UKRAINIAN
IT_SCHOOL

JavaScript Essential

```
if (!Function.prototype.bind) {
// Function.prototype.bind
Function.prototype.bind = function bind(scope) {
        var
        callback = this,
        prepend = Array.prototype.slice.call(arguments, 1),
        Constructor = function () {},
        bound = function () {
        return callback.apply(
                this instanceof Constructor && scope ? this : scope,
                prepend.concat(Array.prototype.slice.call(arguments, 0))
        );
        };
        Constructor.prototype = bound.prototype = callback.prototype;
        return bound;
}
```

JavaScript Essential

События

AddEventListener

Работа с координатами

addEventListener

Фундаментальный недостаток DOM level 0 способов назначения обработчика — невозможность повесить несколько обработчиков на одно событие.

Например, одна часть кода хочет при клике на кнопку делать ее подсвеченной, а другая — выдавать сообщение. Для этого нужно 2 разных обработчика.

При этом новый обработчик будет затирать предыдущий. Например, следующий код на самом деле назначает один обработчик — последний:

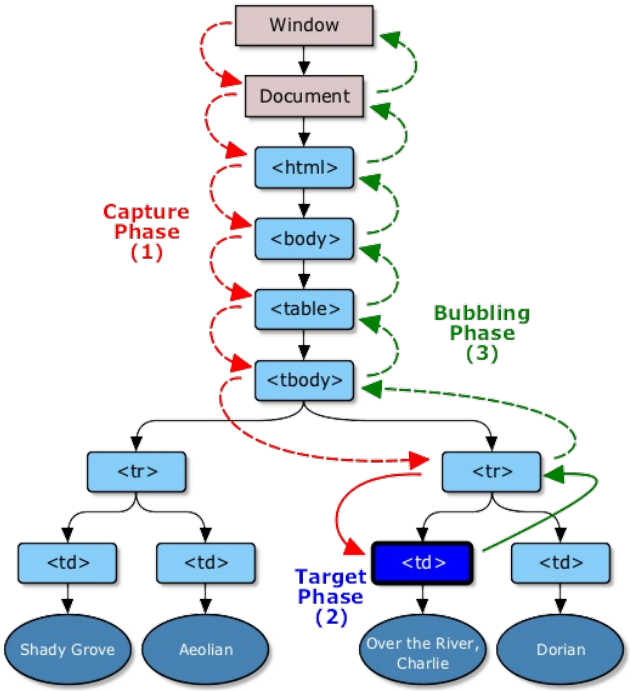
```
input.onclick = function() { alert(1); }
input.onclick = function() { alert(2); } // заменит предыдущий обработчик
```

addEventListener

Элементы DOM могут быть вложены друг в друга. При этом обработчик, привязанный к родителю, срабатывает, даже если посетитель кликнул по потомку.

Это происходит потому, что событие всплывает.

Визуализация событий



Действия, которые нельзя отменить

Есть действия браузера, которые происходят до вызова обработчика.
Такие действия нельзя отменить.

Например, при клике по ссылке происходит фокусировка.
Большинство браузеров выделяют такую ссылку пунктирной границей.
Можно и указать свой стиль в CSS для псевдоселектора :focus.

Фокусировку нельзя предотвратить из обработчика onfocus, поскольку обработчик вызывается уже после того, как она произошла.

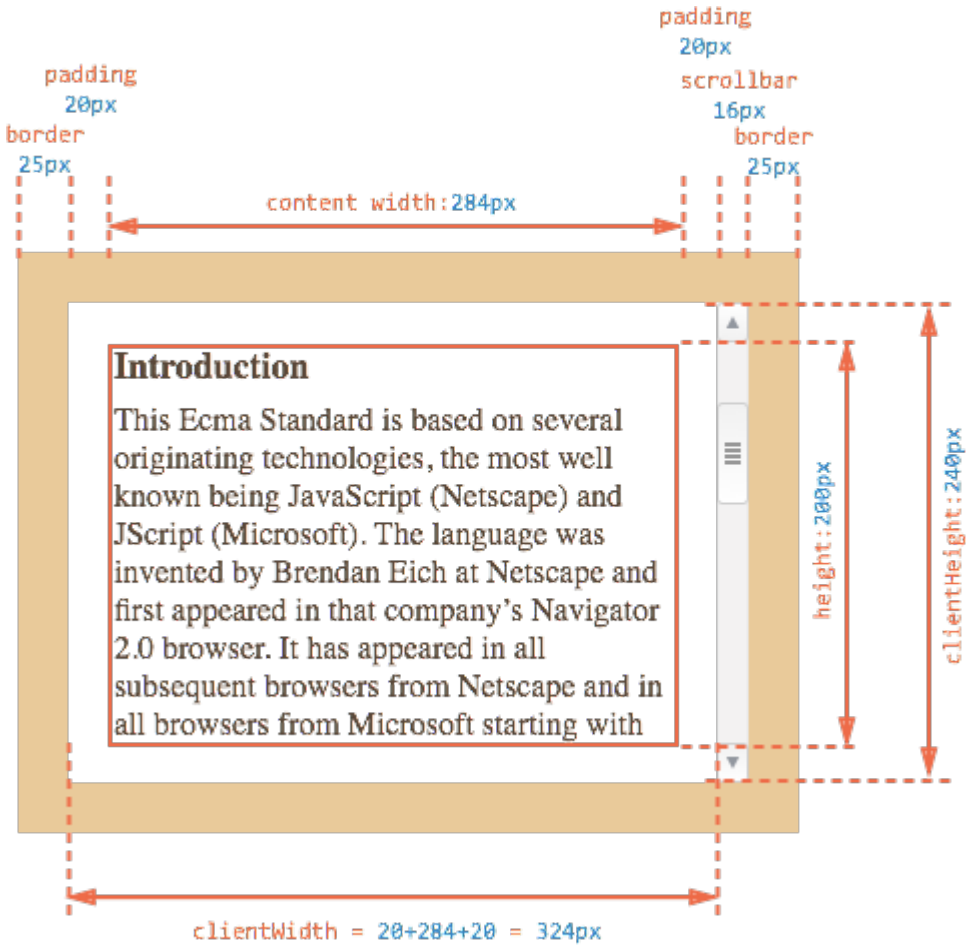
onDOMContentLoaded

Событие onDOMContentLoaded срабатывает при загрузке документа,
после выполнения всех тегов SCRIPT.

В отличие от window.onload, оно не ждёт загрузки дополнительных ресурсов,
поэтому наступает гораздо раньше.

На момент наступления onDOMContentLoaded, дерево DOM полностью построено
и все элементы доступны. Поэтому это событие часто используют для инициализации
интерфейса. Во фреймворках для него отводят специальные функции: $(ready),
Ext.onReady и т.п.

Client Height/Width



HOMEWORK

Задания находятся в папках уроков