



**aninconvenientapi**

THE THEORY OF THE DOM

© 2006 Douglas Crockford

# **The Misconceived Web**

- **The original vision of the WWW was as a hyperlinked document-retrieval system.**
- **It did not anticipate presentation, session, or interactivity.**
- **If the WWW were still consistent with TBL's original vision, Yahoo would still be two guys in a trailer.**

# **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)**

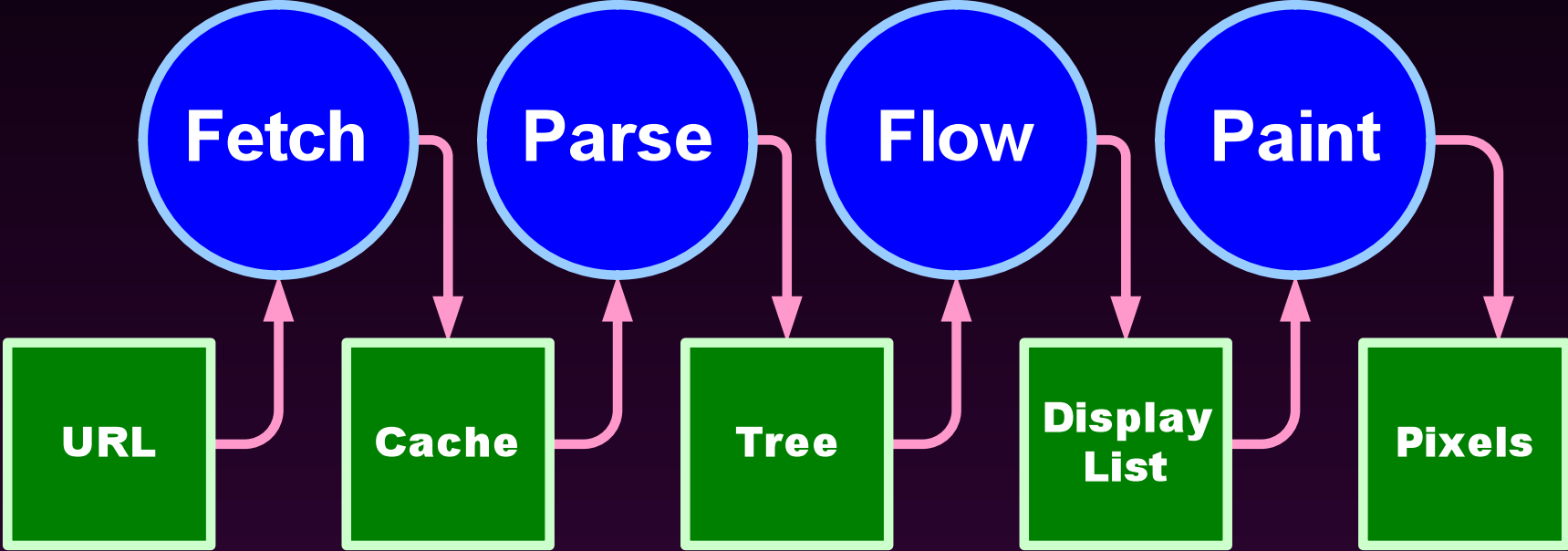
# Proprietary Traps

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

# Pax Microsoft

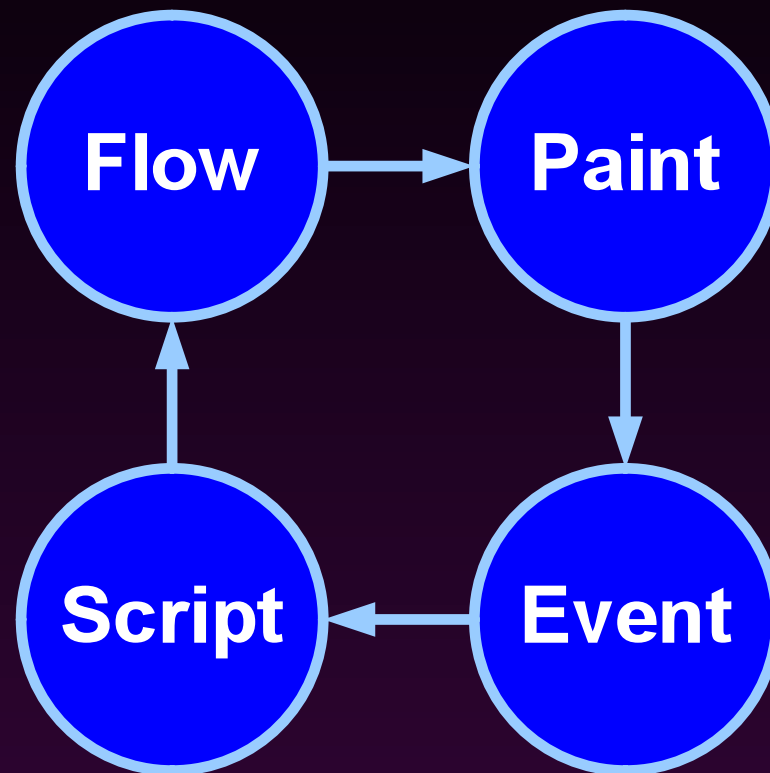
- **In the years since the end of the Browser War, the number of browser variations in significant use fell off significantly.**
- **W3C attempts to unify.**
- **Mozilla abandoned the Netscape layer model in favor of the W3C model.**
