JavaScript. Уровень 2
Расширенные возможности

Валентин Гревцов

Темы курса

- Объектная модель браузера
- Использование элементов HTML
- Объектная модель документа: DOM
- События и их обработка

Модуль 1

Объектная модель браузера

Темы модуля

- Объектная модель браузера
- Объект Window
- Объект Navigator
- Объект Screen
- Объект History
- Объект Location
- Объект Document
- HTML-коллекции

Псевдопротокол javascript: (demo-1-1.html)

- Псевдопротокол изменяет поведение ссылки. Это удобно при создании ссылки на номер телефона для мобильных устройств:
  +7-910-123-45-67

- Так же применяется для запуска функции JavaScript из ссылки

- function foo( ) {
      alert('Ура! Заработало!');
  }
  </script>
  </head>

  <body>
  Выполнить функцию foo( )

Объект Window

- Объект Window
  - Объект Navigator (свойство navigator)
  - Объект Screen (свойство screen)
  - Объект History (свойство history)
  - Объект Location (свойство location)
  - Объект Document (свойство document)
  - другие свойства
  - методы

Объект History

- Ссылается на объект history. Хранит информацию о переходах пользователя по отдельным страницам. Но мы не можем посмотреть где был пользователь

- Свойство **length** вернет кол-во страниц в истории сеанса работы с текущим окном (вкладкой) браузера
  console.log(history.length); // и что нам это дает?

- Методы:
  - **back( );** загружает предыдущий URL в списке истории;
  - **forward( );** загружает следующий URL в списке истории;
  - **go(** number **);** загружает определенный URL из списка истории

```
function goBack( ) {
  history.back( );
}
</script>

<a href='javascript: goBack( );'>Go Back</a>
```

Объект Navigator

- Отвечает за браузер, за его свойства

- Свойства
  - appVersion
  - userAgent
  - cookieEnabled
  - getBattery
  - geolocation

- Результат работы св-ва userAgent (Ноябрь 2019 года) для **стационарного ПК** и **мобильного устройства** (смартфон Motorola E4):

  - Mozilla/5.0 (**Windows NT 6.2**; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36

  - Mozilla/5.0 (**Linux**; **Android 7.1.1**; Moto E (4)) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.92 **Mobile** Safari/537.36

Определяем браузер (устройство) пользователя

- Метод **test( );** объекта RegExp проверяет, есть ли совпадение в строке относительно шаблона. Возвращает true если совпадение обнаружено, иначе, вернет false

- ```
  var ptrn = /Android|webOS|iPhone|iPad|iPod|BlackBerry|IEMobile|Opera Mini/i;

  if ( ptrn.test( navigator.userAgent ) ) {
      document.write( '<mark>Вы зашли с мобильного</mark><br>' );
      document.write( 'Ваш браузер: ' + navigator.userAgent + '' );
  } else {
      document.write( '<mark>Вы зашли с ПК</mark><br>' );
      document.write( 'Ваш браузер: ' + navigator.userAgent + '' );
  }
  ```
  ```

Объект геолокация

- API геолокация позволяет пользователю предоставлять свое местоположение web-приложению, если пользователь согласится предоставить его

- Для получения текущего местоположение пользователя, требуется вызвать метод **getCurrentPosition( );**

- ```
  navigator.geolocation.getCurrentPosition( function( x ) {
          console.log( 'Вы здесь: ' + x.coords.latitude + ', ' + x.coords.longitude );
      }
  );
  ```

- latitude – широта
  longitude – долгота
  ```

