NURE

# *Гипертекст и гипермедиа*

ведущий лектор 1-го потока

*Ерохин Андрей Леонидович*

лектор 2-го потока

*Груздо Ирина Владимировна*

NURE

# Лекция №9

## Обработка событий в JavaScript

# Обработка событий в JavaScript

Название программы-обработчика события обычно состоит из названия собственно события, которому предшествует префикс “**on**”

*Типичный синтаксис:*

<ТЭГ обработчик_ события = " Имя_ программы-обработчика (аргументы) ">

**Пример:**

```
<html> <head>
<script type='text/javascript'>
function myobr(f) {
 if (confirm('Вы уверены?')) f.result.value=eval(f.expr.value)
     else
  alert('Повторите снова')
}
</script> </head>
```

NURE

```
<body>
<form style='background:silver' name='f'>
```
*Введите выражение:*

```
<input type='text' name='expr'  size='15' >
<input type ='button'  value='Переместить'
onClick='myobr(this.form)'>
```

*Результат:*

```
<input type='text' name='result' size='15'>
</form> </body>
```

просмотр примера

NURE

# Обработка событий в JavaScript

- JavaScript выполняется в едином потоке. Современные браузеры позволяют порождать подпроцессы Web Workers, они выполняются параллельно и могут отправлять/принимать сообщения, но не имеют доступа к DOM.
- Обычно события становятся в очередь и обрабатываются в порядке поступления, асинхронно, независимо друг от друга.
- Синхронными являются вложенные события, инициированные из кода.
- Чтобы сделать событие гарантированно асинхронным, используется вызов через setTimeout(func, 0).
- Отложенный вызов через setTimeout(func, 0) используется не только в событиях, а вообще – всегда, когда мы хотим, чтобы некая функция func сработала после того, как текущий скрипт завершится.

NURE

# Объект Event

позволяет скриптовой программе получить подробную информацию о возникшем событии

Особенности :

Объект **Event** доступен только во время самого события

Обращаться к **Event** можно только с обработчиков событий или функций

NURE

# *Свойства объекта Event*

**x, y** - горизонтальная и вертикальная координаты события
**offsetX, offsetY** - координаты события относительно контейнера
**clientX, clientY** - координаты события в клиентских координатах
**screenX, screenY** - координаты события относительно окна
**type** - тип события
**srcElement** - источник события
**srcElement.tagName** – имя тега-источника события
**button** – нажатая кнопка мыши
**keyCode** – код нажатой клавиши
**altKey,ctrlKey,shiftKey** – булевы значения нажатия клавиш Alt, Ctrl, Shift
**cancelBubble** – указывает, всплывает ли событие в иерархии объекта
**fromElement** – сохраняет источник события
**toElement** – сохраняет приемник события

https://developer.mozilla.org/ru/docs/Web/Events - **Event reference**

*Пример:*

*Получить информацию о событиях для всего документа*

```
<html><head>
  <script  type='text/javascript'>
     function InfoEvent()  {
     //event.srcElement придумали в IE. В других браузерах и новых IE это event.target.
      var st=event.srcElement.tagName;
      alert('Источник события - тег:  ' + st);
}
    </script>
</head>
<body onClick='InfoEvent()'>
<div> Первый  div</div>
<img src='#'  alt='Рисунок'  width='100'  height='120'  border='3'>
<div>  Второй div</div>
</body>
</html>
```

# Классификация способов обработки событий

1) Задание кода функции и вызов этой функции с использованием схемы присоединения события к обработчику
***onИмясобытия = Имя_функции***

2) Использование методов *captureEvents(імя*) – захват события
*handleEvent(імя)* – передает событие соответствующему обработчику
*routeEvent(імя*) –передает захваченное событие следующему обработчику

3) Использование схемы задания функции-обработчика

```
function Имя_обьекта.имя_события ( ) {
        код обработчика события
}
```

4) Поддержка общих событий
с использованием атрибутов **FOR и EVENT**

NURE

## *Пример*

```
<script  for='document'
            event='onmousemove()'
         type='text/javascript'>
alert('Мышка движется')
</script>
```

Атрибут FOR – имя или идентификатор ID элемента, для которого создан обработчик события

NURE

# Объект IMAGE

- для задания свойств изображений, включенных в состав страницы
- для загрузки изображений в кэш и их последующего отображения

## Свойства:

**src** - url-адрес рисунка

**complete** – булево выражение, указывает, загруженно ли изображение

**height** - высота бокса рисунка в пикселях

**width** - ширина

**border** - толщина бордера

**hspace** - горизонтальное дополнение рисунка пробелами (readonly)

**vspace** - вертикальное дополнение рисунка пробелами (readonly)

**name** - собственное имя объекта-рисунка

**lowsrc** - src для мониторов с низким разрешением

# Графический индикатор загрузки

```
<html><head>
<script type='text/javascript'>
function progress_load() {
if (document.images['bar'].width < 150) {
document.images['bar'].width+=5;
document.images['bar'].height=15; } else {
clearInterval(loadprogress);
}  }
var loadprogress;
//Событие onload на window срабатывает, когда загружается вся страница, включая
//ресурсы на ней – стили, картинки, ифреймы и т.п.
window.onload=function(){
loadprogress = setInterval('progress_load();', 400);
}
</script></head>
<body> <img src='bar.gif' name='bar'/> </body></html>
```

