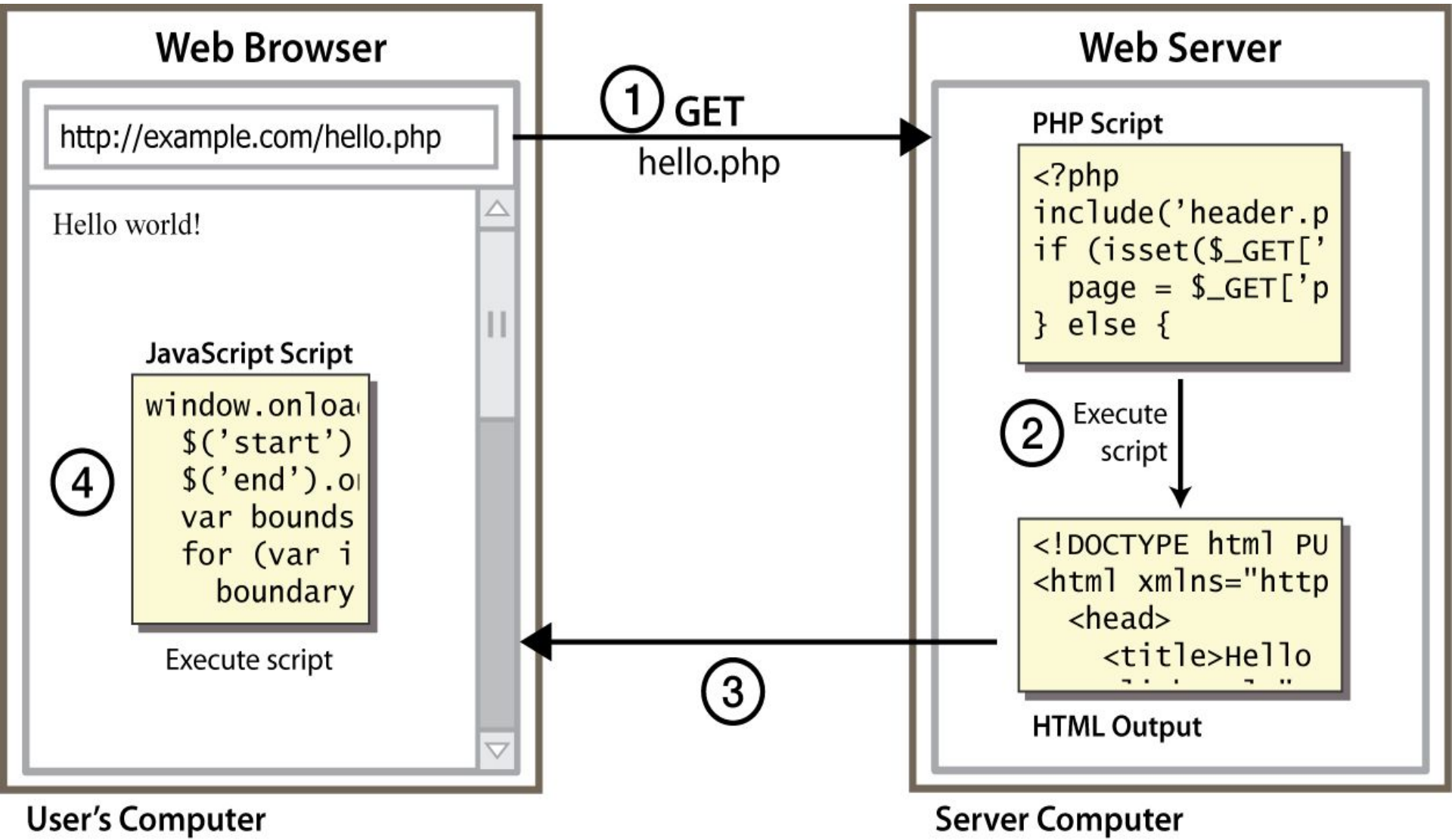
Введение в JavaScript

Сценарии на Стороне Клиента



Зачем использовать клиентское Программирование?

PHP уже позволяет нам создавать динамические веб-страницы. Зачем также использовать клиентские сценарии?

Преимущества сценариев на стороне клиента (JavaScript):

- удобство использования : можно изменить страницу без необходимости отправлять ее обратно на сервер (более быстрый пользовательский интерфейс)
- эффективность: может делать небольшие, быстрые изменения на странице, не дожидаясь сервера.
- управляемый событиями: может реагировать на действия пользователя, такие как щелчки и нажатия клавиш

Зачем использовать клиентское Программирование?

- преимущества программирования на стороне сервера (PHP):
  - безопасность: имеет доступ к личным данным сервера; клиент не может видеть исходный код
  - совместимость: не зависит от совместимости браузера.
  - питание: может записывать файлы, открывать соединения с серверами, подключаться к базам данных , ...

Javascript?

- облегченный язык программирования ("язык сценариев")
  - используется для того, чтобы сделать веб-страницы интерактивными
  - вставка динамического текста в HTML (например: имя пользователя)
  - реагировать на события (например: загрузка страницы пользовательским кликом)
  - получение информации о компьютере пользователя (например, тип браузера)
  - выполнение вычислений на компьютере пользователя (например, проверка формы)

Javascript?

- веб- стандарт (но не поддерживается одинаково всеми браузерами)
- Не связано с Java иначе как по имени и некоторым синтаксическим сходствам

Javascript vs Java

- интерпретируется, а не компилируется
- более расслабленный синтаксис и правила
  - меньшее количество и " более свободные" типы данных
  - переменные не нужно объявлять
  - ошибки часто молчат (мало исключений)
- ключевая конструкция- это функция, а не класс
  - "первоклассные" функции используются во многих ситуациях
- содержится в веб- странице и интегрируется с ее содержимым HTML / CSS

Javascript vs Java

JavaScript vs. PHP

- сходство:
  - оба интерпретируются, а не компилируются
  - оба расслаблены в отношении синтаксиса, правил и типов
  - оба варианта чувствительны к регистру
  - оба имеют встроенные регулярные выражения для мощной обработки текста

JavaScript vs. PHP

- различия:
  - JS более объектно-ориентирован: существительное.глагол (), менее процедурный: глагол(существительное)
  - JS фокусируется на пользовательских интерфейсах и взаимодействии с документом; PHP ориентирован на вывод HTML и обработку файлов / форм
  - Код JS выполняется в браузере клиента; код PHP выполняется на веб- сервере

Основные типы данных

1 целые численные:
в десятичной системе единиц: 0, 29, 70, -147 и т.п.;
в 16-ричной: 0x70 или 0x70, 0XFA7D0 и т.п.;
в 8-ричной: 070, 0710 (***Внимание!!!*** Ведущий ноль воспринимается как символ 8-ричного числа) и т.п.

2 вещественные численные:
0.0, -2.9, 0.7E1, 14.7e-2, 1e+308 (максимальное вещественное число), 1.001e-305 (минимальное по модулю вещественное число, отличное от нуля) и т.п.;

3 логические (булевские): true и false;

4 строковые: "Привет, все!", "ОК", 'Слово "Привет!" с кавычками внутри строки', "Другой вариант 'Привет' с кавычками внутри строки" и т.п. (допускаются оба типа кавычек и многократное использование таких пар внутри друг друга). Специальные символы обозначаются комбинацией символа \ и буквы (или последовательности цифр), например: \b — "забой", \n — перевод на новую строку, \" — "кавычка".

5 null — специальное значение для обозначения “пустого множества” значений.

Размещение кода JavaScript на HTML странице

- Гипертекстовая ссылка (схема URL)
- Обработчик события (handler)
- Подстановка (Entity)
- Элемент разметки Script

Переменные. Приведение типов

```
var myVariable=0.1
var B=false

myVariable="Теперь это текстовая переменная"
```

Операторы: арифметических действий, присваивания, инкрементные, декрементные. Условные выражения

Сложение "+",
вычитание "-",
умножение "*",
деление "/",
остаток от целочисленного деления "%".

y+=x эквивалентно y=y+x
y-=x эквивалентно y=y-x
y*=x эквивалентно y=y*x
y/=x эквивалентно y=y/x
y%=x эквивалентно y=y%x – остаток от деления нацело y на x.

Операторы сравнения

"= =" -"равно";
">" -"больше";
"<" -"меньше";
">=" -"больше или равно";
"<=" -"меньше или равно";
"!=" -"не равно".