Объект Screen: свойства

- Отвечает за экран монитора. Можно узнать размер, найти середину

- Свойства
  - width
  - height
  - availWidth
  - availHeight
  - colorDepth

- Свойства **availWidth** и **availHeight** возвращают ширину и высоту экрана, исключая полосы прокрутки, панель инструментов и т.д.

- ```
  console.log( "Ширина экрана: " + screen.width );
  console.log( "Высота экрана: " + screen.height );
  ```
  ```

Объект Location: свойства

- Отвечает за адресную строку

- https://my-site.ru/index.html?user_name=John#top

- Свойства
  - protocol // https:
  - host // my-site.ru
  - hostname // my-site.ru
  - port
  - pathname // /index.html
  - search // ?user_name=John
  - hash // #top
  - href // https://my-site.ru/index.html?user_name=John#top

- console.log("host: " + location.host); // host: my-site.ru

- Переход на другой ресурс:
  - location.href = 'http://mail.ru';

Объект Location: методы (demo-1-2.html)

- Загружает документ заново.
  Флаг Boolean, при значении true указывает, что страница должна быть перезагружена с сервера. Если установлен в false или не указан, браузер может загрузить страницу из кэша
  - **reload(** true **);**

- Загружает новый документ в том же окне браузера
  - **assign(** 'http://site.ru' **);**

- Позволяет перейти на новую страницу, не запомнив текущую
  - **replace(** 'http://site.ru' **);**

- **toString( );** возвращает DOMString, содержащий URL целиком

Объект Document: обзор (demo-1-3.html)

- Содержит внутри себя html

- Свойства
  - lastModified
  - title
  - domain
  - body
  - …
    - document.body.style.background = '#cfc';

- Методы (создание нового окна браузера)
  - **open( );** открывает входной поток
  - **write(** string, [string[,…]] **);** пишет строку в поток документа
  - **close( );** завершает запись в документ

Объект Window (demo-1-4.html)

- Самый большой глобальный объект. Создается для каждого окна появляющегося на экране

- Свойства
  - frames
  - innerWidth
  - innerHeight

- Методы
  - Диалоговые окна
  - Таймеры
  - Манипуляции с окном

  - **print( );**
    - открывает диалоговое окно для печати текущего документа

  - **stop( );**
    - прекращает загрузку страницы

Диалоговые окна. Методы

- Выводит для пользователя окно предупреждения, заставляя щелкнуть на кнопке OK
  - alert('Привет');

- Предоставляет пользователю возможность подтверждения или отмены. Возвращает true или false
  - var x = confirm('Привет');
    console.log(x);

- Выводит окно для ввода информации. Возвращает string или null
  - var y = prompt('Введите имя', 'John');
    console.log(y);

- Нет возможности изменить внешний вид этих окон. Поэтому, можно либо написать свои, либо воспользоваться готовыми библиотеками. Например, библиотека jquery.com и плагин к ней jqueryui.com/dialog

Таймеры. Методы (demo-1-5.html)

- **setTimeout( )**; позволяет выполнить функцию JavaScript по истечении определенного промежутка времени и только один раз
  - `var x = setTimeout( code, 1000 );`

- clearTimeout( ); прекращает вызов метода setTimeout( )
  - `clearTimeout( x );`

- **setInterval( )**; используется для повторного выполнения ф-ции JavaScript через заданный интервал времени
  - `var y = setInterval( code, 1000 );`

- clearInterval( ); прекращает работу метода setInterval( )
  - `clearInterval( y );`

Таймеры. Передача аргументов

- В вызов функции вызываемой в таймере часто требуется передать параметры. Но, если написать **( )**, то функция будет вызвана сразу.
  Чтобы этого избежать можно воспользоваться функцией-оболочкой

- ```
  var txt = 'Вася';

  function username( name ) {
      alert( 'Привет, ' + name );
  }

  setTimeout( function( ) { username( txt ) }, 2000 );
  ```
  ```

Создание нового окна (demo-1-6.html)

- Свойства
  - closed – указывает, открыто ли целевое окно или нет
  - opener – возвращает ссылку на окно, которое открыло окно, используя open( )

- Методы
  - **open(** 'url', 'name', 'params', history **);** позволяет открыть с помощью JS новое окно. Метод получает 3 параметра, последний из которых устанавливает множество настроек, таких как

    ```
    open( 'https://bmstu.ru/', 'name', 'width=300, height=300, left=100, top=100' );
    ```

  - **close( );** закрывает текущее окно
  - **focus( );** переносит текущее окно на передний план
  - **blur( );** переносит текущее окно на задний план

Параметры нового окна

- width (число)
- height (число)
- left (число)
- top (число)
- location (1|0 или yes|no)
- menubar (1|0 или yes|no)
- scrollbars (1|0 или yes|no)
- toolbar (1|0 или yes|no)
- status (1|0 или yes|no)
- resizable (1|0 или yes|no)

Методы манипуляции окном

- Перемещает новое окно на новое место. Может выполняться несколько раз
  - **moveTo(** x, y **);**

- Перемещает новое окно на новое место. Может выполняться только один раз
  - **moveBy(** x, y **);**

- **resizeTo(** x, y **);**
- **resizeBy(** x, y **);**

- Прокрутка документа до указанных координат
  - **scrollTo(** x, y **);**

- Прокручивает документ на указанные величины
  - **scrollBy(** x, y **);**

Лабораторная работа – 1

- Использование свойств и методом объектов браузера

HTML-коллекция frames

- Интерфейс HTMLCollection является обобщённой коллекцией (объектом, ведущим себя подобно массиву) элементов (в порядке упоминания в документе) и предоставляет методы и свойства для получения хранящихся в нём элементов

- <iframe name='top' src='top.html'></iframe>
  <iframe name='bottom' src='bottom.html'></iframe>

- frames[0] //как у элемента массива
  frames['top'] //как к свойству объекта
  frames.top //как к свойству объекта

Отношения между окнами (frames)

- window
- self
- parent – доступ к родительскому окну
- top – доступ к самому верхнему окну, если несколько вложенных друг в друга фреймов
- name

- Запрет на открытие страницы во фрейме
  - onload = function( ) {
    if(top.frames.length > 0)
    top.location.href = 'http://ya.ru';
    };

Выводы

- Объектная модель браузера
- Объект Window
- Объект Navigator
- Объект Screen
- Объект History
- Объект Location
- Объект Document
- HTML-коллекции

Модуль 2

Использование элементов HTML

Темы модуля

- Свойства и методы элемента Form
- Свойства и методы элемента Input
- Свойства и методы элемента TextArea
- Свойства и методы элемента Select
- Свойства и методы элемента Option
- Свойства и методы элемента Image
- Свойства и методы элемента Table

HTML-коллекции

- forms
- images
- links

- <form name='search'>
  …
  </form>

- Доступ к элементу
  - document.forms[0]; // по индексу элемента в массиве
  - document.forms['search']; // по имени (как к свойству объекта)
  - document.forms.search; // по имени (как к свойству объекта)

- document.links[0].style.color = 'red';

Правила именования свойств

- element.style.padding = '10px';

- Правила именования (camelCase)
  - border-top-width => borderTopWidth

- Разрешение конфликта имен атрибутов
  - for => htmlFor
  - float => cssFloat
  - Исключение
    - class => className

