CSC309: Introduction to Web Programming

Lecture 5

Wael Aboulsaadat

The Misconceived Web

- The original vision of the WWW was as a hyperlinked document-retrieval system.
- It did not anticipate presentation, session, or interactivity.

How We Got Here

- Rule Breaking
- Corporate Warfare
- Extreme Time Pressure

The Miracle

- It works!
- Java didn't.
- Nor did a lot of other stuff.

The Scripted Browser

- Introduced in Netscape Navigator 2 (1995)
- Eclipsed by Java Applets
- Later Became the Frontline of the Browser War
- Dynamic HTML
- Document Object Model (DOM)

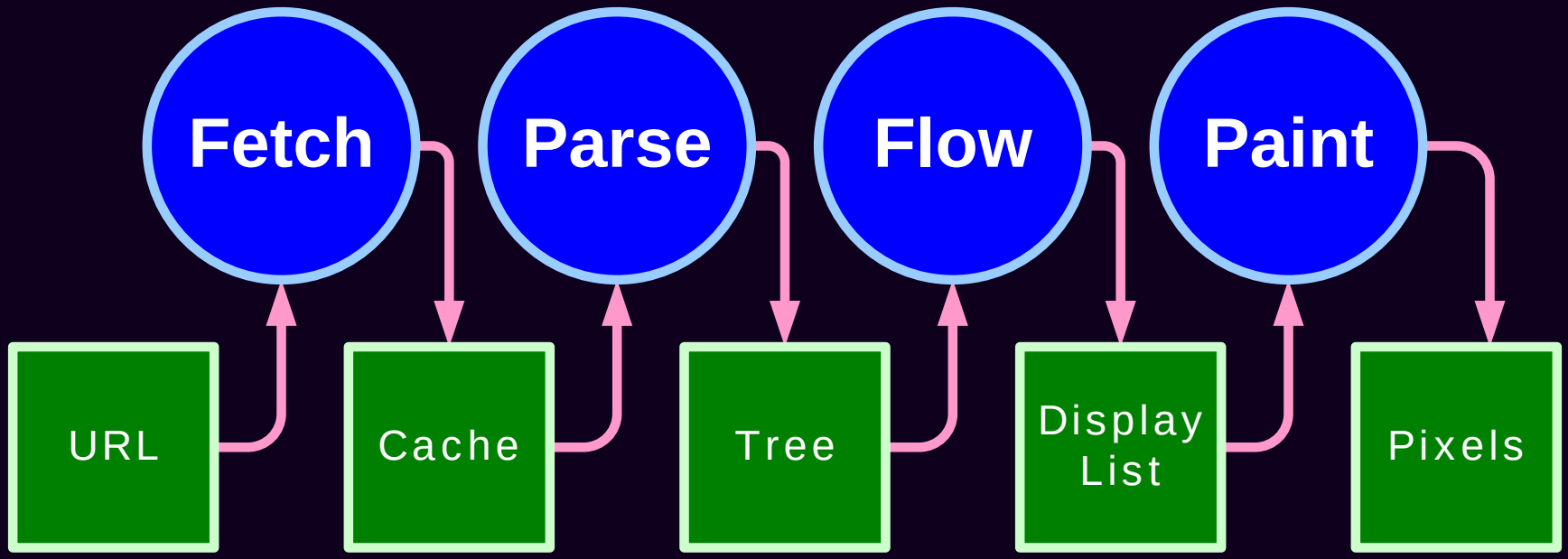
Proprietary Traps

- Netscape and LiveWire
- Microsoft and Internet Information Services
- Both server strategies frustrated by Apache
- Browser-dependent sites

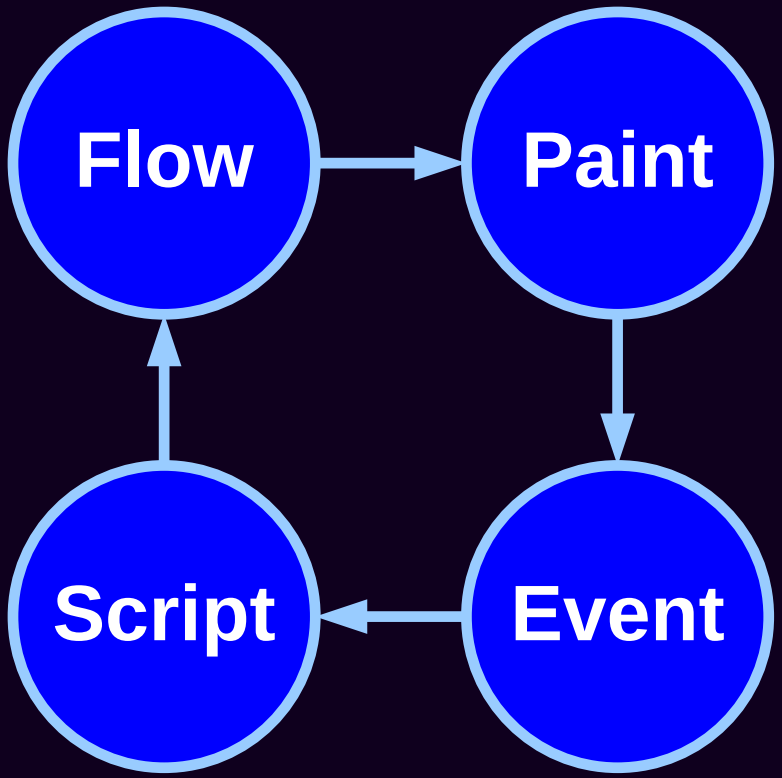
Pax Microsoft

- In the years since the end of the Browser War, the number of browser variations in significant use fell off significantly.
- W3C attempts to unify.
- Mozilla abandoned the Netscape **<layer>** model in favor of the W3C model.
- The browser platform becomes somewhat stable.
- HTML + CSS + JavaScript becomes known as DHTML
- Since 2004, DHTML becomes known as Ajax.

Browser



Scripted Browser



The A List

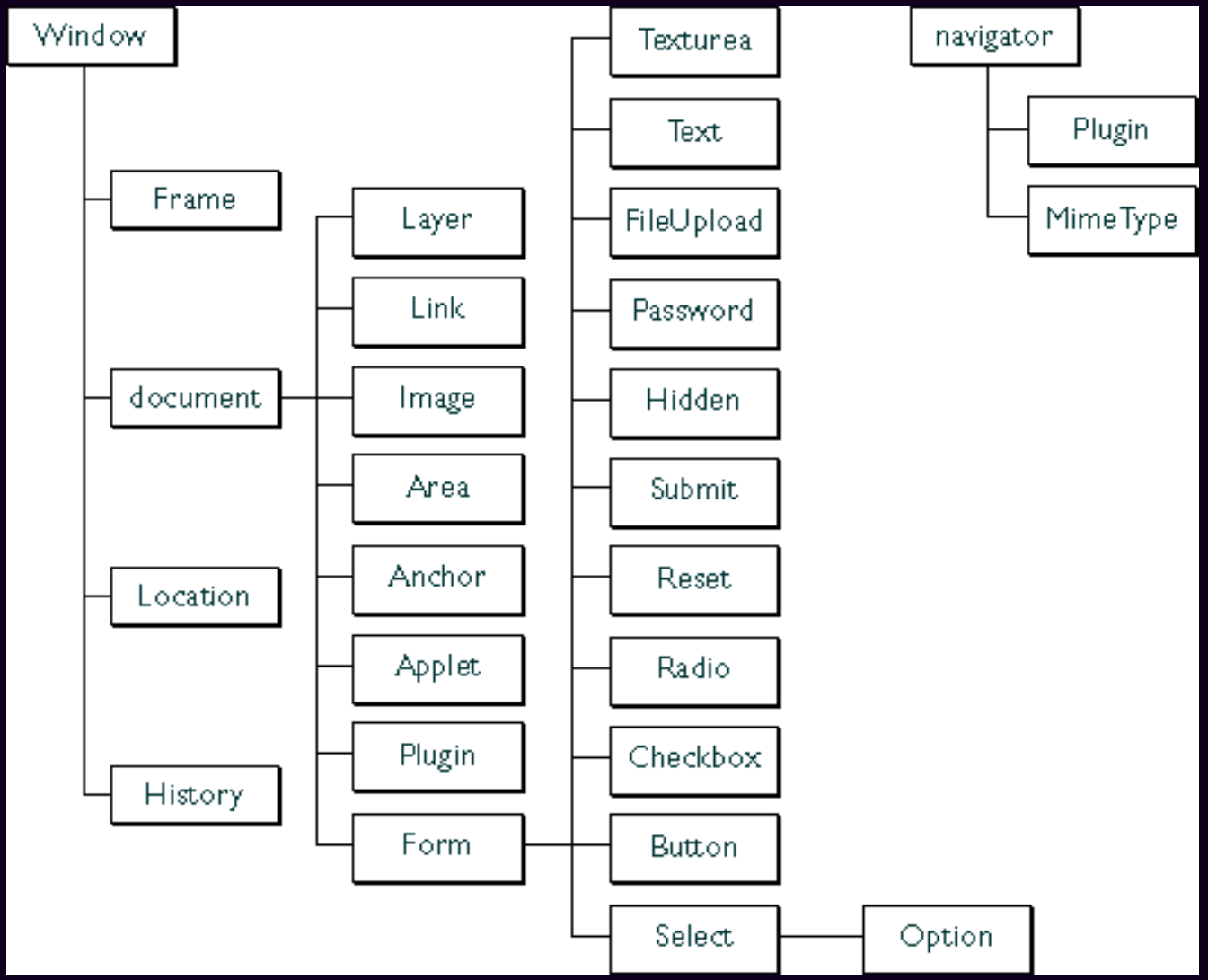
- Firefox 1.5
- Firefox 2.0
- Safari 2
- IE 6
- IE 7
- Opera 9

http://developer.yahoo.com/yui/articles/gbs/gbs_browser-chart.html

document.write

- Allows JavaScript to produce HTML text.
- Before onload: Inserts HTML text into the document.
- After onload: Uses HTML text to replace the current document.
- Not recommended.

The DOM


Window
Frame
document
Location
History
Layer
Link
Image
Area
Anchor
Applet
Plugin
Form
Texturea
Text
FileUpload
Password
Hidden
Submit
Reset
Radio
Checkbox
Button
Select
Option
navigator
Plugin
MimeType

Retrieving Nodes

```
document.getElementById(id)

document.getElementsByName(name)

node.getElementsByTagName(tagName)
```

name v id

- **name=**

  Identifies values in form data

  Identifies a window/frame

- **id=**

  Uniquely identifies an element

- They used to be interchangeable.

`document.all`

- Microsoft feature, rejected by W3C and most other browsers.
- It acts as a function or array for accessing elements by position, **`name`**, or **`id`**.
- Avoid it.

Collections

- `document.anchors`
- `document.applets`
- `document.embeds`
- `document.forms`
- `document.frames`
- `document.images`
- `document.plugins`
- `document.scripts`
- `document.stylesheets`

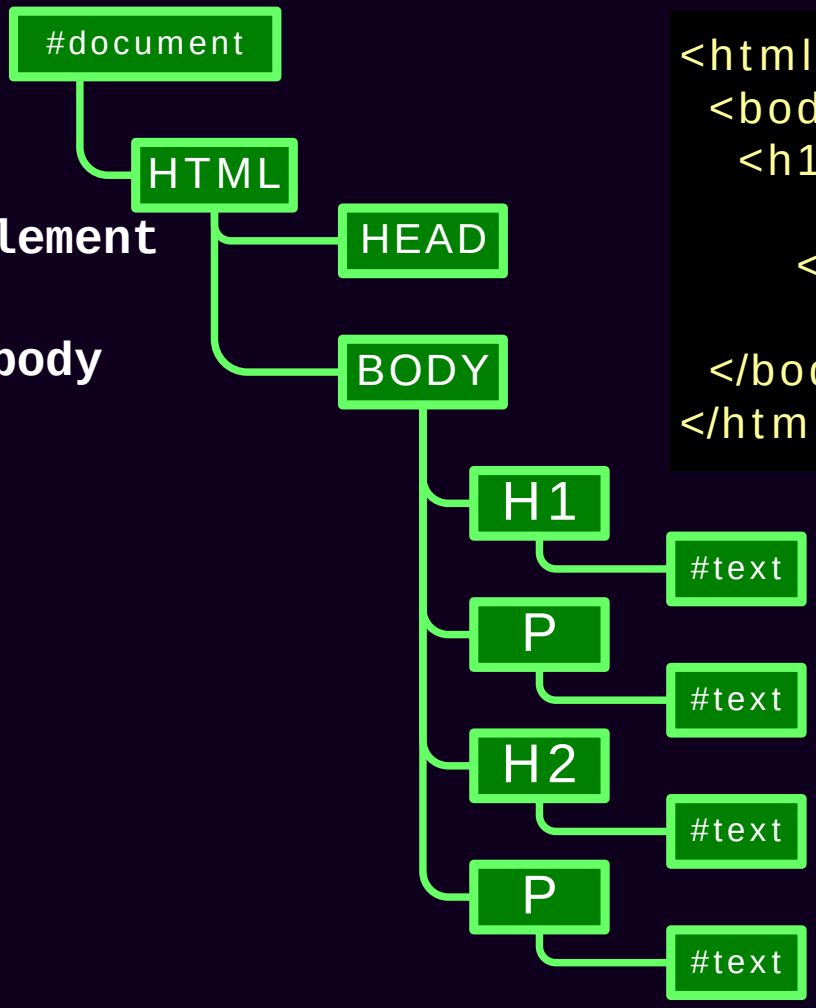
- Avoid these.

