Динамическое содержание документа

Ранее мы неоднократно встречались с методом `document.write()`, позволяющим менять содержание документа. В настоящем разделе мы рассмотрим другие способы манипулирования содержанием, которые обеспечиваются четырьмя свойствами (`innerText, innerHTML, outerText, outerHTML`) и двумя методами (`insertAdjacentHTML, insertAdjacentText`). Эти свойства и методы применимы к любому элементу документа и предоставляют наиболее простые способы управления содержанием.

Свойства динамического содержания

Что означают слова «изменение содержания документа»? На языке HTML это значит, что, во-первых, могут изменяться, добавляться или удаляться некоторые теги, а, во-вторых, может меняться содержимое тегов. Такие изменения могут определяться с помощью свойств `innerText, innerHTML, outerText, outerHTML`. Чтобы пояснить каждое из этих свойств, рассмотрим простой элемент:

```
<DIV id=agr>Зайдите на этот сайт</DIV>
```

Свойство `innerText` представляет содержание элемента, то есть имеет значение `"Зайдите на этот сайт"`. Это нетрудно проверить с помощью скрипта:

```
<SCRIPT LANGUAGE="javascript">
  alert(document.all.agr.innerText);
</SCRIPT>
```

Свойство `innerHTML` также представляет содержание элемента, но оно включает HTML-разметку всех дочерних элементов. Рассмотрим документ с элементом `<DIV>`, в который вложен элемент `<SPAN>`:

```
<HTML>
 <HEAD>
   <TITLE>Свойство innerHTML</TITLE>
  </HEAD>
 <BODY>
  <DIV id=agr>Если у вас есть время,
   <SPAN style="font-weight:bold; color:blue">зайдите на этот сайт
   </SPAN>
  </DIV>
  <SCRIPT language="javascript">
    alert(document.all.agr.innerHTML);
  </SCRIPT>
 </BODY>
</HTML>
```

В результате на экране появится сообщение (рис.1), в котором вы прочтете значение свойства `innerHTML` для объекта `agr`.

Рис.1. Сообщение, в котором указано значение свойства `innerHTML` для объекта `agr`

Таким образом, свойства `innerText` и `innerHTML` – это внутренние свойства элемента HTML. В отличие от них, внешние свойства `outerText` и `outerHTML` относятся ко всему элементу, а не к его содержанию. Так, для приведенного выше элемента `<DIV>` значением свойства `outerHTML` обудет содержание элемента вместе с внешними тегами:

```
<DIV id=agr>Если у вас есть время,<SPAN style="font-
weight:bold; color:blue">зайдите на этот сайт</SPAN></DIV>
```

Другое внешнее свойство, `outerText`, возвращает то же значение, что и внутренне свойство `innerText`. Разница заключается в том, что эти свойства по-разному себя ведут при присвоении им новых значений. Так, установив новое значение для `innerText`, вы измените содержание элемента, в то время как установка нового значения для `outerText` заменит весь элемент новым текстом.

Исчезновение элемента (свойство outerText)

Различие между свойствами динамического содержания лучше всего изучать на примерах. Рассмотрим операцию установки свойства `outerText` для HTML-элемента, в результате которой удаляются теги самого элемента:

```
<HTML>
 <HEAD>
   <TITLE>Присвоение свойства outerText</TITLE>
   <SCRIPT language="javascript">
      function change() {
         document.all.agr.outerText="Вы заменили весь элемент
заголовка данным текстом";
      }
   </SCRIPT>
 </HEAD>
 <BODY>
  <H1 id=agr style="cursor:hand" onclick='change()'>Щелкните на
этом заголовке
  </H1>
 </BODY>
</HTML>
```

