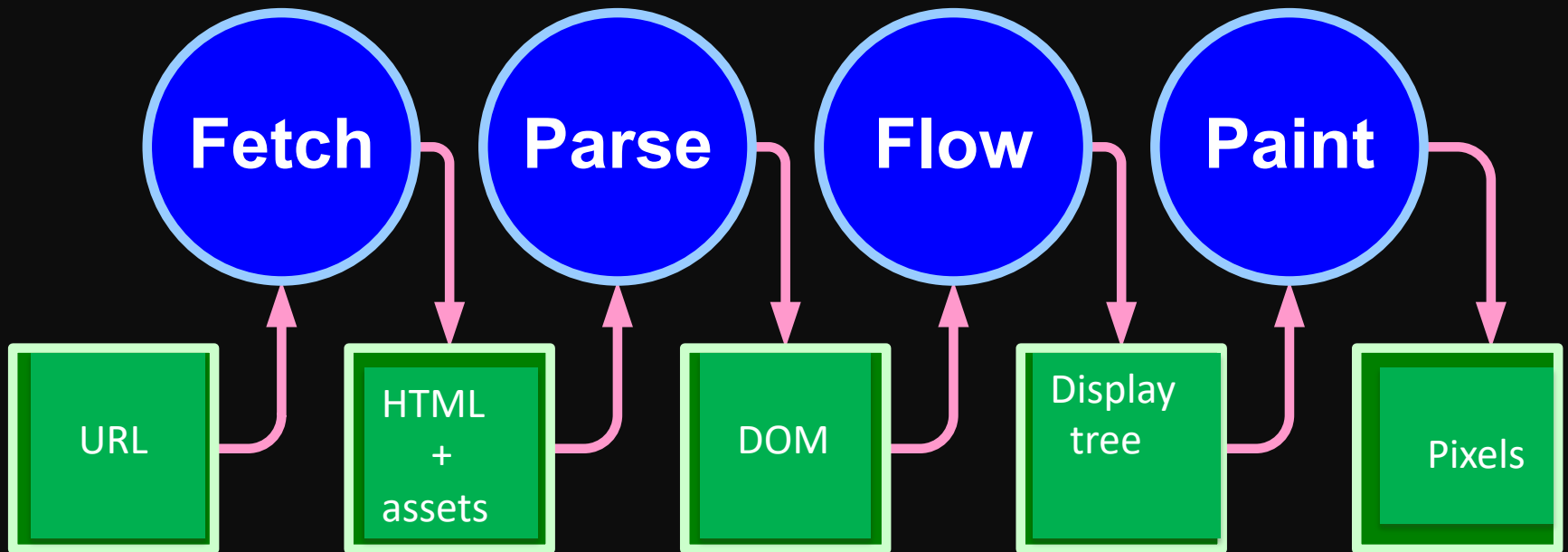


The Web Browser

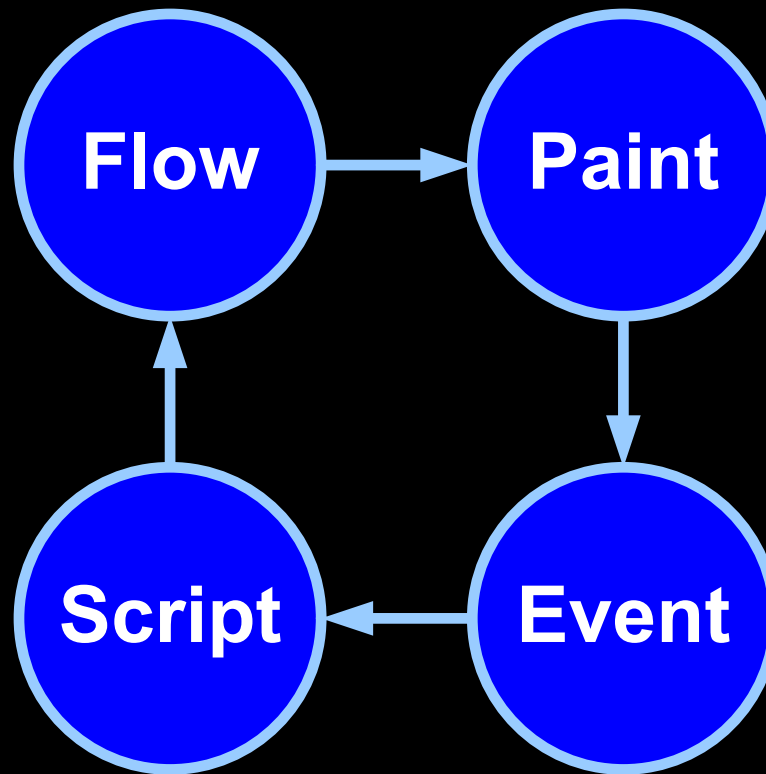