Document Tree Structure

document

document.
documentElement

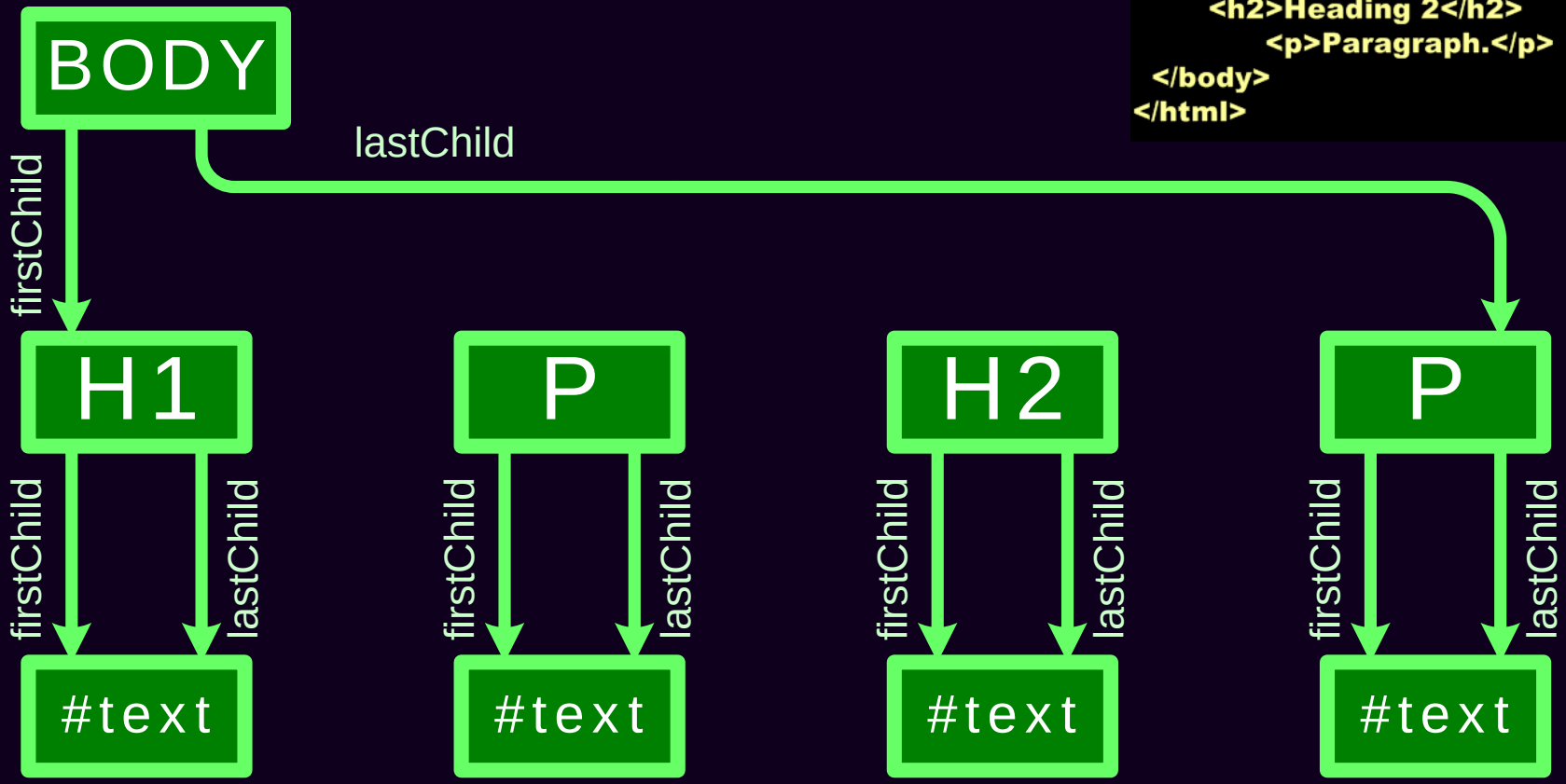
document.body




```
<html>
 <body>
  <h1>Heading 1</h1>
        <p>Paragraph.</p>
     <h2>Heading 2</h2>
        <p>Paragraph.</p>
 </body>
</html>
```

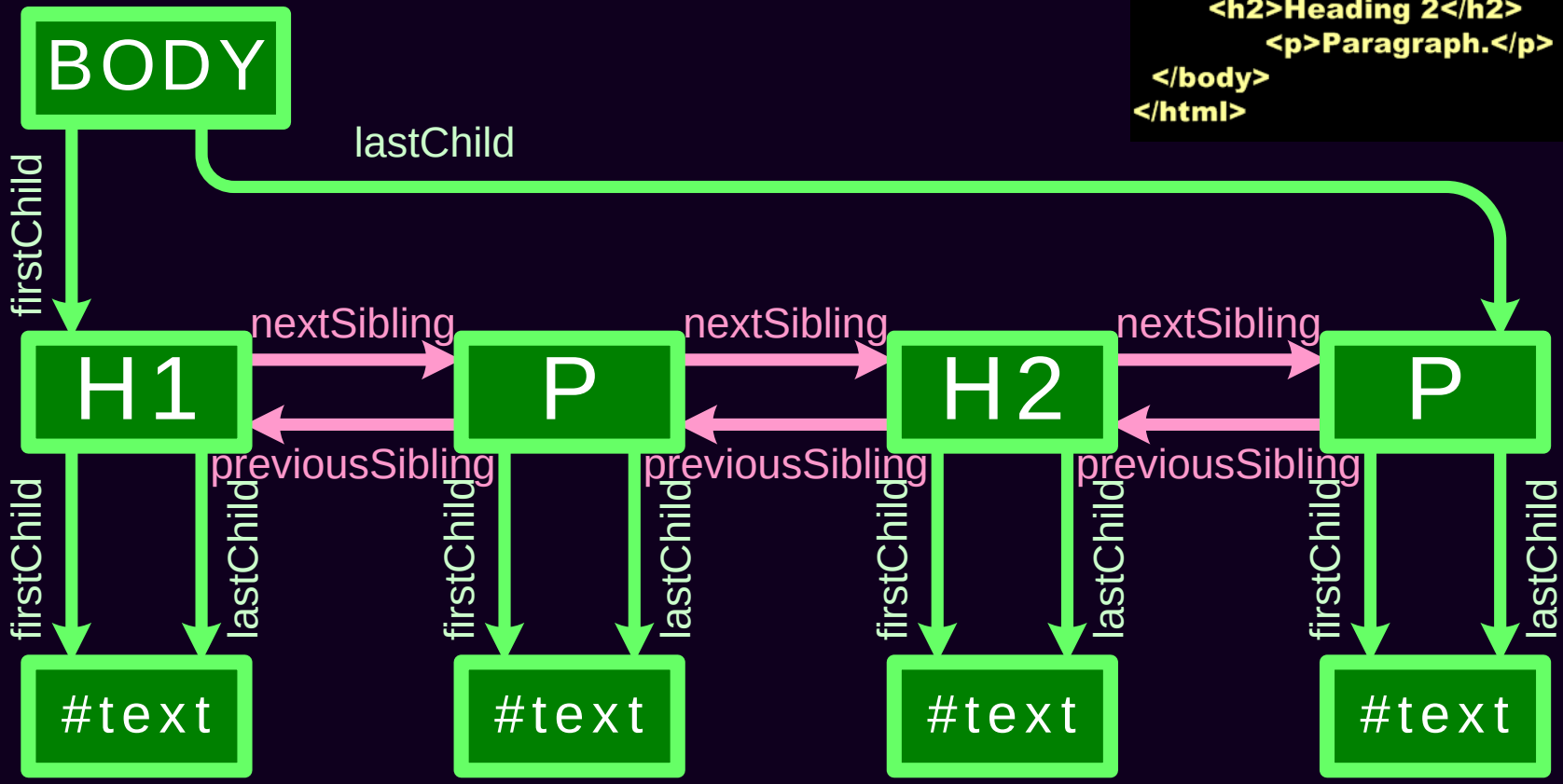
child, sibling, parent

```
<html>
  <body>
    <h1>Heading 1</h1>
        <p>Paragraph.</p>
    <h2>Heading 2</h2>
        <p>Paragraph.</p>
  </body>
</html>
```



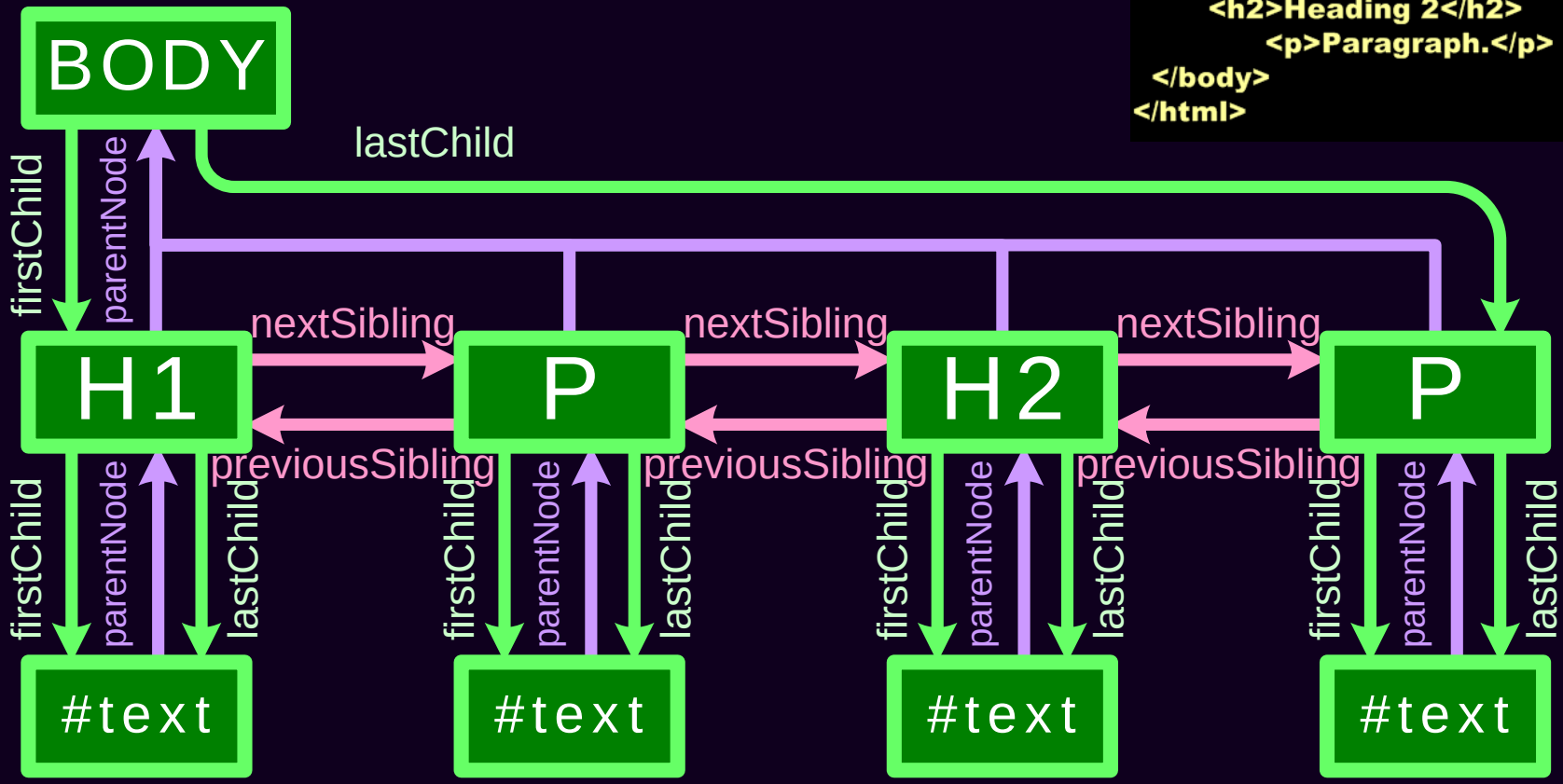
child, sibling, parent

```
<html>
  <body>
    <h1>Heading 1</h1>
        <p>Paragraph.</p>
    <h2>Heading 2</h2>
        <p>Paragraph.</p>
  </body>
</html>
```