Старшинство операций

условное выражение: "?:";
логическое "ИЛИ": "||";
логическое "И": "&&";
побитовое "ИЛИ": "|";
побитовое "XOR": "^";
побитовое "И": "&";
сравнение на равенство "= =", "!=";
сравнение: "<", "<=", ">", ">=";
побитовый сдвиг: "<<", ">>", ">>>";
сложение, вычитание: "+", "-";
умножение, деление: "*", "/";
отрицание, инкремент, декремент: "!", "~", "++", "--";
скобки: "( )", "[ ]".

Функции

```
function myFunction(arg1, arg2, ...)
{
        ...
    последовательность операторов
    ...
}
```

Условный оператор if

```
if(условие)
{
        утверждение1
}
else
{
        утверждение2
}
```

Цикл for

```
for(секция инициализации; секция условия; секция изменения счетчиков)
{
        утверждение
}
```

Цикл while

```
while(условие)
{
        выражение
}
```

Оператор with для работы с объектами

```
with(объект)
{
        утверждение
}
```

Правила работы с объектами

```
function IndividualPlan(student, yearCount)
{
        this.person=student;
        this.year=yearCount;
}

PaulPlan=new IndividualPlan(Paul,5);
PaulPlan.person.name.
```

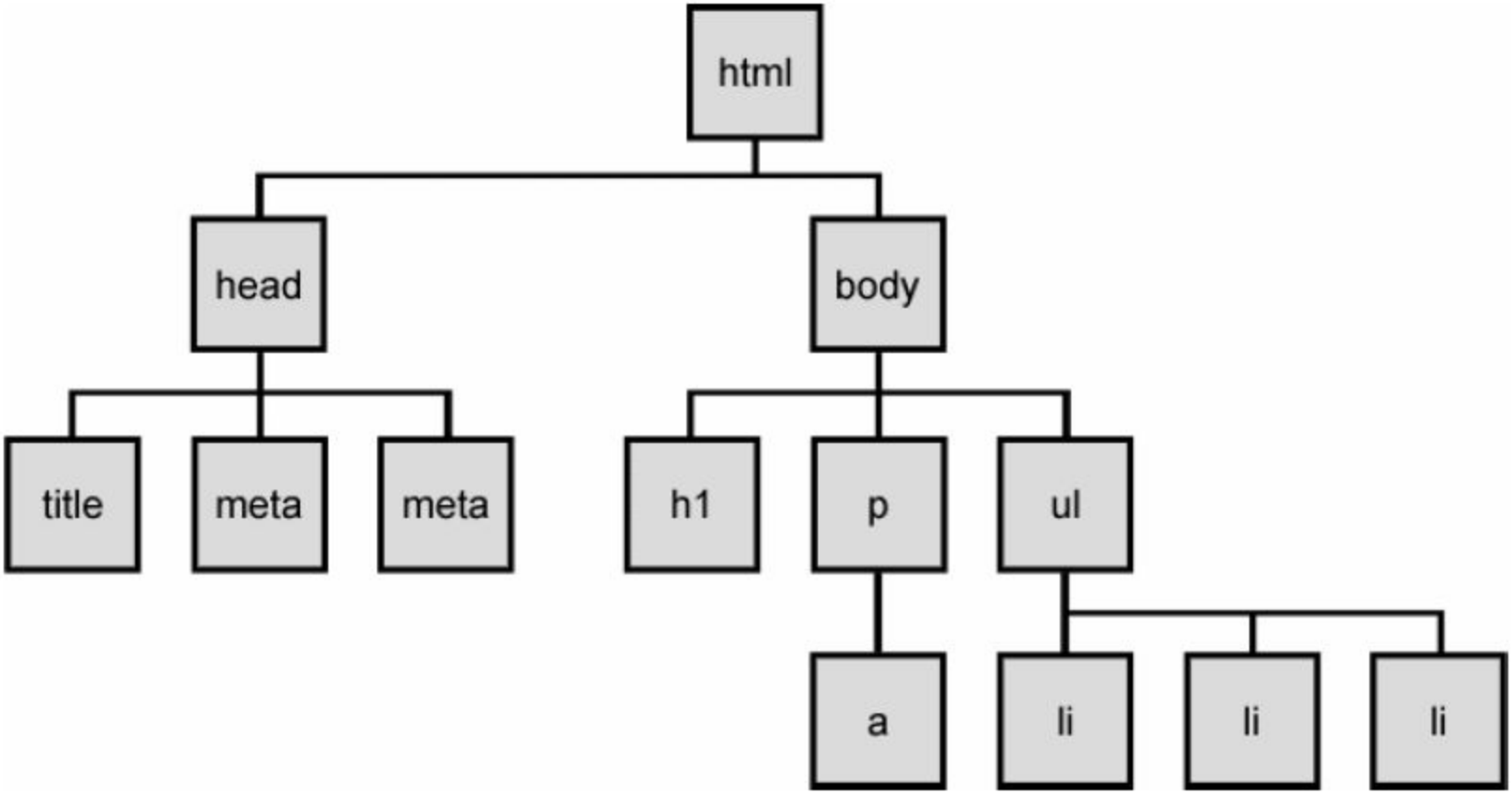
Классы и объекты языка JavaScript

Класс Global (задан неявно)
Класс Math
Класс Window
Коллекция фреймов (window.frames)
Класс Document (window.document)
Класс Location (window.location)
Класс Link (document.link)
Класс History
Класс MimeType
Класс Navigator

Экранные элементы

Класс Form (document.forms[i])
Классы Button
Класс Checkbox
Класс Radio
Класс Reset
Классы Text и Password
Класс Textarea
Классы Select и Option

The DOM tree



Типы DOM

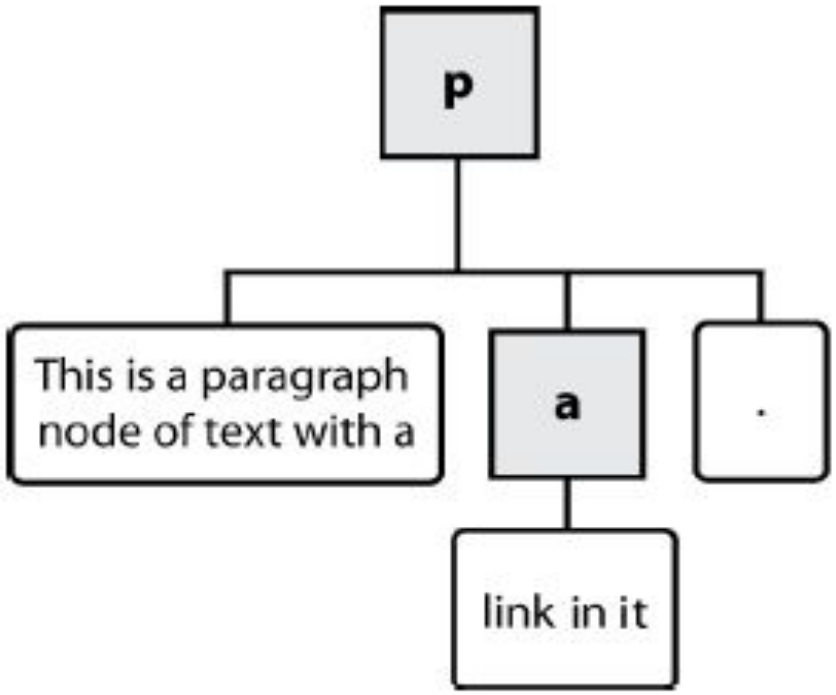
```
<p>
This is a paragraph of text with a
<a href="/path/page.html">link in it</a>.
</p>
```
HTML

- узлы элементов (HTML- тег)
  - может иметь детей и / или атрибуты
- текстовые узлы (текст в элементе блока)
- узлы атрибутов (пара атрибут / значение)
  - текст / атрибуты являются дочерними элементами в узле элемента
  - не может иметь детей или атрибутов
  - обычно не показывается при рисовании дерева DOM

Типы DOM

```
<p>
This is a paragraph of text with a
<a href="/path/page.html">link in it</a>.
</p>
```