An event-driven environment

Browser

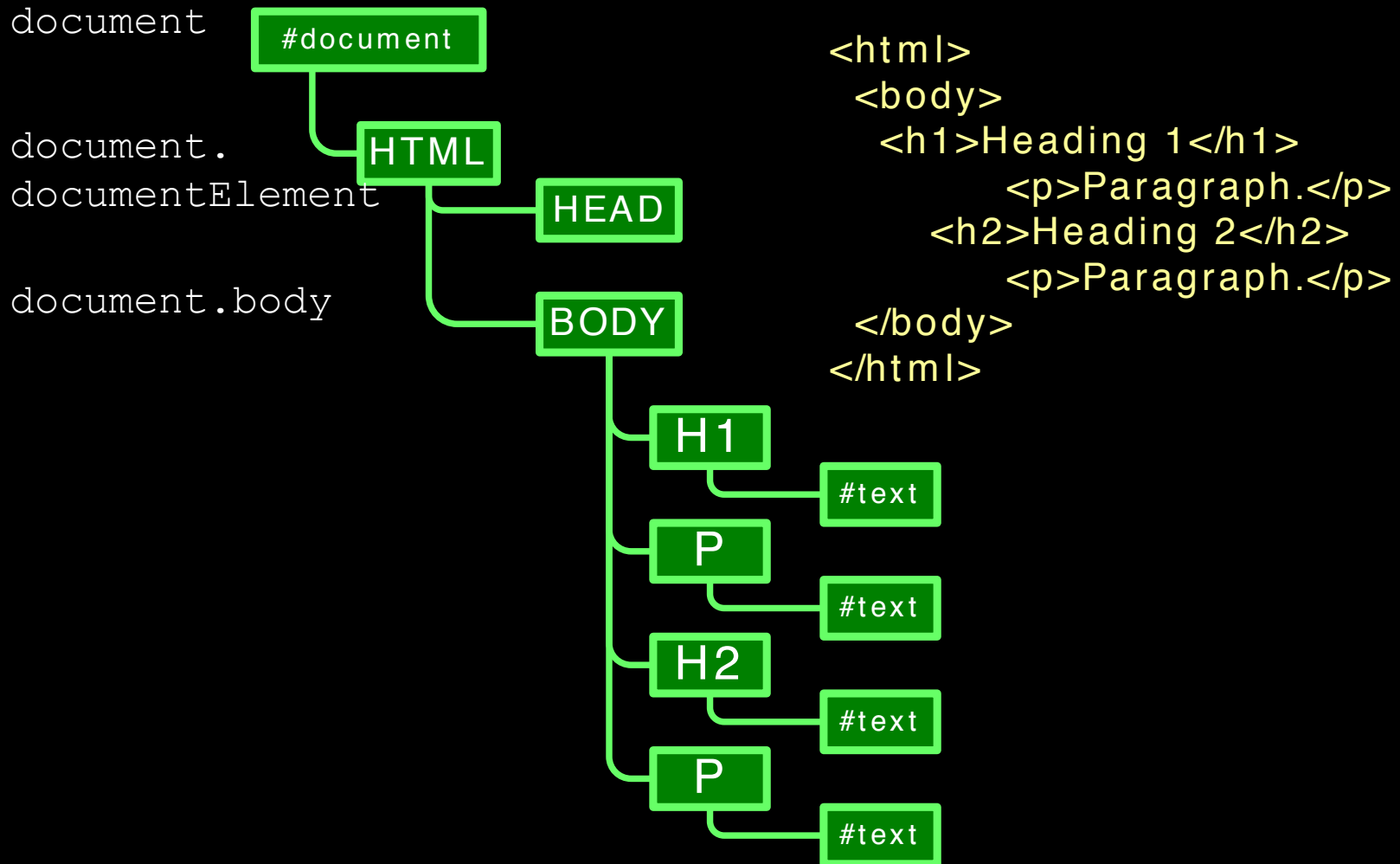
The Theory Of The DOM
(Douglas Crockford)