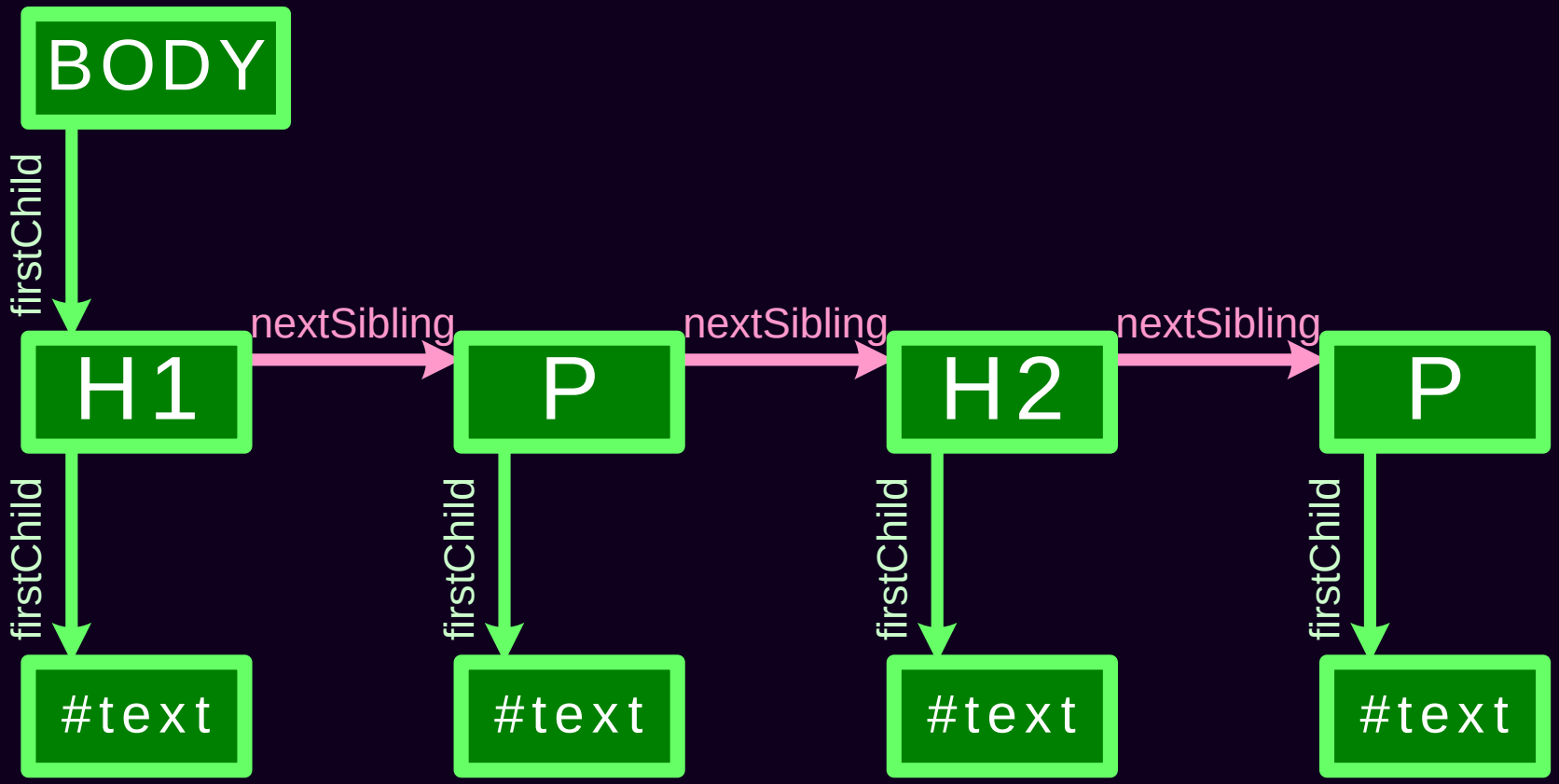


child, sibling, parent

```
<html>
  <body>
    <h1>Heading 1</h1>
        <p>Paragraph.</p>
    <h2>Heading 2</h2>
        <p>Paragraph.</p>
  </body>
</html>
```



child, sibling, parent



Walk the DOM

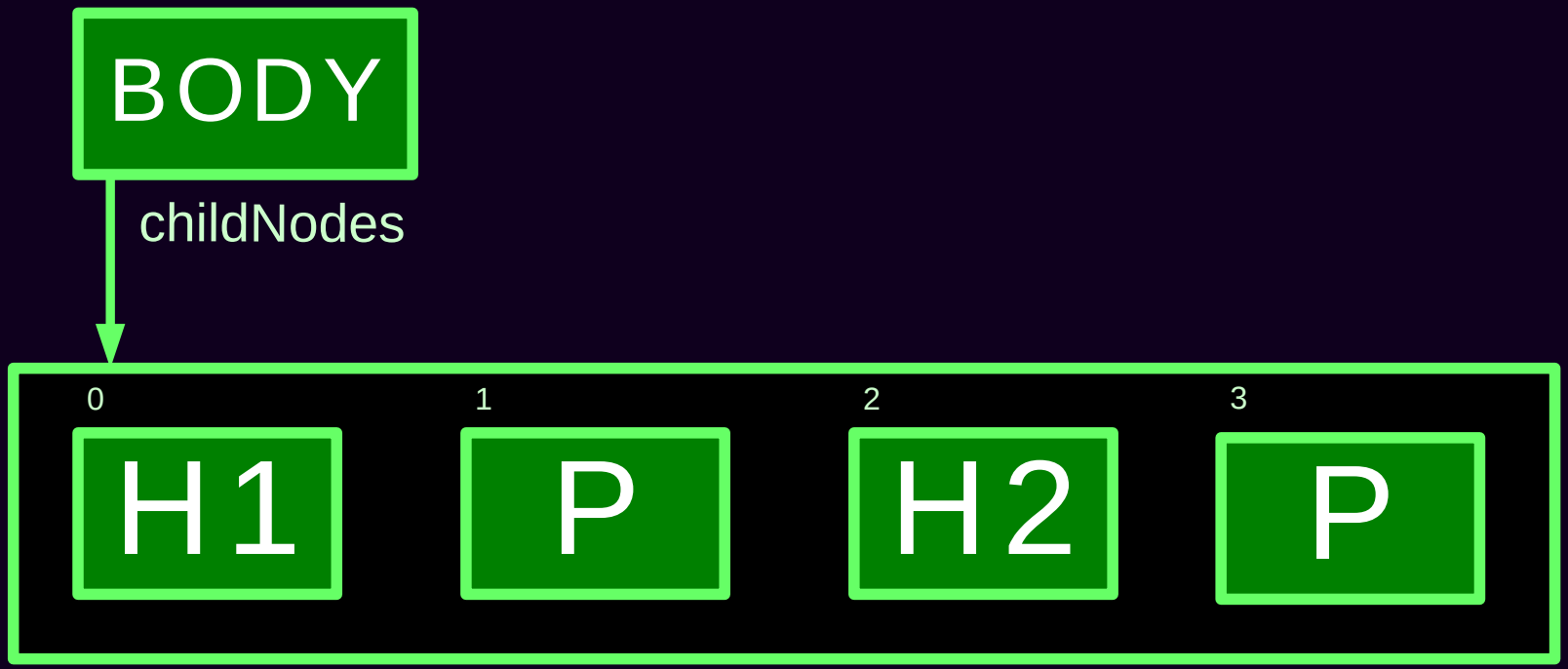
- Using recursion, follow the **firstChild** node, and then the **nextSibling** nodes.

```
function walkTheDOM(node, func) {
    func(node);
    node = node.firstChild;
    while (node) {
        walkTheDOM(node, func);
        node = node.nextSibling;
    }
}
```

getElementsByClassName

```
function getElementsByClassName(className) {
    var results = [];
    walkTheDOM(document.body, function (node) {
        var a, c = node.className, i;
        if (c) {
            a = c.split(' ');
            for (i = 0; i < a.length; i += 1) {
                if (a[i] === className) {
                    results.push(node);
                    break;
                }
            }
        }
    });
    return results;
}
```

childNodes



Manipulating Elements

IMG has these properties:

- **align** **'none'**, **'top'**, **'left'**, ...
- **alt** string
- **border** integer (pixels)
- **height** integer (pixels)
- **hspace** integer (pixels)
- **id** string
- **isMap** boolean
- **src** url
- **useMap** url
- **vspace** integer (pixels)
- **width** integer (pixels)

node.*property* **=** *expression*;

Manipulating Elements

- Old School

```
if (my_image.complete) {
    my_image.src = superurl;
}
```

- New School

```
if (my_image.getAttribute('complete')) {
    my_image.setAttribute('src', superurl);
}
```

Style

```
node.className

node.style.stylename

node.currentStyle.stylename    (Only IE)

document.defaultView().
    getComputedStyle(node, "").
    getPropertyValue(stylename);
```

Style Names

CSS	JavaScript
background-color	backgroundColor
border-radius	borderRadius
font-size	fontSize
list-style-type	listStyleType
word-spacing	wordSpacing
z-index	zIndex

Making Elements

`document.createElement(`*`tagName`*`)`

`document.createTextNode(`*`text`*`)`

`node``.cloneNode()`

Clone an individual element.

`node``.cloneNode(true)`

Clone an element and all of its descendents.

- The new nodes are not connected to the document.

Linking Elements

```
node.appendChild(new)

node.insertBefore(new, sibling)

node.replaceChild(new, old)

old.parentNode.replaceChild(new, old)
```

Removing Elements

***node*.removeChild(*old*)**

It returns the node.

Be sure to remove any event handlers.

***old.parentNode*.removeChild(*old*)**

innerHTML

- The W3C standard does not provide access to the HTML parser.
- All A browsers implement Microsoft's **`innerHTML`** property.

Command-line driven vs. event-driven

Command-line model (e.g., UNIX shell, DOS)

- **Interaction controlled by system**
- **User queried when input is needed**

Event-driven model (e.g., GUIs)

- **Interaction controlled by the user**
- **System waits for user actions and then reacts**
- **More complicated programming and architecture**

Events

User input is modeled as “events” that must be handled by the system

Examples?

Mouse

button down, button up, button clicked, entered, exited, moved, dragged

Keyboard

key down, key up, key pressed

Window

movement, resizing

Anatomy of an Event

An event encapsulates the information needed for handlers to react to the input

1. **Event type** **(mouse button down, key up, etc.)**
2. **Event target** **(component in which event occurred)**
3. **Timestamp**
4. **Modifiers** **(Ctrl, Shift, Alt, etc.)**
5. **Type-specific content**
   - Mouse: x,y coordinates, # clicks
   - Keyboard: key code

Event Handlers

Events are dispatched to components

Application developers can specify code to be executed when the event occurs (callbacks)

Built-in components will have code to handle most keyboard and mouse events

Buttons handle mouse up/down to change graphic

Text boxes update their contents on key press

Built-in components often generate new "high-level" events from combinations of low-level events

- Text boxes generate "change" events when contents changes and focus is lost
- Sliders generate "change" events when thumb is dragged

Event Loop

Input Devices → Event Queue → Event Loop

mouse up (10,20)
key down ('h')
key up ('h')
key down ('i')

```
while(!done) {
    evt = dequeue_event();
    dispatch_event(evt);
    repaint_screen();
}
```

Exists in every application

Usually handled for you by UI framework

Event Loop

Input Devices → Event Queue → Event Loop

mouse up (10,20)
key down ('h')
key up ('h')
key down ('i')

```
while(!done) {
    evt = dequeue_event();
    dispatch_event(evt);
    repaint_screen();
}
```

