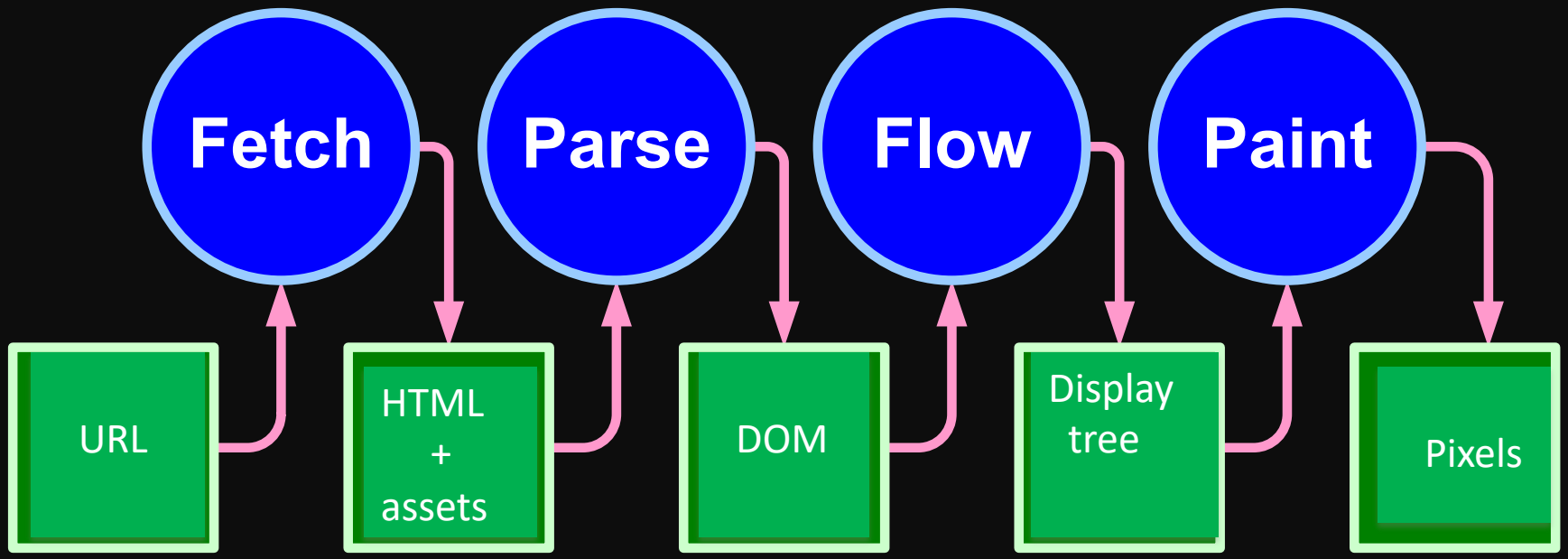
The Web Browser

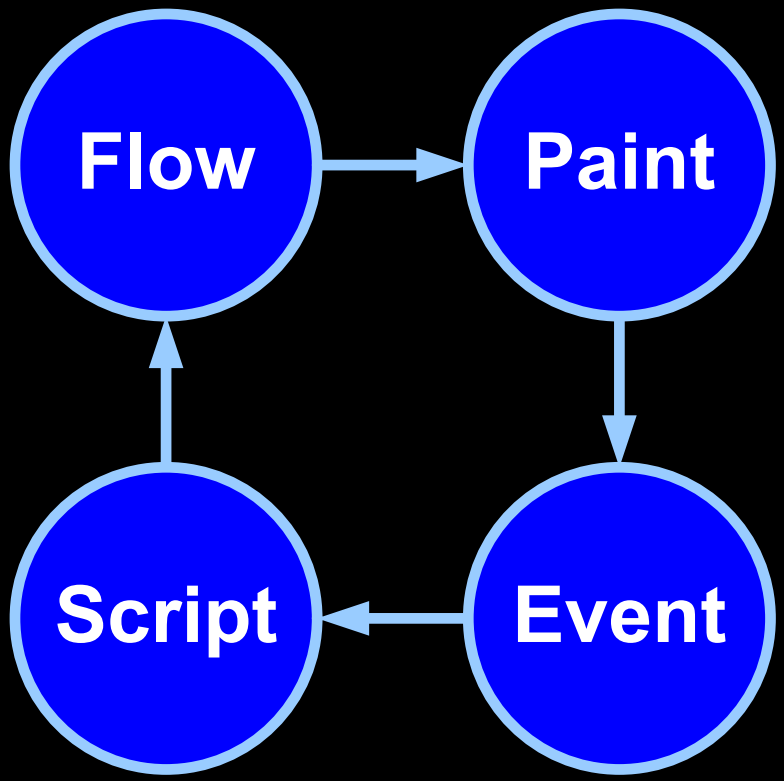
An event-driven environment

Browser

The Theory Of The DOM
(Douglas Crockford)

