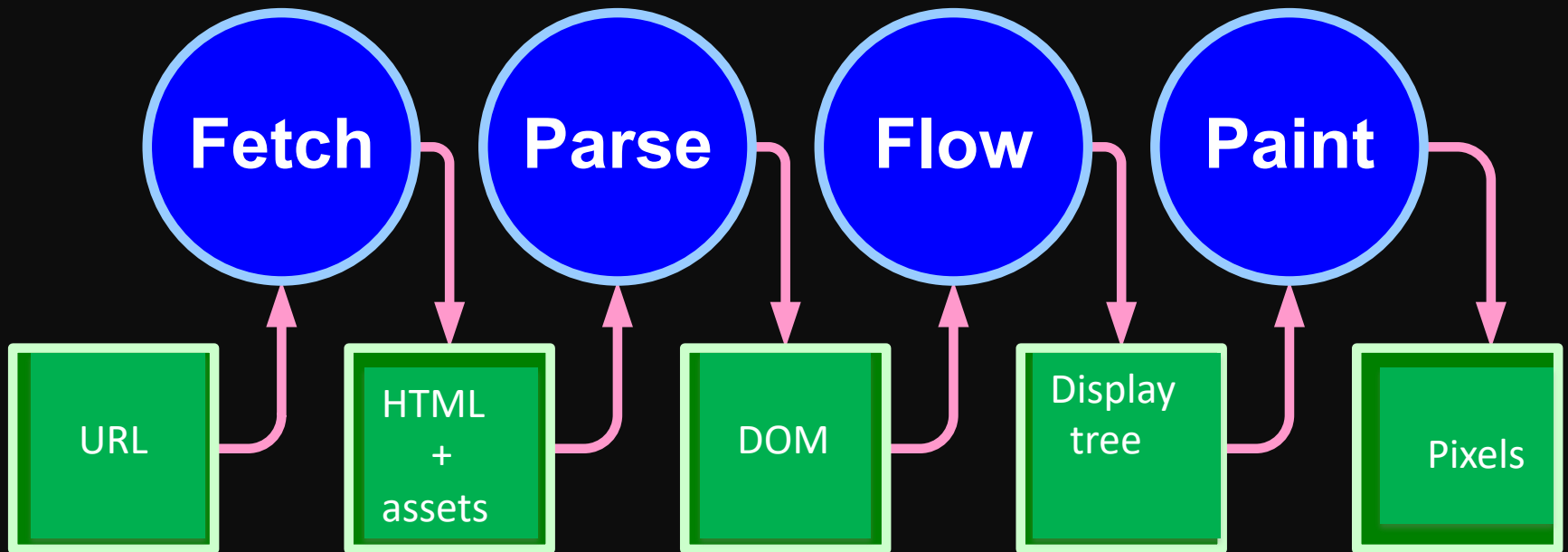


The Web Browser

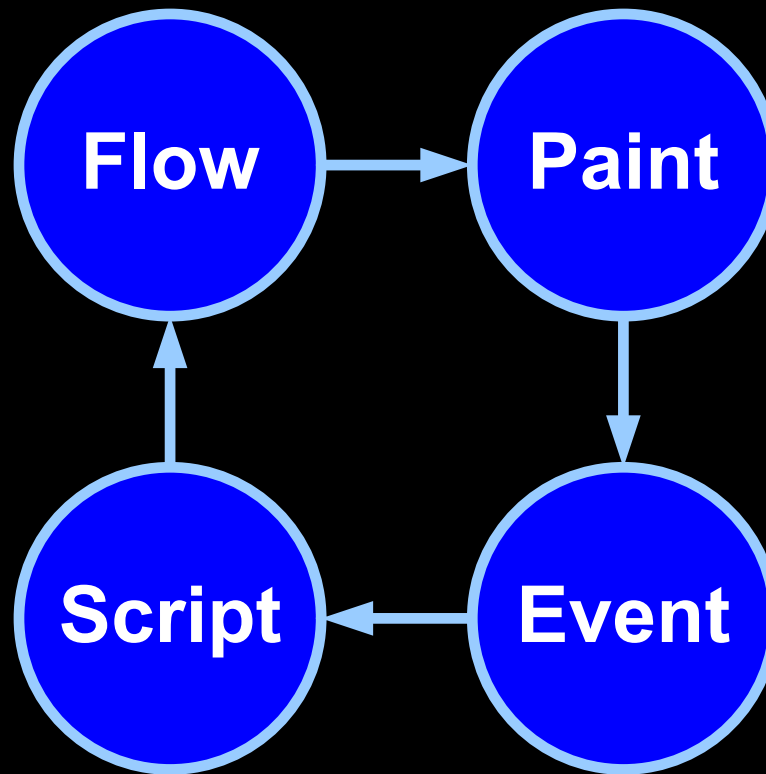