- **The browser platform becomes somewhat stable.**
- **DHTML becomes 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](http://developer.yahoo.com/yui/articles/gbs/gbs_browser-chart.html)

# `<script></script>`

- `<!-- // -->`  
Hack for Mosaic and Navigator 1.0.
- `language=javascript`  
Deprecated.
- `src=URL`  
Highly recommended.  
Don't put code on pages.
- `type=text/javascript`  
Ignored.

# `<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.**

# 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.**

# Collections

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

**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.**



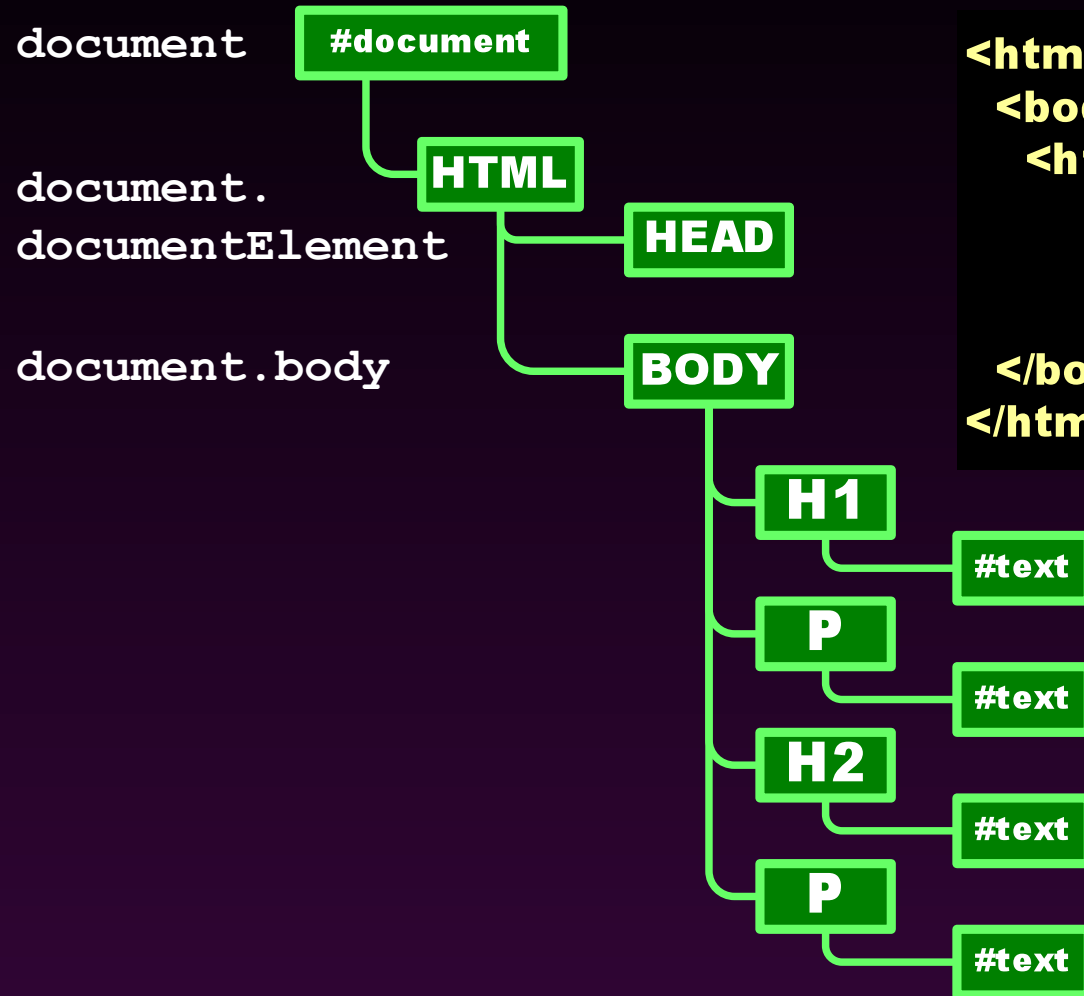
# Retrieving Nodes

```
document.getElementById(id)
```

```
document.getElementsByName(name)
```

