В каких единицах измерения задается значение атрибута `size`?
4 С помощью какого тега можно увеличить размер шрифта?
5 При использовании какого тега появляются вертикальные отступы?
6 Чем отличаются абсолютные и относительные ссылки?

Практическое занятие № 3. HTML. Создание списков и таблиц

Цель занятия:
– освоить синтаксис HTML;
– изучить порядок использования списков;
– изучить порядок создания и использования таблиц при разработке Web-страниц.

Порядок выполнения

1 Ознакомиться с основами применения списков и таблиц в HTML.
2 Выполнить задание, полученное у преподавателя.
3 Добавить на страницу список трех определений из практической работы № 1.
4 Оформить отчет в виде html-файла.

Основные теоретические положения

1 Списки в HTML

В HTML используются списки следующих видов:
– упорядоченные (нумерованные);
– неупорядоченные (ненумерованные);
– определений.

Перед пунктами неупорядоченных списков обычно ставятся символы-маркеры (bullets), например, точки, ромбики и т. п., в то время как пунктам упорядоченных списков предшествуют их номера. Теги, используемые для создания списков, показаны в таблице 1.

2 Таблицы в HTML

Таблица состоит из строк и столбцов ячеек, которые могут содержать текст и рисунки. Обычно таблицы используются для упорядочения и представления данных, однако возможности таблиц этим не ограничиваются. С помощью таблиц удобно верстать макеты страниц, расположив нужным образом фрагменты текста и изображений.

Таблица 1 – Теги списков

Тег	Назначение
** - **	Создает неупорядоченный список
** - **	Создает упорядоченный список
** - **	Создает пункт списка внутри тегов **ol** или **ul**
<dl> - </dl>	Открывает и закрывает список определений
<dt> - </dt>	Создает термин в списке определений внутри элемента **dl**
<dd> - </dd>	Создает определение термина внутри элемента **dl**

Для добавления таблицы на web-страницу используется тег-контейнер **<table>.** Таблица должна содержать хотя бы одну строку и колонку.

Для добавления строк используются теги **<tr>** и **</tr>**. Чтобы разделить строки на колонки, применяются теги **<td>** и **</td>.**

Для изменения вида и свойств таблицы используются атрибуты, которые добавляются в тег **<table>** (таблица 2).

<table атрибут1=... атрибут 2=...>

Таблица 2 – Атрибуты таблицы и их значения

Атрибут	Значение	Описание	Пример
align=	**left right center**	Выравнивание таблицы	`align=center`
background=	**URL**	Фоновый рисунок	`background="pic.gif"`
bgcolor=	**#rrggbb**	Цвет фона таблицы	`bgcolor=#FF9900`
border=	**n**	Толщина рамки в пикселях	`border=2`
cellpadding=	**n**	Расстояние между ячейкой и ее содержимым	`cellpadding=7`
cellspacing=	**n**	Дистанция между ячейками	`cellspacing=3`
colspan=	**n**	Число ячеек, объединяемых по горизонтали	`colspan=2`
rowspan=	**n**	Число ячеек, объединяемых по вертикали	`rowspan=3`
nowrap		Запрещает переносы строк в тексте	`<table nowrap>`
frame=	**void above below lhs rhs hsides vsides box**	Задание типа рамки таблицы	`frame=hsides`
valign=	**top bottom**	Выравнивание по высоте	`valign=top`
width=	**n, n%**	Минимальная ширина таблицы в пикселях или процентах	`width=90%`
height=	**n, n%**	Минимальная высота таблицы в пикселях или процентах	`height=18`

Вопросы для контроля

1 Какие существуют виды списков?
2 Чем отличаются нумерованные и маркированные списки?
3 С помощью каких атрибутов можно изменить порядок нумерации списка?
4 Какие существуют значения атрибута `type` для маркированных списков?
5 Как создать маркированный список без вертикальных отступов?
6 Расшифруйте теги `<dl>`, `<dt>`, `<dd>`. Для чего они предназначены?
7 Какие теги предназначены для создания заголовка и подвала таблицы?
8 Каким способом можно создать рамку вокруг ячейки таблицы?
9 Для чего применяются атрибуты colspan и rowspan?
10 С помощью какого атрибута можно задать пространство между границей и содержимым ячейки?
11 Что произойдет, если для двух смежных ячеек задать разные значения ширины или высоты?
12 Для чего предназначен тег <caption>?