HTML



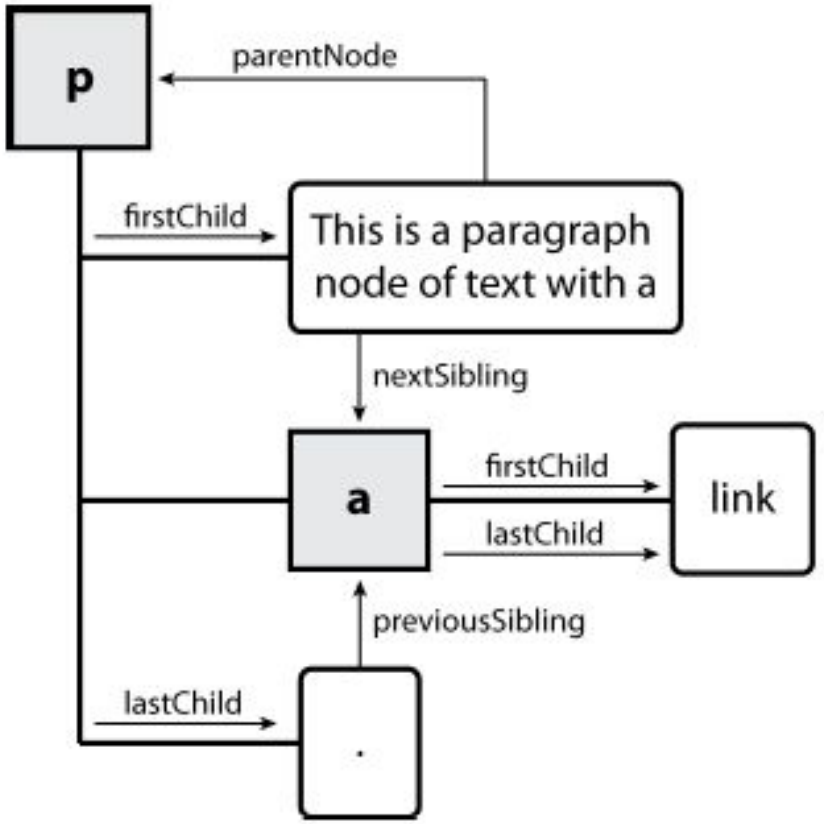
Обход дерева DOM

Наименование	Описание
firstChild, lastChild	start/end of this node's list of children
childNodes	array of all this node's children
nextSibling, previousSibling	neighboring nodes with the same parent
parentNode	the element that contains this node

DOM tree traversal example

```
<p id="foo">This is a paragraph of text with a
<a href="/path/to/another/page.html">link</a>.</p>
```

HTML



Выбор групп объектов DOM

- методы в документе и других объектах DOM для доступа к потомкам:

Наименованите	Описание
getElementsByTagName	returns array of descendents with the given tag, such as "div"
getElementsByName	returns array of descendants with the given name attribute (mostly useful for accessing form controls)

Изменение дерева DOM

Наименование	Описание
appendChild(node)	places given node at end of this node's child list
insertBefore(new, old)	places the given new node in this node's child list just before old child
removeChild(node)	removes given node from this node's child list
replaceChild(new, old)	replaces given child with new node

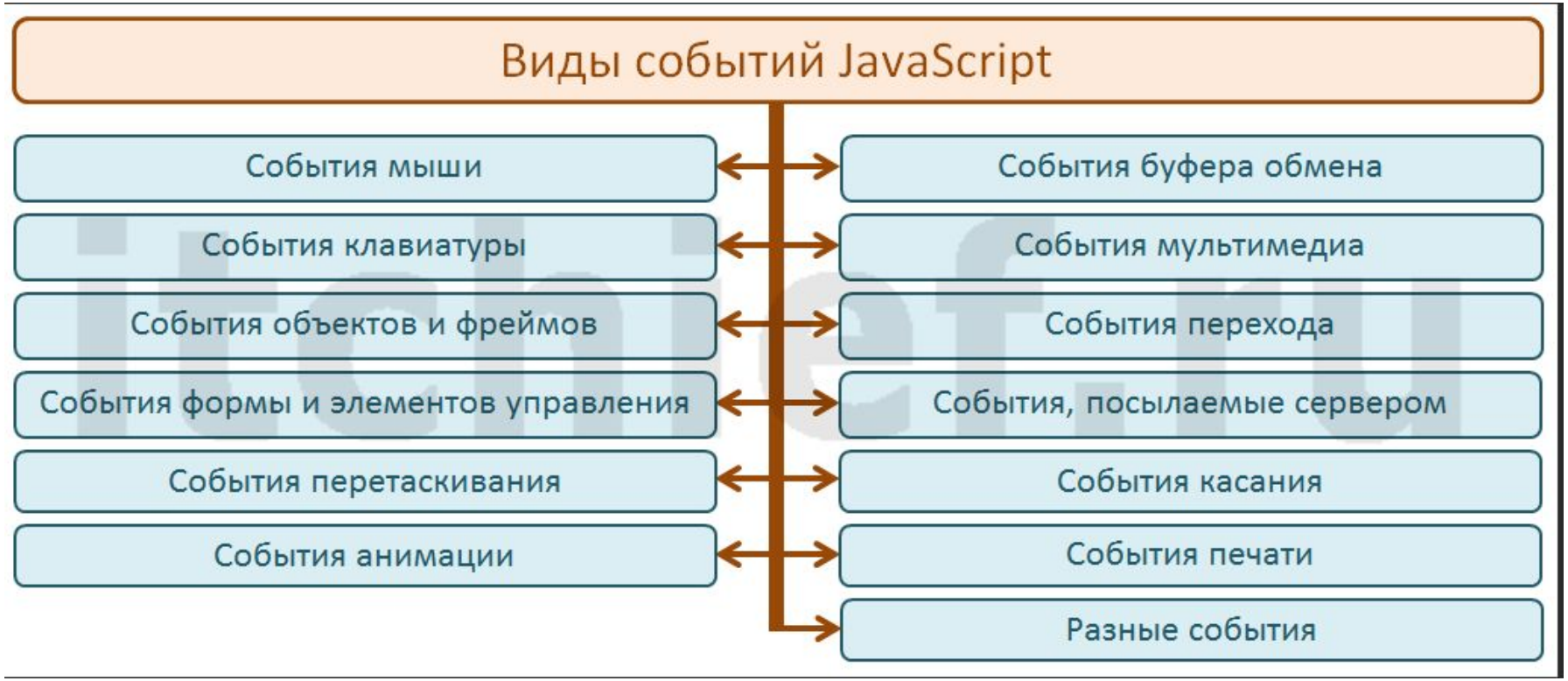
```
var p = document.createElement("p");
p.innerHTML = "A paragraph!";
$("main").appendChild(p);
```
JS

Удаление узла со страницы

```
function slideClick() {
    var bullets = document.getElementsByTagName("li");
    for (var i = 0; i < bullets.length; i++) {
        if (bullets[i].innerHTML.indexOf("children") >= 0)
{
            bullets[i].remove();
        }
    }
}
```

JS

- каждый объект DOM имеет метод removeChild для удаления его дочерних объектов со страницы
- Прототип добавляет метод remove для узла, чтобы удалить себя


Виды событий JavaScript
События мыши
События клавиатуры
События объектов и фреймов
События формы и элементов управления
События перетаскивания
События анимации
События буфера обмена
События мультимедиа
События перехода
События, посылаемые сервером
События касания
События печати
Разные события

события в JavaScript

устройство-зависимые

- onclick
- ondoubleclick
- onmouseover
- onmouseout
- onmousedown
- onmouseup
- onkeypress
- onkeyup
- onkeydown

устр.-независимые

- onfocus
- onblur
- onchange
- onselect
- onclick*

* только для ссылок и элементов форм

Проблемы с JavaScript

JavaScript- это мощный язык, но он имеет много недостатков:

- дом может быть неуклюжим в использовании

- один и тот же код не всегда работает одинаково в каждом браузере

Фреймворк JQuery

Функция $()

$(html)
$(elems)
$(fn)
$(expr, context)

$(html)

Позволяет создать html-элементы «на лету» из «чистого» HTML. Например, мы можем создать элемент div, содержащий параграф с текстом «Ба-бах!» и добавить его к элементу с id="body" таким образом:

```
var my_div = $("<div><p>Ба-бах!</p></div>");
my_div.appendTo("#body");
```

$(elems)

Позволяет «прицепить» всю функциональность jQuery к уже существующим элементам на странице (а именно к элементам из объектной модели документа, из DOM).

```
$(document.body).css( "background-color", "black" );
$( myForm.elements ).hide();
```

$(expr[, context])