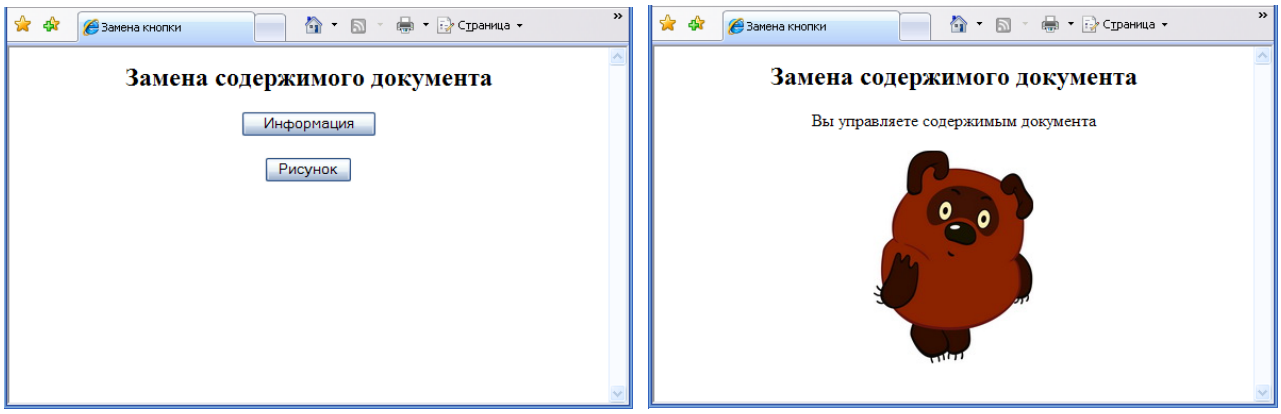
Щелчок на заголовке <H1> в документе вызывает функцию `change()`, которая заменяет элемент <H1> простым текстом. Если вместо `outerText` в функции `change()` использовать свойство `innerText`, теги заголовка будут сохранены, и оформление текста останется прежним.

Замена элемента (свойство outerHTML)

Если на странице вам нужно обеспечить динамическую замену одного элемента другим, для этого проще всего воспользоваться свойством `outerHTML`. Приведем простой пример страницы, на которой имеются две кнопки: щелчок на одной из них приводит к исчезновению кнопки и появлению на ее месте текста, а щелчок на другой кнопке приводит к замене кнопки на картинку:

```
<HTML>
 <HEAD>
   <TITLE>Замена кнопки</TITLE>
 </HEAD>
 <BODY>
  <CENTER>
  <H2>Замена содержимого документа</H2>
  <INPUT type=button value="Информация"
onClick="this.outerHTML='Вы управляете содержимым документа'">
  <BR><BR>
  <INPUT type=button value="Рисунок" onClick="this.outerHTML='<IMG
src=winnie-the-pooh.jpg>'">
  </CENTER>
 </BODY>
</HTML>
```

Исходное содержание документа представлено на рис.2,а. На соседнем рисунке приведен вид документа после того, как нажаты обе кнопки.



а б

Рис. 2. Пример замены кнопки на различные элементы документа: а – исходный документ; б – документ после замены

Электронные часы (свойство innerText)

Ранее рассматривался пример создания цифровых часов в строке состояния. Теперь опишем часы, которые выводятся в текстовый элемент `<SPAN>`. Содержимое этого элемента обновляется с помощью свойства `innerText` каждые 1000 миллисекунд. Полный листинг HTML-документа выглядит следующим образом:

```
<HTML>
  <HEAD>
   <TITLE>Электронные часы</TITLE>
   <STYLE type="text/css">
     #tictac {color:blue; font-size:110%}
   </STYLE>
   <SCRIPT language="javascript">
      function ahead0(x) {
         return(x<10)?"0"+x.toString():x;
       }
      function currClock(){
         var time=new Date();
         var hour=time.getHours();
         var minut=time.getMinutes();
         var sec=time.getSeconds()
         return ahead0(hour)+":"+ahead0(minut)+":"+ahead0(sec);
      }
      function tick(){
         document.all.tictac.innerText=currClock();
      }
    </SCRIPT>
   </HEAD>
 <BODY onUnload="if(null!=window.tmr)clearInterval(window.tmr);"
onload="window.tmr=setInterval('tick()',1000);">
  <CENTER>
  <H2>Электронные часы</H2>
  <P>Текущее время:
  <SPAN id="tictac">
  <SCRIPT language="javascript">
     document.write(currClock());
  </SCRIPT>
  </CENTER>
 </BODY>
</HTML>
```

Страница в окне браузера изображена на рис.3.

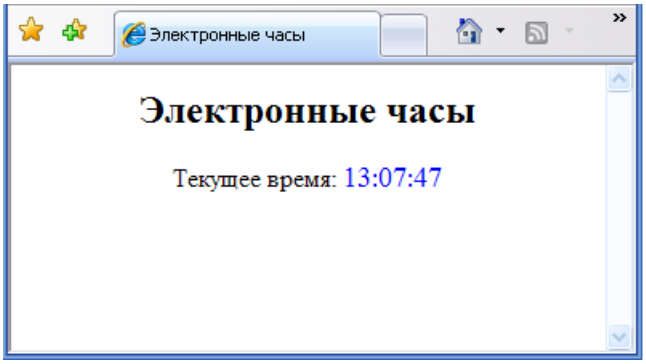


Рис.3. Электронные часы. Пример обновления содержимого документа с помощью свойства `innerText`