Практическое занятие № 4. CSS. Основные понятия. Каскадирование

Цель занятия:
– изучить порядок создания и использования CSS при разработке Web-страниц;
– изучить механизм каскадирования в CSS.

Порядок выполнения

1 Ознакомиться с основами применения CSS.
2 Оформить практические работы № 2–3 с использованием CSS.
3 Использовать ***все типы*** селекторов и ***все способы*** подключения CSS.
4 Сделать вывод о размере кода HTML с применением CSS, а также о целесообразности применения CSS в конкретном случае.
5 Подготовить Web-документы (за основу можно взять предыдущие работы), при форматировании которых применяется механизм каскадирования.
6 В применяемых CSS механизм каскадирования пояснить комментариями.

Основные теоретические положения

1 CSS. Основные понятия

CSS (Cascading Style Sheets, Каскадные Таблицы Стилей) – это набор правил форматирования Web-страницы, который может быть применен к различным элементам страницы.

В HTML для присвоения какому-либо элементу определенных свойств, таких как цвет, размер, положение на странице и т. п., приходится каждый раз

описывать эти свойства.

Применяя CSS, можно один раз описать свойства и их значения и определить это описание как *стиль*, а в дальнейшем просто указывать, что элемент, который надо оформить соответствующим образом, должен принять эти свойства стиля.

Описание стиля можно сохранить не в тексте страницы, а в отдельном файле – это позволит использовать описание стиля на любом количестве Web-страниц.

Описания стилей в Web-странице должны находиться в тегах **<style></style>**, которые размещаются между тегами **<head></head>**.

`Example.css` – это CSS-файл, содержащий описание применяемых стилей. Если он находится в другом каталоге, нужно указать к нему путь. Создается CSS-файл в любом текстовом редакторе, например, в Блокноте, нужно только изменить расширение текстового файла на CSS. В CSS-файле не должны указываться теги **<style></style>**.

Можно определить стиль для любого тега отдельно. Для этого нужно в тег добавить атрибут **style=""** и описать его стиль в кавычках.

Следующий пример отображает слово «Пример» шрифтом Verdana, размером 150 % и красным цветом.

```
<h3 style="font-family:Verdana; font-size:150%; color:red">Пример</h3>
```

Некоторые свойства CSS и их назначение показаны в таблице 3.

Таблица 3 – Свойства CSS

Свойства CSS	Назначение
Свойства шрифта	
`font-family`	Используется для указания шрифта или шрифтового семейства, которым будет отображаться элемент. Пример: `p {font-family: Verdana, sans-serif;}`
`font-weight`	Определяет степень насыщенности шрифта: bold|bolder|lighter|normal|100|200|300|400|500|600|700|800|900 Пример: `b {font-weight: bolder;}`

`font-size`	Устанавливает размер шрифта. Параметр может указываться в процентах, пикселях или сантиметрах. Примеры использования для тегов `h1`, `h2`, `h3`:
	`h1 {font-size: 200%;}`
	`h2 {font-size: 150px;}`
	`h3 {font-size: 400pt;}`

Свойства текста	
`text-decoration`	Устанавливает эффекты оформления шрифта, такие как подчеркивание или зачеркивание текста. Пример использования для тега `h4`: `h4 {text-decoration: underline;}` (`подчеркивание`) `h4 {text-decoration: line-through;}` (`зачеркивание`)