An event-driven environment

Browser

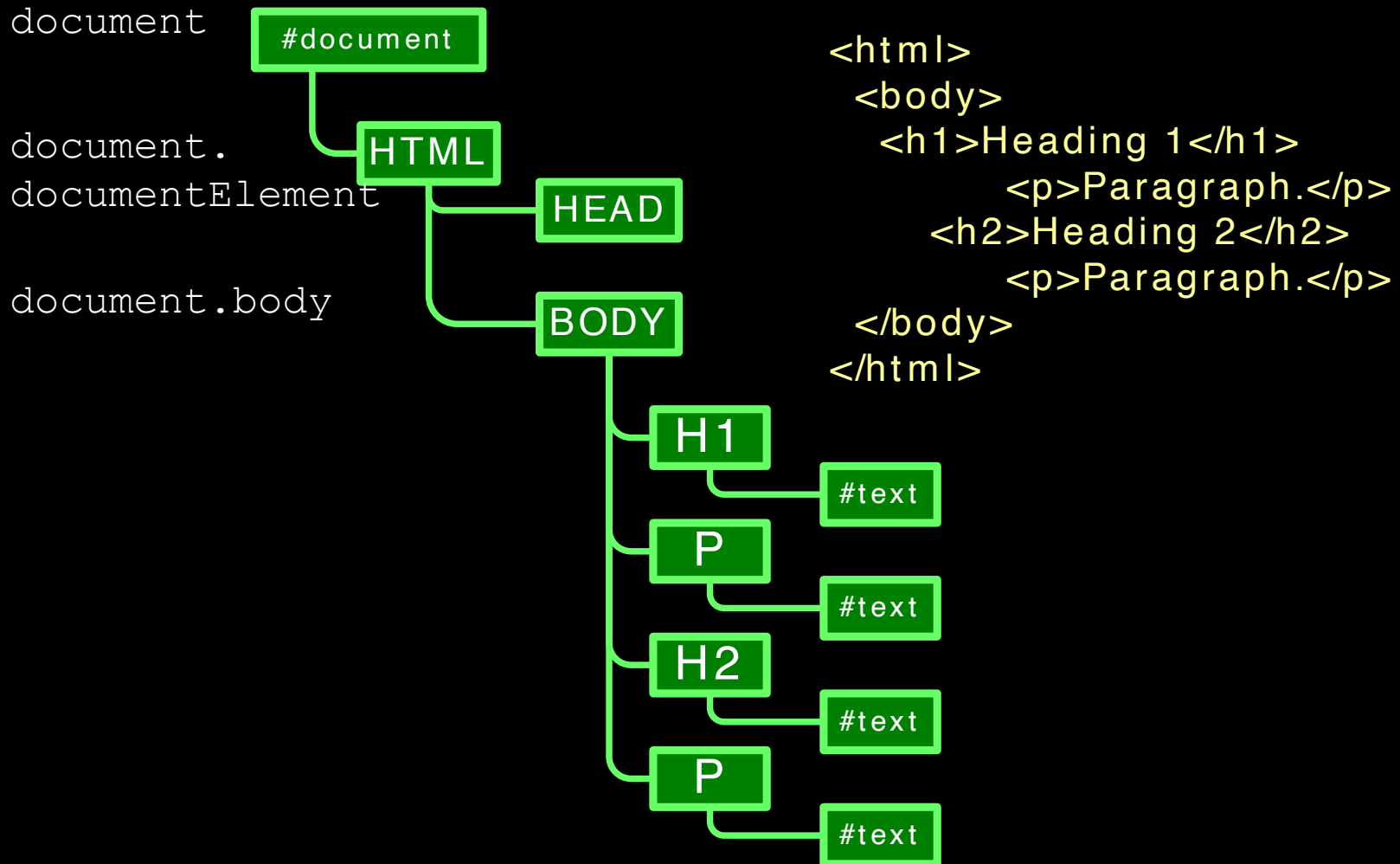
The Theory Of The DOM
(Douglas Crockford)