Blocks until an event arrives

Event Loop

Input Devices → Event Queue → Event Loop

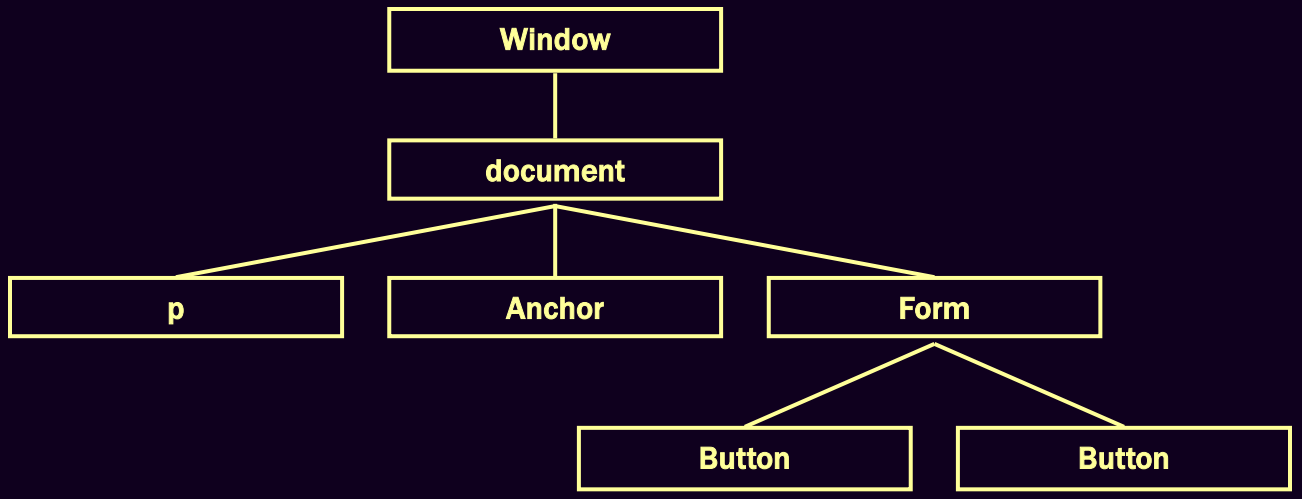
mouse up (10,20)
key down ('h')
key up ('h')
key down ('i')

```
while(!done) {
    evt = dequeue_event();
    dispatch_event(evt);
    repaint_screen();
}
```

Most of the work happens here

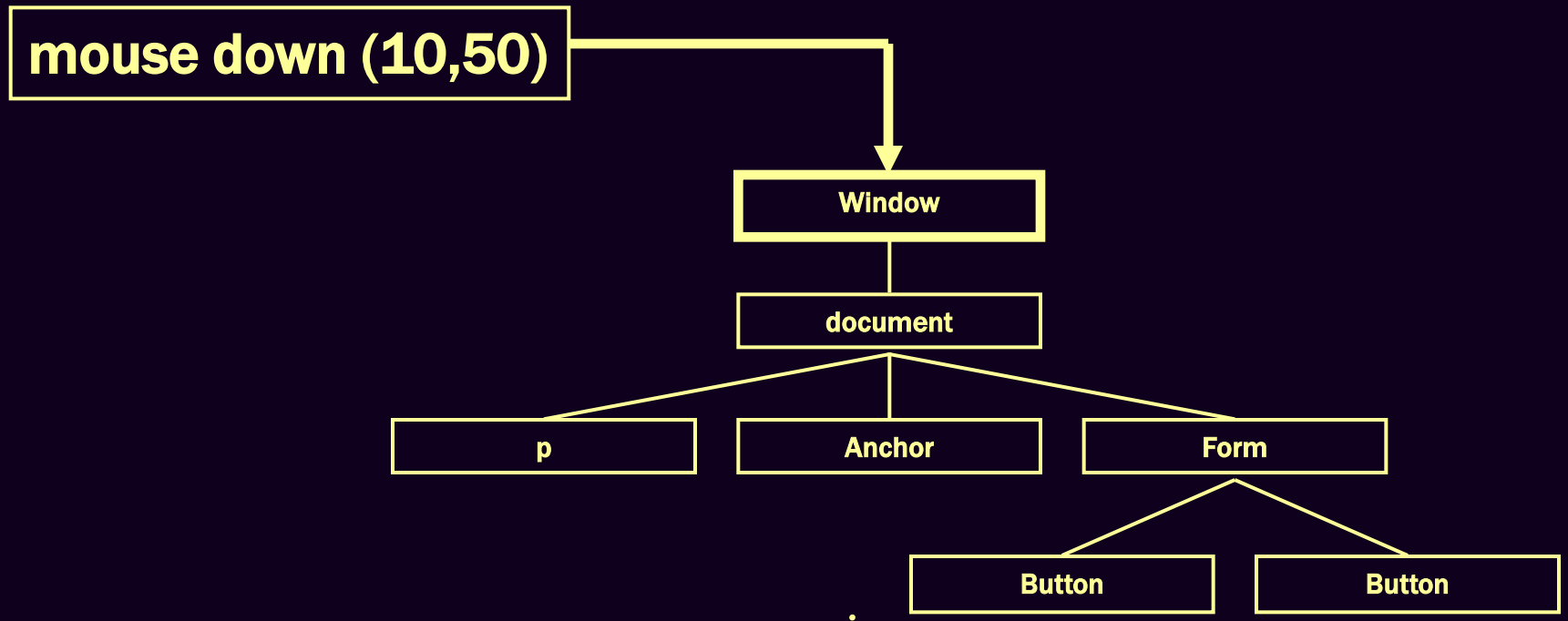
Dispatching Events

mouse down (10,50)



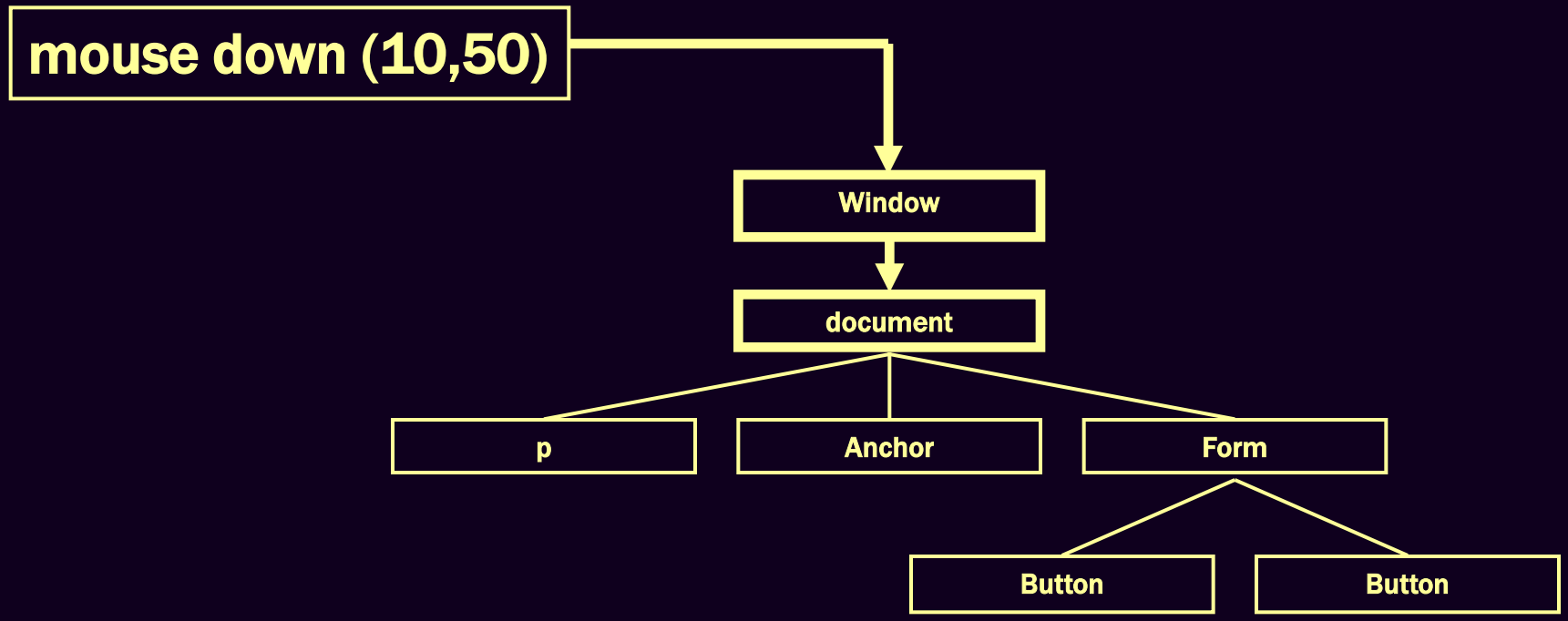

```
function onMouseDown(evt) {
    // do something...
}
```

Dispatching Events



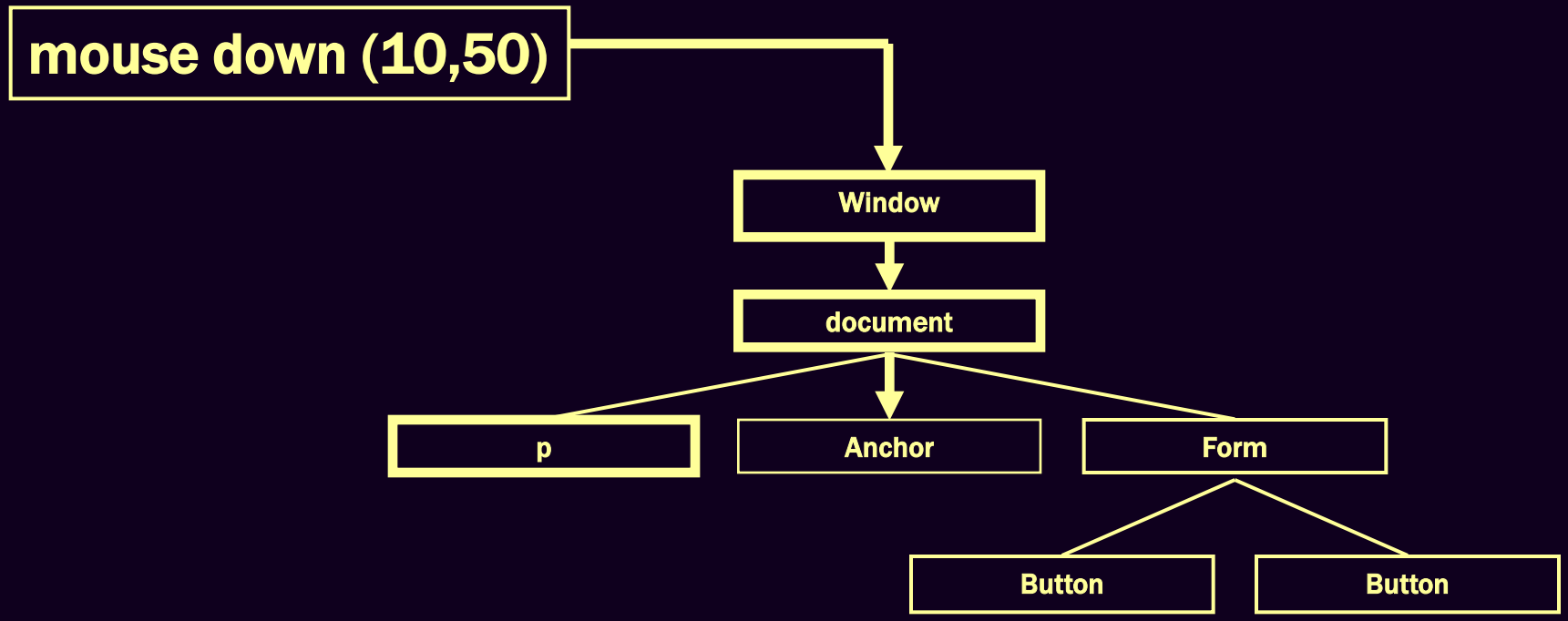

```
function onMouseDown(evt) {
    // do something...
}
```