Элементы HTML

- HTMLFormElement
  - HTMLInputElement
  - HTMLTextAreaElement
  - HTMLSelectElement
  - HTMLOptionalElement

- HTMLImageElement

- HTMLTableElement
  - HTMLTableSectionElement
  - HTMLTableRowElement
  - HTMLTableCellElement

HTMLFormElement (demo-2-1.html)

- Свойства
  - elements
  - length
  - name
  - action
  - enctype
  - method
- Методы
  - submit( )
  - reset( )

- ```
  <form action=''>
      <input type='text' name='user_name'> Введите имя<br>
      <input type='submit' value='Поехали'>
  </form>

  <script>
  var myForm = document.forms[ 0 ];       // ссылка на форму
  console.log( myForm.length );           // 2
  var first_elem = myForm.elements[ 0 ];
  console.log( first_elem );              // <input type='text' name='user_name'>
  ```
  ```

HTMLInputElement (demo-2-2.html)

- Свойства
  - **form**
  - **defaultValue**
  - **defaultChecked**
  - **checked**
  - maxLength
  - type
  - **value**
  - size
  - **name**

- Методы
  - select( )
  - click( )
  - focus( )
  - blur( )

HTMLTextAreaElement

- Свойства
  - **defaultValue**
  - **form**
  - **value**
  - cols
  - rows
  - readOnly
  - tabIndex
  - disabled
- Методы
  - select( )
  - focus( )
  - blur( )

Лабораторная работа – 2-1

- Проверка заполнения полей формы перед отправкой

HTMLSelectElement (demo-2-3.html)

- Свойства
  - **options**
  - **length**
  - **type**
  - **selectedIndex**
    - показывает порядковый номер выбранного элемента <option>. Значение -1 означает, что ни один из элементов не выбран
  - **value**
  - **form**
  - disabled и multiple
  - name
  - size и tabIndex
- Методы
  - add( )
  - remove( )
  - focus( )
  - blur( )

HTMLOptionElement

- Свойства
  - **form**
  - **text**
  - defaultSelected
  - **index**
    - положение опции в списке опций, к которым она принадлежит, в древовидном порядке
  - **value**
  - **selected**
  - disabled
  - label

Конструктор Option

- var newOpt = new Option(text, value, defaultSelected, selected);

- Добавление созданного элемента option внутрь select
  - select.add(newOpt, cutOpt);

- Удаление элемента option
  - select.remove(index);

- ```
  <select id='mySelect'>
      <option value='apple'>Яблоко</option>
  </select>

  <script>
  // создаем новый элемент option
  var option = new Option( 'Груша', 'pear', false, true );

  // null – добавление после последней option (для новых браузеров)
  mySelect.add( option, null );
  ```
  ```

Лабораторная работа – 2-2

- Добавление и удаление элементов списка

HTMLImageElement (demo-2-4.html)

- Это работа с изображениями. Главное его назначение – предварительная загрузка изображения. При этом изображение не выводится, а сохраняется в памяти

- `var img = new Image( 300, 200 );`

- Свойства
  - **src**

  - alt
  - width
  - height
  - border

- Большая картинка из интернета:
  - https://s1.1zoom.ru/big3/3/Canada_Lake_Scenery_507880.jpg

HTMLTableElement

- Свойства
  - **tHead**
  - **tFoot**
  - **tBodies[ ]**
  - **rows[ ]**

  - caption
  - width
  - border
  - cellPadding
  - cellSpacing

- Методы
  - createCaption( )
  - deleteCaption( )
  - **createTBody( )** // IE9+
  - createTHead( )
  - deleteTHead( )
  - createTFoot( )
  - deleteTFoot( )
  - insertRow(index)
  - deleteRow(index)

HTMLTableSectionElement

- Элемент HTML Table Body **<tbody>** инкапсулирует набор строк **<tr>** элементов таблицы, указывая, что они составляют тело таблицы **<table>**. Элемент **<tbody>** вместе со своими братьями **<thead>** и **<tfoot>** обеспечивает полезную семантическую информацию, которая может быть использована при визуализации для любого экрана или принтера, а также для доступности целей
  https://developer.mozilla.org/ru/docs/Web/HTML/Element/tbody

- Свойства
  - **rows**
  - align
  - vAlign

- Методы
  - insertRow(index)
  - deleteRow(index)

HTMLTableRowElement

- Свойства
  - **cells**
  - **rowIndex**
  - **sectionRowIndex**
  - vAlign
  - bgColor
- Методы
  - insertCell(index)
  - deleteCell(index)

HTMLTableCellElement (demo-2-5.html)

- Свойства
  - cellIndex
  - colSpan
  - rowSpan
  - align
  - vAlign
  - width
  - height

  - **textContent**
    - позволяет задавать или получать **текстовое содержимое** элемента и его потомков

  - **innerHTML**
    - устанавливает или получает HTML или XML **разметку** дочерних элементов

Геометрия (geometry.html)

- Определение координат элемента относительно:
  - окна браузера;
  - документа;
  - родителя

- Метод
  - **elem.getBoundingClientRect( )**; – возвращает (объект DOMRect) координаты сторон (и размер) элемента относительно окна браузера

- Свойства
  - **elem.offsetLeft** и **elem.offsetTop** – возвращают координаты Левой и Верхней сторон элемента относительно документа. Однако, если у родителя задано css свойство position: НЕ static; то дочерний элемент будет определять свое положение относительно родительского элемента