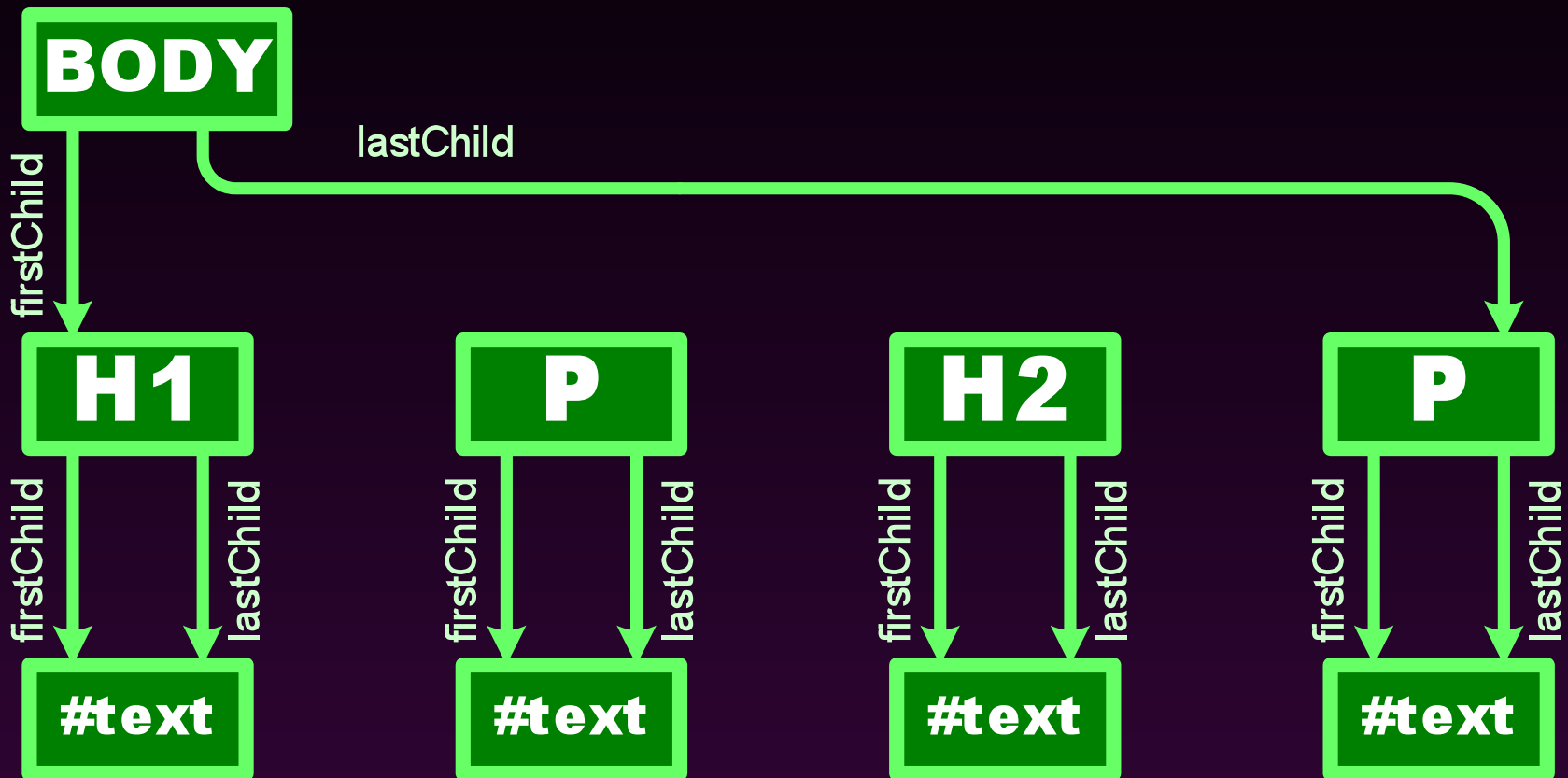
```
node.getElementsByTagName(tagName)
```

