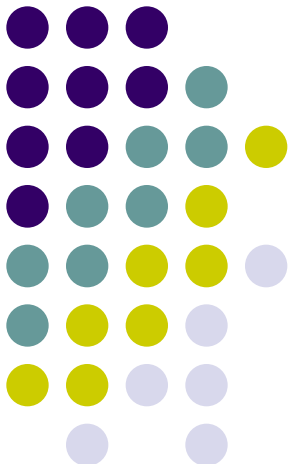
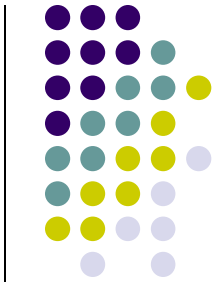
Sergey Miroshnyk
smiroshnick@gmail.com
0677223233

CPU + Память

Responsive интерфейсы



Оптимизация интерфейсов

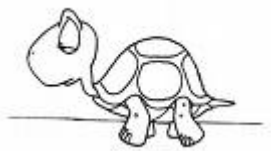


- Браузер работает медленно
  - уменьшаем количество операций
    - throttle onscroll/onmousemove
  - упрощаем операции
    - Перетаскивать иконку а не окно
  - при необходимости изменяем UI
    - при перетаскивании перемещать не объект, а иконку
    - не всю ветку дерева, а верхний узел

- Классические принципы оптимизации
  - оптимизация только узких мест
  - не вредить архитектуре
  - профилируй то, что получилось

Проще операции
CSS-селекторы

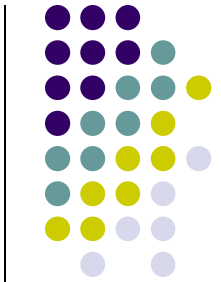
- Справа-налево
  - .menu div { }
  - *#my span { }*
  - ul li a {...}



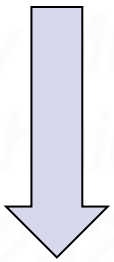
ключевой селектор: очень медленно.

- Библиотека тоже ищет справа-налево
  - $(‘.me .user .email’)

Оптимизация циклов
Меньше обращений к length



```
for(var i=0; i<data.length; i++) {
    work(data[i])
}
```




```
for(var i=data.length;i--;) {
    work(data[i])
}
```

только одно обращение к length

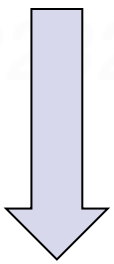
Оптимизация циклов
Почему?

Оно живое!

И привязано к документу…

```
var nodes = document.getElementsByTagName("...")

for(var i=0; i<nodes.length; i++) {
    work(nodes[i])
}
```

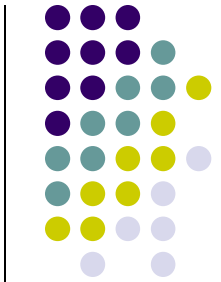


```
for(var i=nodes.length;i--;) {
    work(nodes[i])
}
```

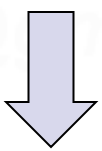
Бенчмарк

Оптимизация циклов

Выносим лишнее



```
for(var i=0; i<elements.length; i++) {
    elements[i].onclick = function() {
        alert("click")
    }
    elements[i].number = i
}
```

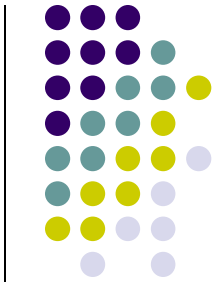



```
var handler = function() { alert("click“) }

for(var i=elements.length; i--;) {
    var elem = elements[i]
    elem.onclick = handler
    elem.number = i
}
```

- Бенчмарк

Миф «обращение к [[Prototype]] - долго»



```
function A() {
    this.method = {
        ....
    }
}
```