  - **elem.scrollTop** – считывает или устанавливает количество пикселей, прокрученных от верха элемента
    - document.documentElement.scrollTop = 50;

Выводы

- Свойства и методы элемента Form
- Свойства и методы элемента Input
- Свойства и методы элемента TextArea
- Свойства и методы элемента Select
- Свойства и методы элемента Option
- Свойства и методы элемента Image
- Свойства и методы элемента Table

Модуль 3

Объектная модель документа: DOM

Темы модуля

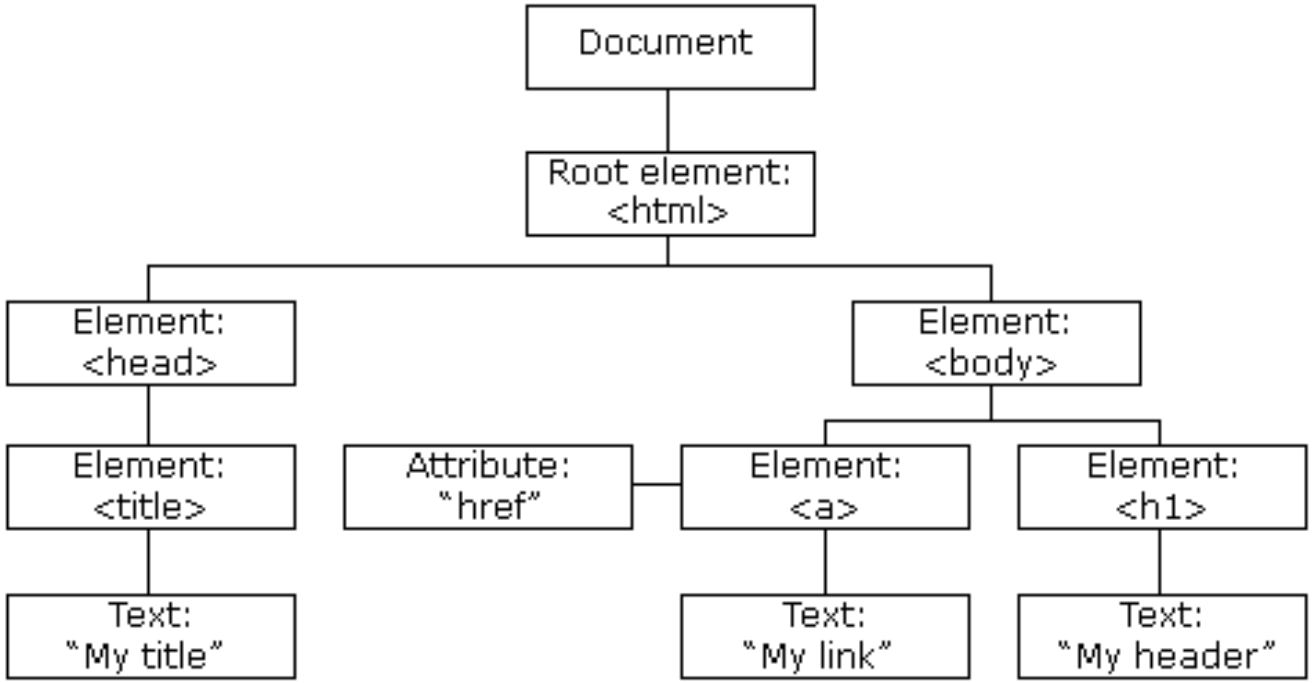
- DocumentObjectModel (DOM)
- Типы узлов
- Связи между объектами
- Свойства и методы интерфейса Node
- Свойства и методы интерфейса Element
- Свойства и методы интерфейса Document
- Нестандартные свойства и методы HTML – элементов
- Свойства и методы коллекции styleSheets

Уровни DOM

DOM 0	Традиционная модель JavaScript. Эквивалент модели, предложенной в Netscape 3.0 и IE 3.0 Поддерживает общие коллекции объектов документа: forms[ ], images[ ], anchors[ ], links[ ], applets[ ]
DOM 1	Обеспечивает возможность работы со всеми элементами документа посредством стандартного набора функций
DOM 2	Обеспечивает более совершенные возможности доступа к элементам страницы, а так же возможности доступа к таблицам стилей
DOM 3	Улучшение поддержки XML (на основе XPath). Обеспечение возможностей для обмена содержимыми файлов

Document Object Model

```
<html>
<head>
    <title>My title</title>
</head>
<body>
    <h1>My header</h1>
    <a href='#'>My link</a>
</body>
</html>
```



узел элемента (element)
текстовый узел (text node)

