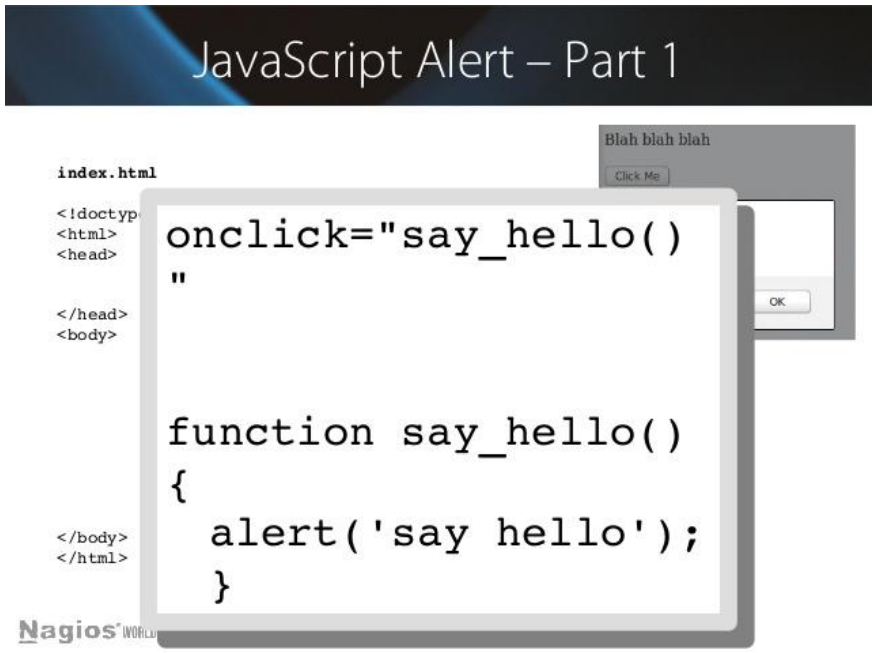
# Подробнее о событиях JS



# Типы событий мыши

## Простые события

mousedown

Кнопка мыши нажата над элементом.

mouseup

Кнопка мыши отпущена над элементом.

mouseover

Мышь появилась над элементом.

mouseout

Мышь ушла с элемента.

mousemove

Каждое движение мыши над элементом генерирует это событие.

## Комплексные события

click

Вызывается при клике мышью, то есть при mousedown, а затем mouseup на одном элементе

contextmenu

Вызывается при клике правой кнопкой мыши на элементе.

dblclick

Вызывается при двойном клике по элементу.

# Получение информации о кнопке: which

При обработке событий, связанных с кликами мыши, бывает важно знать, какая кнопка нажата.

**Для получения кнопки мыши в объекте event есть свойство which.**

На практике оно используется редко, т.к. обычно обработчик вешается либо onclick – только на левую кнопку мыши, либо oncontextmenu – только на правую.

Возможны следующие значения:

- event.which == 1 – левая кнопка
- event.which == 2 – средняя кнопка
- event.which == 3 – правая кнопка

# Правый клик: oncontextmenu

Это событие срабатывает при клике правой кнопкой мыши:

```
<div>Правый клик на этой кнопке выведет "Клик".</div>
<button oncontextmenu="alert('Клик!');">Правый клик сюда</button>
```

Правый клик на этой кнопке выведет "Клик".

Правый клик сюда

При клике на кнопку выше после обработчика `oncontextmenu` будет показано обычное контекстное меню, которое браузер всегда показывает при клике правой кнопкой. Это является его действием по умолчанию.

Если мы не хотим, чтобы показывалось встроенное меню, например потому что показываем своё, специфичное для нашего приложения, то можно отменить действие по умолчанию.

В примере ниже встроенное меню показано не будет:

```
<button oncontextmenu="alert('Клик!');return false">Правый клик сюда</button>
```

Правый клик сюда

# Модификаторы shift, alt, ctrl и meta

Во всех событиях мыши присутствует информация о нажатых клавишах-модификаторах.

Соответствующие свойства:

- `shiftKey`
- `altKey`
- `ctrlKey`
- `metaKey` (для Mac)

Например, кнопка ниже сработает только на Alt+Shift+Клик:

```
<button>Alt+Shift+Кликни меня!</button>

<script>
  document.body.children[0].onclick = function(e) {
    if (!e.altKey || !e.shiftKey) return;
    alert( 'Ура!' );
  }
</script>
```

Alt+Shift+Кликни меня!