Dispatching Events



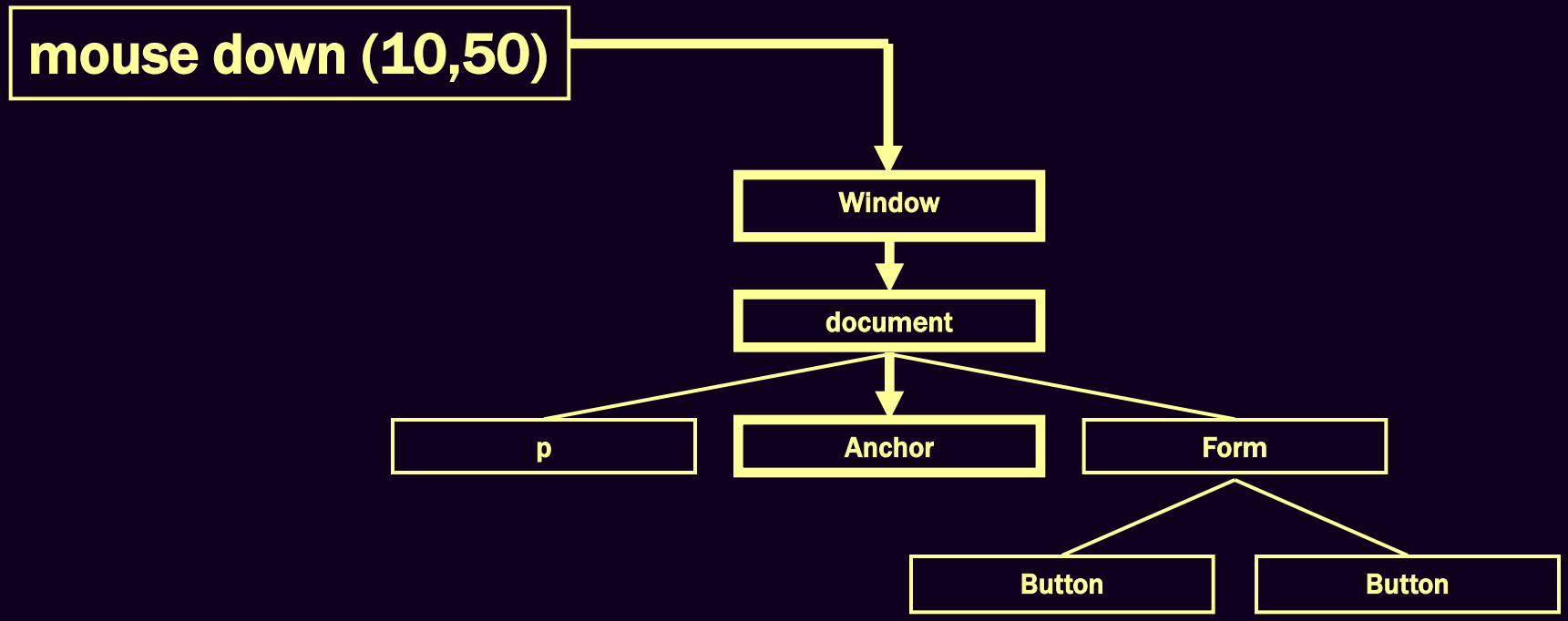

```
function onMouseDown(evt) {
    // do something...
}
```

Dispatching Events



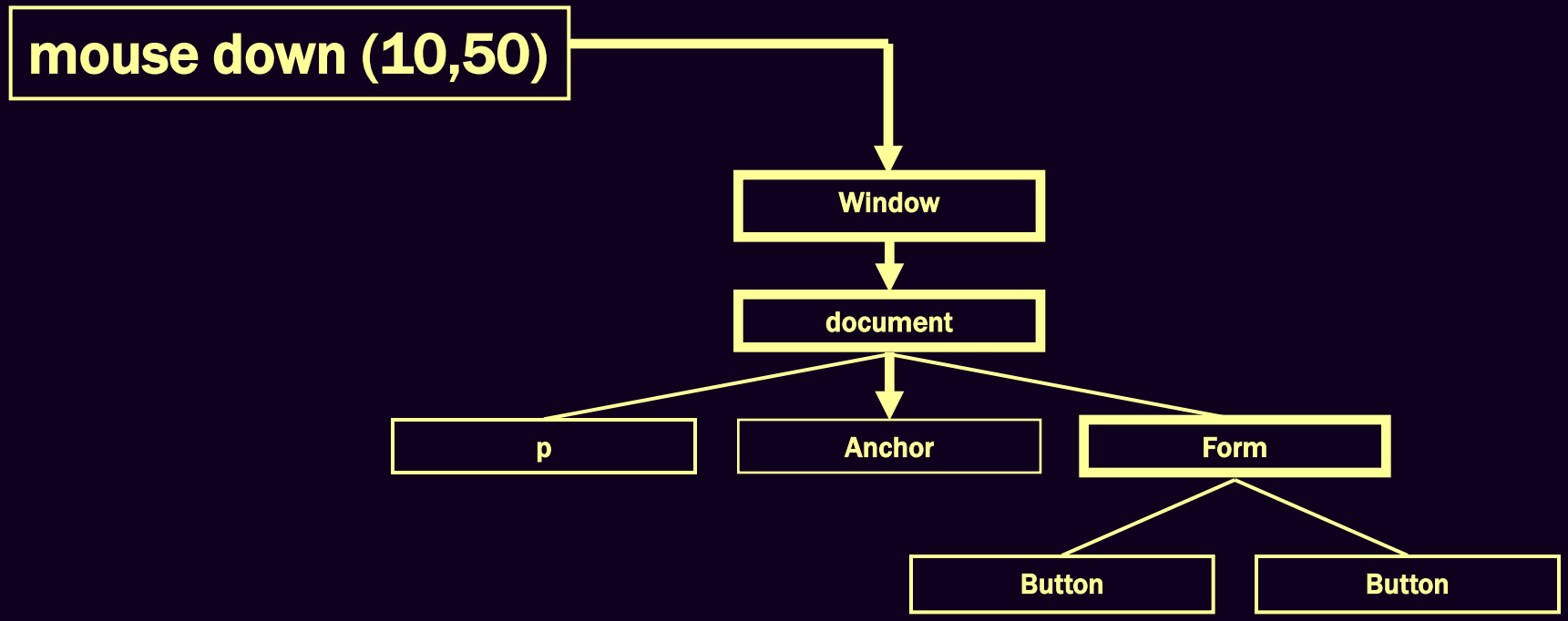

```
function onMouseDown(evt) {
    // do something...
}
```

Dispatching Events



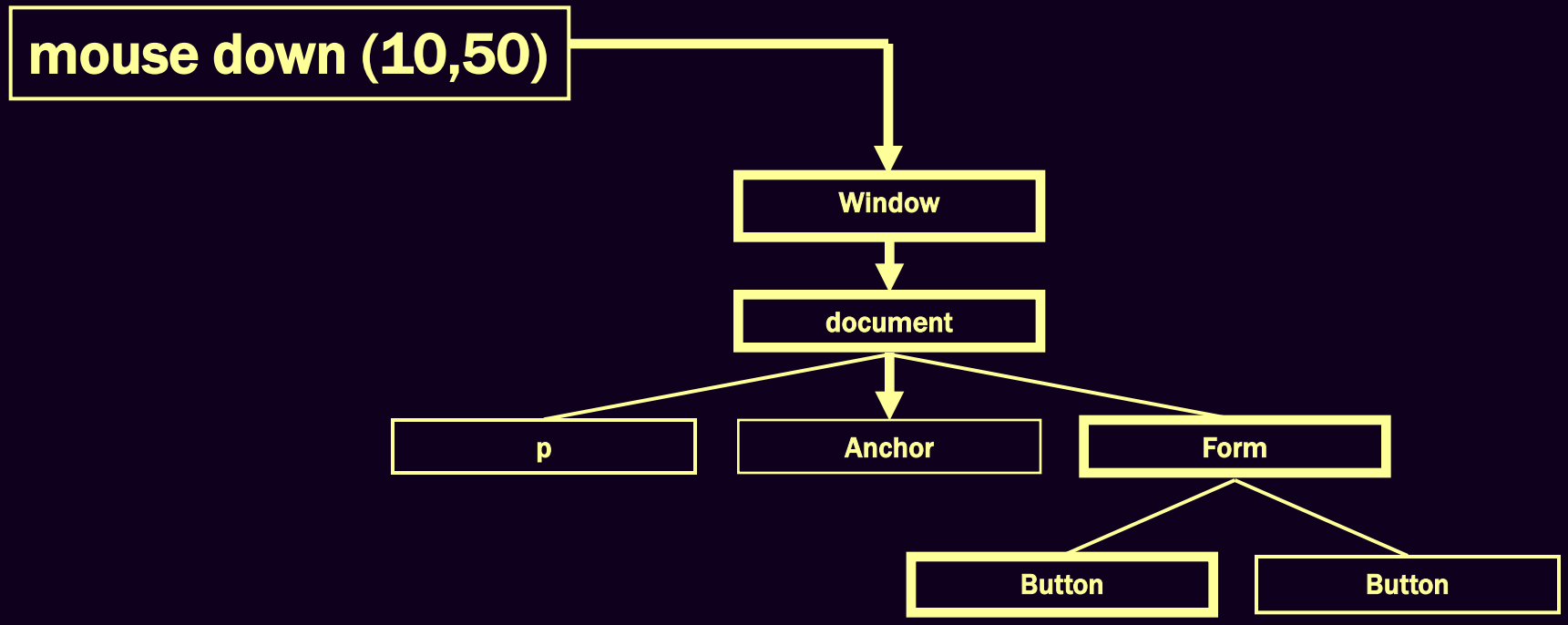

```
function onMouseDown(evt) {
    // do something...
}
```

Dispatching Events




```
function onMouseDown(evt) {
    // do something...
}
```

Dispatching Events




```
function onMouseDown(evt) {
    // do something...
}
```

Events

- The browser has an event-driven, single-threaded, asynchronous programming model.
- Events are targeted to particular nodes.
- Events cause the invocation of event handler functions.

Mouse Events

- The target is the topmost (z-index) node containing the cursor.

- click
- dblclick
- mousedown
- mousemove
- mouseout
- mouseover
- mouseup

Input Events

- The target is the node having focus.

- blur
- change
- focus
- keydown
- keypress
- keyup
- reset
- submit

Event Handlers

- Classic

```
node["on" + type] = f;
```

- Microsoft

```
node.attachEvent("on" + type, f);
```

- W3C

```
node.addEventListener(type, f, false);
```

Event Handlers

- The handler takes an optional event parameter.

  Microsoft does not send an event parameter, use the global event object instead.

Event Handlers

```
function (e) {
    e = e || event;
    var target =
        e.target || e.srcElement;
    ...
}
```

Trickling and Bubbling

- Trickling is an event capturing pattern which provides compatibility with the Netscape 4 model. Avoid it.

- Bubbling means that the event is given to the target, and then its parent, and then its parent, and so on until the event is canceled.

Why Bubble?

- Suppose you have 100 draggable objects.
- You could attach 100 sets of event handlers to those objects.
- Or you could attach one set of event handlers to the container of the 100 objects.

Cancel Bubbling

- Cancel bubbling to keep the parent nodes from seeing the event.

```
e.cancelBubble = true;
```

Memory Leaks

- Memory management is automatic.
- It is possible to hang on to too much state, preventing it from being garbage collected.

Memory Leaks on IE 6

- Explicitly remove all of your event handlers from nodes before you discard them.
- The IE6 DOM uses a reference counting garbage collector.
- Reference counting is not able to reclaim cyclical structures.
- You must break the cycles yourself.

Garbage Collection

An automatic memory management scheme implemented by the runtime environment

Analyzes usage of heap and recover pieces of storage no longer reachable from user pointers and references

Pros:

- Simplify programming,
- Shorten development lifecycle (less memory problems…)

Cons:

- Execution time cost traded for easier job for user
- Unsuitable for real time systems

E.g.:

- Scheme, Java, ML, Ada, Modula, Perl, JavaScript, VisualBasic

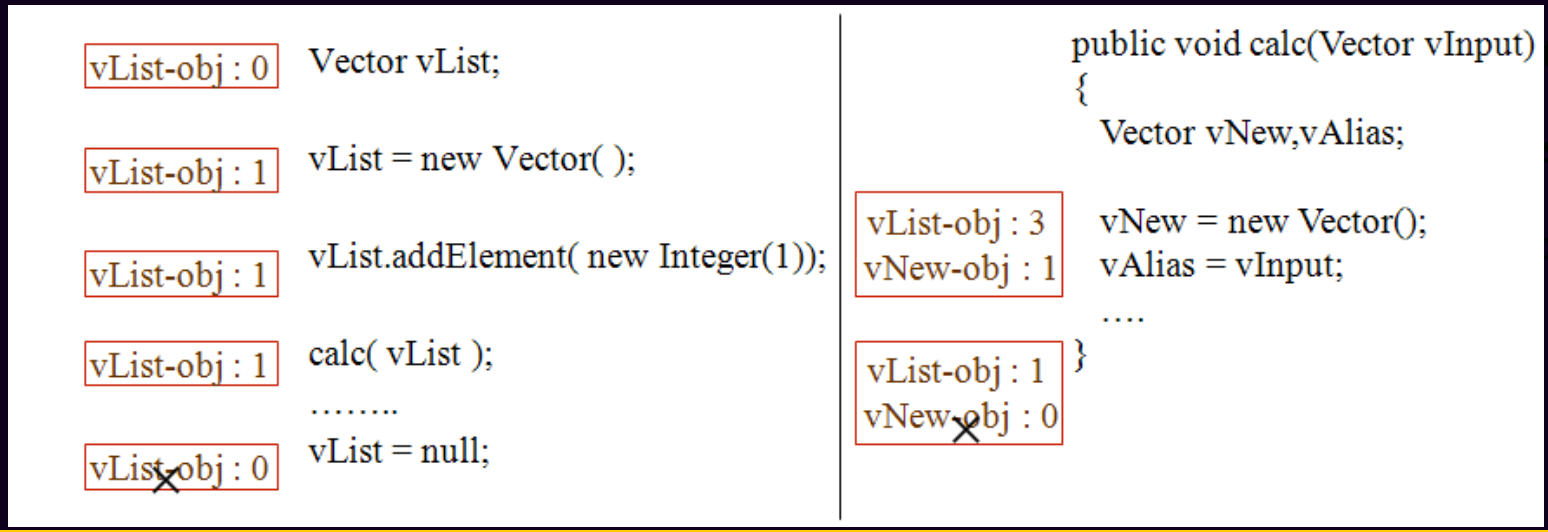
Reference Counting

Maintain total number of pointers to a storage block

- Each variable has an additional attribute (a counter) telling how many pointers are pointing to that variable.Every time a pointer is disconnected, decrement counter by 1 and check for 0
- Every time a pointer is connected, increment the counter by 1
- **If counter is 0, delete the variable.**