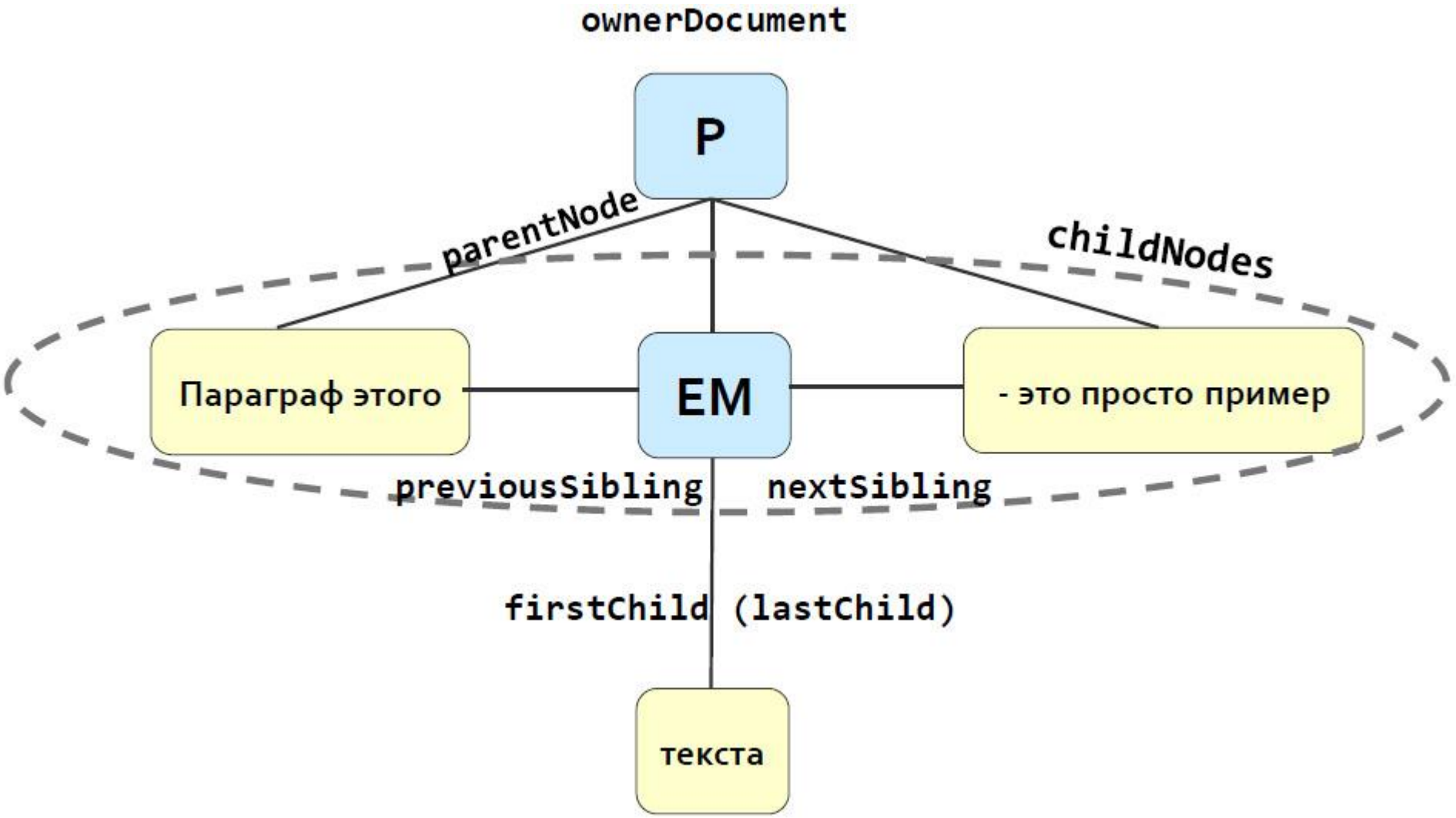
Отношения между узлами:
родительский – дочерний (parent – child), например **html** и **head**
предок – потомок (ancestor – descendant), например **html** и **title**
смежный (sibling), например **head** и **body**, **a** и **h1**

Типы узлов

Код типа	Тип узла	Описание	Пример
1	**ELEMENT**	**Элемент**	** … **
2	ATTRIBUTE	Атрибут элемента	charset='utf-8'
3	**TEXT**	**Тестовый узел**	**Это текст**
8	COMMENT	Комментарий	<!-- comment -->
9	DOCUMENT	Узел документа	document
10	DocumentType	Декларация типа документа	<!doctype html>

Связи между объектами

Параграф этого текста – это просто пример



Node – узел документа (demo-3-1.html)

- **Node** это интерфейс, от которого наследуют несколько типов DOM, он так же позволяет различным типам быть обработанными (или протестированными) https://developer.mozilla.org/ru/docs/Web/API/Node

- Свойства
  - parentNode
  - childNodes
  - firstChild
  - lastChild
  - nextSibling
  - previousSibling
  - ownerDocument

  - nodeName
  - nodeType
  - nodeValue

- Методы
  - node.hasChildNodes( )
  - node.cloneNode(bool)
  - parentNode.appendChild(e)
  - parentNode.insertBefore(e, p)
  - parentNode.replaceChild(e, p)
  - parentNode.removeChild(e)

Element – элемент документа (demo-3-2.html)

- Интерфейс **Element** представляет собой один из объектов в Document. Этот интерфейс описывает методы и свойства, общие для всех видов элементов https://developer.mozilla.org/ru/docs/Web/API/Element

- Свойства
  - Свойства Node
    +
  - tagName
  - className
  - **classList**

  - HTMLElement.**dataset** – предоставляет доступ как для чтения, так и для изменения всех пользовательских дата-атрибутов (**data-***), установленных у элемента

- Методы
  - Методы Node
    +
  - getElement**s**ByTagName( )
  - getElement**s**ByClassName( )
  - getElement**s**ByName( )

  - querySelector(css)
  - querySelector**All**(css)

  - elem.hasAttribute(a)
  - elem.getAttribute(a)
  - elem.setAttribute(a, v)
  - elem.removeAttribute(a)

Свойство classList (IE10)

