

secrets of javascript closures

fronteers, september 2008

stuart langridge



what's a closure?



one of the great mysteries



confusion





zen

In computer science, a closure is a function that is evaluated in an environment containing one or more bound variables. When called, the function can access these variables.



In computer science, a closure is a function that is evaluated in an environment containing one or more bound variables. When called, the function has access to these variables.

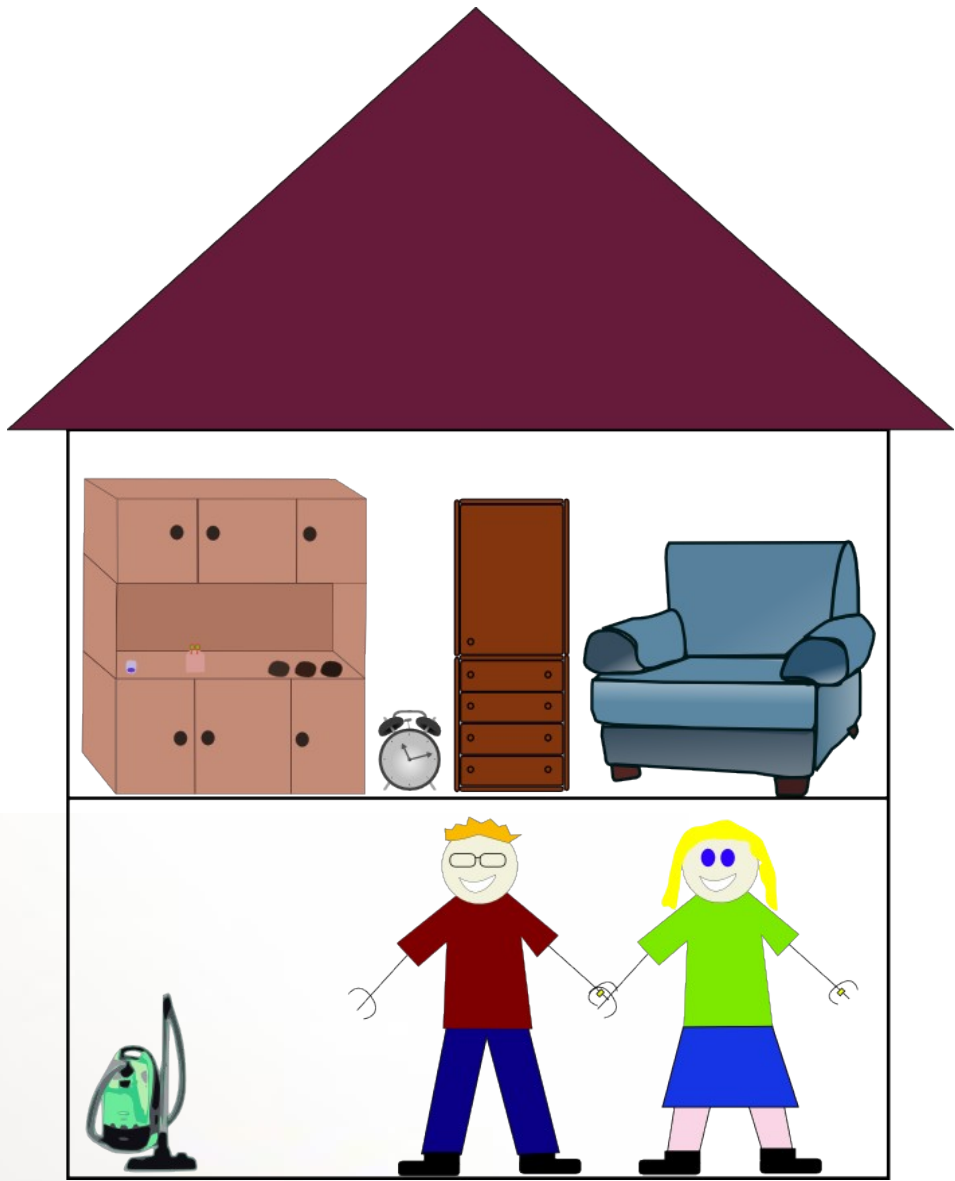


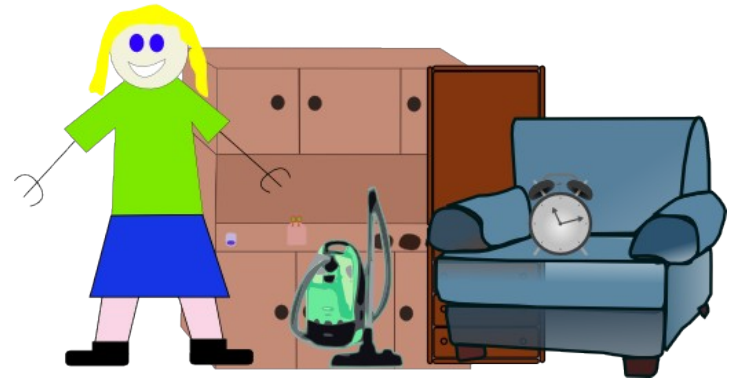
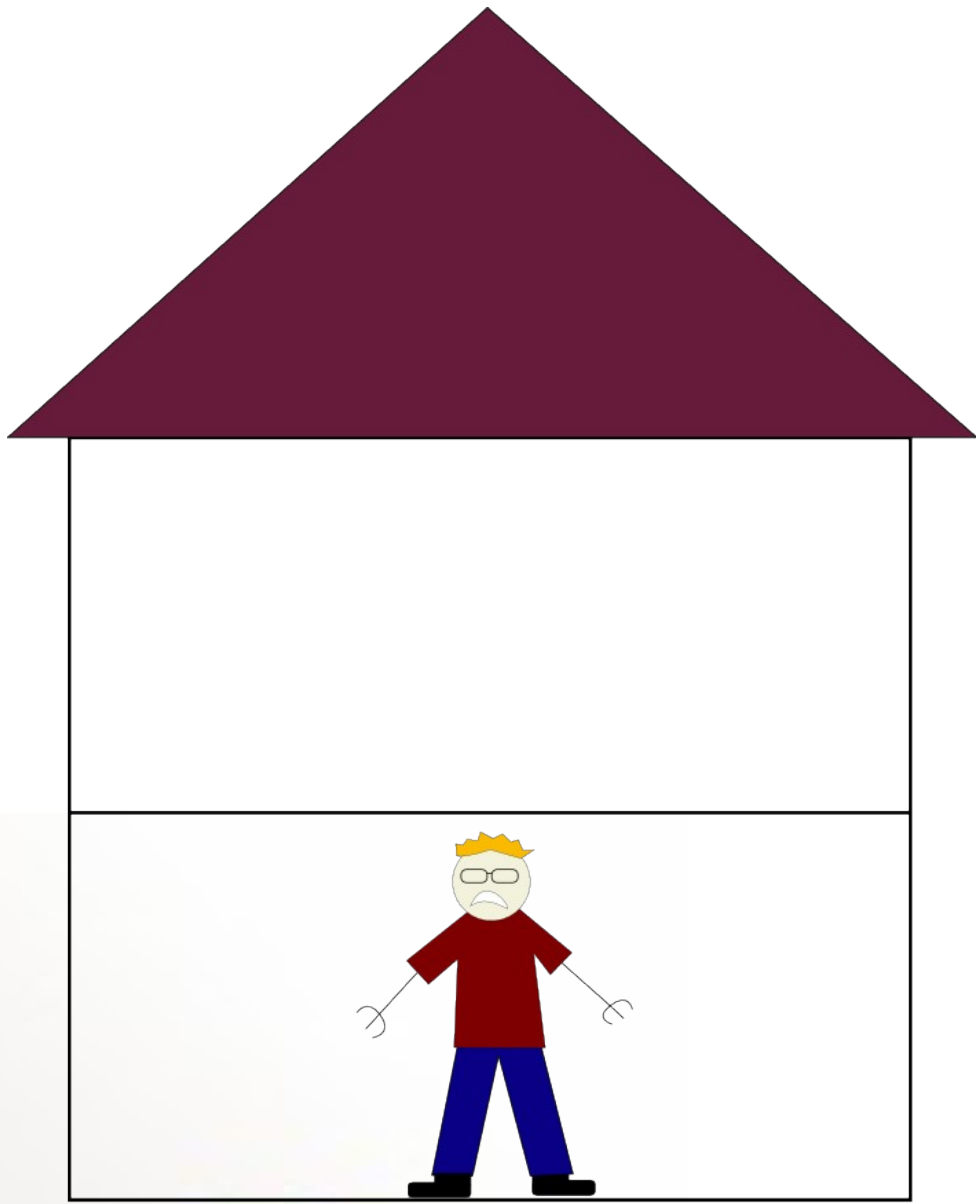
~~dictionary~~