# Координаты

Все мышиные события предоставляют текущие координаты курсора в двух видах: относительно окна и относительно документа.

Пара свойств `clientX/clientY` содержит координаты курсора относительно текущего окна.

При этом, например, если ваше окно размером 500x500, а мышь находится в центре, тогда и `clientX` и `clientY` будут равны 250.

Можно как угодно прокручивать страницу, но если не двигать при этом мышь, то координаты курсора `clientX/clientY` не изменятся, потому что они считаются относительно окна, а не документа.

Проведите мышью над полем ввода, чтобы увидеть `clientX/clientY`:

```
1 <input onmousemove="this.value = event.clientX+':'+event.clientY">
```

727:733

В той же системе координат работает и метод `elem.getBoundingClientRect()`, возвращающий координаты элемента, а также `position:fixed`.

### Относительно документа: pageX/Y

Координаты курсора относительно документа находятся в свойствах `pageX/pageY`.

Так как эти координаты – относительно левого-верхнего узла документа, а не окна, то они учитывают прокрутку. Если прокрутить страницу, а мышь не трогать, то координаты курсора `pageX/pageY` изменятся на величину прокрутки, они привязаны к конкретной точке в документе.

В IE8- этих свойств нет, но можно получить их способом, описанным в конце главы.

Проведите мышью над полем ввода, чтобы увидеть `pageX/pageY` (кроме IE8-):

```
1 <input onmousemove="this.value = event.pageX+':'+event.pageY">
```

687:5110

В той же системе координат работает `position:absolute`, если элемент позиционируется относительно документа.

# События mouseover/mouseout, свойство relatedTarget

Событие `mouseover` происходит, когда мышь появляется над элементом, а `mouseout` – когда уходит из него.


При этом мы можем узнать, с какого элемента пришла (или на какой ушла) мышь, используя дополнительное свойство объекта события `relatedTarget`.

Например, в обработчике события `mouseover`:

- `event.target` – элемент, на который пришла мышь, то есть на котором возникло событие.
- `event.relatedTarget` – элемент, с которого пришла мышь.

Для `mouseout` всё наоборот:

- `event.target` – элемент, с которого ушла мышь, то есть на котором возникло событие.
- `event.relatedTarget` – элемент, на который перешла мышь.

http://learn.javascript.ru/article/mousemove-mouseover-mouseout-mouseenter-mouselea

# Мышь: Drag'n'Drop

Drag’n’Drop – это возможность захватить мышью элемент и перенести его. В свое время это было замечательным открытием в области интерфейсов, которое позволило упростить большое количество операций.

Основной алгоритм Drag’n’Drop выглядит так:

- Отслеживаем нажатие кнопки мыши на переносимом элементе при помощи события mousedown.
- При нажатии – подготовить элемент к перемещению.
- Далее отслеживаем движение мыши через mousemove и передвигаем переносимый элемент на новые координаты путём смены left/top и position:absolute.
- При отпускании кнопки мыши, то есть наступлении события mouseup – остановить перенос элемента и произвести все действия, связанные с окончанием Drag’n’Drop.

# Мышь: Drag'n'Drop

```
var ball = document.getElementById('ball');

ball.onmousedown = function(e) { // 1. отследить нажатие

  // подготовить к перемещению
  // 2. разместить на том же месте, но в абсолютных координатах
  ball.style.position = 'absolute';
  moveAt(e);
  // переместим в body, чтобы мяч был точно не внутри position:relative
  document.body.appendChild(ball);

  ball.style.zIndex = 1000; // показывать мяч над другими элементами

  // передвинуть мяч под координаты курсора
  // и сдвинуть на половину ширины/высоты для центрирования
  function moveAt(e) {
    ball.style.left = e.pageX - ball.offsetWidth / 2 + 'px';
    ball.style.top = e.pageY - ball.offsetHeight / 2 + 'px';
  }

  // 3, перемещать по экрану
  document.onmousemove = function(e) {
    moveAt(e);
  }

  // 4. отследить окончание переноса
  ball.onmouseup = function() {
    document.onmousemove = null;
    ball.onmouseup = null;
  }
}
```

# Задача

1. Реализовать пример с таблицей. Дополнительная подсветка осуществляется с зажатым Ctrl

2. Реализовать пример с перетаскиванием. Доработать недостатки.