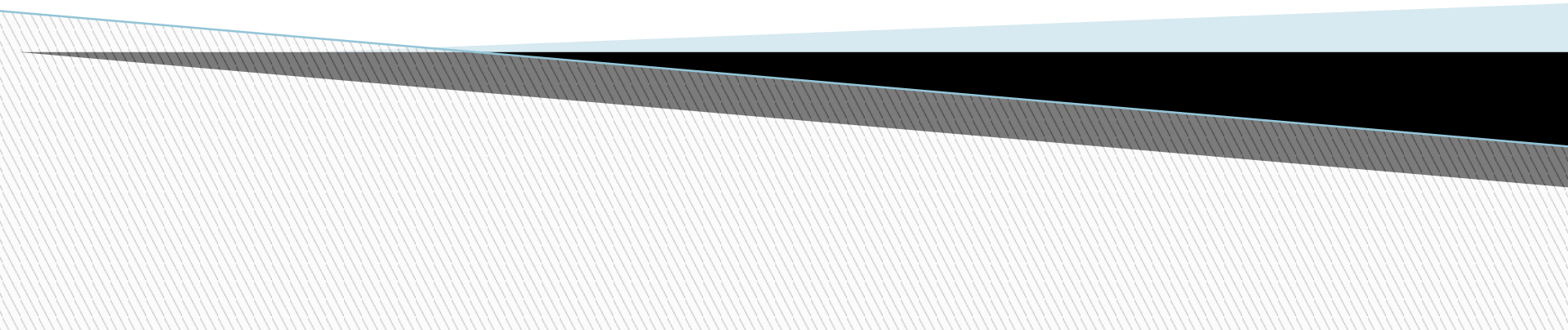


Лекция 4.

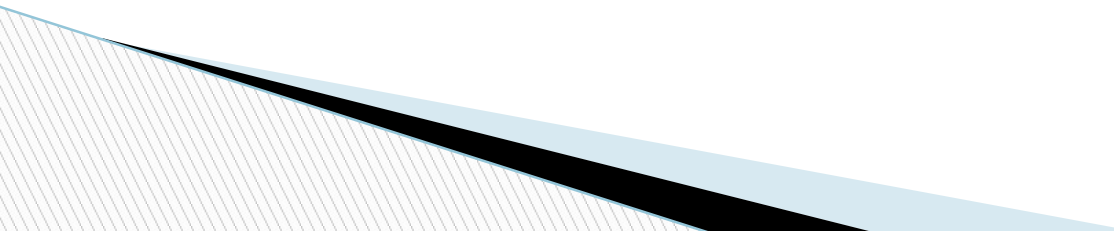
Программирование свойств окна браузера

Ст. преподаватель Еремеев А.А.
YeremeevAA@mpei.ru



Объект `window`

Класс объектов `Window` - это самый старший класс в иерархии объектов JavaScript. Объект `window` создается только в момент открытия окна. Все остальные объекты, которые порождаются при загрузке страницы, есть свойства объекта `window`.