Окончание таблицы 3

Свойства CSS	Назначение
`text-align`	Определяет выравнивание элемента. Пример: `p {text-align: left}` (`выравнивание по левому краю`) `p {text-align: right}` (`выравнивание по правому краю`) `p {text-align: justify}` (`выравнивание по ширине`) `p {text-align: center}` (`выравнивание по центру`)
`text-indent`	Устанавливает отступ первой строки текста. Чаще всего используется для создания параграфов с табулированной первой строкой. Пример использования: `h1 {text-indent: 60pt;}`
`line-height`	Управляет интервалами между строками текста. Пример: `p {line-height: 50%}`
Цвет	
`color`	Определяет цвет элемента. Пример использования для тега `h3`: `h3 {color: #0000FF;}`

`background-color`	Устанавливает цвет фона для элемента.
	Пример использования для тега `h3`:
	`<h3 style="background-color:gold; color:brown;">`
	`Пример </h3>`

Свойства границ	

`margin-left`
`margin-right`
`margin-top`
`margin-bottom`
	Устанавливают значения отступов вокруг элемента.
	Пример использования для рисунка:
	`img {margin-left: 20pt}`
	`img {margin-right: 20pt}`
	`img {margin-top: 20pt}`
	`img {margin-bottom: 20pt}`

2 Каскадирование

В соответствии со спецификацией CSS имеется несколько способов подключения таблиц стилей к документу. И может получиться, что возникнет конфликт – на одно свойство конкретного элемента претендуют несколько разных значений, пришедших из разных правил (таблиц стилей).

Для разрешения конфликтов значений свойств в CSS определен механизм (порядок, алгоритм) каскадирования:

– по важности (явной приоритетности);
– по источнику;
– по специфичности;
– по расположению.

2.1 Важность (явная приоритетность).

Важность объявления настолько велика, что перевешивает все остальные факторы. CSS2.1 называет их (по вполне понятным причинам) *важными объявлениями (important declarations)* и предоставляет возможность отмечать их путем введения в объявление ключевого слова **!important** прямо перед завершающей точкой с запятой:

```
p.dark {color: #333 !important; background: white;}
```

Здесь значение цвета **#333** отмечено как **!important**, тогда как цвет фона **white** – нет. Если надо сделать важными оба объявления, каждому из них понадобится собственный маркер **!important**:

```
p.dark {color: #333 !important;
  background: white !important;}
```

Необходимо правильно размещать **!important**, иначе объявление будет признано недействительным. Ключевое слово **!important** всегда располагается *в конце объявления*, прямо перед точкой с запятой. Это особенно важно, когда дело доходит до свойств (например, **font**), значения которых могут состоять из нескольких ключевых слов:

```
p.light (color: yellow;
  font: smaller Times,
  serif !important;}
```

Если бы **!important** было расположено где-либо в другом месте объявления **font**, то все объявление было бы признано недействительным и ни один из его стилей не был бы применен.

Объявления, отмеченные как **!important**, не имеют особого значения специфичности, они рассматриваются отдельно от остальных. Фактически все объявления **!important** группируются вместе, и тогда уже их конфликты специфичностей разрешаются относительно друг друга. Аналогично группируются все остальные объявления, и конфликты свойств разрешаются с помощью специфичностей.

2.2 Источник правила.

Существуют *три* возможных источника правил: ***автор***, ***читатель*** и ***браузер*** пользователя. Причем на их приоритетность влияет **инструкция !important**. Таким образом, с точки зрения *приоритетности* объявлений выделяют *пять* уровней. В порядке ***уменьшения приоритетности*** это:

1) важные объявления пользователя;
2) важные объявления автора;
3) обычные объявления автора;

4) обычные объявления пользователя;
5) объявления браузера пользователя.

2.3 Специфичность.

Для каждого правила браузер пользователя вычисляет специфичность (*specificity*) *селектора* и прикрепляет ее к каждому *объявлению* правила.

Специфичность селектора определяется компонентами селектора.

Значение специфичности состоит из четырех частей ***a***, ***b***, ***c*** и ***d***: `0,0,0,0`.

Если стиль встроенный (*inline*), ***a*** = 1.

Значение ***b*** равно общему количеству селекторов идентификаторов.