- наиболее часто используемая функция. Первый, обязательный, параметр – это выражение, которое позволит jQuery найти элемент на странице. Второй, необязательный, параметр указывает, где искать этот элемент (по умолчанию jQuery будет искать по всей странице).
Например: Найдем все элементы p, находящиеся внутри всех элементов div на странице

$("div > p");

Найдем все радиокнопки в первой форме на странице

$("input:radio", document.forms[0]);

Найдем все элементы div в XML, что прислан с сервера с помощью технологии AJAX:

$("div", xml.responseXML);

Найдем все видимые элементы LI внутри элемента UL

$("ul > li:visible"); // CSS + селектор

$(fn)

- выполняет код в тот момент, когда доступна объектная модель документа, то есть когда браузер уже получил исходный код страницы полностью, но, возможно, еще не подгрузил различное мультимедийное содержимое (рисунки, видео, флэш).

```
$(document).ready(
  function() { /* ... */}
);
```

или его сокращенным вариантом, $(fn):

```
$(
  function(){/* ... */}
);
```

Пример:

```
$(document).ready(
  function()
  {
    // добавим ко всем ссылкам на странице некий текст
    $("a").append("<span>(opens in new window)</span>");
```

Функции JQuery

append(content)/prepend(content) - Добавить переданный элемент или выражение в конец/в начало выбранного элемента.

appendTo(expr)/prependTo(expr) - Добавить выбранный элемент в конец/в начало переданного элемента.

attr(name) - Получить значение атрибута.

attr(params) - Установить значение атрибутов. Атрибуты передаются в виде {ключ1:значение1[, ключ2:значение2[, …]]}

attr(name, value) - Установить значение одного атрибута.

css(name)/css(params)/css(name, value) - Получить/установить значение отдельных параметров CSS. Аналогично функции attr().

text()/text(val) - Получить/задать текст элемента. В случае ввода текста специальные символы HTML заменяются символьными примитивами (entities, например, знак ">" будет заменен >).

Empty() - Удалить все подэлементы текущего элемента.

Пример:

Код
```
<div id="my-div">
  <a href="http://google.com/" id="my-link">Ссылка</a>
</div>
```

$("#my-div").html();
Вернет Ссылка

$("#my-div").append("Полужирный текст");
// или
$("Полужирный текст").appendTo($("#my-div"));
HTML станет таким:
```
<div id="my-div">
  <a href="http://google.com/" id="my-link">Ссылка</a>
  <strong>Полужирный текст</strong>
</div>
```

$("#my-div").empty();

HTML станет таким:
```
<div id="#my-div"></div>
```

Пример:

HTML
Hello, how are you?

JQuery
$("p").find("span").html("Ahoy").end().append(" I'm fine ");

Код	Текущая коллекция элементов
$("p")	[..]
.find("span")	[..]
.html("Ahoy")	[..]
.end()	[..]
.append(...);	[..]

Результат HTML:

Ahoy, how are you? I'm fine

Пример:
Цвет всех ссылок на странице станет зеленым:
$("a").css("color", "green");

Индексы и get([index])

Возвращаемая функцией $() коллекция является «псевдо»-массивом DOM-элементов. Поэтому можно обратиться к индивидуальным элементам по их индексу:
$("p")[0].className = "test";

$("p"); // Получили объект jQuery
$("p").get(); // Получили соответствующую коллекцию DOM-элементов

$("a").get(0); // Получили первую из всех ссылок на странице

each(fn)

Иногда нужно пробежаться по всем элементам коллекции и выполнить над ними какие-то действия. Для этого нам понадобится функция each. Эта функция принимает в качестве аргумента другую функцию. each() работает в контексте найденных элементов и поэтому каждый раз, когда выполняется переданная ей функция, в этой функции доступна переменная this, указывающая на соответствующий DOM-элемент.

Animate

Ключевой функцией, на которой базируются все остальные, является функция animate:

animate(params, speed, easing, callback);

Здесь:

- params – свойства, которые участвуют в анимации в виде пар {ключ: значение}. Например: {height: "show"} или {opacity: 50, width: 100, height: 200}.
- speed – скорость в миллисекундах.
- easing – замедлениие анимации (замедляется ли к концу, "easein", или, наоборот, ускоряется, "easeout". Дополнительные типы доступны в модулях расширения).
- callback – функция, которая будет вызвана после завершения анимации.

Пример.

Пусть у нас есть элемент div с каким-нибудь текстом. Мы хотим скрыть этот текст, заменить новым, и показать обновленный текст.

```
$("#mydiv")
  .animate({height: "hide"}, 300)
```

Эффекты

Метод animate является основой большинства, если не всех, эффектов jQuery и плагинов. Например, jQuery предлагает следующие методы для показа и скрытия элементов:

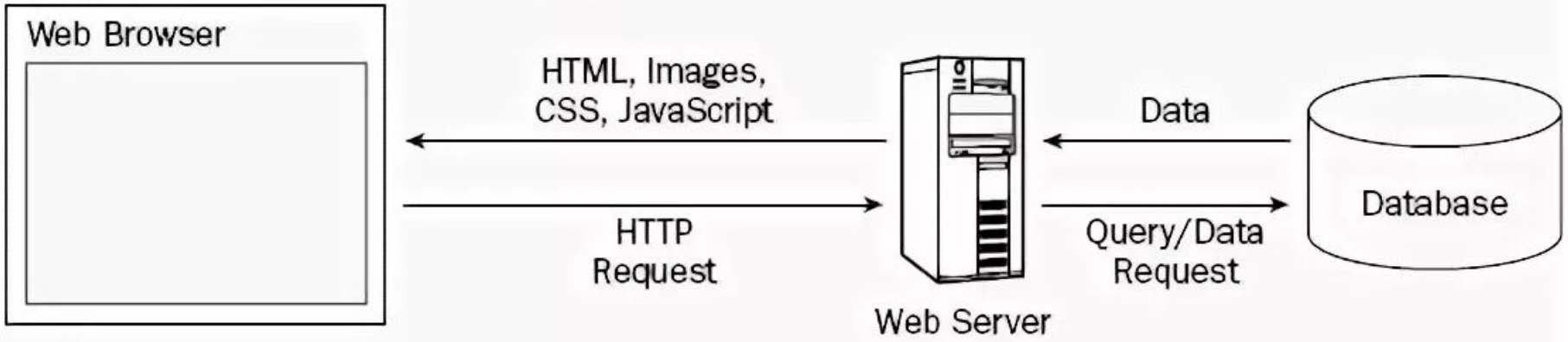
- show([speed[, callback]]) – показать элемент;
- hide([speed[, callback]]) – скрыть элемент;
- fadeIn(speed[, callback]) – показать элемент путем изменения его прозрачности;
- fadeOut(speed[, callback]) – скрыть элемент путем изменения его прозрачности;
- slideDown(speed, callback) – показать элемент, спустив его сверху;
- slideUp(speed, callback) – показать элемент, подняв его снизу;

где

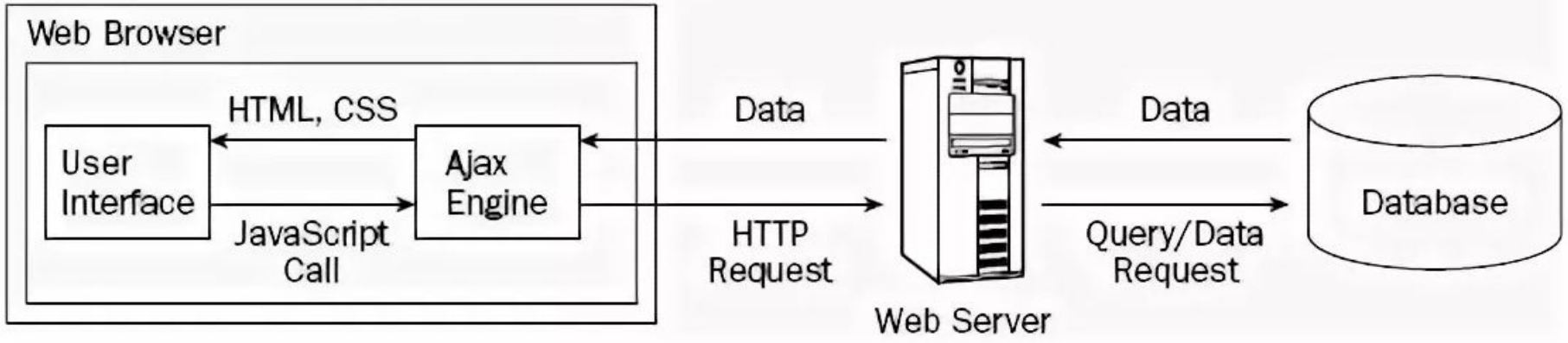
speed – скорость в миллисекундах или одно из "slow" (600 миллисекунд) или "fast" (200 миллисекунд);

callback – функция, которая будет вызвана после выполнения анимации.