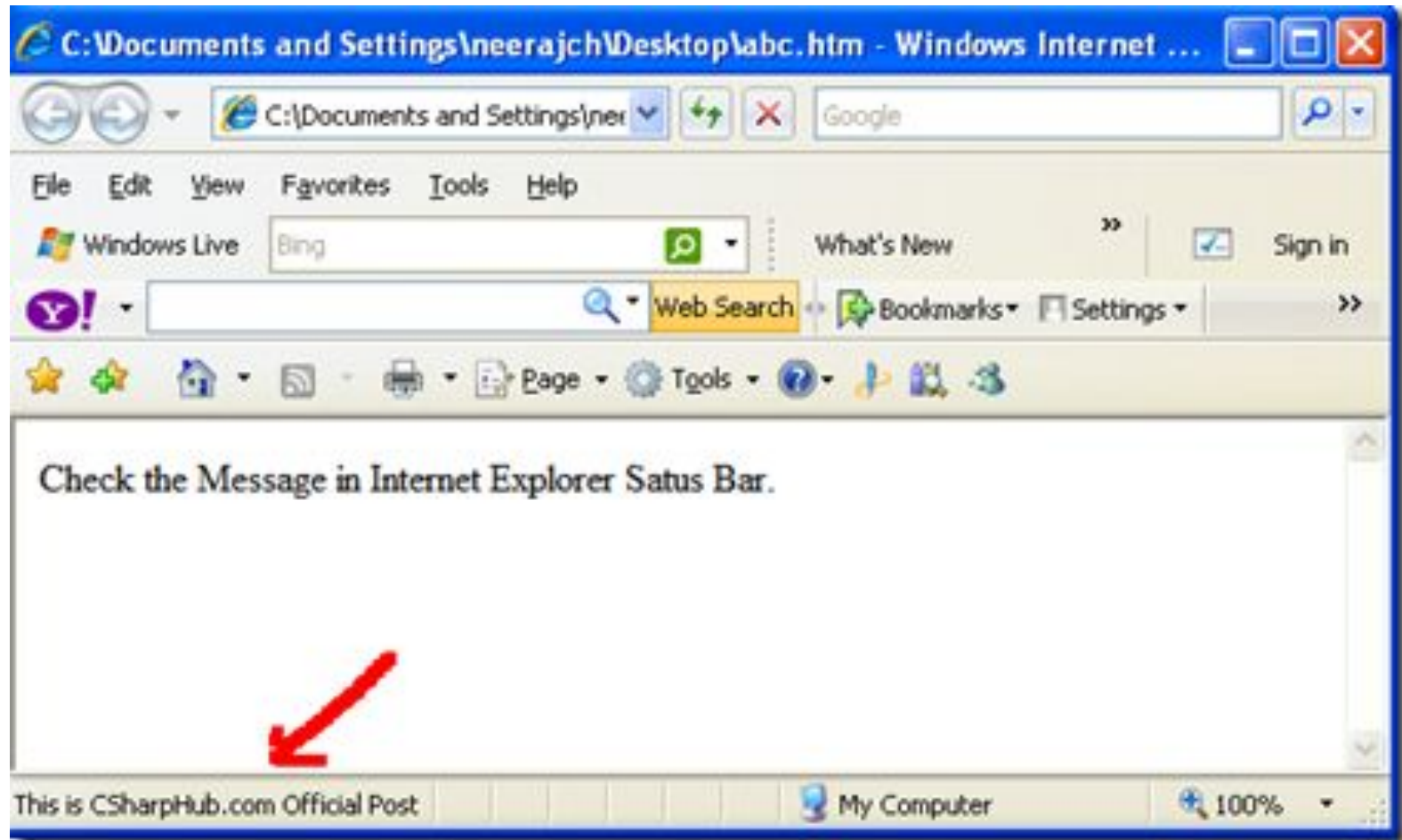
Использование `window`

Поскольку объект `window` является самым старшим, то в большинстве случаев при обращении к его свойствам и методам приставку "`window.`" можно опускать. Например:

можно писать `alert('Привет')` вместо `window.alert('Привет')`.

Исключениями из этого правила являются вызовы методов `open()` и `close()`, у которых нужно указывать *имя окна*, с которым работаем (родительское в первом случае и дочернее во втором).

Поле статуса и свойство `window.status`



Поле статуса

- ▣ `window.status` - значение поля статуса;
- ▣ `window.defaultStatus` - значение поля статуса по умолчанию.

Разница между этими двумя свойствами заключается в их поведении: если свойству `status` присвоить пустую строку: `window.status=""`, то в поле статуса автоматически будет отображено значение `defaultStatus`. Обратного же не происходит.

Программирование **status**

Свойство *status* связано с отображением сообщений о событиях, отличных от простой загрузки страницы.

Например, в *Internet Explorer* при наведении указателя мыши на ссылку обработчик `onMouseOver` помещает в поле статуса значение URL, указанное в атрибуте *HREF* этой ссылки (при этом никак не меняя значения свойств *status* и `defaultStatus`).

Пример

```
<A onMouseOver="window.status='Мышь над  
ссылкой';return true;"  
onMouseOut="window.status='Мышь увели со  
ссылки';"
```

```
  HREF="http://site.com/">Наведите мышь на  
ссылку и следите за полем статуса</A>
```

Программирование defaultStatus

Дополним предыдущий пример изменением этого свойства в момент окончания загрузки документа, т. е. в обработчике onLoad:

```
<BODY onLoad="window.defaultStatus='Значение по умолчанию';">
```

```
<A onMouseOver="window.status='Мышь над ссылкой';return true;"
```

```
onMouseOut="window.status='Мышь увели со ссылки'; alert('Ждем');"
```

```
  HREF="http://site.com/">Наведите мышь на ссылку и следите за полем статуса</A>
```

```
</BODY>
```


Поле адреса и свойство `window.location`

Поле адреса в браузере обычно располагается в верхней части окна и отображает URL загруженного документа. Если пользователь хочет вручную перейти к какой-либо странице (набрать ее URL), он делает это в поле адреса.