==

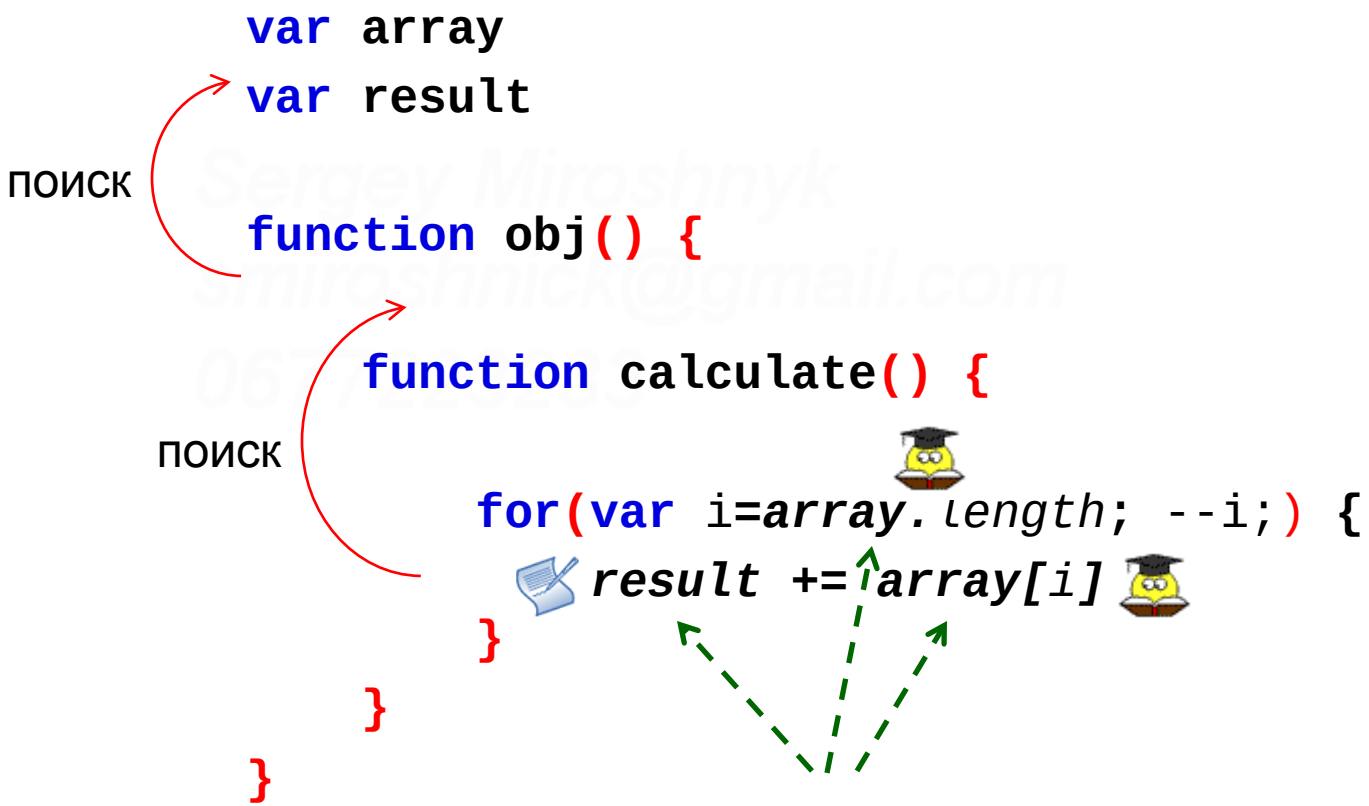
```
function A() { }
A.prototype.method = {
    ...
}
```


Первый доступ кеширует местонахождение свойства

... кроме IE

Правда «обращение к [[Scope]] - долго»

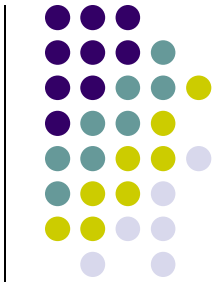
- Получение из scope – тормозит?
  - http://nczonline.net/experiments/javascript/performance/identifier-depth/



Полного кеширования нет,

но разные браузеры стараются оптимизировать доступ

Ветвление на уровне функции

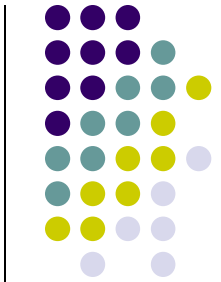


```
elem.onmousemove = function() {
    if (navigator.userAgent.test(/MSIE/))
        ...
    } else {
        ...
    }
}
```



```
if (navigator.userAgent.test(/MSIE/) {
    elem.onmousemove =  function() {
        ...
    }
} else {
    elem.onmousemove = function() {
        ...
    }
}
```

Мемоизация

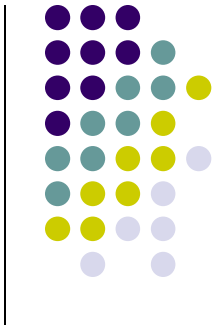


- Замена функции на результат
  - Способ прозрачного кеширования

```
var func = function(a,b) {
    var res = heavyCalcOnce()
    func = function(a,b) {
        … use res …
        return …
    }
    return func(a,b)
}
…
func()
```

  - + переопределение в зависимости от условия

documentFragment



- Одна сущность
  - много узлов
  - поддерживает cloneNode
  - ограничен функционал вне DOM (получить ширину)

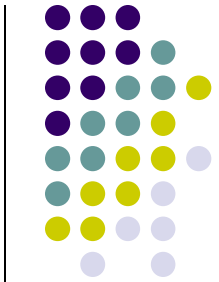
```
function makeHtml(data) { // вернуть много узлов как один
    var fragment = document.createDocumentFragment()
    for (var i = 0; i<data.length; i++) {
        var td = document.createElement('td')
        td.innerHTML = data[i]
        fragment.appendChild(td)
    }
    return fragment
}
...
// documentFragment исчезнет
tr.appendChild(makeHtml())
```