# Document Tree Structure

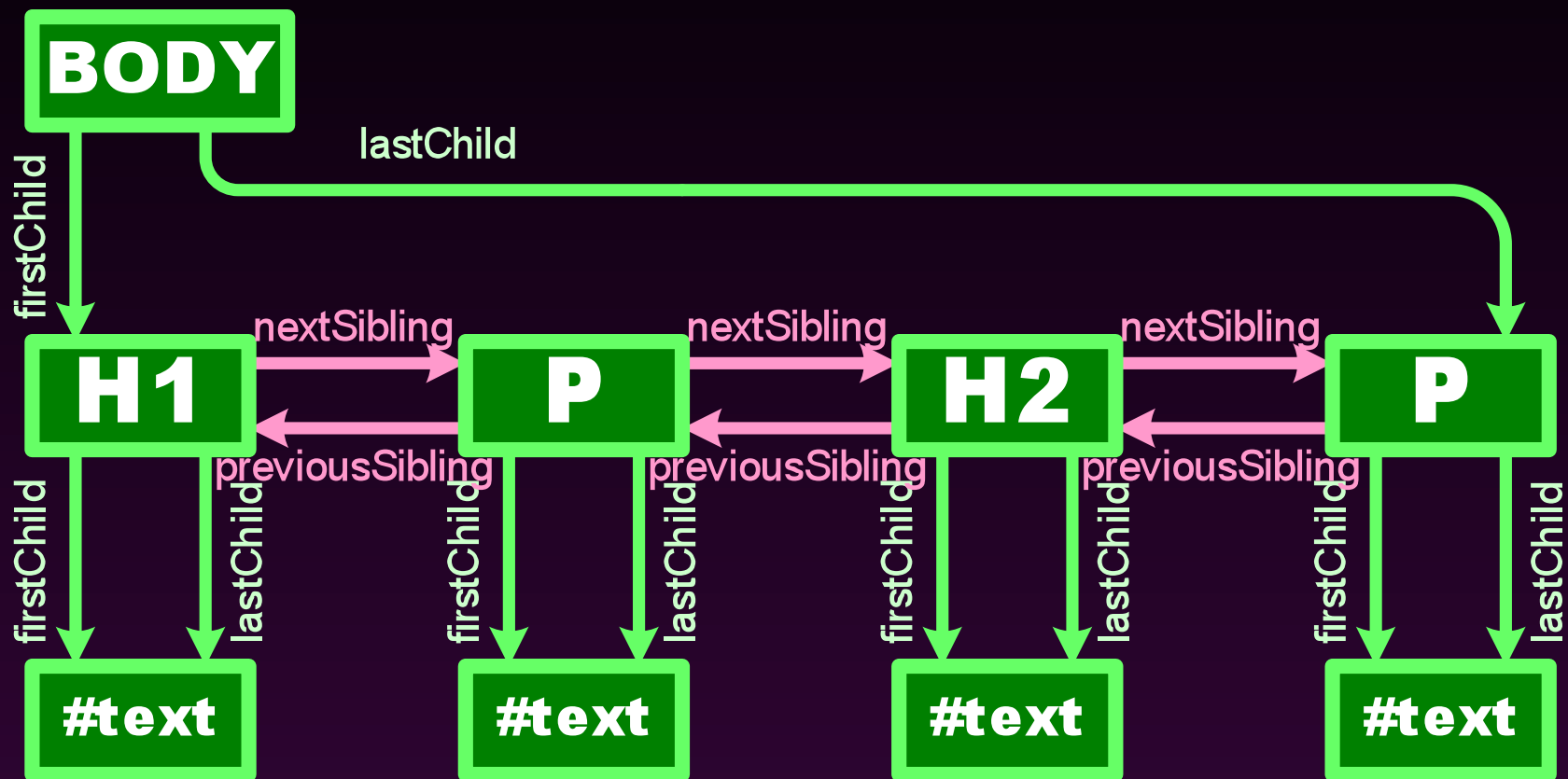


```
<html>  
  <body>  
    <h1>Heading 1</h1>  
    <p>Paragraph.</p>  
    <h2>Heading 2</h2>  
    <p>Paragraph.</p>  
  </body>  
</html>
```