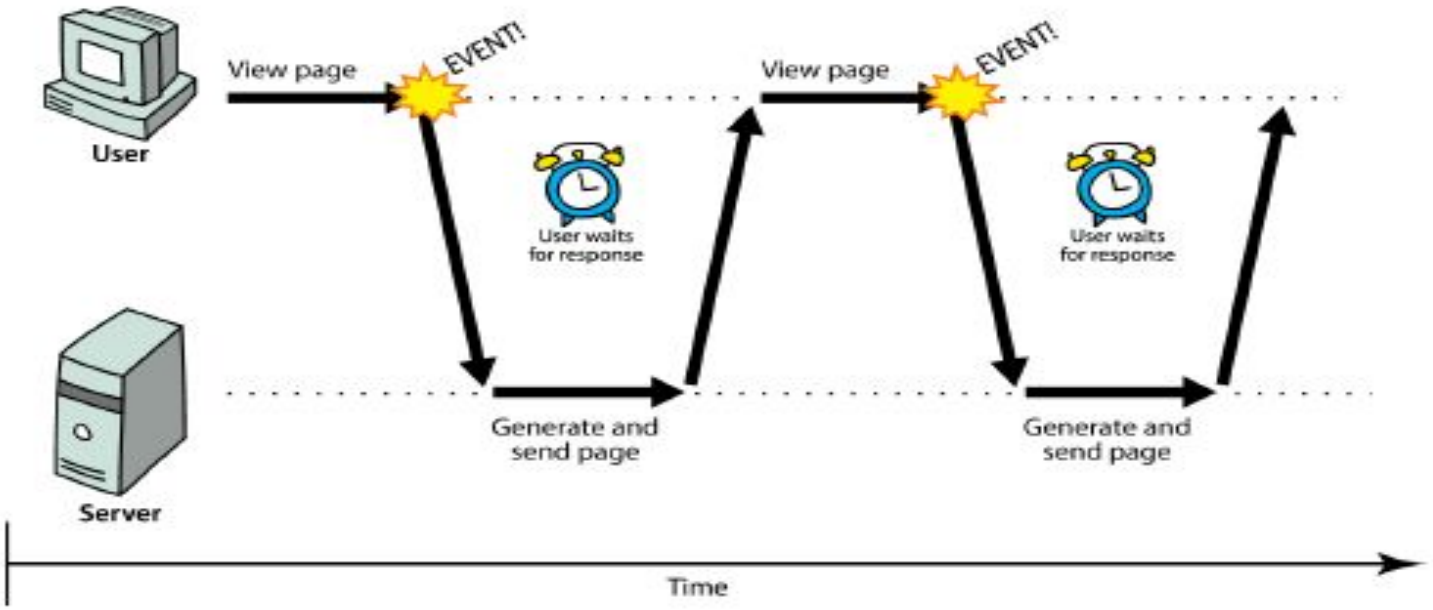
Традиционное веб-приложение



Веб-приложение с применением AJAX



Синхронная веб-связь



- синхронно: пользователь должен ждать загрузки новых страниц
  - типичный шаблон коммуникации, используемый на веб- страницах (нажмите, подождите, обновите)

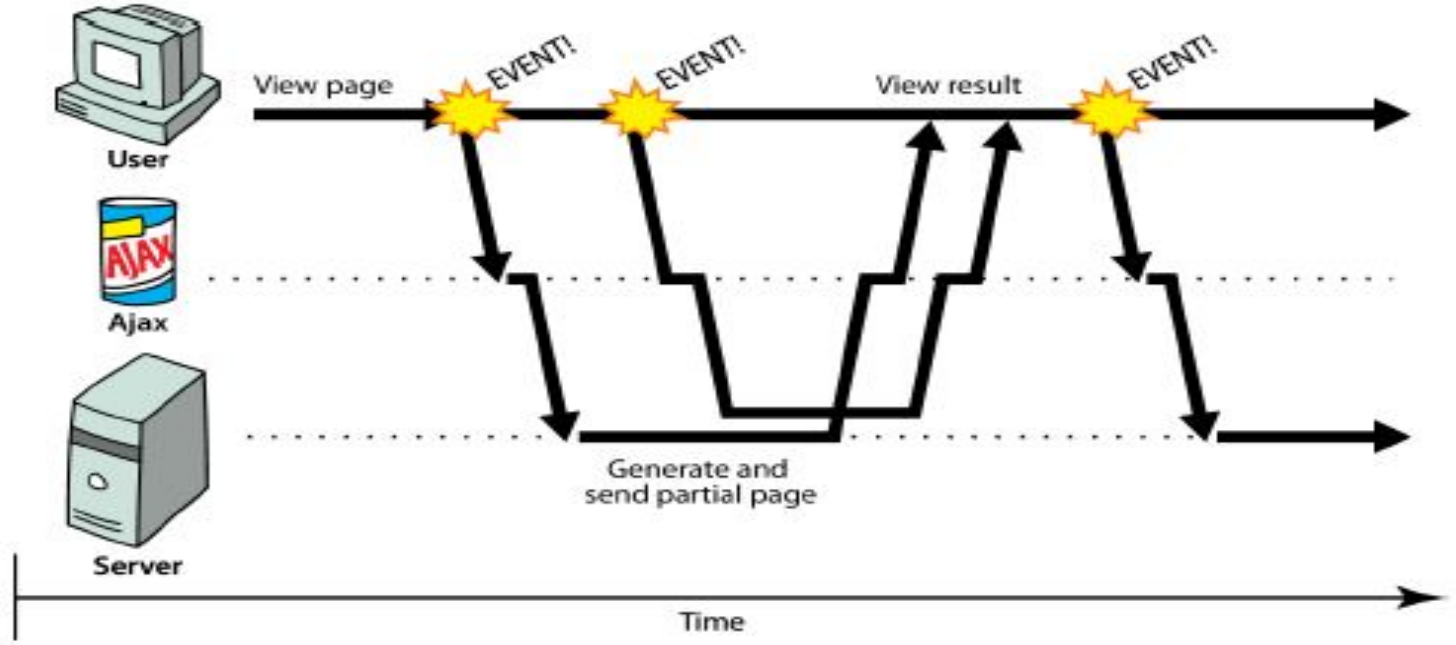
Веб-приложения и Ajax

- веб- приложение: динамический веб-сайт, который имитирует ощущение настольного приложения
  - представляет непрерывные пользовательские действия, а не разрозненные страницы
  - примеры: Gmail, Google Maps, Google Docs and Spreadsheets, Flickr, A9

Веб- приложения и Ajax

- Ajax: асинхронный JavaScript и XML
  - не язык программирования, а особый способ использования JavaScript
  - загрузка данных с сервера в фоновом режиме
  - позволяет динамически обновлять страницу, не заставляя пользователя ждать
  - во избежание "нажал-подождал-обнов шаблон
  - Пример: Google Suggest

Асинхронная веб- связь



- асинхронный: пользователь может продолжать взаимодействовать со страницей во время загрузки данных.
  - коммуникационный паттерн, ставший возможным благодаря Ajax

XMLHttpRequest

- JavaScript включает объект XMLHttpRequest, который может извлекать данные с веб-сервера
  - поддерживается в IE5+, Safari, Firefox, Opera, Chrome и т. д. (с незначительными совместимостями)
- он может делать это асинхронно (в фоновом режиме , прозрачно для пользователя)
- содержимое выбранного файла может быть помещено на текущую веб- страницу с помощью DOM

Типичный AJAX- запрос

1. пользователь щелкает мышью , вызывая обработчик событий
2. код обработчика создает объект XMLHttpRequest
3. XMLHttpRequest объект запрашивает страницу с сервера
4. сервер получает соответствующие данные, отправляет их обратно
5. XMLHttpRequest запускает событие при поступлении данных

   - это часто называется обратным вызовом вы можете присоединить функцию обработчика к этому событию
6. обработчик событий обратного вызова обрабатывает данные и отображает их

Методы и свойства прототипа Ajax

Свойства	Описание
method	how to fetch the request from the server (default "post")
parameters	query parameters to pass to the server, if any
asynchronous (default true), contentType, encoding, requestHeaders	

options that can be passed to the Ajax.Request constructor

Методы и свойства прототипа Ajax

События	**Описание**
onSuccess	request completed successfully
onFailure	request was unsuccessful
onException	request has a syntax error, security error, etc.