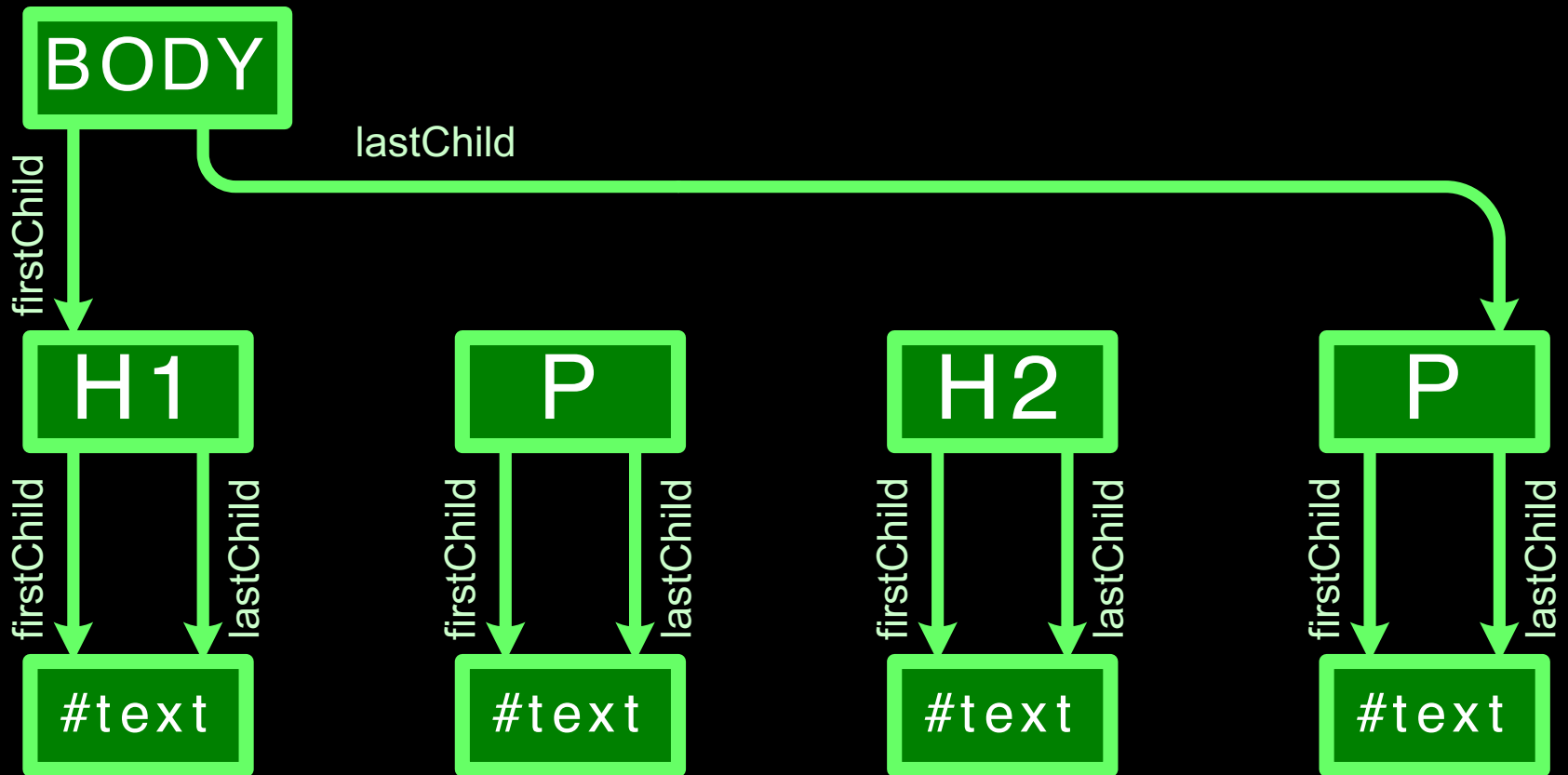
Scripted Browser



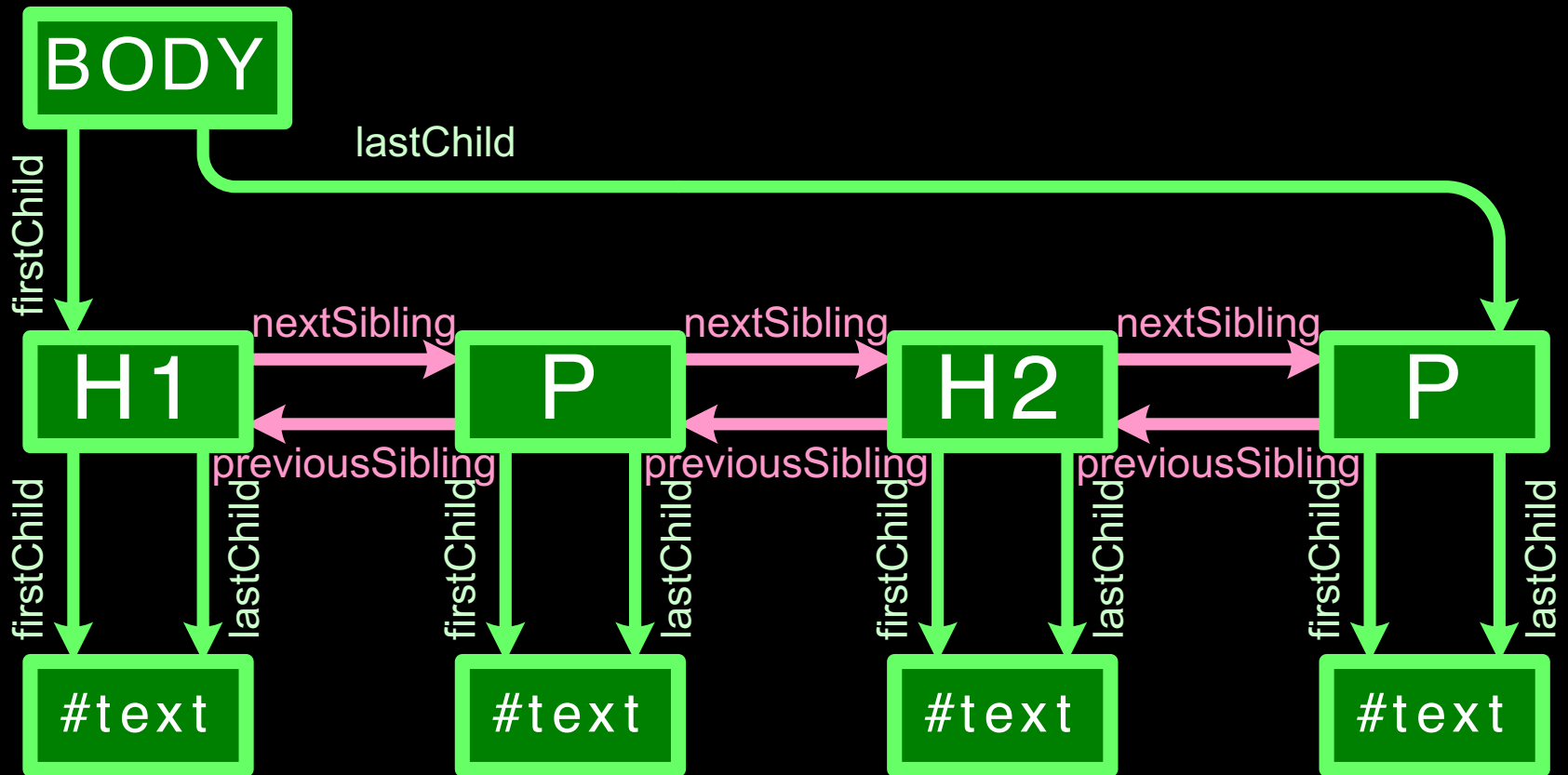
Document Tree Structure (aka DOM)