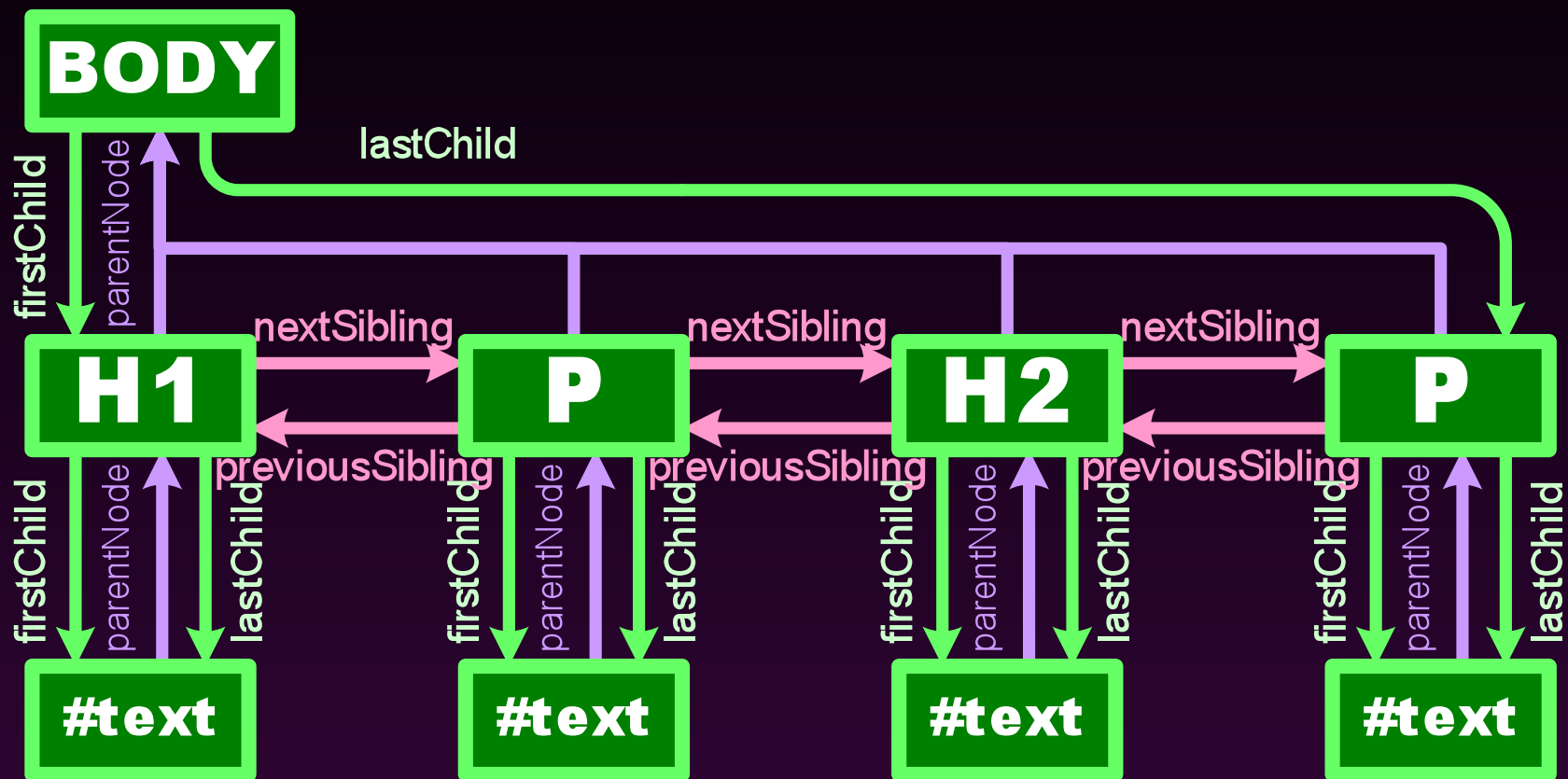
# child



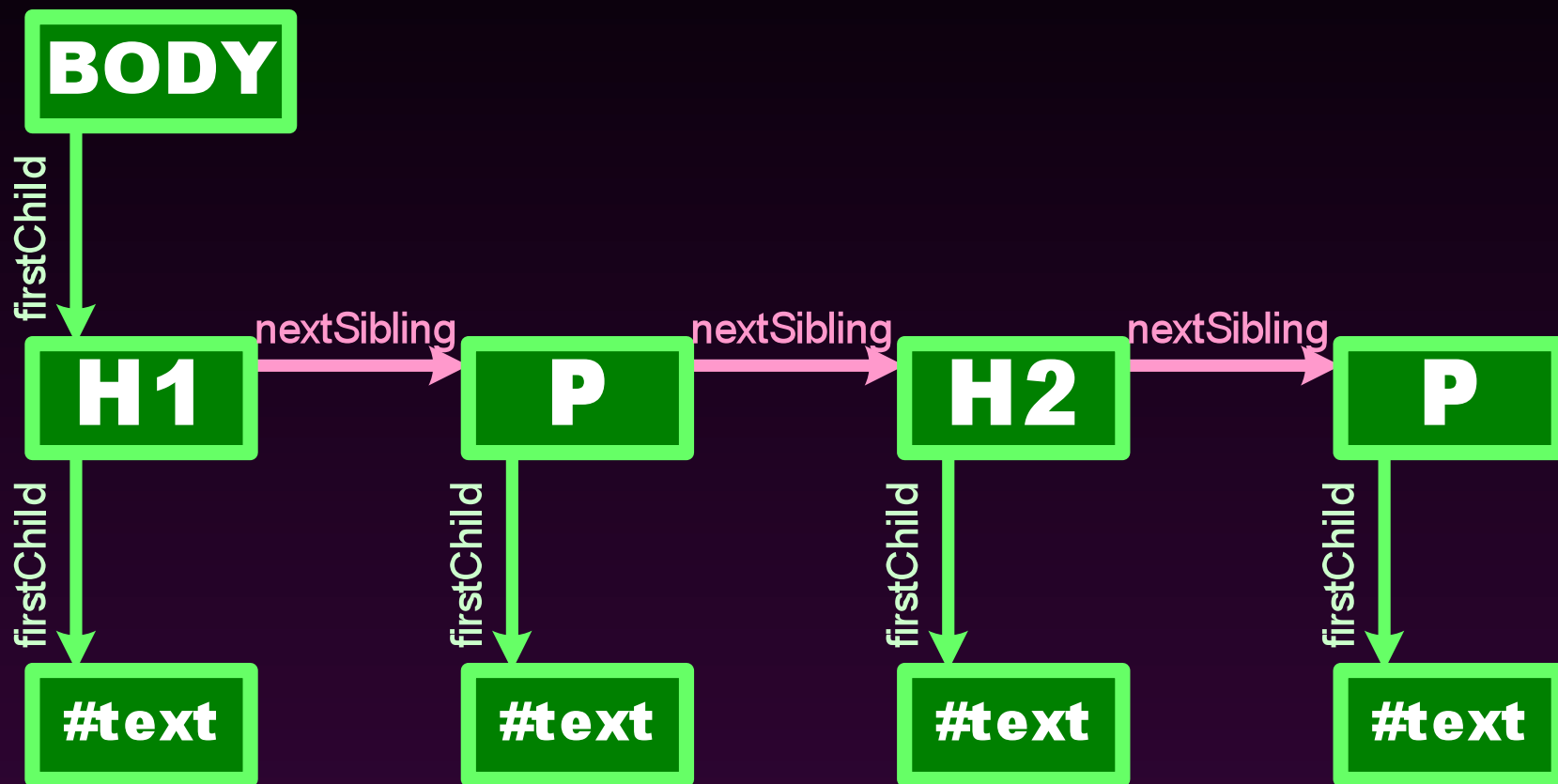
# child, sibling



# child, sibling, parent



# child, sibling, parent