Значение ***c*** равно количеству классов, псевдоклассов и селекторов атрибутов.

Значение ***d*** равно количеству селекторов типов и псевдоэлементов.

2.4 Порядок расположения.

Если два правила имеют совершенно одинаковые *приоритетность* и *специфичность*, то побеждает то из них, которое расположено в таблице стилей *позже*.

С этих позиций принимается, что стили, определенные в атрибуте **`style`** элемента, находятся в самом конце таблицы стилей документа, т. е. размещаются после всех остальных правил и поэтому имеют более высокую *специфичность*, чем любой селектор таблицы стилей.

2.5 Упрощенный порядок каскадирования.

Упрощенный порядок определения приоритетность правил (от низшего к высшему):

1) связанная таблица стилей (элемент `<link>`);
2) импортируемая таблица стилей (инструкция `@import`);
3) правило с элементом HTML в качестве селектора (элемент `<style>`);
4) правило с параметром `class` в качестве селектора;
5) правило с параметром `id` в качестве селектора;
6) встроенное в тег HTML правило (атрибут `style`).

Вопросы для контроля

1 Какова структура правила CSS?
2 Что такое стиль и таблица стилей?
3 Способы подключения таблиц стилей CSS.
4 Какие существуют типы селекторов?
5 Для чего используется группирование в CSS?
6 У какого селектора комбинатор #?
7 Когда применяется механизм каскадирования?
8 Как задать явную приоритетность?
9 По какому компоненту правила вычисляется специфичность?
10 Какие существуют источники правил?

11 Как подключить пользовательский стиль в браузере?
12 Назовите упрощенный порядок каскадирования от низшего к высшему.

Практическое занятие № 5. CSS. Модель визуального форматирования

Цель занятия: изучить возможности модели визуального форматирования в CSS.

Порядок выполнения

1 Ознакомиться с основами применения блочной модели в CSS.
2 Выполнить задание, полученное у преподавателя.
3 При этом использовать:
– максимальное количество CSS-свойств форматирования текста;
– различные типы и комбинации селекторов;
– возможности свойства display для отображения элементов.
4 Оформить отчет в виде html-файла.

Основные теоретические положения

1 Типы отображения элементов

Css-свойство `display` задаёт тип отображения элемента. На русский язык его можно перевести фразой: «Веди себя как». То есть берём любой элемент `html` и говорим ему: веди себя как… блок, строка, таблица или вообще как будто тебя нет.

Наиболее часто используемые значения свойства `display`:
– `none`;
– `block`;
– `inline`;
– `inline-block`;
– `list-item`.

Свойство **`display`** изменяет поведение элемента, но не классовую принадлежность.

2 Замещаемые и незамещаемые элементы

CSS определяется элементами, но не все элементы создаются одинаково. Например, изображения и абзацы – это элементы разных типов, так же как `<span>` и `<div>`. В CSS элементы разделяются на *замещаемые* и *незамещаемые*.

Замещаемыми (replaced) называются те элементы, содержимое которых замещается чем-то, что не содержится непосредственно в документе. Наиболее очевидный пример из HTML – элемент `img`, замещаемый файлом изображения,

который является внешним по отношению к документу. Он фактически не имеет содержимого, как видно из простого примера:

```
<img src="howdy.gif">
```

Этот фрагмент разметки не имеет реального содержимого, а только имя элемента и атрибут. Данный элемент ничего не представляет, пока нет указания на внешнее содержимое (в данном случае изображение, заданное атрибутом `src`).

Элемент `input` замещается кнопкой, переключателем или полем ввода текста в зависимости от его типа. Замещаемые элементы также генерируют блоки в своем визуальном представлении.

Основная масса элементов HTML – *незамещаемые* (nonreplaceable). Например, элемент `<span>эй там</span>` – незамещаемый элемент, и агент пользователя (user agent) будет отображать текст «эй там». Это выполняется для абзацев, заголовков, ячеек таблиц, списков и почти всех остальных элементов HTML.