Создание элемента с помощью метода write()

Вы уже имеете представление, как с помощью свойств динамического содержания создается теговая структура документа HTML. Воспроизведению полученной таким образов структуры на Web-странице можно выполнить с помощью метода `document.write()`. Приведем пример создания заголовка:

```
<HTML>
  <HEAD>
   <TITLE>Создание элемента</TITLE>
   <SCRIPT language="javascript">
      function buildElement(){
         var tag="<H1>";
         tag+="Текст заголовка";
         tag+="</H1>"
         return tag;
       }
    </SCRIPT>
   </HEAD>
 <BODY>
  <SCRIPT language="javascript">
     document.write(buildElement());
  </SCRIPT>
 </BODY>
</HTML>
```

В этом документе решается простейшая задача создания заголовка <H1>. В дальнейшем мы будет пользоваться описанным способом для создания более сложного динамического одержания.

Методы Adjacent

В JavaScript для динамического управления содержанием документа предусмотрены методы `insertAdjacentHTML` и `insertAdjacentText`, которые предназначены для вставки соответственно кода или текста в контейнеры HTML. Их удобно применять, когда необходимо вставить содержание, которое не повлияет на остальное содержание документа. Например, вы хотите изменить стиль определенных элементов или добавит содержание при некотором событии.

Различные позиции вставки содержания

Рассматриваемые методы содержат два аргумента, например

```
insertAdjacentHTML("arg1","arg2")
```

и аналогично записывается метод `insertAdjacentText()`. Первый аргумент `arg1` определяет место вставки содержания, а второй аргумент `arg2` – собственно содержание. Возможны четыре места вставки содержания и соответственно значения первого аргумента:

- `beforeBegin` – перед открывающим тегом;

- `afterBegin` – после открывающего тега;
- `beforeEnd` – перед закрывающим тегом;
- `afterEnd` – после закрывающего тега.

Различные значения `arg1` проиллюстрируем следующим примером. Пусть на исходной странице имеется один элемент – заголовок `<H2>`. Составим сценарий, который будет выполнять вставку контейнера верхних индексов `<SUP>` в различные позиции относительно тегов заголовка `<H2>…</H2>`. Для удобства место вставки будет отображаться текстом, содержащимся в контейнере `<SUP>` 9значение аргументов `arg1` и `arg2` одинаковы). Результирующая страница будет описываться кодом:

```
<HTML>
  <HEAD>
   <TITLE>Первый аргумент в методе insertAdjacentHTML</TITLE>
  </HEAD>
  <BODY>
   <CENTER>
   <H2 id=zag>Место вставки<BR>
     изменить можно...
   </H2>
  <SCRIPT language="javascript">
  zag.insertAdjacentHTML("beforeBegin","<SUP>"+"beforeBegin"+"</SUP>");
  zag.insertAdjacentHTML("afterBegin","<SUP>"+"afterBegin"+"</SUP>");
  zag.insertAdjacentHTML("beforeEnd","<SUP>"+"beforeEnd"+"</SUP>");
  zag.insertAdjacentHTML("afterEnd","<SUP>"+"afterEnd"+"</SUP>");
  </SCRIPT>
  </CENTER>
 </BODY>
</HTML>
```

Внешний вид полученной страницы изображен на рис.4.

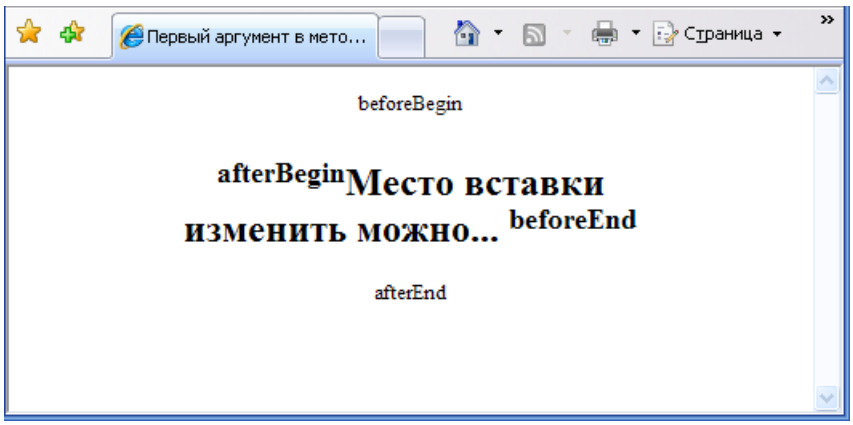


Рис.4. Примеры различных значений аргументов метода `insertAdjacentHTML`