# Walk the DOM

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

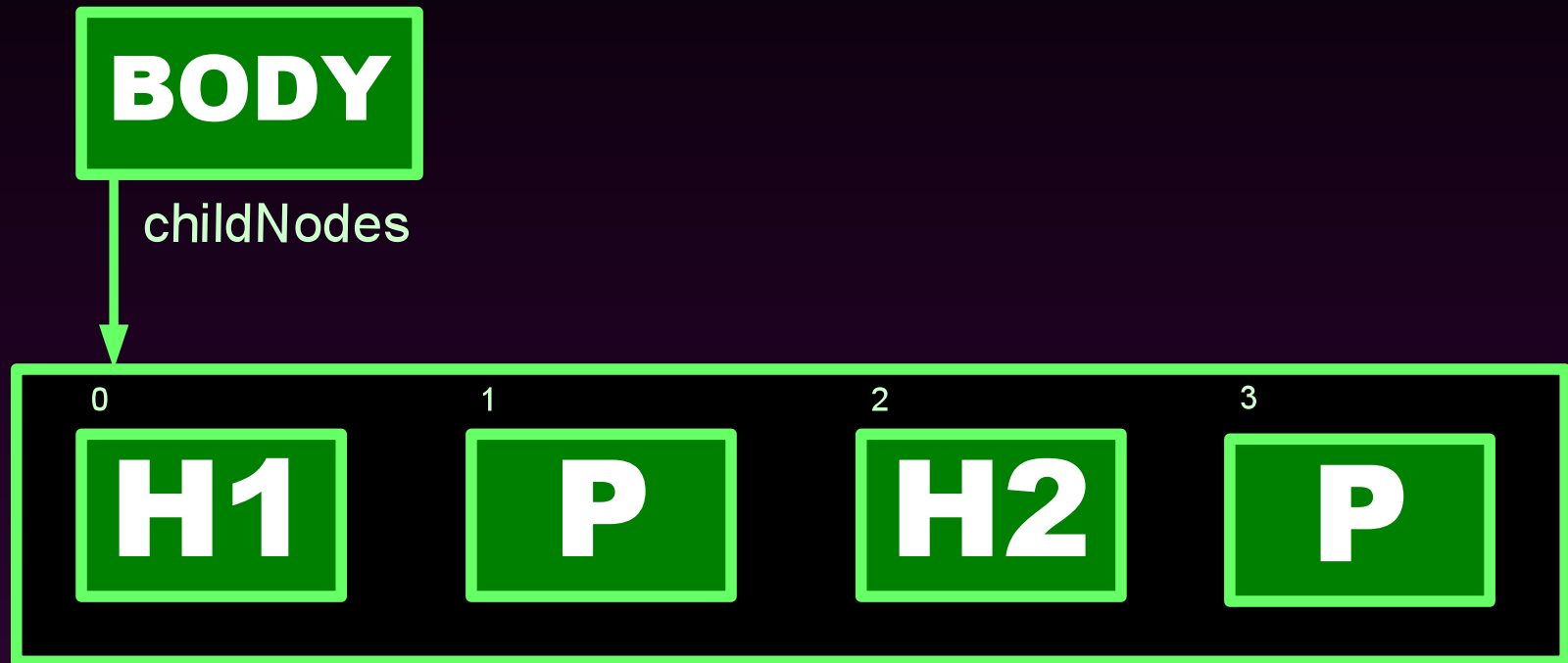
```
function walkTheDOM(node, func) {  
    func(node);  
    node = node.firstChild;  
    while (node) {  
        walkTheDOM(node, func);  
        node = node.nextSibling;  
    }  
}
```