- https://developer.mozilla.org/ru/docs/Web/API/Element/classList

- Свойство classList возвращает псевдомассив DOMTokenList, содержащий все классы элемента
- У classList есть примитивная альтернатива – свойство className, которое содержит значение атрибута class элемента
- Синтаксис
  - `var elementClasses = elem.classList;`

- Методы:
  - **add(** String [, String] **);** добавляет элементу указанные классы;
  - **remove(** String [, String] **);** удаляет у элемента указанные классы;
  - **toggle(** String [, Boolean] **);** если класс у элемента отсутствует то добавляет, иначе убирает. Когда вторым параметром передано false то удаляет указанный класс, а если true добавляет

- Свойство length, возвращает количество классов у элемента

Пример работы с элементом

- <img src="img_1.jpg" id="kartinka" class="first" alt="">

```
<script>
// получаем ссылку на HTML-элемент по id
var pict = document.getElementById( "kartinka" );

var a = pict.getAttribute( "src" );
console.log( a );  // img_1.jpg

// устанавливаем атрибута
pict.setAttribute( "title", "Привет" );

// добавляем класс
pict.classList.add( "second" );

var b = pict.getAttribute( "class" );
console.log( b );  // first second
```

Интерфейс Document

- Свойства Node
  +
  - **documentElement**
  - **body**
  - styleSheets
  - doctype

- Методы Node
  +
  - getElementById(id) – этот метод относится только к объекту Document
  - getElement**s**ByTagName( )
  - getElement**s**ByClassName( )
  - getElement**s**ByName( )
  - querySelector(css)
  - querySelector**All**(css)

```
<div id='d'>Элемент div</div>

<script>
// получаем ссылку на элемент (как селектор css). Удобно и привычно
var div = document.querySelector( 'body #d' );
console.log( div.textContent ); // Элемент div
```

Интерфейс Document (пример использования)

- <p>Первый параграф в body

  <p>Первый параграф в div
  <p>Второй параграф в div

  <script>
  // всего параграфов в документе
  var allP_in_body = **document**.getElementsByTagName('p');
  console.log(allP_in_body.length); // 3

  // получаю ссылку на элемент div
  var div = document.getElementById('d');
  console.log(div); // …

  // всего параграфов в div
  var allP_in_div = **div**.getElementsByTagName('p');
  console.log(allP_in_div.length); // 2

Лабораторная работа – 3-1

- Использование свойств и методов интерфейса Element

Создание узлов (demo-3-3.html)

- Методы интерфейса Document:
  - **createElement(** element_name **);** создает HTML элемент;
  - **createTextNode(** string **);** создает новый текстовый узел;
  - **createDocumentFragment( );** создает новый пустой DocumentFragment;
  - **createComment(** string **);** создает новый комментарий

- <h1>Заголовок H1</h1>

```
<script>
var p = document.createElement( 'p' );

p.setAttribute( 'title', 'Привет' );
p.textContent = 'Я параграф';

// добавляем параграф в конец дочерних элементов parentElem
document.body.appendChild( p );
```

Лабораторная работа – 3-2

- Создание элементов DOM

Получение фактических (вычисляемых) св-в CSS

- Метод **getComputedStyle( );** получает все фактические (вычисляемые) свойства CSS и значения указанного элемента
- Синтаксис:
  - window.getComputedStyle(element, pseudoElement);

- Вычисляемый стиль – это стиль, который фактически используется при отображении элемента после применения стилей из нескольких источников
- Источники стилей могут включать: внутренние таблицы стилей, внешние таблицы стилей, унаследованные стили и стили браузера по умолчанию

- <h2 id='test' style='display: none; padding-left: 100px;'>Test H2</h2>

```
<script>
var elem = document.getElementById( 'test' );
var x = window.getComputedStyle( elem, null ).getPropertyValue( 'display' );
var y = window.getComputedStyle( elem, null ).getPropertyValue( 'padding-left' );
var z = window.getComputedStyle( elem, null ).getPropertyValue( 'margin-top' );
console.log( x, y, z ); // none 100px 19.92px
```

Выводы

- DocumentObjectModel (DOM)
- Типы узлов
- Связи между объектами
- Свойства и методы интерфейса Node
- Свойства и методы интерфейса Element
- Свойства и методы интерфейса Document
- Нестандартные свойства и методы HTML – элементов
- Свойства и методы коллекции styleSheets

Модуль 4

События и их обработка

Темы модуля

- Список событий
- Модели событий
- Назначение обработчика событий
- Работа с обработчиками событий
- Получение ссылки на событие
- Отмена действий по умолчанию
- Кроссбраузерные свойства события
- Получение ссылки на элемент
- Использование фазы всплытия событий
- Отмена всплытия событий

Типы событий 1

- Срабатывают на большинстве элементов
    - onclick (можно отменить)
    - onmousedown
    - onmouseup
    - onmousemove
    - onmouseover
    - onmouseout

Типы событий 2

- <body>, <input>, <textarea>
  - onkeydown (можно отменить)
  - onkeypress (можно отменить)
  - onkeyup

- <body>, <a>, <button>, <input>, <label>, <select>, <textarea>
  - onfocus
  - onblur

Типы событий 3

- <body>, <iframe>, <img>
  - onload

- <body>
  - onunload

- <img>
  - onabort

- <body>, <img>
  - onerror

- <body>, <iframe>
  - onresize

Типы событий 4

- <form>
  - onsubmit (это т.н. перехватчик, можно отменить)
  - onreset (можно отменить)