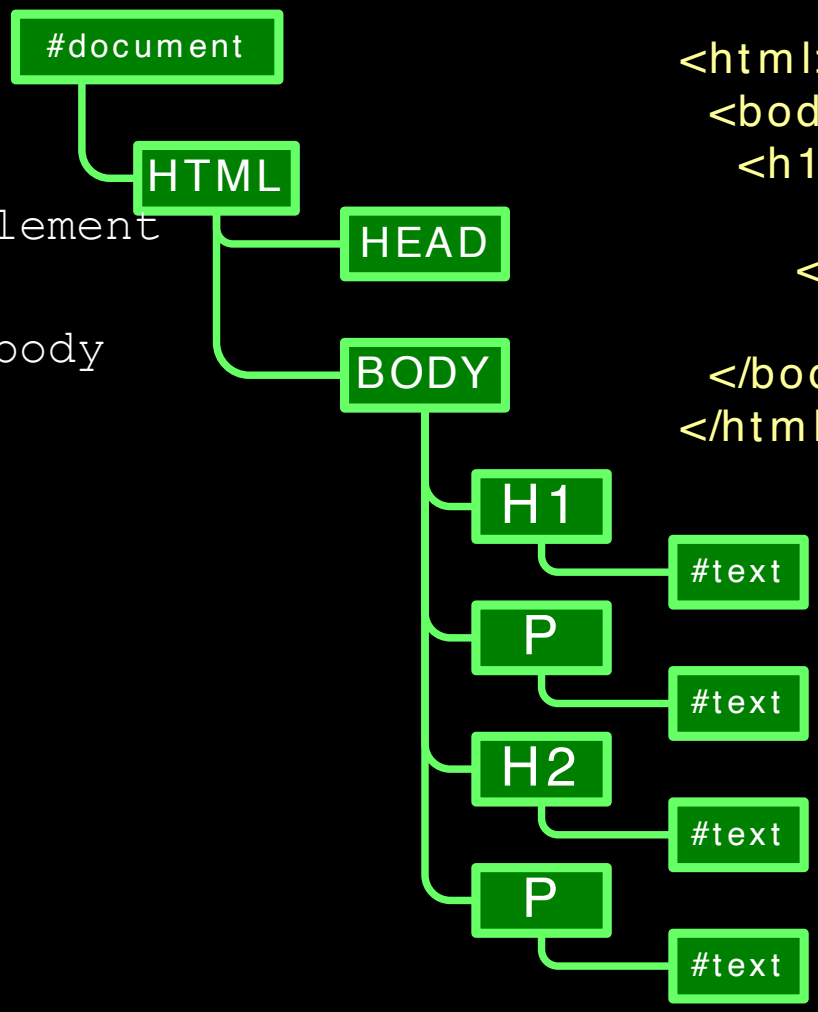


Scripted Browser



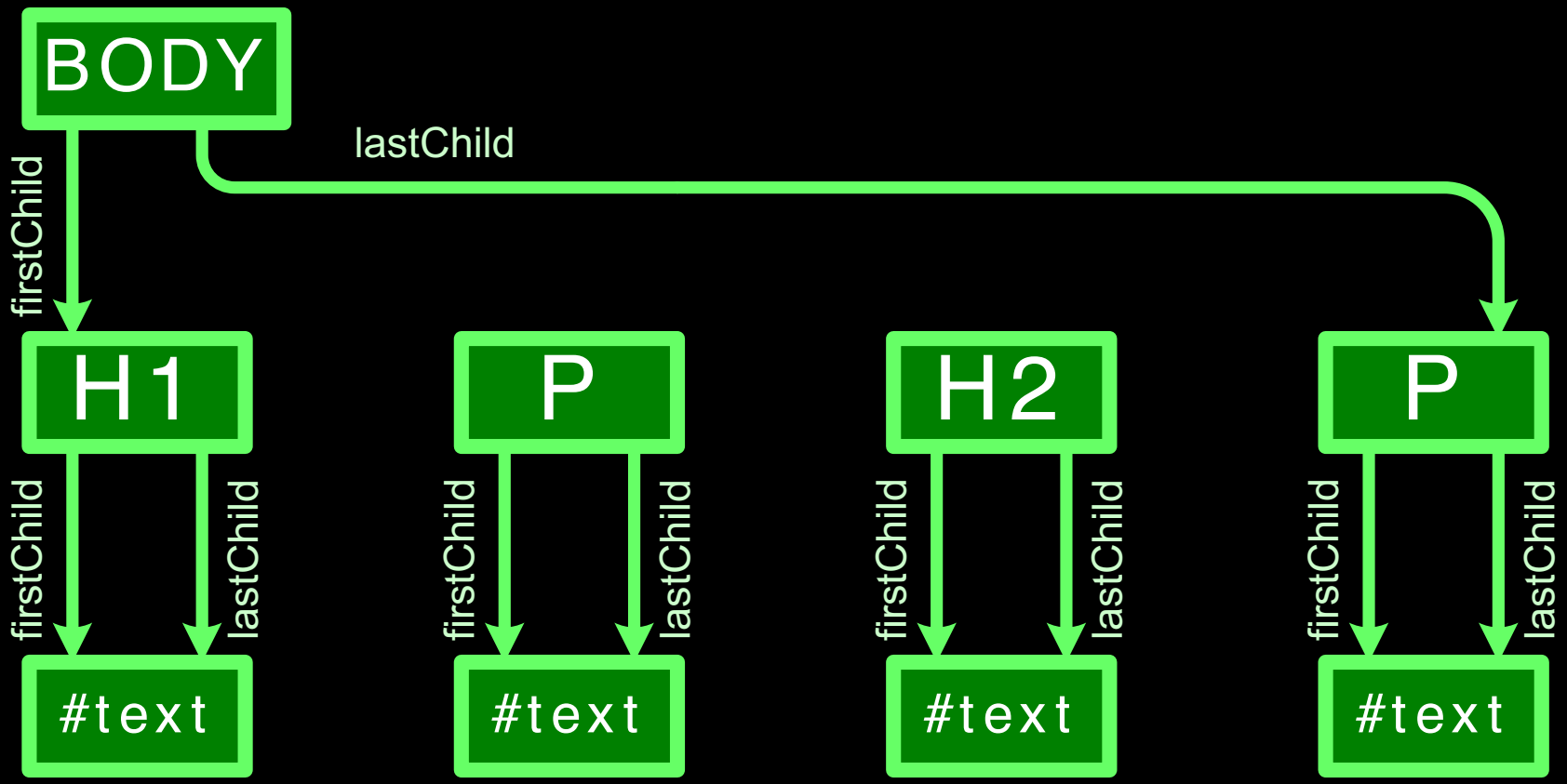
Document Tree Structure (aka DOM)

```
document

document.
documentElement

document.body
```