closure:
where a function
remembers what
happens around it







one function defined inside another



```
function outer() {  
  ...  
  function inner() {  
    ...  
  }  
  ...  
}
```



```
function outer() {  
  var x = 5;  
  function inner() {  
    alert(x);  
  }  
  inner();  
}
```



```
function outer() {  
  var x = 5;  
  function inner() {  
    alert(x);  
  }  
  setTimeout(inner,  
5000);  
}
```



power



things called later on



setTimeout
setInterval
Ajax callbacks
event handlers



```
function main() {  
  var mv = document.getElementById("mover");  
  var counter = 0;  
  var tick = setInterval(function() {  
    mv.style.top = counter + "px";  
    counter += 1;  
    if (counter > 100) {  
      clearInterval(tick);  
    }  
  }, 100);  
}
```

```
main();
```



```
function main(mv) {  
  var counter = 0;  
  var tick = setInterval(function() {  
    mv.style.top = counter + "px";  
    counter += 1;  
    if (counter > 100) {  
      clearInterval(tick);  
    }  
  }, 100);  
}
```

```
main(document.getElementById("mv"));
```



```
function main(mv, start, finish, inc) {  
  var tick = setInterval(function() {  
    mv.style.top = start + "px";  
    start += inc;  
    if (start > finish) {  
      clearInterval(tick);  
    }  
  }, 100);  
}
```

```
main(document.getElementById("mv"), 0, 100, 1);
```