3 Схлопывание (сворачивание) вертикальных полей (margin)

Есть две ситуации в верстке, когда вертикальные поля схлопываются:
1) соседство двух элементов с вертикальными полями;
2) наличие вертикальных полей у родительского и дочернего элементов.
Внешне такой эффект выглядит так, будто поля накладываются друг на друга.

4 Строчные (внутристрочные, inline) элементы

Строчными называются элементы web-страницы, которые являются непосредственно частью строки. К строчным элементам относятся элементы `<img>`, `<span>`, `<a>`, `<q>`, `<code>` и другие. В основном они используются для изменения вида текста.

Характерные особенности строчных элементов:

– внутрь строчных элементов допустимо помещать текст или другие строчные элементы. Вставлять блочные элементы внутрь строчных ***запрещено***;

– эффект схлопывания полей не действует. Если рядом стоят два строчных элемента с определенными полями, то эти поля ***суммируются***;

– свойства, связанные с размерами (width, height), ***не применимы***. Их просто нет. На то она и строка;

– ширина элемента равна содержимому плюс значения отступов, полей и границ;

– несколько строчных элементов, идущих подряд, располагаются на одной строке друг за другом и переносятся на следующую строку при необходимости;

– строчные элементы можно выравнивать по вертикали при помощи css-свойства `vertical-align`.

Вопросы для контроля

1 Какие существуют концепции визуального форматирования?
2 В чем отличие замещаемого элемента от незамещаемого?
3 Какие есть типы отображения элементов?
4 Что нужно сделать, чтобы не происходило схлопывание полей?
5 Как сделать элемент невидимым?
6 Можно ли в строчные элементы вкладывать блочные? А наоборот?

Практическое занятие № 6. CSS. Позиционирование и свободное перемещение

Цель занятия: изучить возможности основных типов позиционирования в CSS.

Порядок выполнения

1 Ознакомиться с основами позиционирования в CSS.
2 Выполнить задание, полученное у преподавателя.
3 Использовать все типы позиционирования.
4 Оформить отчет в виде HTML- и CSS-файлов.

Основные теоретические положения

Сущность позиционирования состоит в том, чтобы любой элемент расположить в необходимом месте web-страницы. Позиционирование определяется свойством `position`, которое может иметь следующие значения:

`position: static;` (значение по умолчанию для всех элементов в нормальном потоке);

```
position: absolute;
position: fixed;
position: relative;
```

Указание точного месторасположения элементов осуществляется ***свойствами смещения***: `top`, `right`, `bottom`, `left`.

`top` – расстояние от верхнего края окна блока-контейнера (или браузера).
`bottom` – расстояние от нижнего края окна блока-контейнера (или браузера).
`left` – расстояние от левого края окна блока-контейнера (или браузера).
`right` – расстояние от правого края окна блока-контейнера (или браузера).

Свойства смещения работают со всеми позиционированными элементами, кроме имеющих значение `static`.

Прежде чем приступать к работе с позиционированием, необходимо разобраться с таким понятием, как *нормальный поток*.

Нормальный поток:

1) *текста* (*контента*) – это размещение символов слева направо и сверху вниз (для западных языков);

2) *порядок вывода элементов* на экран определяется порядком, в котором они написаны в html-коде документа;

3) *блочные элементы* (`display: block`):

– выводятся сверху вниз по странице;

– занимают всю доступную ширину родительского элемента;

– высота элемента определяется его содержимым;

– всегда начинаются с новой строки;

– могут содержать *блочные* и *строковые* элементы.

К блочным элементам относятся `<div>, <h1>, <p>` и др.;

4) *строковые элементы* (`display: inline`):

– выводятся слева направо по строке;

– располагаются друг за другом в одной строке;

– при необходимости строка переносится;

– до и после строчного элемента отсутствуют переносы строки;

– ширина и высота строчного элемента зависят только от его содержания, задать размеры с помощью CSS нельзя;

– могут содержать *только строковые* элементы.

Абсолютное позиционирование

Абсолютное позиционирование задаётся при установке свойства элемента `position: absolute`.