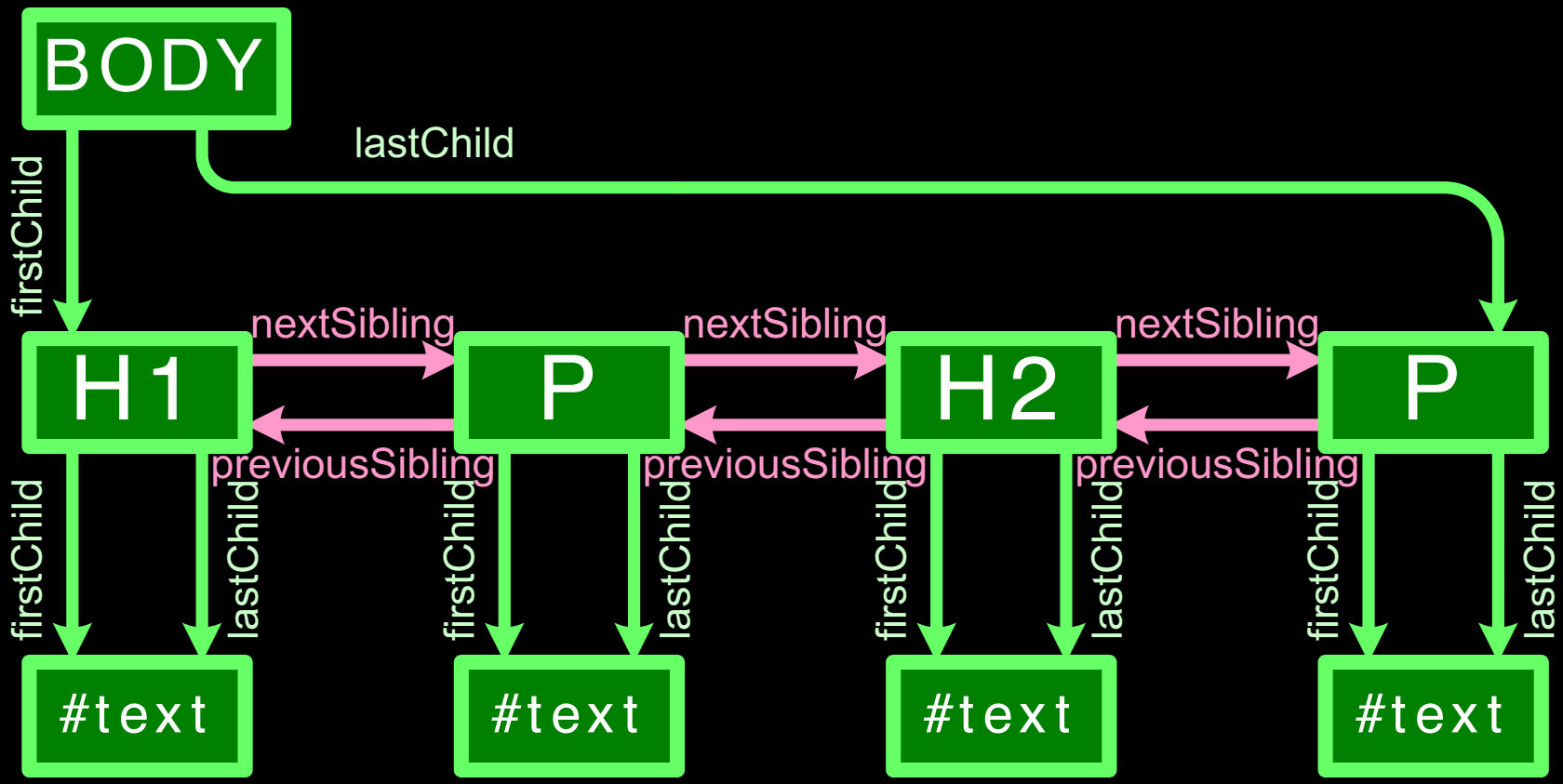



```
<html>
  <body>
    <h1>Heading 1</h1>
        <p>Paragraph.</p>
      <h2>Heading 2</h2>
        <p>Paragraph.</p>
  </body>
</html>
```