NURE

# пример:
# изменение изображений
# по циклу

-реакция на событие load

- функция изменения изображений *changepic()*

- используем *setTimeout*

# Изменение изображений по циклу

```
<html><head><title> Изменение изображений </title>
<script type='text/javascript'>
var i=1;
function changepic() {
var x=document.getElementById('mypic');
if (i<=4)
   {if (i==1) {x.src='1.gif'; i++}
        else if (i==2) {x.src='2.gif'; i++}
        else if (i==3) {x.src='3.gif'; i++}
        else if (i==4) {x.src='4.gif'; i=1}
setTimeout('changepic()',1000)}}
</script></head>
<body><div style='margin:10'> Изменение изображений по циклу</div>
<div style='margin:20' onmouseover='changepic()'>
<img src='1.gif'  id='mypic'  width='150'  height='150' border='2'>
</div></body></html>
```

просмотр примера

NURE

# *Упреждающая загрузка изображений*

**<img src="img1.gif" name="mypic"**
**width="150" height="150" >**
загружается изображение *img1.gif*
И получает имя **mypic**

***document. mypic.src= “img2.gif”;***
предыдущее изображение
*img1.gif* заменяется на новое - *img2.gif*

Новое изображение всегда получает тот же размер, который был у старого, и уже невозможно изменить размер бокса, в котором размещается изображение

NURE

Недостаток такого подхода:

после записи в **src** нового адреса начинается процесс загрузки соответствующего изображения

*Решение: упреждающая загрузка изображений*

NURE

```
hiddenImg= new Image();
//создается новый объект Image

hiddenImg.src= "img3.gif";
// адрес изображения, которое далее представлено с
//помощью объекта hiddenImg
```

Запись нового адреса в атрибуте **src** заставляет браузер загружать изображения с указанного адреса

NURE

После того, как браузер завершит загрузку, изображение на экране не появится.
Оно будет сохранено в кэше для последующего использования.

Чтобы подать изображения на экран:

***document.myImage.src= hiddenImg.src;***

// изображение извлекается из кэша и показывается на экране

NURE

# Изменение изображений в соответствии с событиями, которые инициированы пользователем

```
<html> <head><title>Изменение изображений </title></head>
<body>
<a href='#'
  onMouseOver="document.myImage2.src='1.jpg' "
  onMouseOut="document.myImage2.src='6.gif'">

<img src="1.jpg" name="myImage2" width='120'
    height='120' border='0'></a>
</body>
</html>
```

[просмотр примера]

NURE

# *Использование Cookies*

*Cookie является решением одной из проблем HTTP протокола - непостоянство соединение между клиентом и сервером:*

- для каждого документа (или файла) при передаче по HTTP протоколу посылается отдельный запрос)

- транзакция завершается после того, как браузер выполнил запрос, а сервер вернул соответствующий ответ

- сразу после этого сервер "забывает" о пользователе и каждый раз следующий запрос этого же пользователя считает новым пользователем

# Использование Cookies

Куки обычно используют в следующих целях:

- Авторизация пользователя;
- Хранения настроек пользователя;
- Ведения статистики о пользователях.

Как правило, реклама отображаемая на сайтах использует ваши куки для определения того, какой контент вам показывать.

NURE

**Cookies** — это механизм, позволяющий серверу хранить информацию на клиентском компьютере и при необходимости извлекать ее

С помощью механизма cookies сервер может хранить на клиентском компьютере некоторый именуемый информационный элемент

(имя пользователя, информация о налаживании, служебная информация, используемая в данной сессии и т.п.)

Механизм cookies поддерживается с помощью свойства *cookie* обьекте ***document***

NURE

- Используя **cookie**, можно эмулировать сессию по HTTP протоколу

*принцип эмуляция сессии*:

- при первом запросе выдается соответствующее значение cookie

- при каждом следующем запросе это значение считывается с переменной окружения HTTP_COOKIE и соответствующим образом обрабатывается

- document.cookie. Однако, оно представляет собой не объект, а строку в специальном формате, для удобной манипуляций с которой нужны дополнительные функции.