- <input>, <textarea>
  - onselect

- <input>, <textarea>, <select>
  - onchange

- <body>, элементы с прокруткой
  - onscroll

Типы событий: DOM 2

- Window
  - onfocus
  - onblur
  - **onload**
  - onunload
  - onerror
  - onresize
  - onscroll
- Document
  - onkeydown
  - onkeypress
  - onkeyup
  - **DOMContentLoaded**
    - происходит когда весь HTML был полностью загружен и пройден парсером, не дожидаясь окончания загрузки таблиц стилей

```
function foo( ) {
    var div = document.getElementById( 'd' );
    console.log( div.textContent ); // Я элемент div
}

document.addEventListener( "DOMContentLoaded", foo );
</script>

<div id='d'>Я элемент div</div>
```

Типы моделей событий (demo-4-1.html)

- Базовая (исходная)
  - Например, позволяет в качестве атрибута элемента повесить обработчик
    Текст

- W3C DOM
  - Применяется для современных браузеров

- Internet Explorer
  - Применяется для старых Internet Explorer (IE8-)

Базовая (исходная) модель (demo-4-2.html)

- Прежде всего нас интересует отмена действия по умолчанию, например запрет отправки данных из формы при нажатии на кнопку <input type="submit">, если пользователь ввел некорректные данные (потребуется двойной return)

- ```
  function foo( )  {
      if( document.forms[ 0 ].text.value == '' ) {     // если пустая строка
          alert( 'Данные НЕ отправляются' );
          return false;
      } else
          alert( 'Данные отправляются' );
  }
  </script>

  <form action='http://ya.ru/yandsearch/' onsubmit='return foo( );'>
      <input type='text' name='text'> Ваше имя<br>
      <input type='submit' value='Поехали'>
  </form>
  ```
  ```

Передача ссылки на элемент

- Это мы уже смотрели в файле demo-2-5.html, когда строили таблицу

- ```
  function foo( el ) {
      alert( el.nodeName );  // P
      alert( el.textContent ); // Я обычный параграф
  }
  </script>

  <body>
  Я обычный параграф
  ```
  ```

Обработчик как свойство

- ```
  <form action='http://ya.ru/yandsearch/'>
      <input type='text' name='text'> Ваше имя<br>
      <input type='submit' value='Поехали'>
  </form>

  <script>
  // получаем ссылку на саму форму
  var myForm = document.forms[ 0 ];

  function foo( ) {
      alert( this ); // [object HTMLFormElement]
      // если текстовое поле пустое (т.е. пустая строка), данные не отправляем
      if( myForm.elements[ 0 ].value == '' ) return false;
  }

  // навешиваем обработчик
  myForm.onsubmit = foo;
  ```
  ```

Другие модели событий (demo-4-3.html)

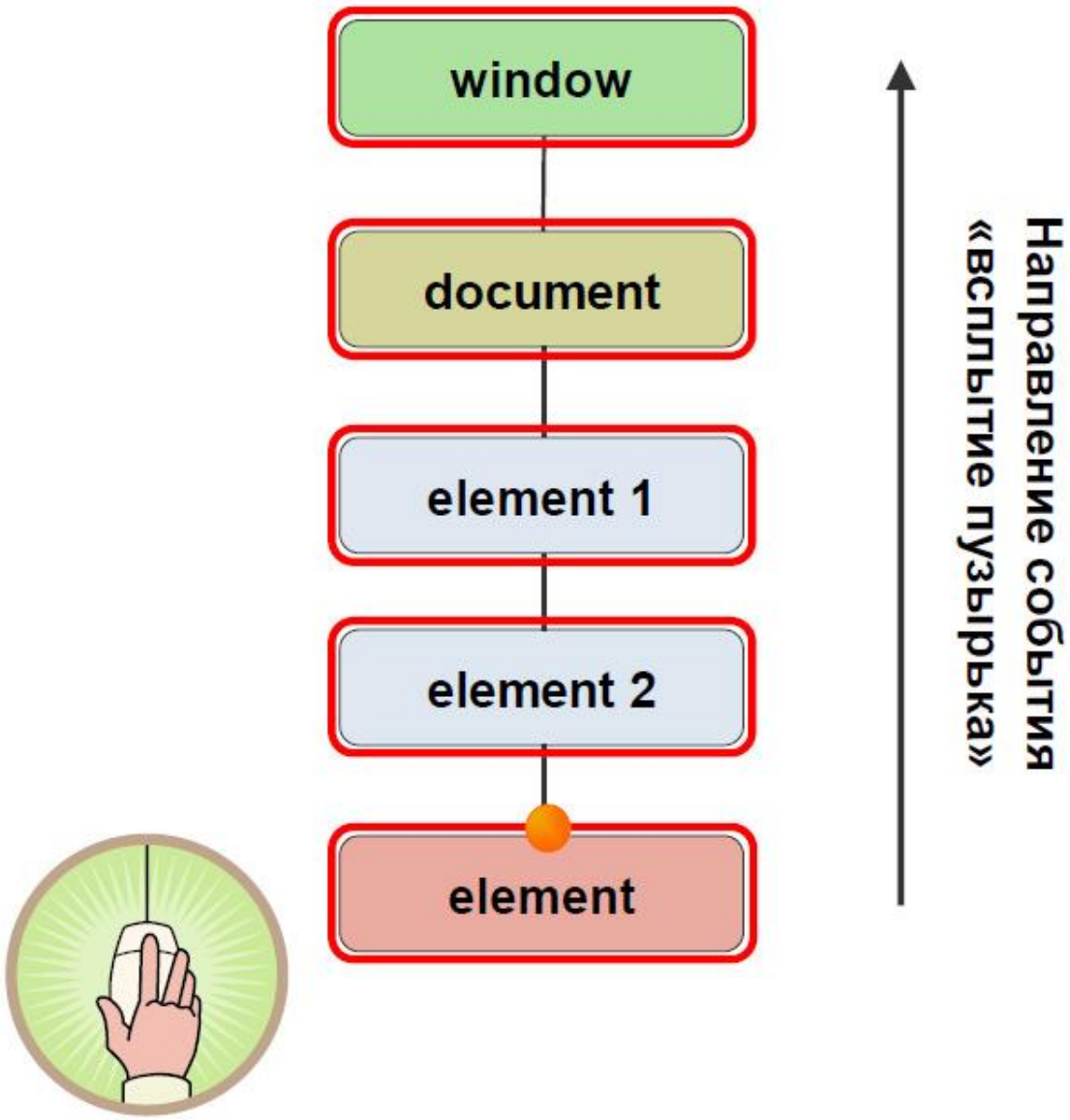
- При наступлении события обработчики сначала срабатывают на самом вложенном элементе, затем на его родителе, затем выше и так далее, вверх по цепочке вложенности, до элемента <html>
  Всплывают почти все события. Например, событие focus не всплывает

  body > div > p > span

- Модель W3C DOM
  - Фаза перехвата и распространения
  - Фаза регистрации
  - Фаза всплытия

- Модель IE
  - Фаза регистрации
  - Фаза всплытия

Модель обработки событий



Регистрация событий (demo-4-4.html)

- Метод **elem.addEventListener(** 'click', foo **);** регистрирует определенный обработчик события (название события без 'on'), вызванного на elem

- <h1 id='h1'>Я заголовок H1</h1>