Базовый прототип Ajax шаблона

Свойства	Описание
status	the request's HTTP error code (200 = OK, etc.)
statusText	HTTP error code text
responseText	the entire text of the fetched page, as a String
responseXML	the entire contents of the fetched page, as an XML DOM tree (seen later)

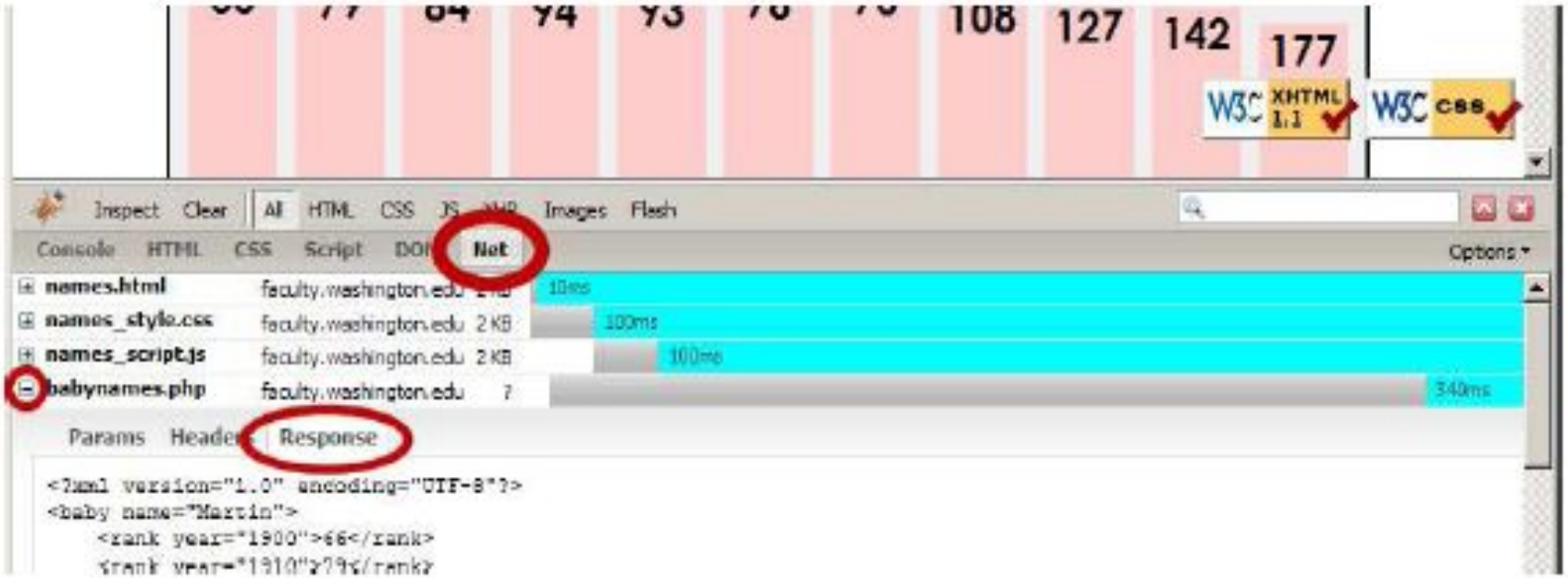
```
function handleRequest(ajax) {
   alert(ajax.responseText);
}
```
JS

Создание запроса POST

```
new Ajax.Request("url",
{
    method: "post", // optional
    parameters: { name: value, name: value, ..., name:
value },
    onSuccess: functionName,
    onFailure: functionName,
    onException: functionName
}
);
```

JS

Отладка Ajax- кода



- Net tab shows each request, its parameters, response, any errors
- expand a request with + and look at Response tab to see Ajax result

AJAX JQuery

Базовыми функциями для работы с AJAX являются post() и get() (есть еще более низкоуровневая, ajax(), но мы ее не будем рассматривать):

$.post(url[, params[, callback]])
$.get(url[, params[, callback]])

Здесь:

- url – адрес страницы, на которую будет отправлен запрос;
- params – параметры, передаваемые в запросе в виде пар «ключ : значение»;
- callback – функция, которая будет вызвана в случае успешного завершения вызова.

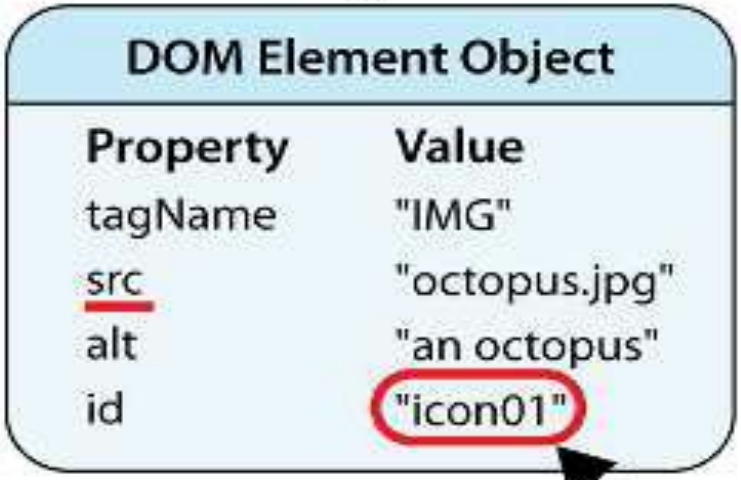
Пример:

```
$.post(
  '/ajaxtest.php',
  {
    type: "test-request",
    param1: "param1",
    param2: 2
  },
  onAjaxSuccess
);
function onAjaxSuccess(data)
{  // Здесь мы получаем данные, отправленные сервером
  alert(data);}
```

Объекты элементов DOM

HTML

```
<p>
  Look at this octopus:
  <img src="octopus.jpg" alt="an octopus" id="icon01" />
  Cute, huh?
</p>
```



JavaScript

```
var icon = document.getElementById("icon01");
icon.src = "kitty.gif";
```

Свойства объекта DOM

```
<div id="main" class="foo bar">
<p>Hello, I am <em>very</em> happy to see you!</p>
<img id="icon" src="images/potter.jpg" alt="Potter"
/>
```

HTML

Property	Description	Example
tagName	element's HTML tag	$("main").tagName is "DIV"
className	CSS classes of element	$("main").className is "foo bar"
innerHTML	content inside element	$("main").innerHTML is "\n <p>Hello, ve...
src	URL target of an image	$("icon").src is "images/potter.jpg"

Свойства DOM для элементов управления формами

```
<input id="sid" type="text" size="7" maxlength="7"
/>
<input id="frosh" type="checkbox" checked="checked"
/> Freshman?
```

HTML

Property	Description	Example
value	the text in an input control	$("sid").value could be "1234567"
checked	whether a box is checked	$("frosh").checked is true
disabled	whether a control is disabled (boolean)	$("frosh").disabled is false
readOnly	whether a text box is read-only	$("sid").readOnly is false

Злоупотребление innerHTML

- innerHTML может вводить произвольное содержимое HTML на страницу
- тем не менее, это склонно к ошибкам и ошибкам и считается плохим стилем

setTimeout пример

```
<button onclick="delayMsg();">Click me!</button>
<span id="output"></span>
```
HTML

```
function delayMsg() {
    setTimeout(booyah, 5000);
    $("output").innerHTML = "Wait for it...";
}
function booyah() { // called when the timer goes off
    $("output").innerHTML = "BOOYAH!";
}
```
JS

setInterval Пример

```
<button onclick="delayMsg();">Click me!</button>
<span id="output"></span>
```
HTML

```
var timer = null; // stores ID of interval timer
function delayMsg2() {
    if (timer == null) {
        timer = setInterval(rudy, 1000);
    } else {
        clearInterval(timer);
        timer = null;
    }
}
function rudy() { // called each time the timer goes off
    $("output").innerHTML += " Rudy!";
}
```
JS