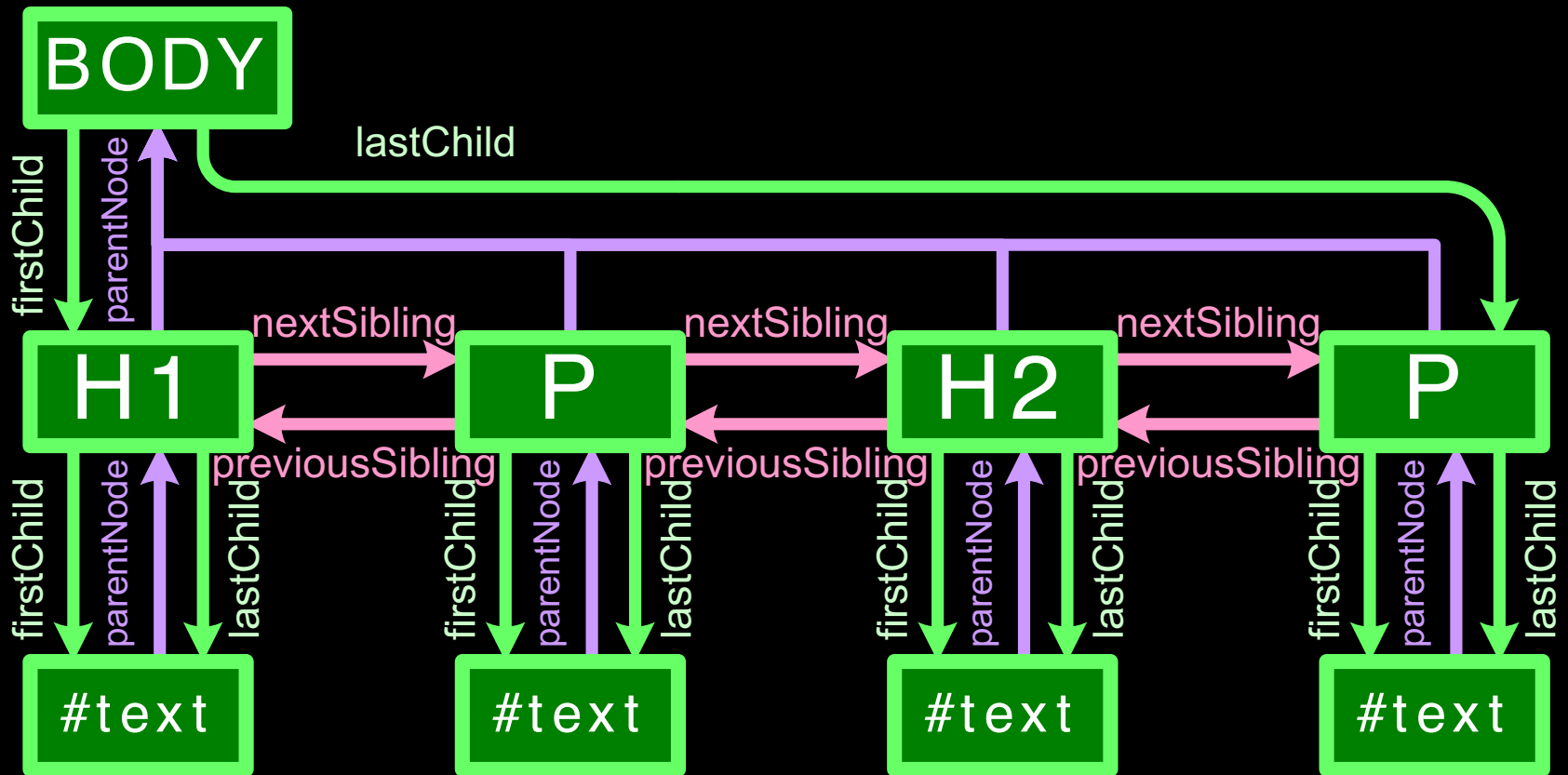
child, sibling, parent



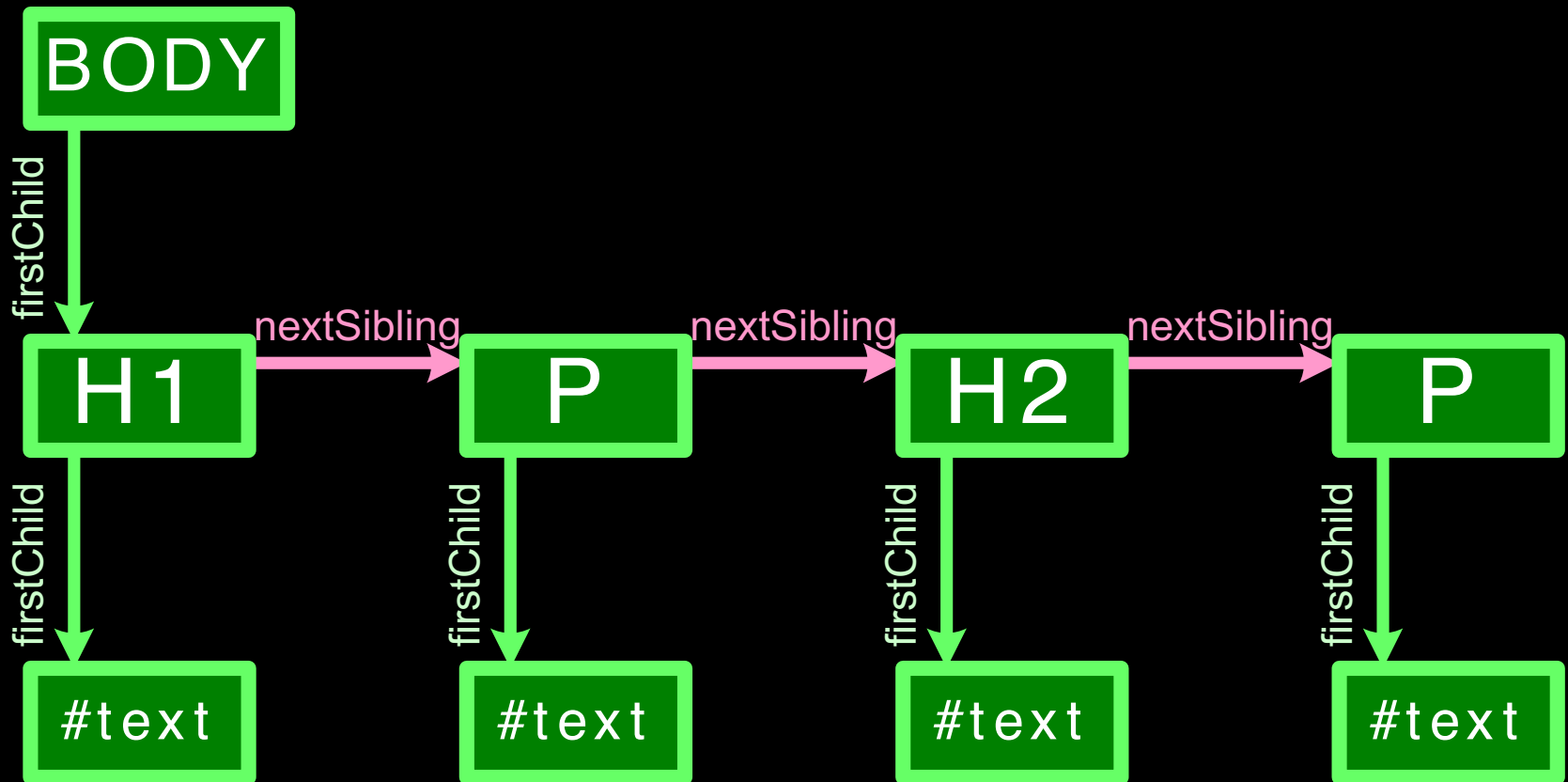
child sibling



child sibling parent



child sibling parent



Sample web page

The screenshot shows a web browser window with the title "Sample web page". The address bar displays the file path: `file:///localhost/Users/diarmuidoconnor/Notes/Common2/JavaScript/fundamentalsJS/browseevent...