this



that



this and that



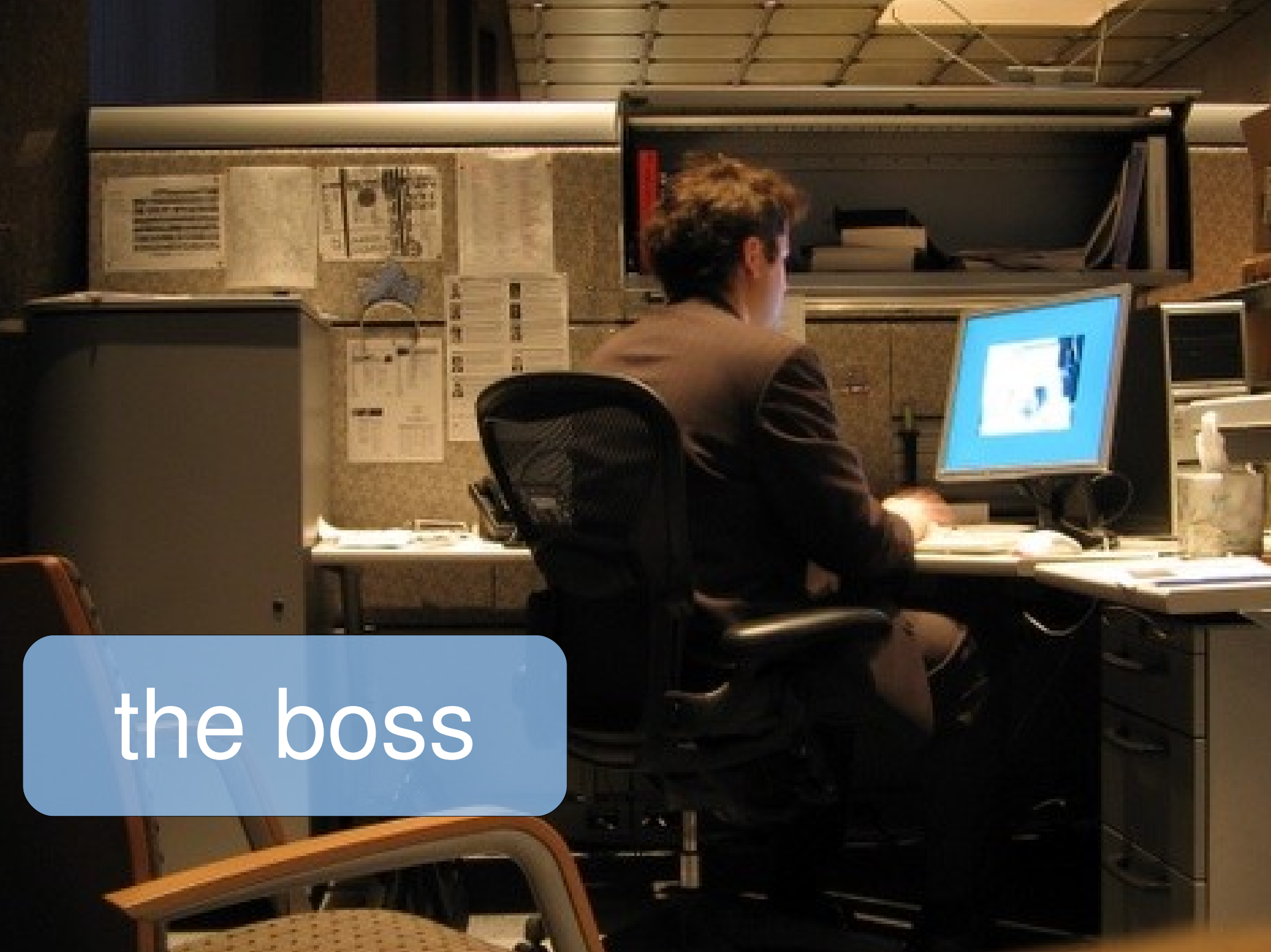
```
function main(link) {
  link.onclick = function(e) {
    var newa = document.createElement("a");
    var tn = document.createTextNode("second");
    newa.appendChild(tn);
    newa.href = "#";
    this.firstChild.nodeValue = "clicked";
    var that = this;
    document.body.appendChild(newa);
    newa.onclick = function(e) {
      that.firstChild.nodeValue = "reset";
      this.parentNode.removeChild(this);
    }
  }
}
```



object oriented

```
s.module = mod;
s.adc = adc;
s.helix = helix;
id.s.channel = chan;
// printf("<%d %d %d %d> = %d\n",
return bsearch(&id, p->channels, p->num_channels,
sizeof(struct hadc_check_channel),
&hadc_check_channel_compare);
int check(const int16_t *buf,
```





the boss

public
private
privileged



```
function Container(param) {
  function dec() {
    if (secret > 0) {
      secret -= 1;
      return true;
    } else { return false; }
  }
  var secret = 3;
  var that = this;
  this.service = function () {
    if (dec()) {
      return param;
    } else { return null; }
  };
}
```