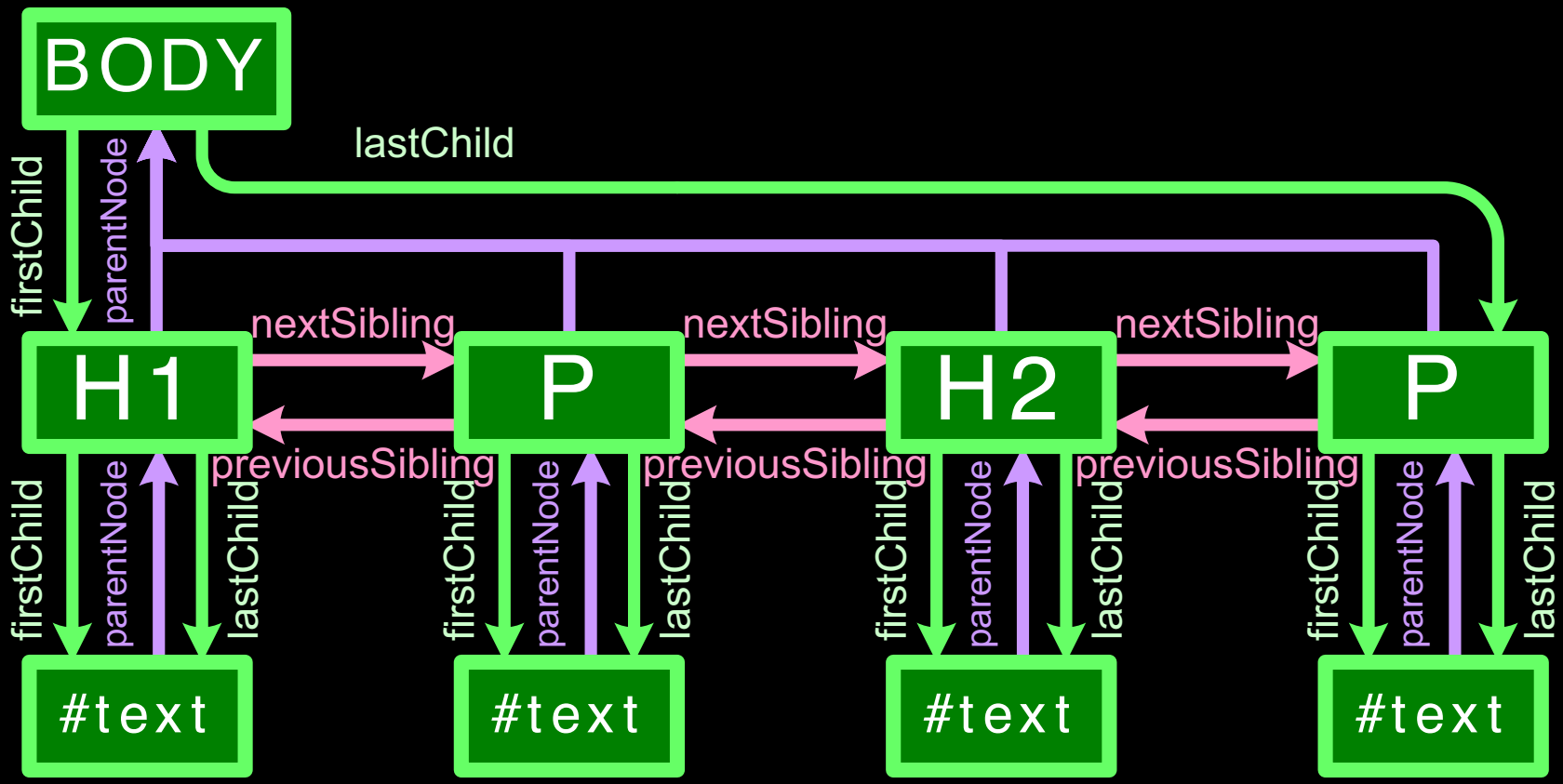
child, sibling, parent



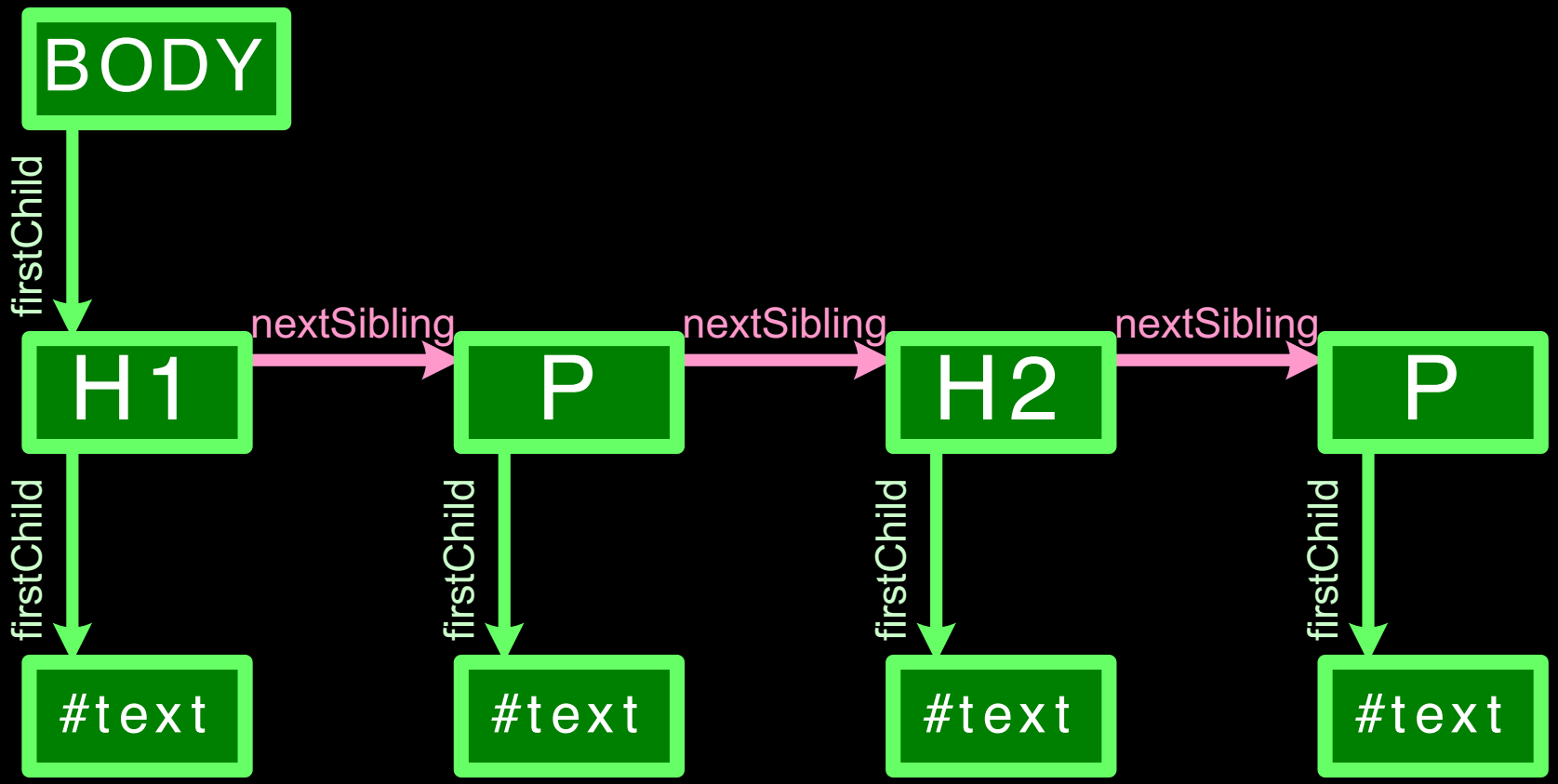
child sibling



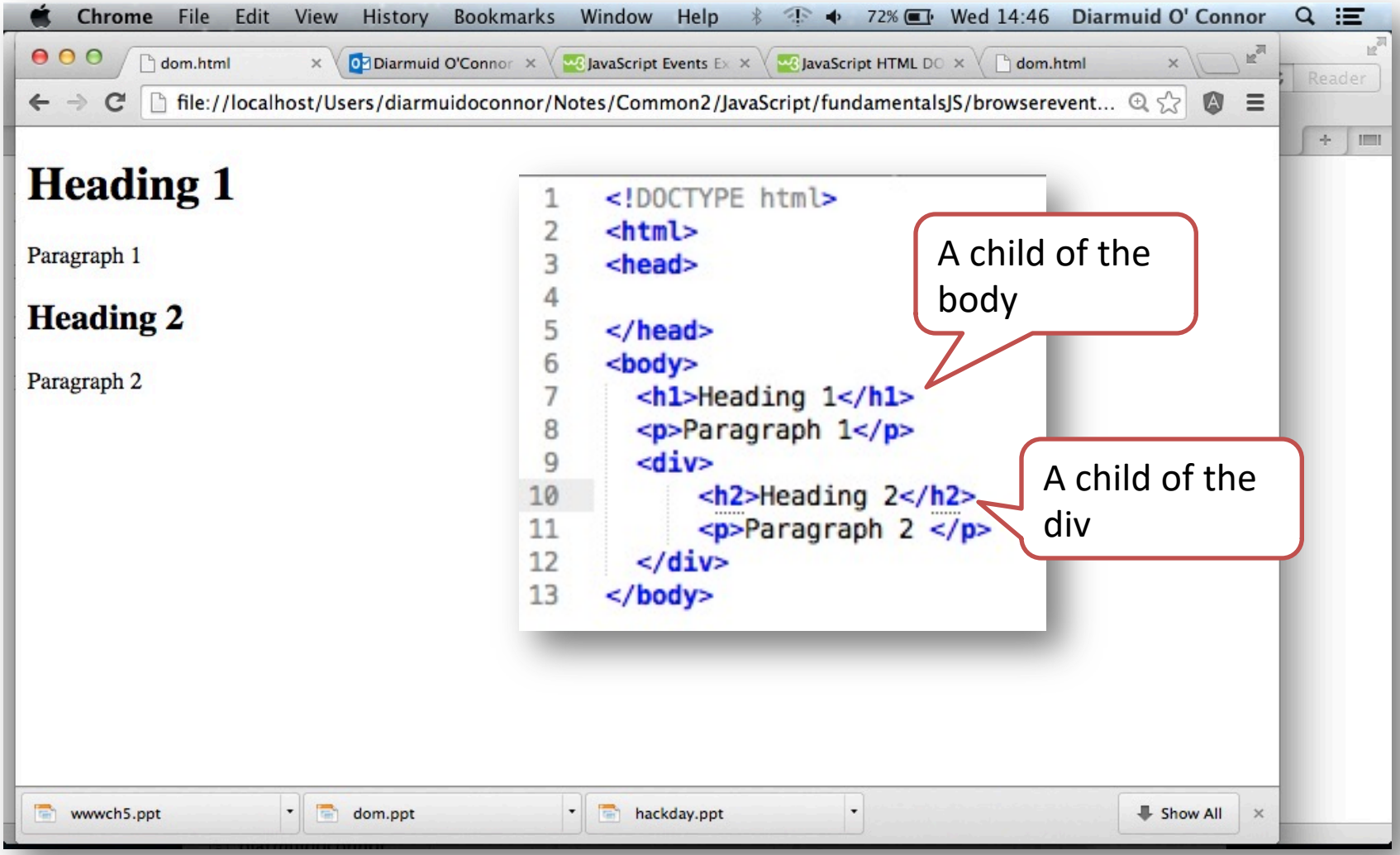
child sibling parent



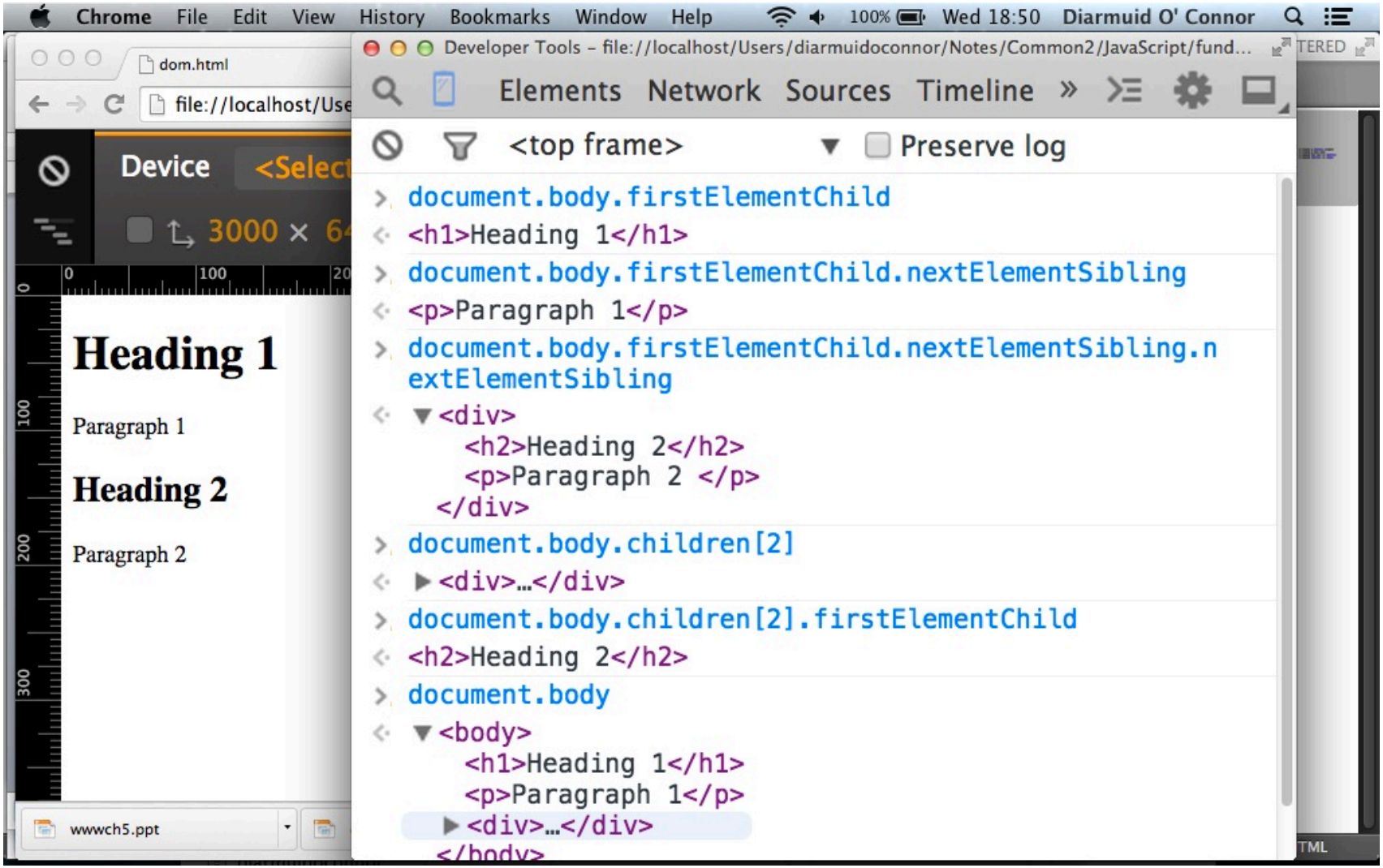
child sibling parent



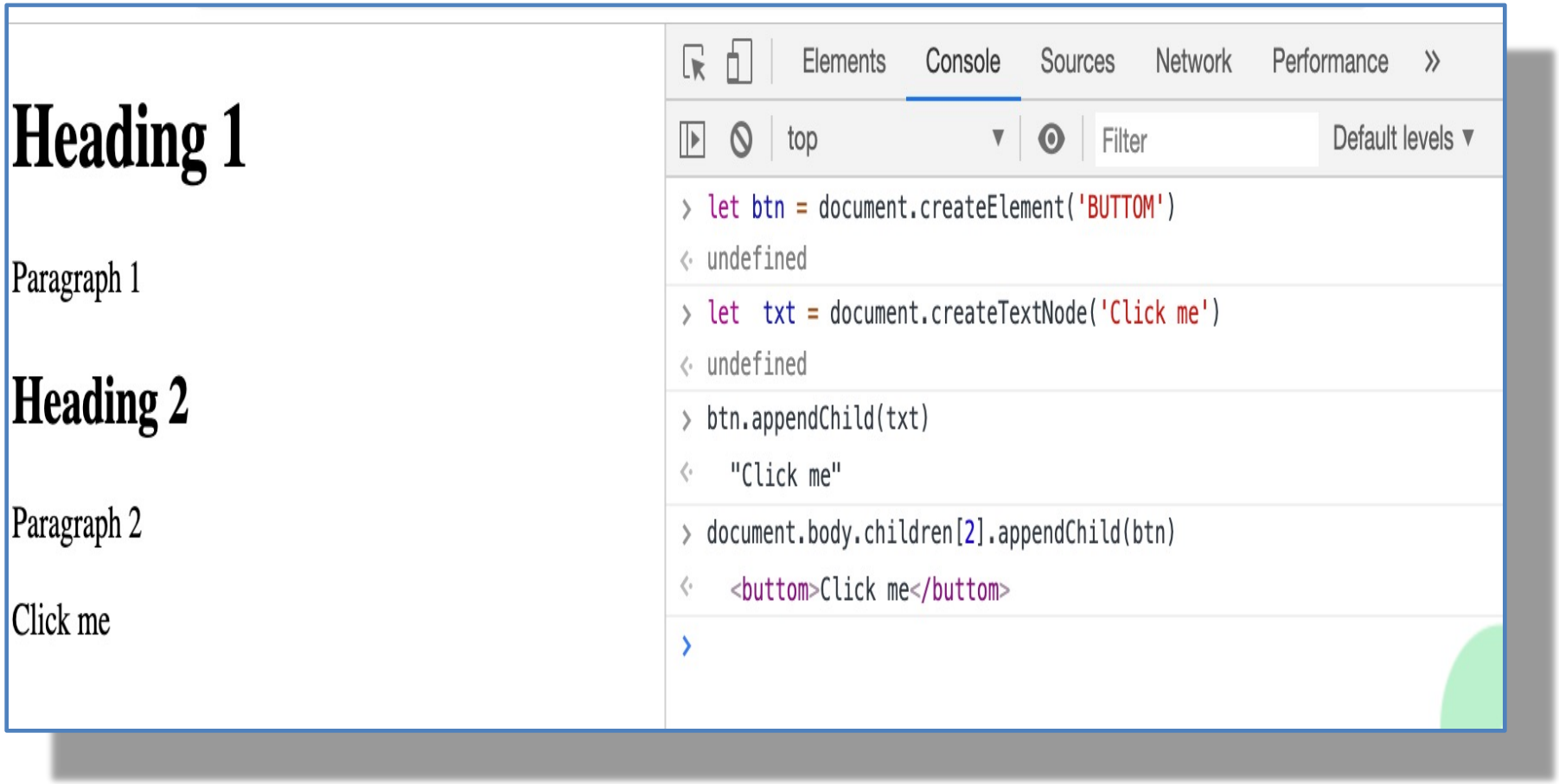
Sample web page



Navigating the DOM



Amending the DOM