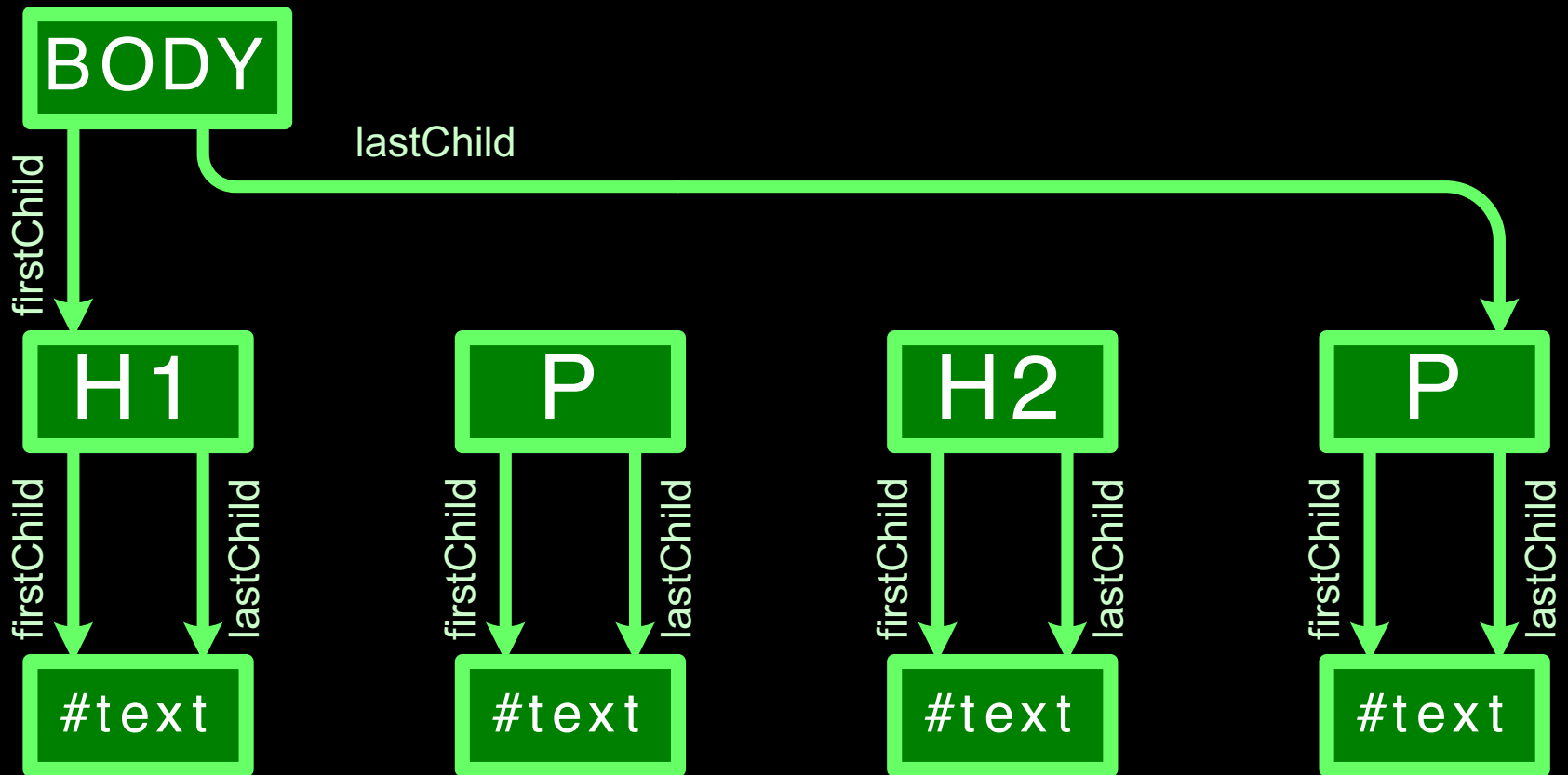
Scripted Browser