```
<script>
var h1 = document.getElementById( 'h1' );

function f1( ) { alert( 'Раз' ); }
function f2( ) { alert( 'Два' ); }

// вешаем на один элемент несколько обработчиков
h1.addEventListener( 'click', f1 );
h1.addEventListener( 'click', f2 );
```

Отмена регистрации событий

- Метод **elem.removeEventListener(** 'click', foo **);** удаляет обработчик события, который был зарегистрирован при помощи elem.addEventListener( )

- <h1 id='h1'>Я заголовок H1</h1>

```
<script>
var h1 = document.getElementById( 'h1' );

function f1( ) { alert( 'Раз' ); }
function f2( ) { alert( 'Два' ); }

h1.addEventListener( 'click', f1 );
h1.addEventListener( 'click', f2 );

// удаляем один из обработчиков
h1.removeEventListener( 'click', f2 );
```

Объект событие

- Что бы хорошо обработать событие, недостаточно знать о том, что это 'click' или 'нажатие клавиши'. Могут понадобиться детали: координаты курсора, введенный символ и другие, в зависимости от события

- Детали произошедшего браузер записывает в 'объект событие', который передается первым аргументом в ф-цию обработчик

- **event.type** – тип события
- **event.target** – возвращает элемент инициализировавший событие (для IE8- используется св-во event.srcElement)
- **event.curentTarget** – элемент, на котором сработал обработчик
- **event.clientX** / **event.clientY** – координаты курсора (относительно окна) в момент клика
- **event.pageX** / **event.pageY** – координаты курсора (относительно документа) в момент клика

Получение ссылки на событие (demo-4-5.html)

- <h1 id='h1'>Объект событие</h1>

```
<script>
var header = document.getElementById( 'h1' );

header.onclick = function( ev ) {

    // получение ссылки на объект событие для IE8-
    ev = ev || window.event;

    console.log( 'x = ' + ev.clientX );
    console.log( 'y = ' + ev.clientY );
    console.log( 'Элемент: ' + ev.target );

};
```

Отмена действий по умолчанию

- Метод **preventDefault( );** интерфейса Event сообщает User agent, что если событие не обрабатывается явно, его действие по умолчанию не должно выполняться так, как обычно

- ```
  <h1 id='h1'>Объект событие</h1>

  <a href='http://ya.ru' id='link'>Ссылка</a>

  <script>
  link.onclick = function( ev ) {

      document.body.style.background = '#fcc';

      ev.preventDefault( );

  };
  ```
  ```

Лабораторная работа – 4-1

- Раскрывающееся меню в виде списка

Отмена всплытия события

- Получение ссылки на событие
  - ```
    function foo( ev ) {
      ev = ev || event;
    }
    ```

- W3C DOM
  - ev.stopPropagation( );

- ```
  <div id='first' onclick='alert( "first" );'>first
    <div id='second' onclick='alert( "second" ); event.stopPropagation( );'>second
      <div id='third' onclick='alert( "third" );'>third</div>
    </div>
  </div>
  ```

Лабораторная работа – 4-2

- Определение индекса элемента, на котором произошло событие

Выводы

- Список событий
- Модели событий
- Назначение обработчика событий
- Работа с обработчиками событий
- Получение ссылки на событие
- Отмена действий по умолчанию
- Кроссбраузерные свойства события
- Получение ссылки на элемент
- Использование фазы всплытия событий
- Отмена всплытия событий

Благодарю за внимание