}



```
function Container(param) {  
  function dec() {  
    if (secret > 0) {  
      secret -= 1;  
      return true;  
    } else { return false; }  
  }  
  var secret = 3;  
  var that = this;  
  this.service = function () {  
    if (dec()) {  
      return param;  
    } else { return null; }  
  };  
}
```

private

}



```
function Container(param) {
```

```
  function dec() {  
    if (secret > 0) {  
      secret -= 1;  
      return true;  
    } else { return false; }  
  }
```

private

```
  var secret = 3;  
  var that = this;
```

```
  this.service = function () {  
    if (dec()) {  
      return param;  
    } else { return null; }  
  };
```

privileged

```
}
```



```
function Container(param) {
```

```
  function dec() {  
    if (secret > 0) {  
      secret -= 1;  
      return true;  
    } else { return false; }  
  }
```

private

```
  this.member = param;
```

```
  var secret = 3;
```

```
  var that = this;
```

```
  this.service = function () {  
    if (dec()) {  
      return that.member;  
    } else { return null; }  
  }
```

privileged

```
};
```



```
var c = new Container("value");  
console.log(c.service());      →  
"value"  
console.log(c.service());      →  
"value"  
console.log(c.service());      →  
"value"  
console.log(c.service());      → null
```



revealing module pattern

(Christian Heilmann)



```
helpers = function() {
  function reg(c){
    return new RegExp( '\\s|^)+' + c + '\\s|$) ');
  };
  function hasClass(el,c){
    return el.className.match(reg(c));};
  function addClass(el,c){
    if (!hasClass(el,c)) el.className += " " + c;
  };
  function removeClass(el,c) {
    if (hasClass(el,c)) {
      el.className=el.className.replace(reg(c), ' ');
    }
  };
  return { addClass: addClass,
    removeClass: removeClass, hasClass: hasClass }
}();
```



```
helpers = function() {  
  function reg(c){  
    return new RegExp( '\\s|^)+' + c + '(\\s|$)' );  
  };  
  function hasClass(e1,c){  
    return e1.className.match( reg(c) );}};  
  function addClass(e1,c){  
    if (!hasClass(e1,c)) e1.className += " " + c;  
  };  
  function removeClass(e1,c) {  
    if (hasClass(e1,c)) {  
      e1.className=e1.className.replace( reg(c), ' ' );  
    }  
  };  
  return { addClass: addClass,  
    removeClass: removeClass, hasClass: hasClass }  
}();
```





Don't use closures unless you really need closure semantics.

In most cases, non-nested functions are the right way to go.

Eric Lippert, Microsoft





May not entirely
be the truth



```
function AttachEvent(elem) {  
    elem.attachEvent("mouseover",  
        function() {  
            alert(this);  
        });  
}  
AttachEvent(someElement);
```



```
function AttachEvent(elem) {  
  elem.attachEvent ("mouseover",  
    function() {  
      alert(this);  
    });  
}  
AttachEvent(som
```