Свойства объекта `location`

`http://www.site.ru:80/dir/page.cgi?product=phone&id=3#mark`

Тогда свойства объекта *location* примут следующие значения:

- `window.location.href` = `"http://www.site.ru:80/dir/page.cgi?product=phone&id=3#mark"`
- `window.location.protocol` = `"http:"`
- `window.location.hostname` = `"www.site.ru"`
- `window.location.port` = `80`
- `window.location.host` = `"www.site.ru:80"`
- `window.location.pathname` = `"dir/page.cgi"`
- `window.location.search` = `"?product=phone&id=3"`
- `window.location.hash` = `"#mark"`

Методы объекта `location`

Методы объекта *location* предназначены для управления загрузкой и перезагрузкой страницы. Это управление заключается в том, что можно либо перезагрузить текущий документ (метод *reload()*), либо загрузить новый (метод *replace()*).

Используя объект *location*, перейти на новую страницу можно двумя способами:

- ❑ `window.location.href="http://www.newsite.ru/";`
- ❑ `window.location.replace("http://www.newsite.ru/");`

История посещений (**history**)

История посещений в JavaScript трансформируется в объект `window.history`.

Этот объект указывает на массив URL-страниц, которые пользователь посещал и которые он может получить, выбрав из меню браузера режим Go. Методы объекта `history` позволяют загружать страницы, используя URL из этого массива.

```
□ <FORM><INPUT TYPE="button" VALUE="Назад"
  onClick="history.back()"></FORM>
```

Данный код отображает кнопку "Назад", нажав на которую, мы вернемся на предыдущую страницу. Аналогичным образом действует метод `history.forward()`.

Тип браузера (**navigator**)

Часто возникает задача настройки страницы на конкретную программу просмотра (браузер). Для определения типа браузера на стороне клиента в арсенале объектов JavaScript существует объект *window.navigator*.

Свойство	Описание
userAgent	Основная информация о браузере. Передается серверу в HTTP-заголовке при открытии пользователем страниц
appName	Название браузера
appCodeName	Кодовое название браузера
appVersion	Данные о версии браузера и совместимости

Пример

Пример определения типа программы просмотра:

```
<FORM>
```

```
<INPUT TYPE=button VALUE="Тип навигатора"  
onClick="alert(window.navigator.userAgent);">
```

```
</FORM>
```



Применения navigator

У объекта *navigator* есть еще несколько интересных с точки зрения программирования применений. Например, чтобы проверить, поддерживает ли браузер клиента язык Java, достаточно вызвать метод `navigator.javaEnabled()`, возвращающий значение `true`, если поддерживает, и `false` в противном случае.

Можно проверить, какие форматы графических файлов поддерживает браузер, воспользовавшись свойством `navigator.mimeTypes` (оно представляет собой массив всех типов *MIME*, которые поддерживаются данным браузером).

Методы объекта `window`

▣ **alert()**

Метод `alert()` позволяет выдать окно предупреждения, имеющее единственную кнопку "OK":

```
<A HREF="javascript:window.alert('Внимание')">Повторите запрос!</A>
```

▣ **confirm()**

Метод `confirm()` позволяет задать пользователю вопрос, на который тот может ответить либо положительно (нажав кнопку "OK"), либо отрицательно (нажав кнопку "Отмена" или "*Cancel*", либо просто закрыв окно запроса). В соответствии с действиями пользователя метод `confirm()` возвращает значение `true` либо `false`.

Методы объекта window

▣ **prompt()**

Метод `prompt()` позволяет принять от пользователя строку текста. Синтаксис его таков:

```
prompt("Строка вопроса", "Строка ответа по умолчанию")
```

Метод `prompt()` возвращает полученную строчку в качестве значения, которое можно далее присвоить любой переменной и потом разбирать ее в JavaScript-программе.

```
<FORM NAME=f>  
<INPUT TYPE=button VALUE="Открыть окно ввода"  
onClick="document.f.e.value=  
    window.prompt('Введите сообщение', 'Сюда');">  
<INPUT SIZE=30 NAME=e>  
</FORM>
```

Методы объекта `window`

▣ `window.open()`

Метод ***open()*** предназначен для создания новых *окон*. В общем случае его синтаксис выглядит следующим образом:

```
myWin = window.open ("URL","имя_окна",  
"параметр=значение,параметр=значение,...",  
заменить);
```

Параметры метода `window.open()`

Параметр	Значения	Описание
<code>width</code>	число	Ширина окна в пикселах (не менее 100)
<code>height</code>	число	Высота окна в пикселах (не менее 100)
<code>left</code>	число	Расстояние от левого края экрана до левой границы окна в пикселах
<code>top</code>	число	Расстояние от верхнего края экрана до верхней границы окна в пикселах
<code>directories</code>	yes / no	Наличие у окна панели папок (Netscape Navigator)
<code>location</code>	yes / no	Наличие у окна поля адреса
<code>menubar</code>	yes / no	Наличие у окна панели меню
<code>resizable</code>	yes / no	Сможет ли пользователь менять размер окна
<code>scrollbars</code>	yes / no	Наличие у окна полос прокрутки
<code>status</code>	yes / no	Наличие у окна поля статуса
<code>toolbar</code>	yes / no	Наличие у окна панели инструментов