При абсолютном позиционировании элемент:

– выводится из общего потока;

– становится независимым от других элементов;

– может накладываться на другие элементы.

При абсолютном позиционировании положение элемента задаётся только свойствами смещения этого элемента (`bottom`, `left`, `right`, `top`) относительно блока-контейнера.

Блоком-контейнером считается ближайший предок (любой), значение свойства `position` которого отлично от `static`. Если таких предков нет, то блоком-контейнером считается начальный блок-контейнер.

Важные моменты:

– абсолютно позиционированные элементы перемещаются вместе с документом при прокрутке;

– свойства `left` и `top` имеют приоритет выше, чем свойства `right` и `bottom`.

Например, если свойства `left` и `right` противоречат друг другу, то свойство `right` будет проигнорировано. То же относится и к свойствам `top/bottom`;

– элементы можно спрятать. Для этого можно задать отрицательным либо `left`, либо `top`. Элемент выйдет за границы окна браузера, полоса прокрутки при этом не возникнет;

– если задать `left` больше, чем ширина окна браузера, либо отрицательное значение `right`, то возникнет горизонтальная полоса прокрутки. Аналогично будет и с `top`, только полоса прокрутки будет вертикальной.

Фиксированное позиционирование

Фиксированное позиционирование задаётся `position:fixed` и по своей сути очень похоже на абсолютное. Элемент с фиксированным позиционированием также выводится из общего потока и не зависит от расположения других элементов. Положение элемента **не** изменяется при прокрутке. Элемент так же, как и при абсолютном позиционировании, может накладываться на другие. Ещё одной особенностью фиксированного позиционирования является то, что при выходе содержимого за пределы видимой области **не** возникает полос прокрутки.

Фиксированное позиционирование используется для создания меню, вкладок, заголовков, т. е. таких элементов, которые всегда должны быть видны пользователю.

Относительное позиционирование

Относительное позиционирование задается свойством **`position: relative`**. При таком способе позиционирования положение элемента определяется относительно краёв элемента родителя и сам элемент не выводится из основного потока. Расстояние до краёв родительского элемента задаётся свойствами: `bottom`, `left`, `right`, `top`.

Важные моменты:

– данный тип позиционирования не применяется к элементам таблицы;

– если при смещении элемента образовалось пустое место, оно не занимается ниже- или вышележащими элементами;

– относительно позиционированный элемент образует блок-контейнер для других элементов.

Плавающие элементы

Плавающее поведение элемента осуществляется заданием свойства `float`. Оно применяется для создания красивого эффекта обтекания какого-либо элемента содержимым. Для этого закрепляется одна сторона, позволяя другим элементам располагаться вокруг.

Возможные значения свойства `float`:

`none` – без обтекания;

`left` – выравнивание по левому краю, обтекание по правому краю;

`right` – выравнивание по правому краю, обтекание по левому краю.

Свойство z-index

Свойство `z-index` предназначено для работы с элементами страницы в трехмерном пространстве. Для этого вводится третья ось – ось Z. Так как страница видится как двумерная, элементы накладываются слой за слоем друг на друга. Управлять подобным наложением можно с помощью свойства `z-index`.

Возможные значения свойства `z-index`:

`auto` – элементы накладываются друг на друга в том порядке, в каком они были указаны в коде HTML;

`целое число` – чем больше число, тем более высокую позицию он занимает по оси Z и, соответственно, перекрывает все элементы, расположенные ниже по этой оси.

Важные моменты:

– числовое значение `z-index` может быть отрицательным;

– при равном значении `z-index` на переднем плане будет находиться тот элемент, который идёт ниже в html-коде. Это же правило действует и при `z-index`, равном `auto`.

Вопросы для контроля

1 В чем сущность позиционирования? Каким свойством оно определяется?
2 Какие свойства являются свойствами смещения?
3 Какие базовые схемы размещения элементов есть в CSS?
4 Что такое нормальный поток?
5 В чем отличие абсолютного позиционирования от относительного?
6 Каким свойством задаются плавающие элементы?