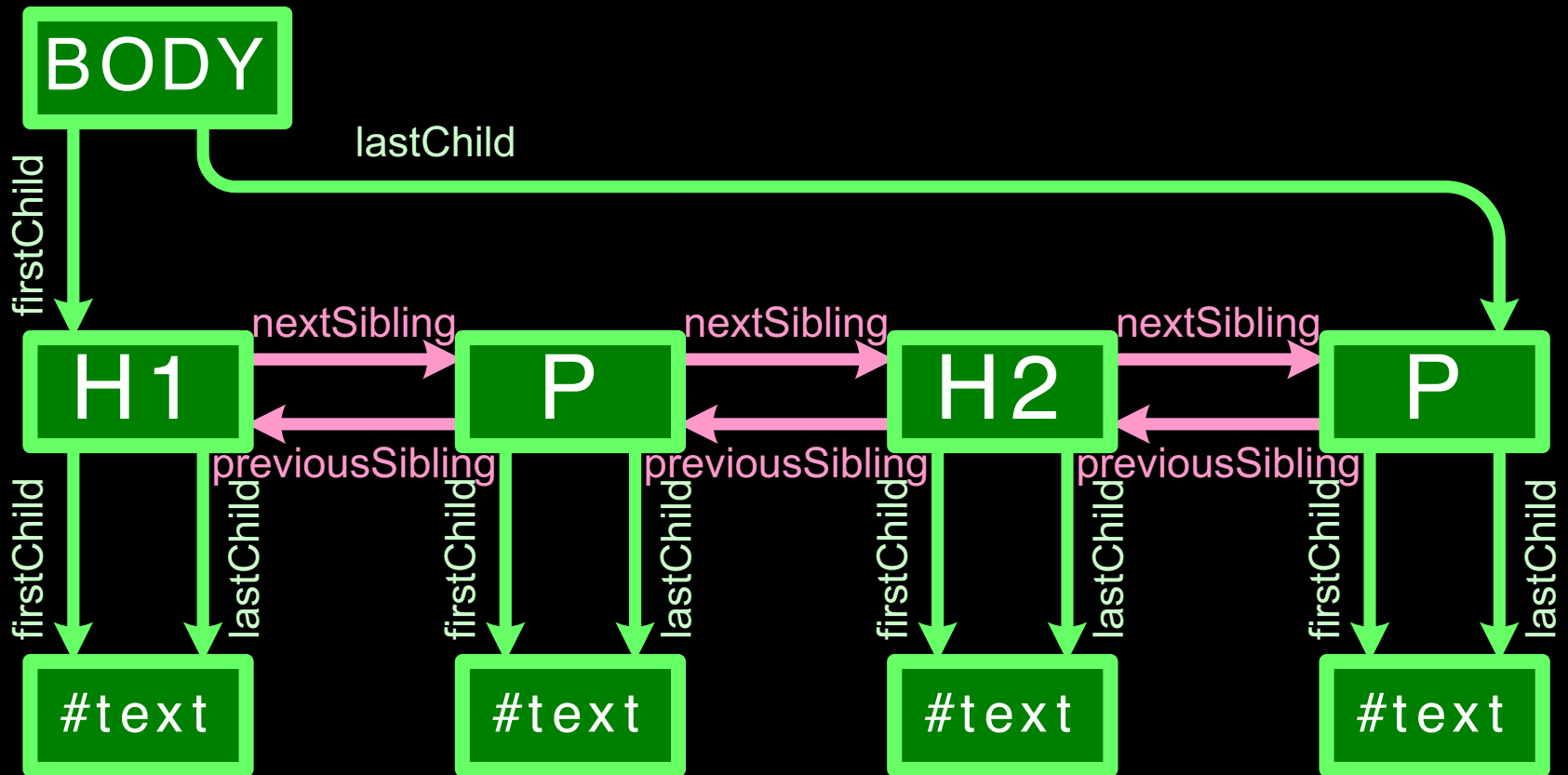
Document Tree Structure (aka DOM)