Методы объекта `window`

▣ `window.close()`

Метод `close()` позволяет закрыть окно. Если необходимо закрыть текущее, то:

```
window.close();
```

```
self.close();
```

Если мы открыли окно с помощью метода `window.open()`, то из скрипта, работающего в новом окне, сослаться на окно-родитель можно с помощью `window.opener`. Поэтому, если необходимо закрыть родительское окно, т.е. окно, из которого было открыто текущее, то:

```
window.opener.close();
```

Если необходимо закрыть произвольное окно, то тогда сначала нужно получить его идентификатор:

```
id=window.open();
```

```
...
```

```
id.close();
```

Методы объекта `window`

▣ Методы `focus()` и `blur()`

Метод `focus()` применяется для передачи фокуса в окно, с которым он использовался. Передача фокуса полезна как при открытии окна, так и при его закрытии, не говоря уже о случаях, когда нужно выбирать окна.

Чтобы увести фокус из определенного окна, необходимо применить метод `myWin.blur()`. Например, чтобы увести фокус с текущего окна, где выполняется скрипт, нужно вызвать `window.blur()`.

Методы объекта `window`

▣ Метод `setTimeout()`

Метод `setTimeout()` используется для создания нового потока вычислений, исполнение которого откладывается на время (в миллисекундах), указанное вторым аргументом:

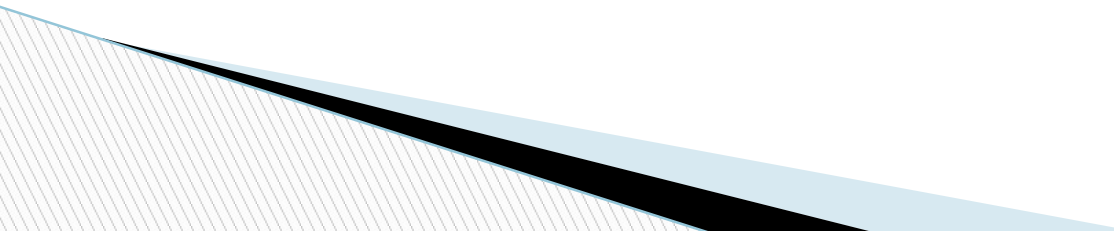
```
idt = setTimeout("JavaScript_код", Time);
```

Типичное применение этой функции - организация периодического изменения свойств объектов. Например, можно запустить часы в поле формы.

▣ Метод `clearTimeout()`

Метод `clearTimeout()` позволяет уничтожить поток, вызванный методом `setTimeout()`.

События объекта **window**

- ▣ *Load* - событие происходит в момент, когда загрузка документа в данном окне полностью закончилась.
 - ▣ *Unload* - событие происходит в момент выгрузки страницы из окна.
 - ▣ *Error* - событие происходит при возникновении ошибки в процессе загрузки страницы.
 - ▣ *Focus* - событие происходит в момент, когда окну передается фокус.
- 

События объекта `window`

- ▣ *Blur* - событие, противоположное предыдущему, происходит в момент, когда данное окно теряет фокус. Это может произойти в результате действий пользователя либо *программными средствами* - вызовом метода `window.blur()`.
- ▣ *Resize* - событие происходит при изменении размеров окна пользователем либо сценарием.

Объект `document`

Объект `document` является важнейшим свойством объекта `window` (т.е. полностью к нему нужно обращаться как `window.document`). Все элементы HTML-разметки, присутствующие на web-странице - текст, абзацы, гиперссылки, картинки, списки, таблицы, формы и т.д. - являются свойствами объекта `document`.

Свойства, методы и события объекта `document`

Свойства	Методы	События
URL	<code>open()</code>	Load
<code>domain</code>	<code>close()</code>	Unload
<code>title</code>		
<code>lastModified</code>	<code>write()</code>	Click
<code>referrer</code>	<code>writeln()</code>	DbClick
<code>cookie</code>		
	<code>getSelection()</code>	MouseDown
<code>linkColor</code>		MouseUp
<code>alinkColor</code>	<code>getElementById()</code>	
<code>vlinkColor</code>	<code>getElementsByName()</code>	KeyDown
	<code>getElementsByTagName()</code>	KeyUp
		KeyPress

Следующая лекция: Программирование формы

12 апреля 2017 года