`. The page content includes a heading "Heading 1", a paragraph "Paragraph 1", a heading "Heading 2", and a paragraph "Paragraph 2". A code editor overlay displays the HTML code for the page, with line numbers 1 through 13. The code is as follows:

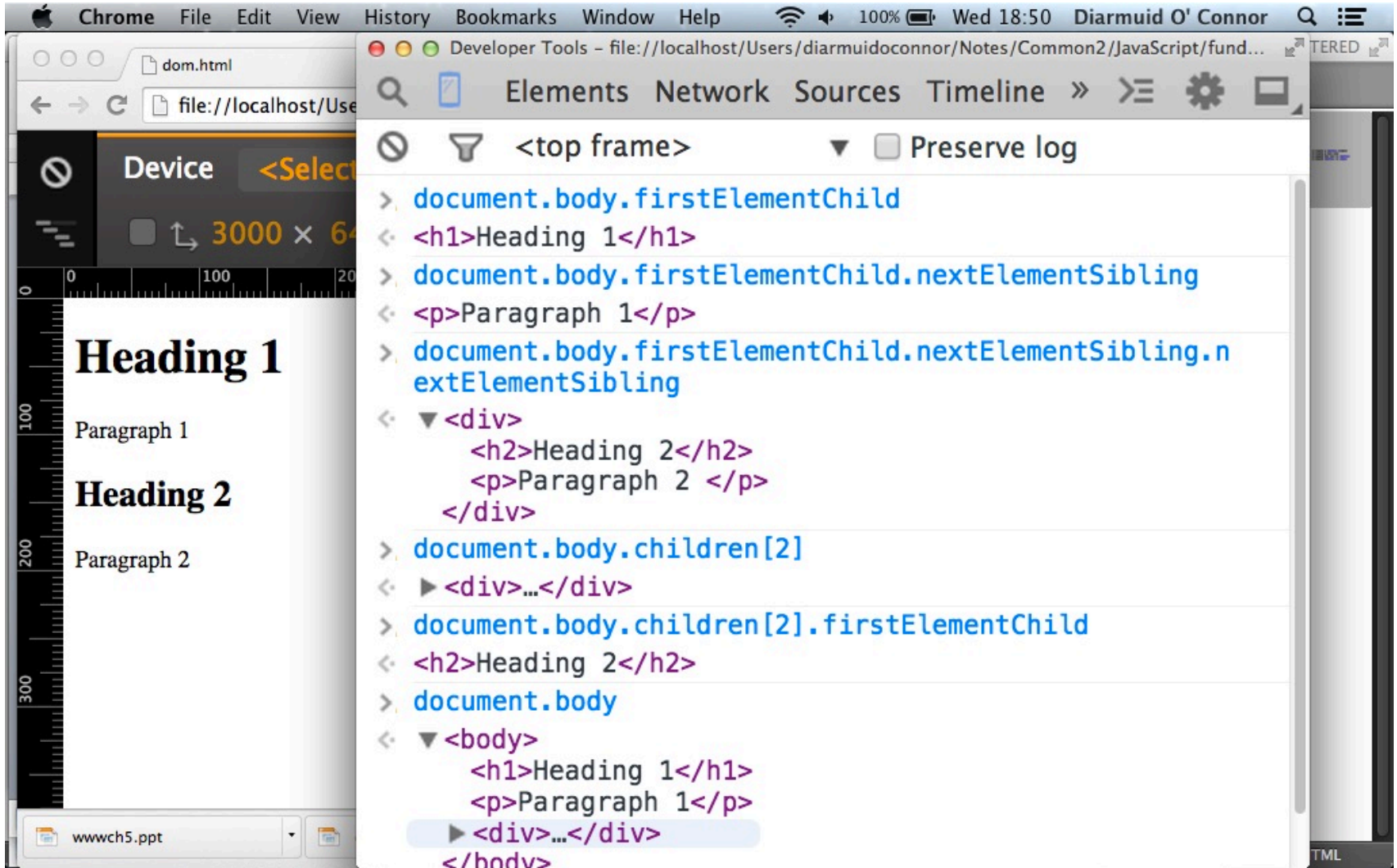
```
1 <!DOCTYPE html>
2 <html>
3 <head>
4
5 </head>
6 <body>
7   <h1>Heading 1</h1>
8   <p>Paragraph 1</p>
9   <div>
10     <h2>Heading 2</h2>
11     <p>Paragraph 2 </p>
12   </div>
13 </body>
```

Two red callout boxes provide context for the code structure:

- A callout box points to the `<div>` tag on line 9, stating: "A child of the body".
- A callout box points to the `<h2>` tag on line 10, stating: "A child of the div".