# 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

- background-color
- border-radius
- font-size
- list-style-type
- word-spacing
- z-index

## JavaScript

- backgroundColor
- borderRadius
- fontSize
- listStyleType
- wordSpacing
- 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.**

# Which Way Is Better?

- **It is better to build or clone elements and append them to the document?**
- **Or is it better to compile an HTML text and use innerHTML to realize it?**
- **Favor clean code and easy maintenance.**
- **Favor performance only in extreme cases.**

# 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;  
if (e.stopPropagation) {  
    e.stopPropagation();  
}
```

- **Or you can use YUI's cancelBubble method.**

# Prevent Default Action

- **An event handler can prevent a browser action associated with the event (such as submitting a form).**

```
e.returnValue = false;  
if (e.preventDefault) {  
    e.preventDefault();  
}  
return false;
```

- **Or you can use YUI's preventDefault method.**

# 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.**

# **Memory Leaks on IE 6**

- **That was not an issue for page view-driven applications.**
- **It is a showstopper for Ajax applications.**
- **It will be fixed in IE7.**

# 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;
            }
        }
    });
}
```

- Or you can use YUI's `purgeElement` method.

# JavaScript

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

**These functions break the asynchronous model.**

**Avoid these in Ajax applications.**

- `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.mozilla.org/docs/web-developer/sniffer/browser\\_type.html](http://www.mozilla.org/docs/web-developer/sniffer/browser_type.html)

# 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;  
}
```

```
YAHOO.util.Event.addListener(node, type, f);
```

- **Support for custom events, and for adding events to object that don't exist yet, and for purging event handlers from objects.**

# Use a Platform Library

- **A platform library insulates the application from the poisonous browsers.**
- **YUI is highly recommended.**
- `http://developer.yahoo.com/yui/`

# 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.**
- **Lacks a compositing model.**
- **Accessibility suffers.**
- **Lacks support for cooperation under mutual suspicion.**

**The Peace is ending.**

# WWW War II

- **Microsoft has awoken. They are beginning to innovate again.**
- **There are now 4 major browser makers.**
- **They will be flooding the web with bugs.**



# **We Will Prevail**

- **We must use our clout to keep the browser makers in line.**
- **We must be players, not pawns.**
- **We must set the standards.**

# References

- **The Document Object Model in Mozilla**  
<http://www.mozilla.org/docs/dom/>
- **MSDN HTML and DHTML Reference**  
[http://msdn.microsoft.com/workshop/author/dhtml/reference/dhtml\\_reference\\_entry.asp](http://msdn.microsoft.com/workshop/author/dhtml/reference/dhtml_reference_entry.asp)
- **Introduction to Safari JavaScript Programming Topics**  
<http://developer.apple.com/documentation/AppleApplications/Conceptual/SafariJSProgTopics/index.html>
- **Document Object Model (DOM) Level 2 Core Specification**  
<http://www.w3.org/TR/DOM-Level-2-Core/>