Урок 9.

Преобразование объектов

Для совершения различных операций объекты конвертируют в примитивные значения. По умолчанию для конвертации применяются два метода `toString` и `valueOf`. Если они не объявлены внутри объекта то они вызываются автоматически на основании встроенных в язык.

- **toString**() - метод вызывается для преобразования объекта в строку. При объявлении должен возвращать любое примитивное значение. Все объекты имеют свою реализацию данного метода.
- **valueOf**() - метод для преобразования объекта в число. Встроенный метод для объектов вызывается в случаях когда объект приводится к числу. Если его нет вызывается **toString**(). При объявлении должен возвращать любое примитивное значение

Утиная типизация, [[Class]], instanceof

Утиная типизация - Если это выглядит как утка, плавает как утка и крякает как утка, то, вероятно, это утка (какая разница, что это на самом деле). Суть данной методики лежит в том чтобы определять принадлежность объекта к тому или иному классу в проверке наличии методов и свойств специфичных только данному классу.

```
var arr = [1, 2, 3];
if (arr.splice) {
    console.log('Это массив');
}
```

У всех встроенных объектов есть скрытое свойство **[[Class]]**. Его значение можно получить вызвав метод `toString` стандартного объекта, позаимствовав его. Этот метод может дать тип только для встроенных объектов. Для пользовательских конструкторов всегда **[[Class]]** = "Object"

```
{}.toString.call( 'string' ); // [object String]
{}.toString.call(12345); // [object Number]
```

instanceof - служит для проверки - экземпляром какого класса (какой функции является объект). Он осуществляет только проверку и не позволяет получить имя ввиде строки.

```
var d = new Date();
console.log(d instanceof Date); //true

function T() {}
var cc = new T();
cc instanceof T; //true
```

querySelector, querySelectorAll, matchesSelector

Методы **querySelector** и **querySelectorAll** осуществляют поиск по **DOM** дереву на основе CSS селекторов. Список CSS селекторов может быть найден здесь. querySelector возвращает первый найденный элемент и останавливает поиск. Если элемент не найден то возвращает **null**. **querySelectorAll** совершает поиск по всему документу. Возможно передавать несколько селекторов одновременно разделяя их запятой. **querySelectorAll** возвращает не только один элемент, совпадающий с CSS-селектором, но и все остальные, в виде «**staticNodeList**». «**StaticNodeList**» — это статичный набор элементов, на которые не влияют последующие изменения, происходящие в дереве документов, например удаление одного из элементов. Список **CSS** селекторов может быть найден здесь - http://www.w3schools.com/cssref/css_selectors.asp . **matchesSelector** - проверяет соответствует ли элемент **CSS** селектору и возвращает **true** или **false**

```
document.querySelectorAll(‘.mygroup'); // поиск всех элементов по классу
document.querySelector('#myheader') //возвращает элемент с ID=«myheader"
document.querySelector('p.description') //возвращает элемент P с class=«description"
document.querySelector('#myform input[type="radio"]:checked') //выбирает отмеченный пункт
```

Работа с формами

Получение и изменение значение элементов формы

- **input, textarea** - element.value - чтение свойства
- input type="**checkbox**", input type=«**radio**" - input.checked - может быть или **true** или **false**
- **select, option** - select.value - получение селекта с одним допустимым значением. Для получения значения из мультиселекта нужен проход цикла и проверка свойства **selected** у каждой option селекта.

События формы

- **change** - Изменение значения любого элемента формы. Для текстовых элементов срабатывает при потере фокуса.
- **input** - срабатывает тут же при изменении значения текстового элемента
- **cut, copy, paste** - срабатывают при вставке/копировании/удалении текста. В них можно отменить действие браузера, и тогда вставке/копирования/удаления не произойдёт.
- **submit** - событие срабатывает при попытке отправки формы на сервер. Может быть сгенерировано при помощи `<input type=«submit» />`. Отправка формы может быть отменена в обработчике события. При вызове метода submit у формы событие submit не генерируется.

data-* атрибуты

Для сохранения связанной информации с DOM элементами HTML поддерживает атрибуты формата **data-***. Для доступа к таким атрибутам используется свойство **dataSet**. Имя атрибута конвертируется в имя с **camel-case**. Свойство **dataSet** поддерживает как запись так и чтение аттрибутов.

```
<div data-custom-name=«some string» id=«some_id» >...</div>
<script>
  var div = document.getElementById(«some_id»);
  console.log( div.dataset.customName ); // some string
</script>
```

jQuery

jQuery — библиотека JavaScript, фокусирующая на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX. Библиотека предоставляет доступ к HTML элементами используя CSS селекторы. Основная цель библиотеки делать код короче и не фокусироваться на поведении в различный браузерах.

Адрес: http://jquery.com/
Документация: http://api.jquery.com/

С помощью библиотеки можно получать доступ к любым элементам и их коллекциям, навешивать обработчики событий, работать с атрибутами элементо, посылать AJAX запросы на сервер и многое другое.

Поиск в jQuery и перебор результатов

Поиск проводится по CSS селекторам. В результате возвращается внутренний объект который схожий по поведению с массивом. Доступ к элементам возможен по индексу, а также с помощью **jQuery** конструкции **$.each** которая принимает два параметра - коллекцию которую нужно перебрать, функция которая будет вызвана на каждом элементе коллекции. Этот объект также поддерживает поиск внутри этого набора результатов. Функция поиска поддерживает также передачу контекста как второго аргумента при поиске. Основная функция для поиска элементов является **$**.

```
var someElement = $('.mygroup');
var someOther = ('#myheader');
var anotherEl  = ('p.description');
var elements = ('#myform input[type=«radio»]:checked');

var firstP = anotherEl[0]; //доступ как в массиве. Не лучший способ
var el1 = elements.get(0); // доступ по индексу. Правильный способ.

//перебор элементов
$.each(anotherEl, function (index, element) {
    console.log(index, element)
});
```