elem has a reference
to the handler



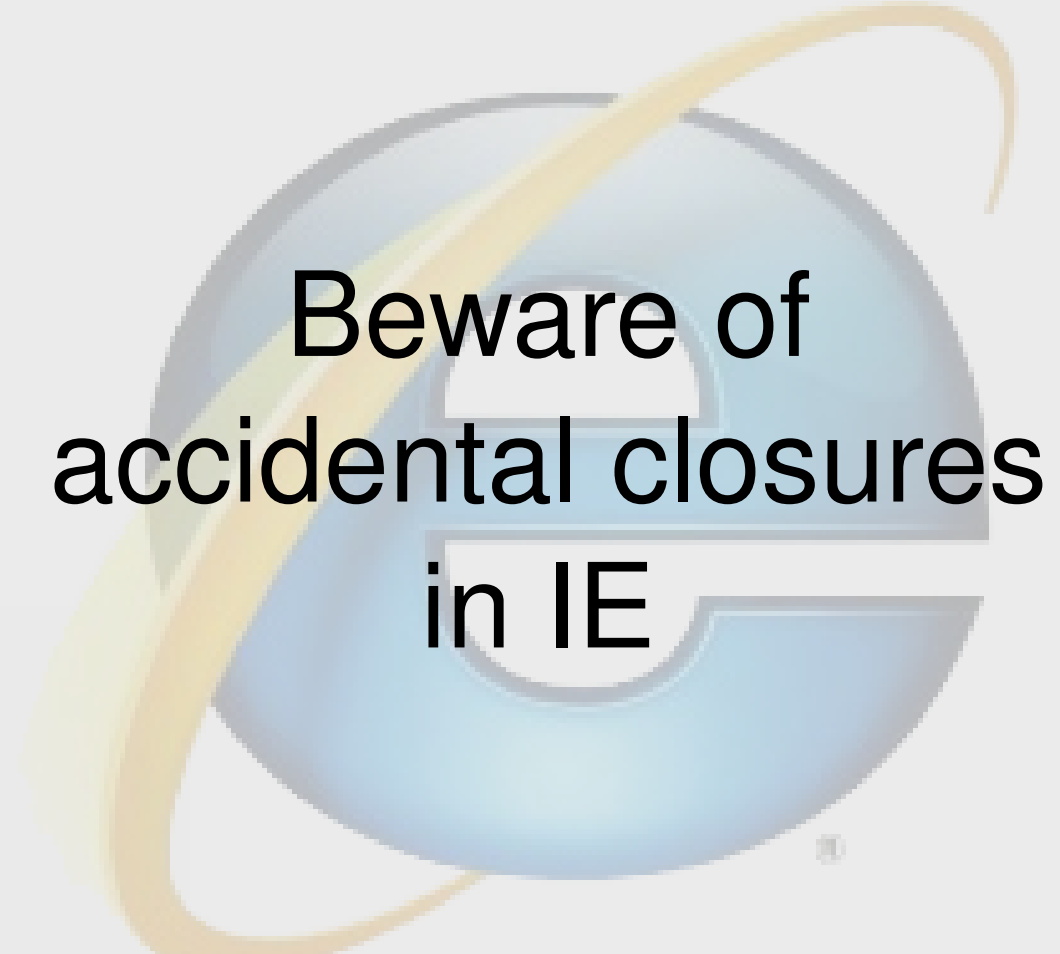
```
function AttachEvent(elem) {  
    elem.attachEvent("mouseover",  
        function() {  
            alert(this);  
        });  
}
```

AttachEvent(some

handler has a reference
to the element
(in the closure)





The background of the slide features the Internet Explorer logo, which consists of a blue globe with a yellow ring orbiting it. The text is centered over this logo.

Beware of accidental closures in IE



loops



```
function main(links) {
  for (var i=0; i<links.length; i++) {
    links[i].onclick = function() {
      alert(i+1);
    }
  }
};
```

```
main(document.getElementsByTagName("a"));
```



```
function main(links) {  
  for (var i=0; i<links.length; i++) {  
    links[i].onclick = function() {  
      alert(i+1);  
    }  
  }  
};
```

```
main(document.getElementsByTagName("a"));
```



FAIL

alerts 6, 6, 6, 6, 6



```
function main(links) {
  for (var i=0; i<links.length; i++) {
    links[i].onclick = function( ) {

      alert(i+1);

    }
  }
};
```

```
main(document.getElementsByTagName("a"));
```



```
function main(links) {
  for (var i=0; i<links.length; i++) {
    links[i].onclick = (function(i) {
      return function() {
        alert(i+1);
      }
    })(i);
  }
};
```

```
main(document.getElementsByTagName("a"));
```