NURE

Для пользователя технология cookie это

- несколько файлов в папке %WINDOWS%\Cookies (Internet Explorer)
- или один файл cookie.txt (другие браузеры))

***Спецификации Cookie:***

- всего Cookies может быть не более 300
- каждый Cookie не может быть больше 4kb
- с одного домена второго уровня (плюс подуровни) не может быть получено более 20 Cookies
- информация по Cookie одного домена второго уровня (плюс подуровни) не может быть прочитана другими доменами
- если документ кэшируется, то информация о cookie не кэшируется
- информация Cookie может передаваться с помощью протокола SSL
- если лимит исчерпывается (4kb), то первые записи удаляются

NURE

## *Атрибуты Cookie:*

***name=value;*** каждый информационный элемент хранится в виде такой пары

***expires=date;*** задает срок «годности» информационного элемента. Если не задан, то срок годности истекает при закрытии браузера

***domain=djmainname;*** задает имя домена, из которого "видно" содержание данного информационного элемента

***path=path;*** задает маршрут, на котором "видно" содержание данного информационного элемента

***secure;*** задает защищенность информации

# Пример

*( из  w3schools.com)*

*Задача: запросить имя пользователя, сохранить его в виде информационного элемента. При последующих посещениях отображать в виде приветствия*

```
<html><head> <script type="text/javascript">
function getCookie(c_name)                          // функция чтения информационного элемента
{
   if (document.cookie.length>0)
  {
   c_start=document.cookie.indexOf(c_name + "=");
      if (c_start!=-1)    {
           c_start=c_start + c_name.length+1 ;
           c_end=document.cookie.indexOf(";",c_start);
   if (c_end==-1) c_end=document.cookie.length
   return unescape(document.cookie.substring(c_start,c_end));
                          }
   }
              return ""
 }
function setCookie(c_name,value,expiredays)  {  // функция установки  информационного элемента
var exdate=new Date();
exdate.setDate(exdate.getDate()+expiredays);
document.cookie=c_name+ "=" +escape(value)+((expiredays==null) ? "" : ";
     expires="+exdate.toGMTString());
                                                  }
```

```
function checkCookie() // функция проверки информационного элемента

{
username=getCookie('username');
if (username!=null && username!="")    {
            alert('Здравствуйте,  '+username+'рады Вас видеть на нашей
      странице!');
                                                  }
           else
 {
       username=prompt('Пожалуйста, представьтесь: ',"");
       if (username!=null && username!="")
       {
          setCookie('username',username,365);
          }
  }
 }
</script>
</head>
<body onLoad="checkCookie()">
</body> </html>
```

просмотр примера

NURE

# Ппроблема небезопасности

*Пример:*

Пользователь зашел на почтовый сайт, заполнил форму с login`ом и паролем, которые записались в cookie (даже через SSL).

Злоумышленник написал письмо пользователю в формате HTML с параметрами чтения cookie с паролями.

Прочитав cookie, HTML-файл запрашивает у пользователя разрешение отослать информацию злоумышленнику, login и пароль будут высланы злоумышленнику.

Злоумышленник также может добавить 0-й фрейм, где будет временно содержаться информация из cookie, которая при ответе на письмо будет добавляться в конец письма.

NURE

# Использование local storage (html 5)

- *при создании сайта удобно использовать локальную базу данных, которая расположена на стороне пользователя*
- *одна из серьезных проблем HTTP - это отсутствие статичности относительно состояния приложений*
- *классический способ сохранить состояние приложения для конкретного пользователя – это cookies*

# Использование local storage (html 5)

Имеется два варианта хранения данных, допустимые HTML5 спецификацией:

- Локальное хранение: позволяет сохранять информацию без ограничений по срокам хранения. Именно этот вариант желательно использовать, когда нужно хранить постоянно данные.
- Использование сеансов: обеспечивает сохранность данных лишь на период одного сеанса, то есть после закрытия пользователем вкладки приложения и повторного ее открытия вся необходимая для дальнейшей работы приложения информация будет удалена.

NURE

# Недостатки *cookies*

- подвешивается при загрузке каждого документа, открытого на домене
- максимальный объем данных для хранения – 4 Кб
- cookies можно незаконно использовать для слежения за поведением пользователя в сети, поэтому пользователи часто отключают cookies (или включают опцию запроса каждый раз, когда сайт пытается поставить cookies)
- неточная идентификация, возможность похищения или подмены, нестабильность между клиентом и сервером

NURE

# Использование локального хранилища в браузерах, совместимых с HTML5

- объект **localStorage** в JavaScript

Можно использовать
методы setItem() и getItem():

```
localStorage.setItem('studname','petrenko');
```

если ввести ключ *studname*,
получаем *petrenko*

```
var taste = localStorage.getItem('studname');
// получаем petrenko
```

Чтобы удалить объект – метод **removeItem()**:

```
localStorage.removeItem('favoriteflav');
var taste = localStorage.getItem('favoriteflav')
// получаем   null
```

NURE

# localStorage

- вместо ***localStorage*** можно использовать ***sessionStorage***,

если необходимо хранить данные только во время одной сессии

(пока пользователь не закроет браузер)

**Проблема «Strings Only»**

- Особенность ***local storage*** – можно использовать в ключах только переменные типа ***string***