Passing parameters to timers

```
function delayedMultiply() {
// 6 and 7 are passed to multiply when timer goes off
    setTimeout(multiply, 2000, 6, 7);
}
function multiply(a, b) {
    alert(a * b);
}
```
JS

- любые параметры передаются в функцию таймера

```
setTimeout(multiply(6 * 7), 2000);
```
JS

Распространенные ошибки

```
setTimeout(booyah(), 2000);
setTimeout(booyah, 2000);
setTimeout(multiply(num1 * num2), 2000);
setTimeout(multiply, 2000, num1, num2);
```

JS

Изменение style:

`element.style`

Attribute	Property or style object
color	color
padding	padding
background-color	backgroundColor
border-top-width	borderTopWidth
Font size	fontSize
Font famiy	fontFamily

Preetify

```
function changeText() {
   //grab or initialize text here

   // font styles added by JS:
   text.style.fontSize = "13pt";
   text.style.fontFamily = "Comic Sans MS";
   text.style.color = "red"; // or pink?
}
```

JS

Variables

```
var name = expression;                    JS
```

```
var clientName = "Connie Client";
var age = 32;
var weight = 127.4;                       JS
```

- variables are declared with the var keyword (case sensitive)
- types are not specified, but JS does have types ("loosely typed")
  - `Number, Boolean, String, Array, Object, Function, Null, Undefined`
  - can find out a variable's type by calling `typeof`

Number type

```
var enrollment = 99;
var medianGrade = 2.8;
var credits = 5 + 4 + (2 * 3);
```
JS

- integers and real numbers are the same type (no int vs. double)
- same operators: + - * / % ++ -- = += -= *= /= %=
- similar precedence to Java
- many operators auto-convert types: "2" * 3 is 6

Comments (same as Java)

```
// single-line comment
/* multi-line comment */
```
JS

- identical to Java's comment syntax
- recall: 4 comment syntaxes
  - HTML: <!-- comment -->
  - CSS/JS/PHP: /* comment */
  - Java/JS/PHP: // comment
  - PHP: # comment

Math object

```
var rand1to10 = Math.floor(Math.random() * 10 + 1);
var three = Math.floor(Math.PI);
```

JS

- methods: `abs, ceil, cos, floor, log, max, min, pow, random, round, sin, sqrt, tan`
- properties: `E, PI`

Special values: null and undefined

```
var ned = null;
var benson = 9;
// at this point in the code,
// ned is null
// benson's 9
// caroline is undefined
```
JS

- `undefined` : has not been declared, does not exist
- `null` : exists, but was specifically assigned an empty or null value
- Why does JavaScript have both of these?

Logical operators

- > < >= <= && || ! == != === !==
- most logical operators automatically convert types:
  - 5 < "7" is true
  - 42 == 42.0 is true
  - "5.0" == 5 is true
- === and !== are strict equality tests; checks both type and value
  - "5.0" === 5 is false

if/else statement (same as Java)

```
if (condition) {
   statements;
} else if (condition) {
   statements;
} else {
   statements;
}
```

js

- identical structure to Java's if/else statement
- JavaScript allows almost anything as a condition

Boolean type

```
var iLike190M = true;
var ieIsGood = "IE6" > 0; // false
if ("web devevelopment is great") { /* true */ }
if (0) { /* false */ }
```
JS

- any value can be used as a Boolean
  - "falsey" values: 0, 0.0, NaN, "", null, and undefined
  - "truthy" values: anything else
- converting a value into a Boolean explicitly:
  - `var boolValue = Boolean(otherValue);`
  - `var boolValue = !!(otherValue);`

for loop (same as Java)

```
var sum = 0;
for (var i = 0; i < 100; i++) {
   sum = sum + i;
}
```
JS

```
var s1 = "hello";
var s2 = "";
for (var i = 0; i < s.length; i++) {
   s2 += s1.charAt(i) + s1.charAt(i);
}
// s2 stores "hheellllloo"
```
JS

while loops (same as Java)

```
while (condition) {
   statements;
}
```
JS

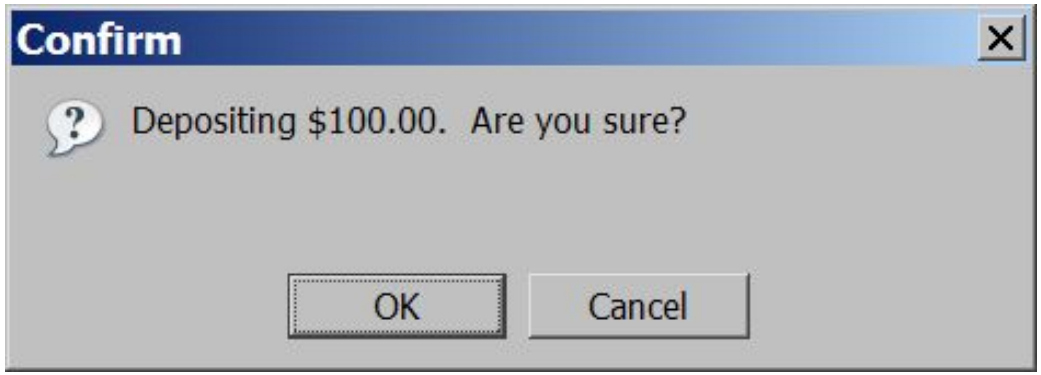
```
do {
   statements;
} while (condition);
```
JS

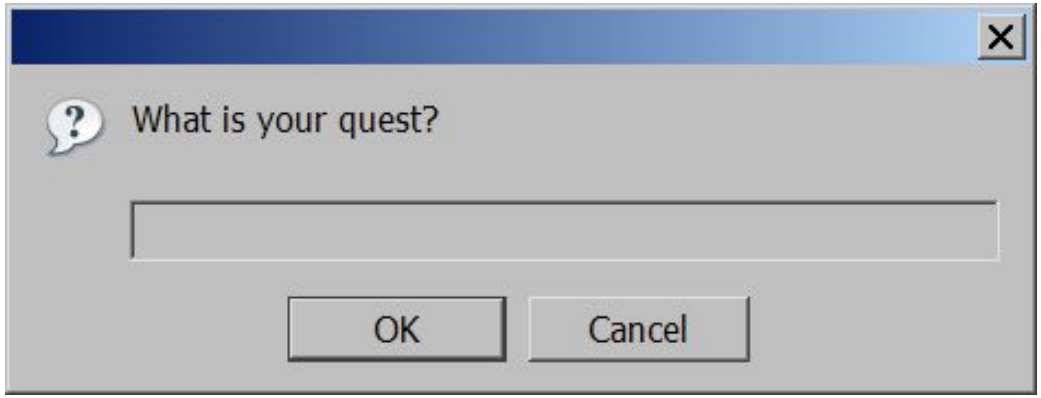
- break and continue keywords also behave as in Java

Popup boxes

```
alert("message"); // message
confirm("message"); // returns true or false
prompt("message"); // returns user input string
```

JS





Arrays

```
var name = []; // empty array
var name = [value, value, ..., value]; // pre-filled
name[index] = value; // store element
```

JS

```
var ducks = ["Huey", "Dewey", "Louie"];
var stooges = []; // stooges.length is 0
stooges[0] = "Larry"; // stooges.length is 1
stooges[1] = "Moe"; // stooges.length is 2
stooges[4] = "Curly"; // stooges.length is 5
stooges[4] = "Shemp"; // stooges.length is 5
```

JS

Array methods

```
var a = ["Stef", "Jason"]; // Stef, Jason
a.push("Brian"); // Stef, Jason, Brian
a.unshift("Kelly"); // Kelly, Stef, Jason, Brian
a.pop(); // Kelly, Stef, Jason
a.shift(); // Stef, Jason
a.sort(); // Jason, Stef
```

JS

- array serves as many data structures: list, queue, stack, ...
- methods: `concat, join, pop, push, reverse, shift, slice, sort, splice, toString, unshift`
  - push and pop add / remove from back
  - unshift and shift add / remove from front
  - shift and pop return the element that is removed

`String` type

```
var s = "Connie Client";
var fName = s.substring(0, s.indexOf(" ")); //
"Connie"
var len = s.length; // 13
var s2 = 'Melvin Merchant';
```

JS

`fromCharCode, indexOf, lastIndexOf, replace, split, substring, toLowerCase, toUpperCase`

  - charAt returns a one-letter String (there is no char type)
- length property (not a method as in Java)
- Strings can be specified with "" or '