Problems:

Costs extra memory and execution time for updates + Circular references

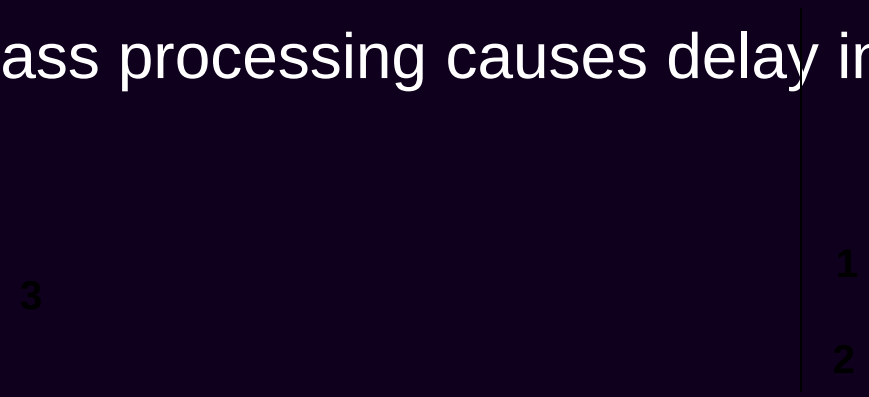


Mark and Sweep

Sweep through entire heap looking for referenced blocks and free unused blocks

Problems:

multi pass processing causes delay in execution of programs

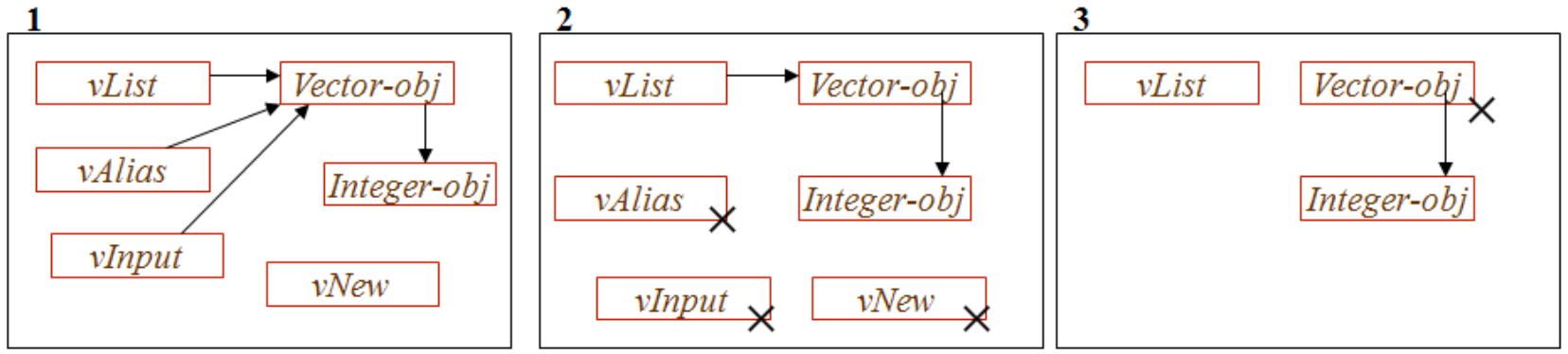


Mark and Sweep

```
Vector vList;
vList = new Vector( );
vList.addElement( new Integer(1));
calc( vList );
……..
vList = null;
```

```
public void calc(Vector vInput)
{
  Vector vNew,vAlias;

  vNew = new Vector();
  vAlias = vInput;
  ….
}
```



```
public class foo
{
  public foo()
  {
    StringBuffer strbufLocal;
    strbufLocal = new StringBuffer( );

    for(int nIndex = 1; nIndex < 20000; nIndex++)
    {
    StringBuffer strbufTemp = new StringBuffer();
    strbufTemp.append("x");
    strbufLocal.append( strbufTemp );
    int nLength = strbufTemp.length();
    }

  }
  public static void main(String strarrArgs[])
  {
    foo f = new foo( );
  }
}
```

```
C:\>java -verbose:gc foo
[GC 511K->182K(1984K), 0.0208331 secs]
[GC 694K->196K(1984K), 0.0045478 secs]
[GC 708K->214K(1984K), 0.0026766 secs]

C:\>java -verbose:gc foo
[GC 511K->182K(1984K), 0.0209680 secs]
[GC 694K->196K(1984K), 0.0046308 secs]
[GC 708K->214K(1984K), 0.0026199 secs]

C:\>java -verbose:gc foo
[GC 511K->182K(1984K), 0.0206537 secs]
[GC 694K->196K(1984K), 0.0045754 secs]
[GC 708K->214K(1984K), 0.0026168 secs]

C:\>java -verbose:gc foo
C:\>
```

Memory Leaks on IE 6

- That was not an issue for page view-driven applications.
- It is a showstopper for Ajax applications (more about this later...)
- It will be fixed in IE7 and IE8....

Memory Leaks on IE 6

- Remove all event handlers from deleted DOM nodes.
- It must be done on nodes before **`removeChild`** or **`replaceChild`**.
- It must be done on nodes before they are replaced by changing **`innerHTML`**.

Breaking Links in the DOM

```
function purgeEventHandlers(node) {
    walkTheDOM(node, function (e) {
        for (var n in e) {
            if (typeof e[n] ===
                    'function') {
                e[n] = null;
            }
        }
    });
}
```

JavaScript

- `alert(`*`text`*`)`
- `confirm(`*`text`*`)`
- `prompt(`*`text, default`*`)`

  Avoid these.

- `setTimeout(`*`func, msec`*`)`
- `setInterval(`*`func, msec`*`)`

window

- The **window** object is also the JavaScript global object.
- Every window, frame, and iframe has its own unique **window** object.
- aka **self**. And sometimes **parent** and **top**.

Inter-window

- **`frames[]`** Child frames and iframes
- **`name`** Text name of window
- **`opener`** Reference to **`open`**
- **`parent`** Reference to parent
- **`self`** Reference to this window
- **`top`** Reference to outermost
- **`window`** Reference to this window

- **`open()`** Open new window

Inter-window

- A script can access another window if

  It can get a reference to it.

  `document.domain === otherwindow.document.domain`

- Same Origin Policy

Cross Browser

- Weak standards result in significant vendor-specific differences between browsers.

- Browser Detection.
- Feature Detection.
- Platform Libraries.

Browser Detection

- Determine what kind of browser that page is running in.
- Execute conditionally.
- The browsers lie.
  `navigator.userAgent Mozilla/4.0`
- Brittle. Not recommended.
- **`http://www.javascriptkit.com/javatutors/objdetect3.shtml`**

Feature Detection

- Using reflection, ask if desired features are present.
- Execute conditionally.

```
function addEventHandler(node, type, f) {
    if (node.addEventListener) {
        node.addEventListener(type, f, false);
    } else if (node.attachEvent) {
        node.attachEvent("on" + type, f);
    } else {
        node["on" + type] = f;
    }
}
```

Feature Detection

- Using reflection, ask if desired features are present.
- Execute conditionally.

```
function addEventHandler(node, type, f) {
    node["on" + type] = f;
}
```

Use a Platform Library

- A platform library insulates the application from the poisonous browsers.

- E.g.:

  DHTMLX

The Cracks of DOM

- The DOM buglist includes all of the bugs in the browser.
- The DOM buglist includes all of the bugs in all supported browsers.
- No DOM completely implements the standards.
- Much of the DOM is not described in any standard.

Coping

1. Do what works.
2. Do what is common.
3. Do what is standard.

The Wall

- Browsers are now being push to their limits.
- Be prepared to back off.
- Reduce your memory requirements.
- Balance of client and server.

The Hole

- The browser was not designed to be a general purpose application platform.