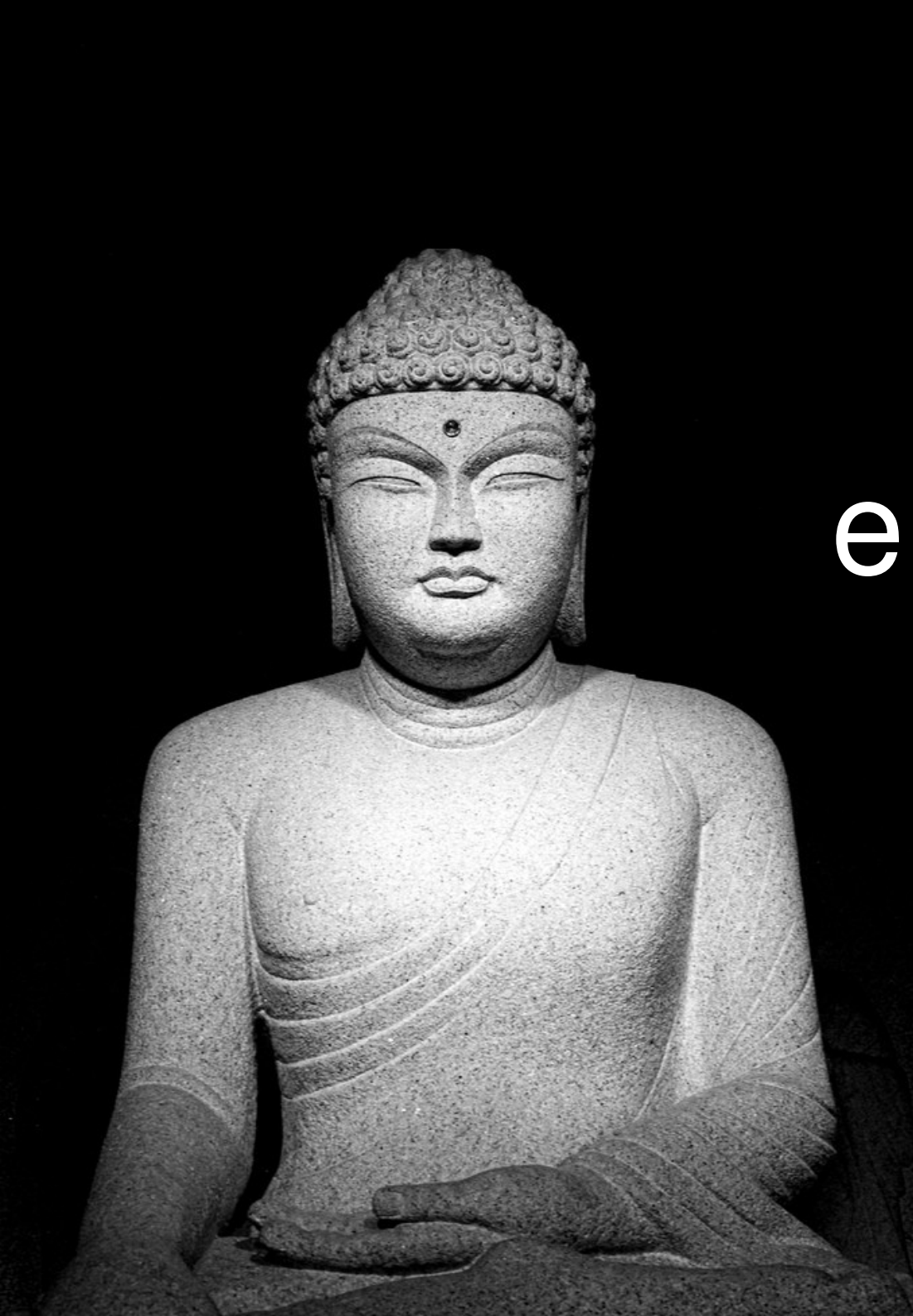
```
function main(links) {
  for (var i=0; i<links.length; i++) {
    links[i].onclick = (function(j) {
      return function() {
        alert(j+1);
      }
    })(i);
  }
};
```

```
main(document.getElementsByTagName("a"));
```



power





enlightenment

secrets of javascript closures
stuart langridge
fronteers, september 2008



fin.

<http://tinyurl.com/jsclosures>

Thanks to
carbonnyc, parhessiastes, judgmentalist, perreira, philip9876,
doug crockford, john resig, chris heilmann