The browser's taskbar at the bottom shows several open files: `wwwch5.ppt`, `dom.ppt`, and `hackday.ppt`. A "Show All" button is also visible.

Navigating the DOM



Amending the DOM

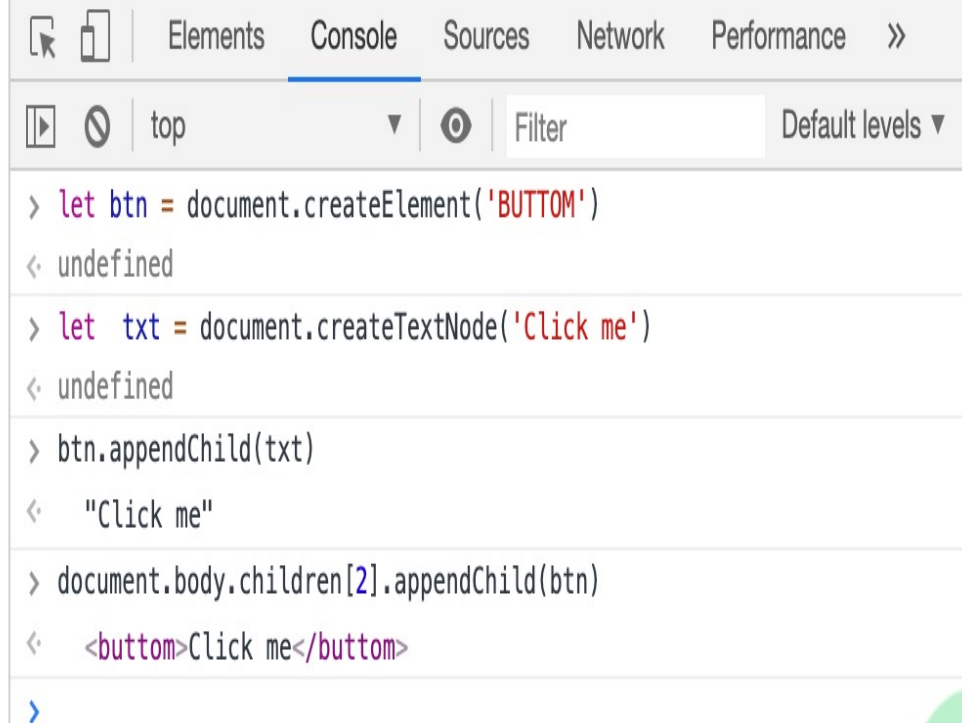
Heading 1

Paragraph 1

Heading 2

Paragraph 2

Click me



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays the following JavaScript code and its output:

```
> let btn = document.createElement('BUTTON')
< undefined
> let txt = document.createTextNode('Click me')
< undefined
> btn.appendChild(txt)
< "Click me"
> document.body.children[2].appendChild(btn)
< <button>Click me</button>
>
```

The code creates a button element, adds the text 'Click me' to it, and appends the button to the third child of the document body. The final output is the HTML string `<button>Click me</button>`.

Events.

The browser has an event-driven, single-threaded, asynchronous programming model.

- **Examples of events:**
 - A mouse click
 - A web page or an image loading
 - ‘Mousing’ over a hot spot on the web page
 - Selecting an input box in an HTML form.
 - Submitting an HTML form
 - A keystroke
- **We can assign an event handler (JS function) to a DOM element for a particular event.**
 - **Browser manages handler execution in an asynchronous manner.**

Event types.

- onabort - Loading of an image is interrupted
- onblur - An element loses focus
- onchange - The content of a field changes
- onclick - Mouse clicks an object
- ondblclick - Mouse double-clicks an object
- onerror - An error occurs when loading a document or an image
- onfocus - An element gets focus
- onkeydown - A keyboard **key is pressed**
- onkeypress - A keyboard key is pressed or held down
- onkeyup - A keyboard key is released
- onload - A page or an image is finished loading
- onmousedown - A mouse button is pressed
- onmousemove - The mouse is moved

Event types.

- onmouseout - The mouse is moved off an element
- onmouseover - The mouse is moved over an element
- onmouseup - A mouse button is released
- onreset - The reset button is clicked
- onresize - A window or frame is resized
- onselect - Text is selected
- onsubmit - The submit button is clicked
- onunload - The user exits the page

Event Handlers.