More about `String`

- escape sequences behave as in Java: \' \" \& \n \t \\
- converting between numbers and Strings:

```
var count = 10;
var s1 = "" + count; // "10"
var s2 = count + " bananas, ah ah ah!"; // "10
bananas, ah ah ah!"
var n1 = parseInt("42 is the answer"); // 42
var n2 = parseFloat("booyah"); // NaN
```
JS

- accessing the letters of a String:

```
var firstLetter = s[0]; // fails in IE
var firstLetter = s.charAt(0); // does work in IE
var lastLetter = s.charAt(s.length - 1);
```
JS

Splitting strings: split and join

```
var s = "the quick brown fox";
var a = s.split(" "); // ["the", "quick", "brown",
"fox"]
a.reverse(); // ["fox", "brown", "quick", "the"]
s = a.join("!"); // "fox!brown!quick!the"
```
JS

- split breaks apart a string into an array using a delimiter
  - can also be used with regular expressions (seen later)
- join merges an array into a single string, placing a delimiter between them

Объекты

Свойства

```
<a href=kuku.htm>kuku</a>
document.links[0].href="kuku1.htm";
```

Методы

```
id=window.open("","example","width=400,height=150");
```

События

```
<input type=button value="Don't click here"
onClick="window.alert('We repeate again: DON\'T
CLICK HERE');">
```

Методы

```
function hello()
{
    id=window.open("","example","width=400,height=150");
id.focus(); id.document.open();
id.document.write("<h1>Hello JavaScript</h1>");
id.document.write("<hr><form>");
id.document.write("<input type=button value='Закрыть
окно' ");
id.document.write("onClick='window.opener.focus();
window.close();'>");
    id.document.close();
}
```

Пример 2


http://webclass.polyn.kiae.su/classes/jsi/html.htm - Wi...
about:blank
Hello JavaScript
Close the window
Интернет
100%

События

```
<input type=button value="Don't click here"
onClick="window.alert('We repeate again: DON\'T
CLICK HERE');">
```

Схема URL

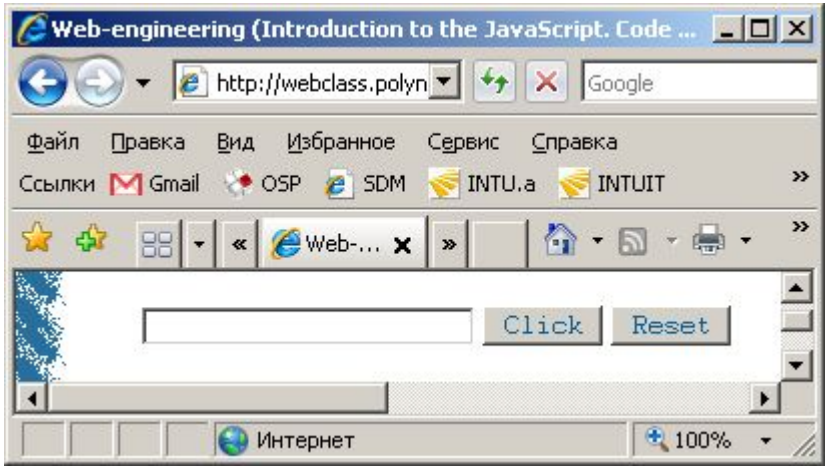
```
<a href="javascript:JavaScript_код">...</a>

<a href="javascript:alert('Attention!!!');">
<font color=red>Внимание!!!</font></a>
```

при нажатии на гипертекстовую ссылку Внимание!!! можно получить окно предупреждения:

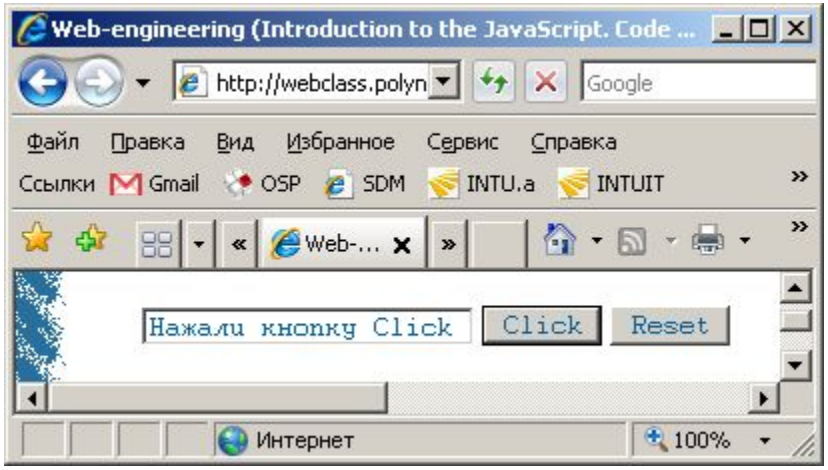
Схема URL

```
<form name=f method=post
action="javascript:window.document.f.i.value='Нажали кнопку
Click';void(0);">
<table border=0>
<tr>
<td><input name=i>
<td><input type=submit value=Click>
<td><input type=reset value=Reset>
</table>
</form>
```

Пример 3



До нажатия на кнопку Click



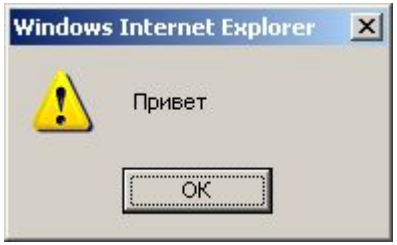
После нажатия на кнопку Click

Обработчик событий

```
<form><input type=button value=Кнопка
onClick="window.alert('Yahoo!!!');"></form>
```

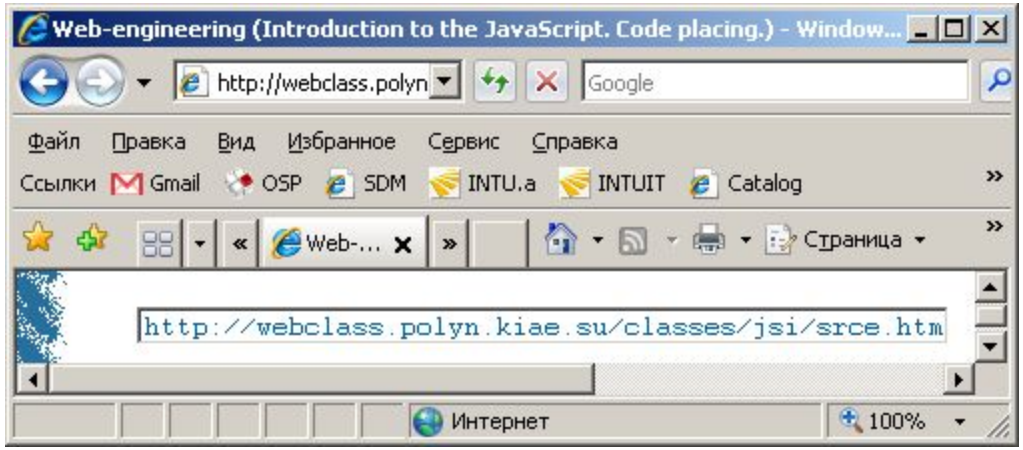
Обработчик событий

```
<form>
<select name=s1 onChange="
window.alert(
document.s.s1.options[document.s.s1.selectedIndex].text);">
<option>Привет
</form>
```



Подстановки

```
<script>
function l()
{
str = window.location.href;
return(str.length);
}
</script>
<form><input value="&{window.location};"
size="&{l();};"></form>
```



Вставка (контейнер SCRIPT)

```
<html>
<head>
<script language=JavaScript>
<!-- Это комментарий
...JavaScript-код...
// -->
</script>
<body>
... Тело документа ...
</body>
</html>
```

Вставка (контейнер SCRIPT)

```
<html><head></head>
<body onLoad=time_scroll()><center>
<h1>Часы в строке статуса</h1>
<form>
<input type=button value="Закрыть окно" onClick=window.close()>
</form>
</center>
<script>
function time_scroll()
{
d = new Date();
window.status =
d.getHours()+":"+d.getMinutes()+":"+d.getSeconds();
setTimeout('time_scroll();',500);
}
</script>
</body></html>
```

Вставка (контейнер SCRIPT)

```
<input type=button value="Изменить поле статуса в окне примера" onClick="id.defaultStatus='Привет';id.focus();">
```