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 asynchronuous 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 releas
- 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

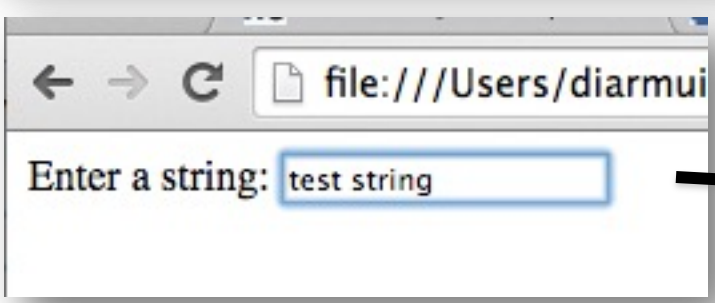
    <tagName on{type}=‘func’ …….>

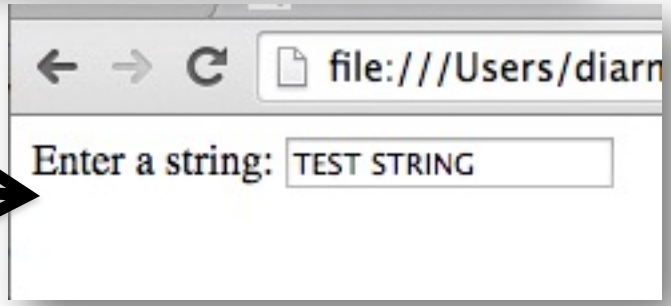
Event Handlers (Declarative style)

```
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