Список литературы

1 **Бройдо, В. Л.** Вычислительные системы, сети и телекоммуникации : учебник / В. Л. Бройдо. – 4-е изд. – Санкт-Петербург : Питер, 2013. – 560 с.

2 **Фрейн, Б.** HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств / Б. Фрейн. – Санкт-Петербург : Питер, 2014. – 304 с: ил.

3 **Гоше, Х. Д.** HTML5. Для профессионалов : пер. с англ. / Х. Д. Гоше. – 2-е изд. – Санкт-Петербург : Питер, 2015. – 560с. : ил.

4 **Дронов, В. А.** HTML 5, CSS 3 и Web 2.0. Разработка современных Web-сайтов / В. А. Дронов. – Санкт-Петербург : БХВ-Петербург, 2011. – 414 с.

5 **Дакетт, Дж.** HTML и CSS. Разработка и дизайн веб-сайтов / Дж. Дакетт. – 2-е изд. – Санкт-Петербург : Эксмо, 2017. – 923 с.

6 **Дунаев, В. В.** HTML, скрипты и стили / В. В. Дунаев. – Санкт-Петербург : БХВ-Петербург, 2011. – 810 с.

7 **Евсеев, Д. А.** Web-дизайн в примерах и задачах : учебное пособие / Д. А. Евсеев, В. В. Трофимов ; под ред. В. В. Трофимова. – Москва : КНОРУС, 2010. – 272 с.

8 **Макфарланд, Д.** Большая книга CSS / Д. Макфарланд. – 2-е изд. – Санкт-Петербург: Питер, 2012. – 560 с: ил.

9 **Мак-Дональд, М.** HTML5. Недостающее руководство : пер. с англ. / М. Мак-Дональд. – Санкт-Петербург : БХВ-Петербург, 2012. – 480 с. : ил.

10 **Матросов, А. В.** HTML 4.0 / А. В. Матросов, А. О. Сергеев, М. П. Чаунин. – Санкт-Петербург : БХВ-Петербург, 2007. – 672 с.: ил.

11 **Мейер, Э.** CSS – каскадные таблицы стилей. Подробное руководство : пер. с англ. / Э. Мейер. – 3-е изд. – Санкт-Петербург : Символ-Плюс, 2010. – 576 с. : ил.

12 **Комолова, Н.** HTML, XHTML и CSS / Н. Комолова, Е. Яковлева. – Санкт-Петербург : Питер, 2012. – 304 с. : ил.

13 **Немцова, Т. И.** Компьютерная графика и web-дизайн: учебное пособие / Т. И. Немцова, Т. В. Казанкова, А. В. Шнякин. – Москва : ФОРУМ ; ИНФРА-М, 2014. – 400 с.

14 **Прохоренок, Н. А.** HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Н. А. Прохоренок. – Санкт-Петербург : БХВ-Петербург, 2010. – 900 с.

15 **Гарольд, Э.** XML : справочник : пер. с англ. / Э. Гарольд, С. Минс. – Санкт-Петербург : Символ-Плюс, 2002. – 576 с. : ил.

16 **Рэй, Э.** Изучаем XML : пер. с англ. / Э. Рэй. – Санкт-Петербург : Символ-Плюс, 2001. – 408 с.: ил.

17 **Старыгин, А. А.** XML : разработка Web-приложений / А. А. Старыгин. – Санкт-Петербург : БХВ-Петербург, 2003. – 592 с. : ил.

18 **Хабибуллин, И. Ш.** Самоучитель XML / И. Ш. Хабибуллин. – Санкт-Петербург : БХВ-Петербург, 2003. – 336 с. : ил.

19 **Холзнер, С.** XML: энциклопедия/ С. Холзнер. – 2-е изд. – Санкт-Петербург : Питер, 2004. – 1101 с. : ил.

20 **Шапошников, И. В.** Справочник Web-мастера. XML / И. В. Шапошников. – Санкт-Петербург : БХВ-Петербург, 2001. – 304 с. : ил.