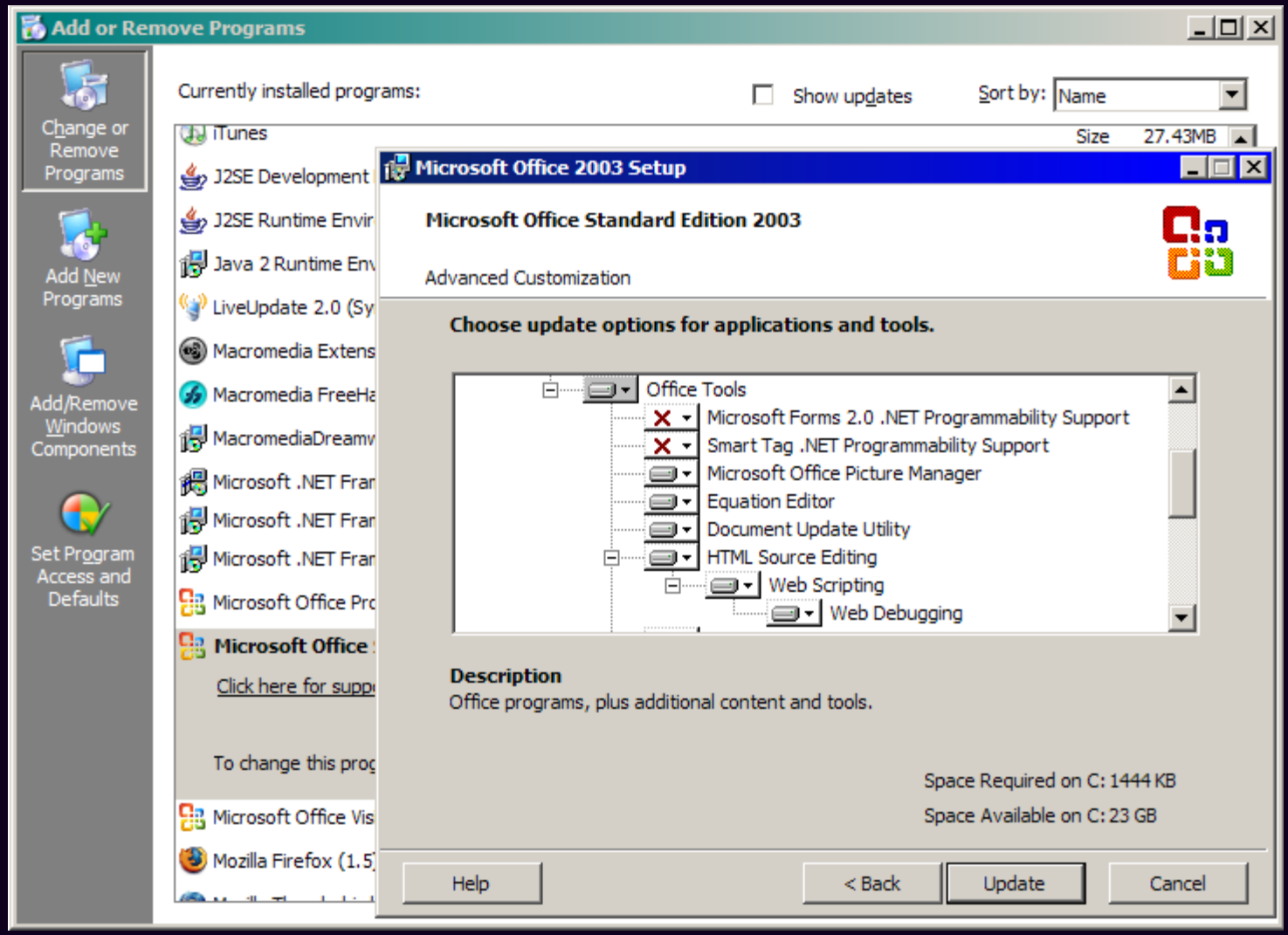
Debugging

- IE
  Microsoft Script Debugger
    Office 2003
    Visual Studio

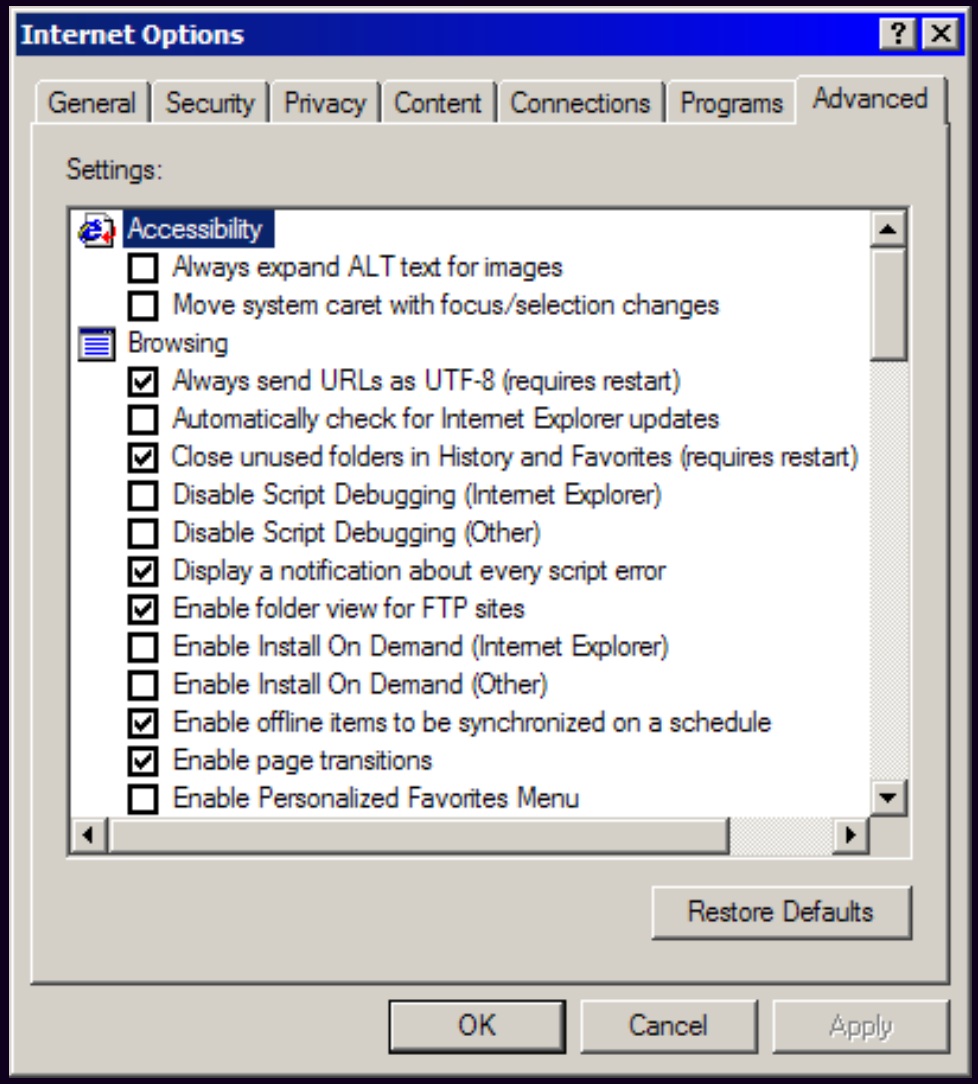
- Mozilla
  Venkman
  Firebug

- Safari
  Drosera

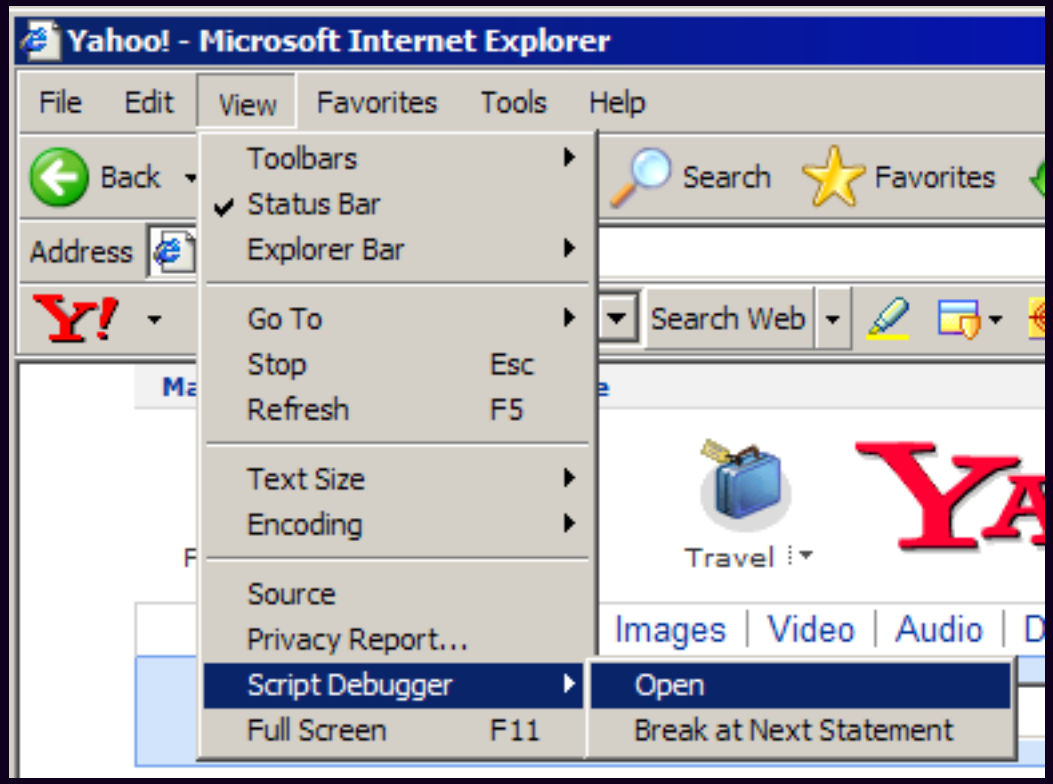
Microsoft Script Debugger



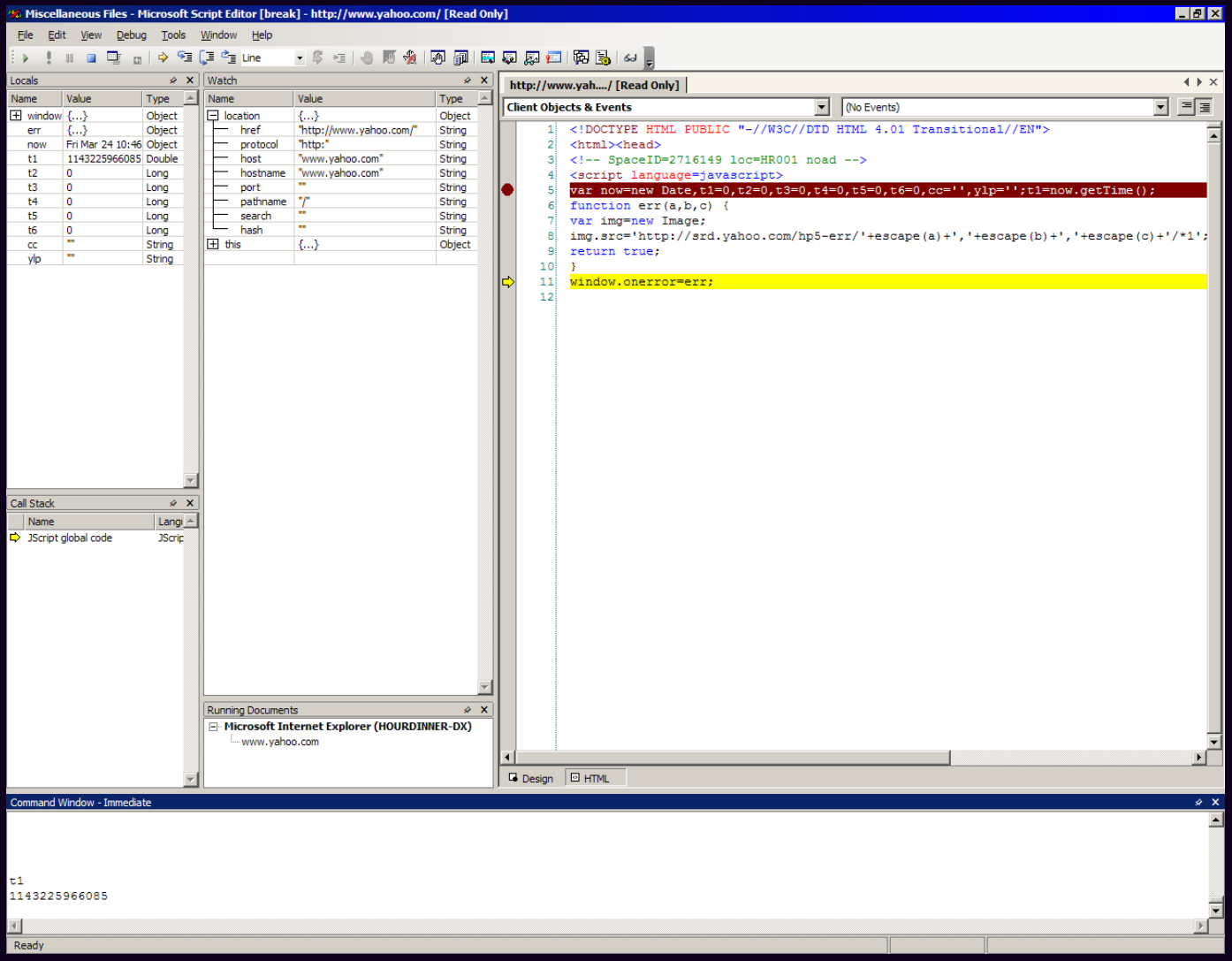
Microsoft Script Debugger



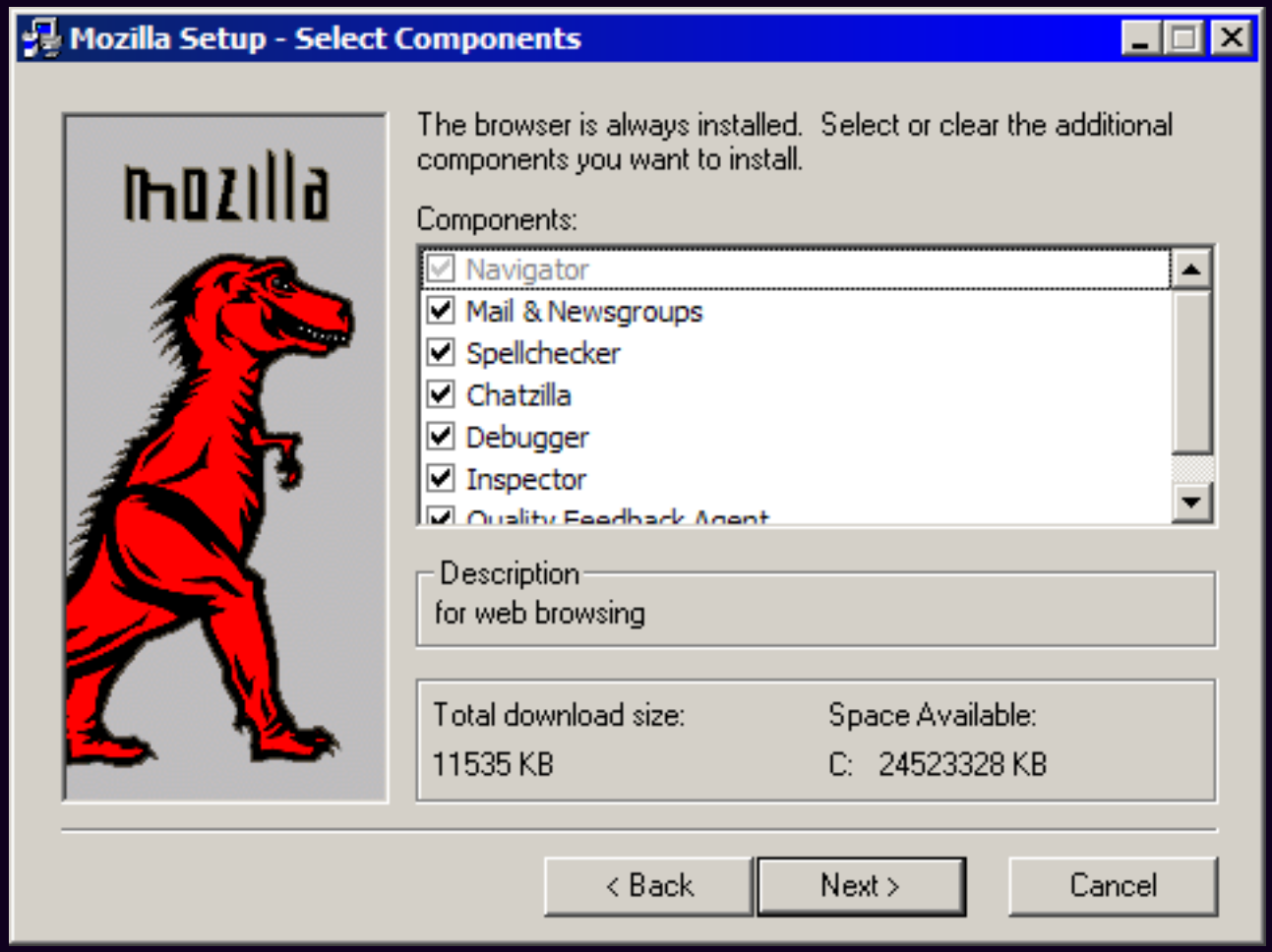
Microsoft Script Debugger



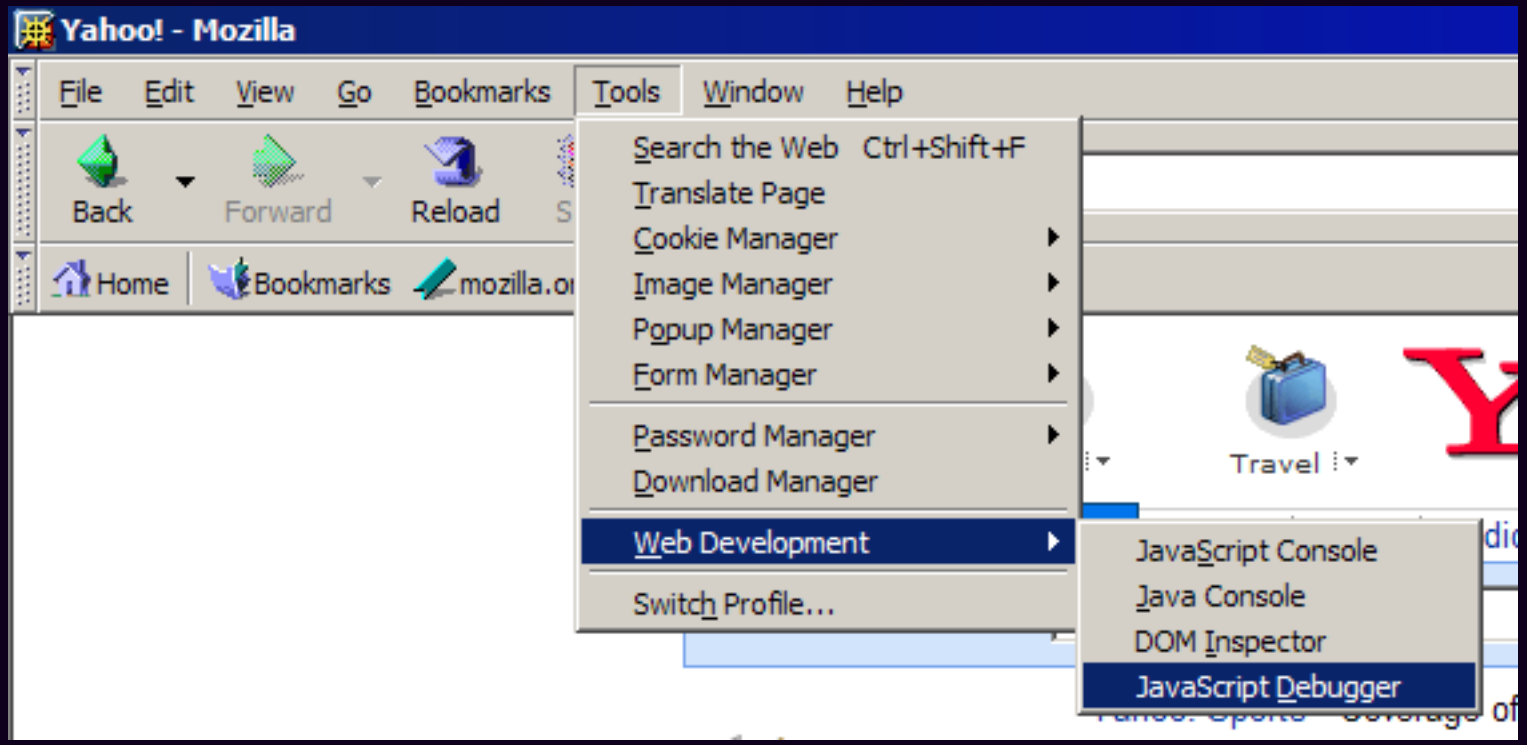
Microsoft Script Debugger



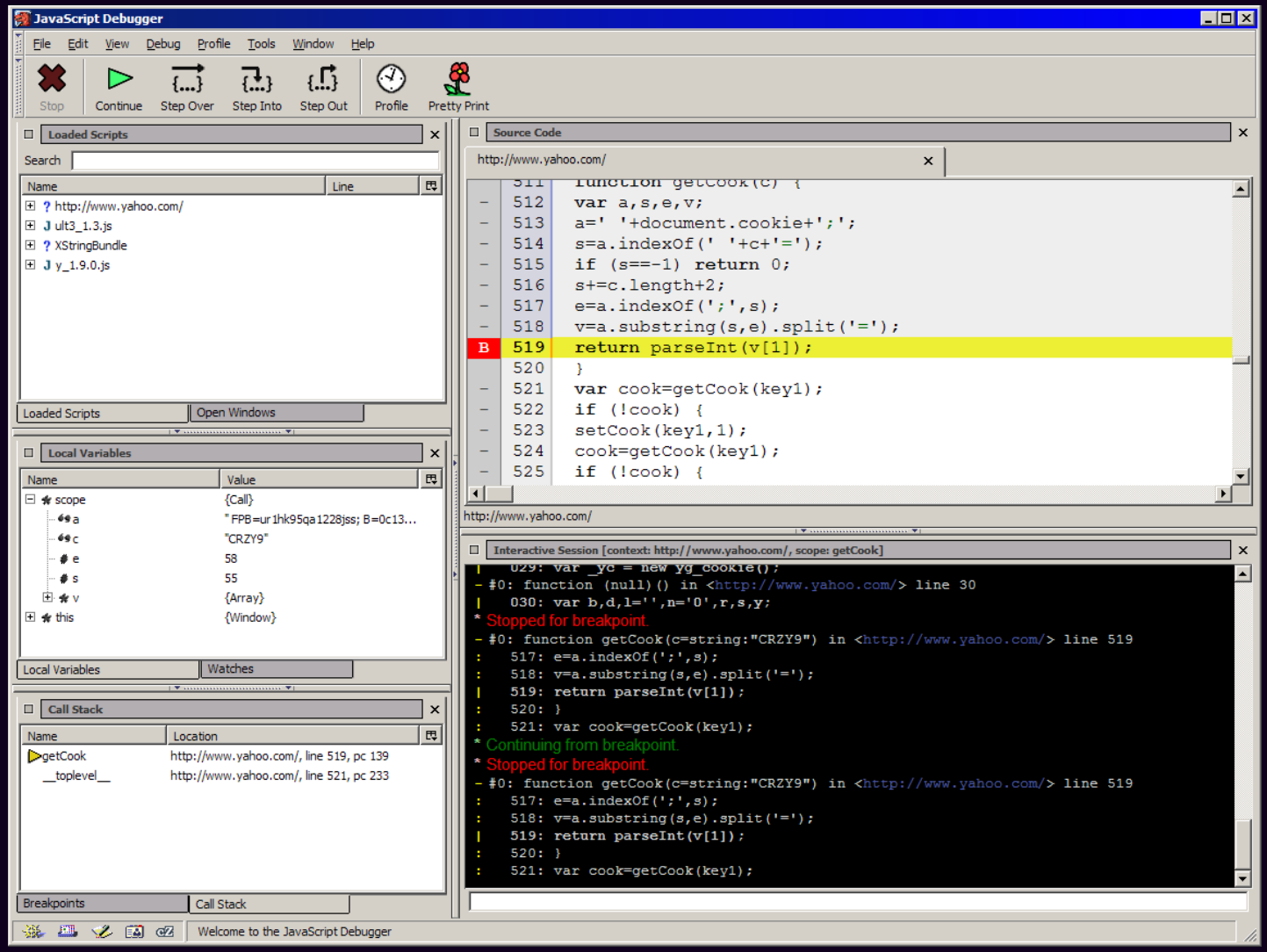
Venkman


Mozilla Setup - Select Components
mozilla
The browser is always installed. Select or clear the additional components you want to install.
Components:
Navigator
Mail & Newsgroups
Spellchecker
Chatzilla
Debugger
Inspector
Description
for web browsing
Total download size:
11535 KB
Space Available:
C: 24523328 KB
< Back
Next >
Cancel

Venkman


Yahoo! - Mozilla
File
Edit
View
Go
Bookmarks
Tools
Window
Help
Back
Forward
Reload
Home
Bookmarks
Search the Web Ctrl+Shift+F
Translate Page
Cookie Manager
Image Manager
Popup Manager
Form Manager
Password Manager
Download Manager
Web Development
Switch Profile...
JavaScript Console
Java Console
DOM Inspector
JavaScript Debugger
Travel

Venkman


JavaScript Debugger
File Edit View Debug Profile Tools Window Help
Stop
Continue
Step Over
Step Into
Step Out
Profile
Pretty Print
Loaded Scripts
Search
Name
Line
http://www.yahoo.com/
ult3_1.3.js
XStringBundle
y_1.9.0.js
Loaded Scripts
Open Windows
Local Variables
Name
Value
scope
{Call}
a
"FPB=ur1hk95qa1228jss; B=0c13...
c
"CRZY9"
e
58
s
55
v
{Array}
this
{Window}
Local Variables
Watches
Call Stack
Name
Location
getCook
http://www.yahoo.com/, line 519, pc 139
__toplevel__
http://www.yahoo.com/, line 521, pc 233
Breakpoints
Call Stack
Source Code
http://www.yahoo.com/
function getCook(c) {
var a,s,e,v;
a=' '+document.cookie+';';
s=a.indexOf(' '+c+'=');
if (s==-1) return 0;
s+=c.length+2;
e=a.indexOf(';',s);
v=a.substring(s,e).split('=');
return parseInt(v[1]);
}
var cook=getCook(key1);
if (!cook) {
setCook(key1,1);
cook=getCook(key1);
if (!cook) {
http://www.yahoo.com/
Interactive Session [context: http://www.yahoo.com/, scope: getCook]
#0: function (null)() in <http://www.yahoo.com/> line 30