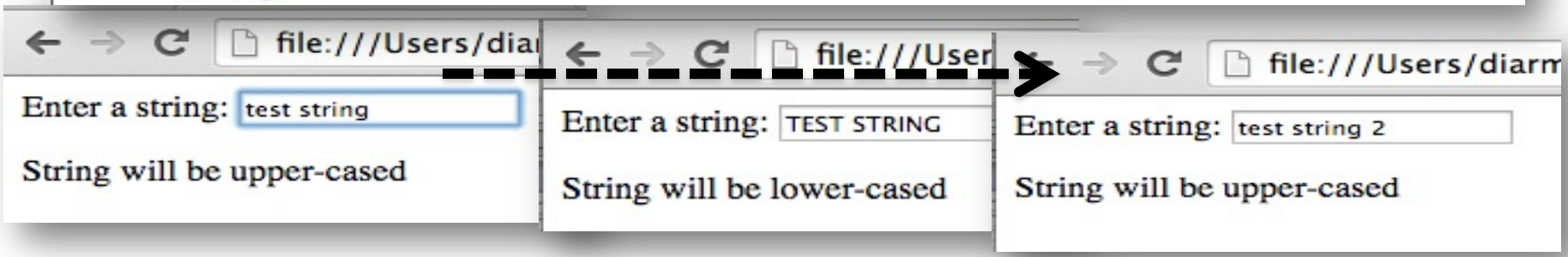




```html
<!DOCTYPE html>
<html>
<head>
<script>
function upperCase() {
    var element = document.getElementById("string") ;
    element.value = element.value.toUpperCase();
    // Switch event handler
    element.removeEventListener('change',upperCase );
    element.addEventListener('change',lowerCase , false);
    document.getElementsByTagName('p')[0].innerHTML =
                                'String will be lower-cased';
}

function lowerCase() {
    var element =  event.srcElement   // event has global scope
    element.removeEventListener('change', lowerCase)
    element.addEventListener('change',upperCase , false)
    element.value = element.value.toLowerCase()
    document.getElementsByTagName('p')[0].innerHTML =
                                'String will be upper-cased'
}
</script>
</head>
<body>
Enter a string: <input type="text" id="string" onchange="upperCase()">
<p>String will be upper-cased</p>
</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 → Simgle 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 inproved developer experience**