- Adding event handlers/listeners to a web page element. Two styles:

- Imperative:

- `dom_node.addEventListener(type, func, false)`

- Declarative

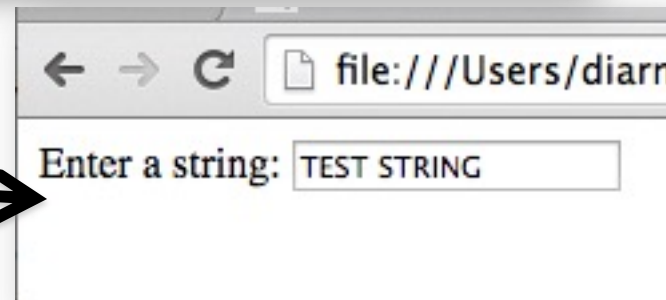
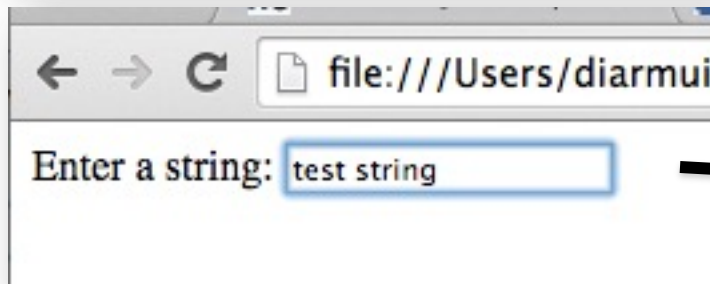
- `<tagName on{type}= 'func'>`

Event Handlers (Declarative style)

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <script>
5 function upperCase() {
6     var element = document.getElementById("string")
7     element.value = element.value.toUpperCase()
8 }
9 </script>
10 </head>
11 <body>
12     Enter a string: <input type="text" id="string" onchange="upperCase()">
13 </body>
```

Event handler / listener

Ref. 02_1_onchange.html



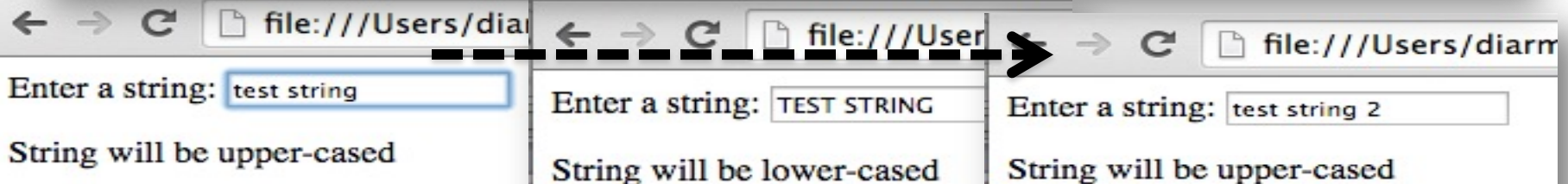

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4  <script>
5  function upperCase() {
6      var element = document.getElementById("string") ;
7      element.value = element.value.toUpperCase();
8      // Switch event handler
9      element.removeEventListener('change',upperCase );
10     element.addEventListener('change',lowerCase , false);
11     document.getElementsByTagName('p')[0].innerHTML =
12         'String will be lower-cased';
13 }
14
15 function lowerCase() {
16     var element = event.srcElement // event has global scope
17     element.removeEventListener('change', lowerCase)
18     element.addEventListener('change',upperCase , false)
19     element.value = element.value.toLowerCase()
20     document.getElementsByTagName('p')[0].innerHTML =
21         'String will be upper-cased'
22 }
23 </script>
24 </head>
25 <body>
26 Enter a string: <input type="text" id="string" onchange="upperCase()">
27 <p>String will be upper-cased</p>
28 </body>

```

Imperative
style

Ref. 02_2_onchange.html



DOM API → JQuery API.

- The DOM API is not developer-friendly.
- The JQuery JS library (Aug., 2006) improved the developer experience (DX) by:
 - Simplifying event binding and DOM manipulation.
 - Providing a common API across multiple browsers.
 - Supporting plug-in modules to extend functionality.
- JQuery is built on top of the DOM API.
- Disadvantage: 'Spaghetti' code; Poor maintainability

JQuery API → Single Page App Frameworks (SPA).

- E.g. React, Angular, Vue, etc
- Improved Code structure → Improved maintainability.
-
- Supported addressability

Summary

- The browser stores the ‘current’ web page as a tree of nodes (JS objects), called the DOM (Document Object Model).
- The browser is an event-driven programming environment.
 - Event handlers can be linked to nodes for specific events.
 - Result: A web page can be dynamic!!
- Browsers provide a native API to navigate the DOM - DOM API.
 - JQuery made working with DOM API more developer-friendly
 - SPA frameworks further improved developer experience