030: var b,d,l='',n='0',r,s,y;
Stopped for breakpoint.
#0: function getCook(c=string:"CRZY9") in <http://www.yahoo.com/> line 519
517: e=a.indexOf(';',s);
518: v=a.substring(s,e).split('=');
519: return parseInt(v[1]);
520: }
521: var cook=getCook(key1);
Continuing from breakpoint.
Stopped for breakpoint.
#0: function getCook(c=string:"CRZY9") in <http://www.yahoo.com/> line 519
517: e=a.indexOf(';',s);
518: v=a.substring(s,e).split('=');
519: return parseInt(v[1]);
520: }
521: var cook=getCook(key1);
Welcome to the JavaScript Debugger

Debugging

- Check the browser's error console message
- With Firefox

  Check the DOM in Venkman (or Firebug)

  Use `console.log("message");` to write to the console

Coding Efficiency Tips

- Common subexpression removal
- Loop invariant removal

Before

```
for (var i = 0; i < divs.length; i += 1) {
    divs[i].style.color = "black";
    divs[i].style.border = thickness +
        'px solid blue';
    divs[i].style.backgroundColor = "white";
}
```

After

```
var border = thickness + 'px solid blue',
    nrDivs = divs.length;

for (var i = 0; i < nrDivs; i += 1) {
    var ds = divs[i].style;
    ds.color = "black";
    ds.border = border;
    ds.backgroundColor = "white";
}
```

Strings

- Concatenation with **+**

  Each operation allocates memory

```
foo = a + b;
```

- Concatenate with array**.join('')**

  The contents of an array are concatenated into a single string

```
foo = [a, b].join('');
```

Minification vs Obfuscation

- Reduce the amount of source code to reduce download time.
- Minification deletes whitespace and comments.
- Obfuscation also changes the names of things.
- Obfuscation can introduce bugs.

<script></script>

- Script files can have a big impact on page loading time.

1. Place **<script src>** tags as close to the bottom of the body as possible. (Also, place CSS **<link>** as high in the head as possible.)

2. Minify and gzip script files.

3. Reduce the number of script files as much as possible.

<script></script>

- `<!--  // -->`
  Hack for Mosaic and Navigator 1.0.

- `language=javascript`
  Deprecated.

- `type=text/javascript`
  Ignored.

- `src=`URL
  Highly recommended.
  Don't put code on pages.