Чем больше элементов в DOM тем дороже работа с живым DOM

Создание элементов в отрыве от DOM и 1 раз appendChild

insertAdjacentHTML

События: используйте всплытие



- Один обработчик на много узлов
  - Дерево
  - Drag'n'drop
  - Таблица

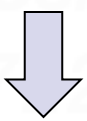
  В одном контейнере много узлов со схожим функционалом
- Меньше памяти
  - один обработчик
- Проще скрипт
  - Не нужно добавлять и удалять обработчики
  - Не нужно беспокоиться об обработчиках при изменениях DOM

Используйте всплытие

Пример: таблица

Много кликабельных элементов

onclick	onclick	onclick	onclick	onclick
onclick	onclick			
		onclick		
			onclick	onclick

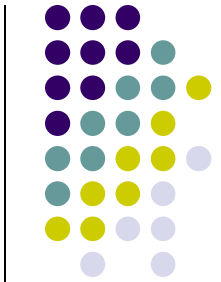


onclick

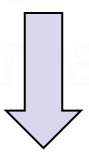
- Найти нужный узел

```
function handler(event) {
  var elem = event.target
  while (elem) {
    if (elem.tagName == 'TH') {
      grid.sort(elem)
      return
    }
    if (elem.tagName == “TD") {
      grid.edit(elem)
      return
    }
    elem = elem.parentNode
  }
}
```

Меньше операций с innerHTML



```
function buildUI()  {
    var baseElement = document.getElementById("target")
    baseElement.innerHTML = ""
    baseElement.innerHTML += buildTitle()
    baseElement.innerHTML += buildBody()
    baseElement.innerHTML += buildFooter()
}
```

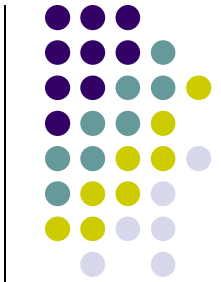



```
function buildUI() {
    var elementText = buildTitle() + buildBody() + buildFooter()
    document.getElementById("target").innerHTML = elementText
}
```

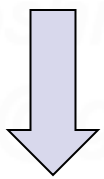
- Тест скорости
  http://javascript.ru/optimize/javascript-performance#obrashcheniya-k-dom

Оптимизация создания узлов

Строки → Array.join IE<8



```
var str = "<table><tr>"
for(var i=0; i<data.length; i++) {
   str += "<td>" + data[i] + "</td>"
}
str += "</tr></table>"
elem.innerHTML = str
```

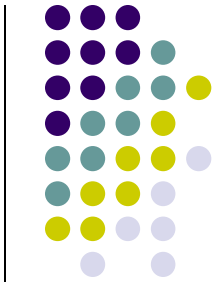



```
var buffer = ["<table><tr>"]
for(var i=0; i<data.length; i++) {
   buffer.push("<td>" + data[i] + "</td>")
}
buffer.push("</tr></table>")
elem.innerHTML = buffer.join("")
```

- Тест скорости

  http://javascript.ru/optimize/javascript-performance#array-join-vmesto-slozheniya-strok

Оптимизация создания узлов
Мини-шаблонка



HTML

```
<script type="text/html" id="user-list-tmpl">
  <% for ( var i = 0; i < users.length; i++ ) { %>
    <li><a href="<%=users[i].url%>"><%=users[i].name%></a></li>
  <% } %>
</script>
```

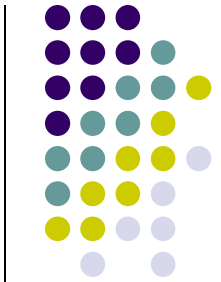
JS

```
data = { /* AJAX -> JSON */
    users : [
        { id: 123, name: "Vasya", url:"http://vasyapage.ru"},
        ...
    ]
}
var userList = document.getElementById("user-list")
usersList.innerHTML = tmpl("user-list-tmpl", data)
```

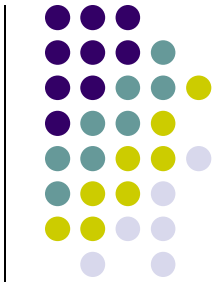
Подробнее

http://javascript.ru/unsorted/templating

RegExp

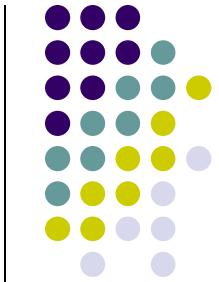


- Скорость регулярных выражений
  - http://javascript.ru/perf/trim/trim.html



Mobile

10. Прелоадинг
    - <script> /* … */ </script>
    - <!-- … -->
    - CSS-анимации вместо обычных



Структура кода

- http://www.amazon.com/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882
- Стив Макконел «Совершенный код»