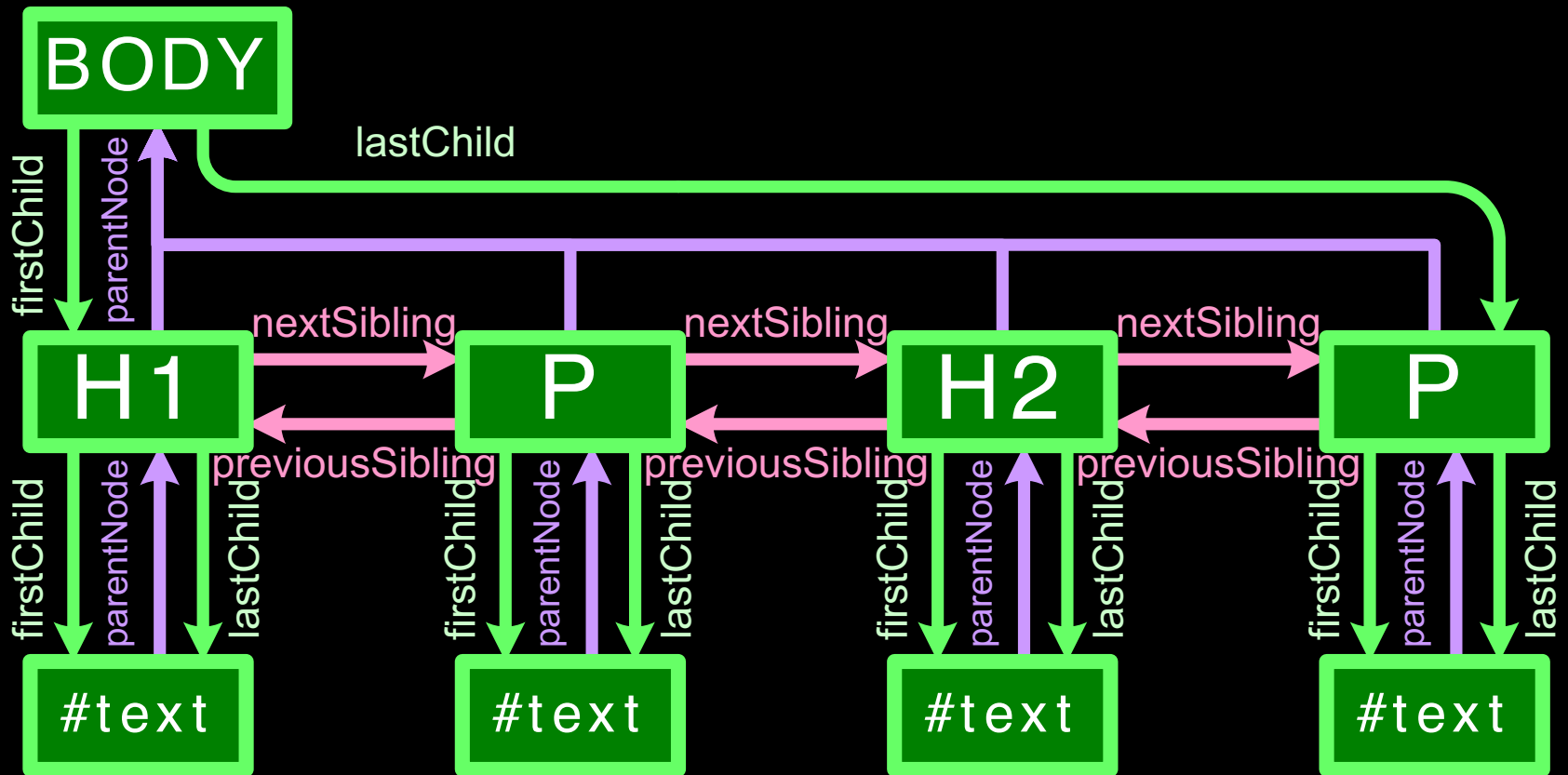
child, sibling, parent



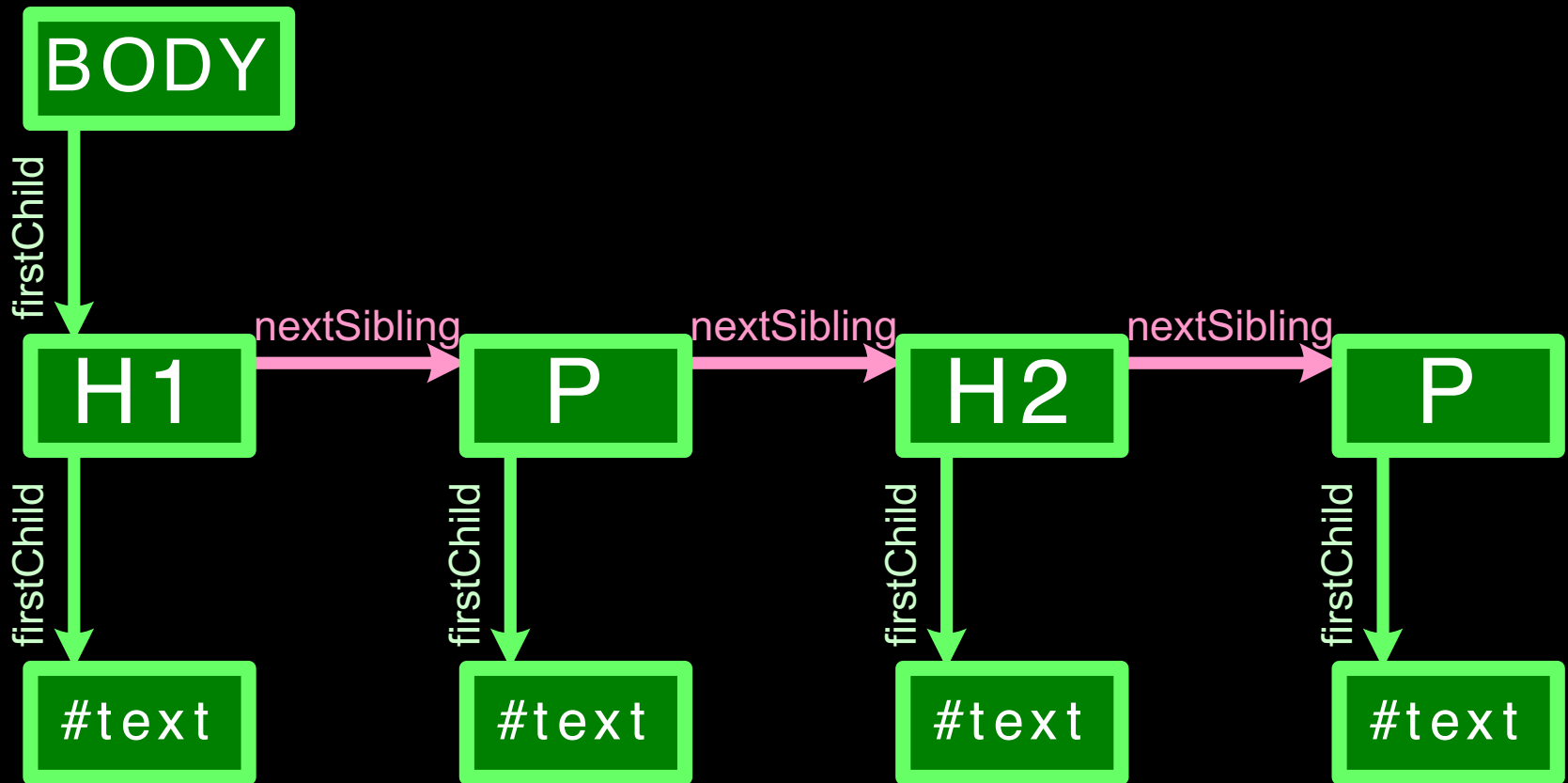
child sibling



child sibling parent



child sibling parent



Sample web page

The screenshot shows a Google Chrome browser window. The address bar displays a local file path: `file:///localhost/Users/diarmuidoconnor/Notes/Common2/JavaScript/fundamentalsJS/browseevent...`. The browser has several tabs open, including `dom.html`, `Diarmuid O'Connor`, `JavaScript Events`, `JavaScript HTML DO`, and another `dom.html`. The main content area of the browser displays a web page with the following structure:

- Heading 1**
- Paragraph 1
- Heading 2**
- Paragraph 2

Overlaid on the right side of the browser window is a code editor showing the HTML source code of the page. The code is as follows:

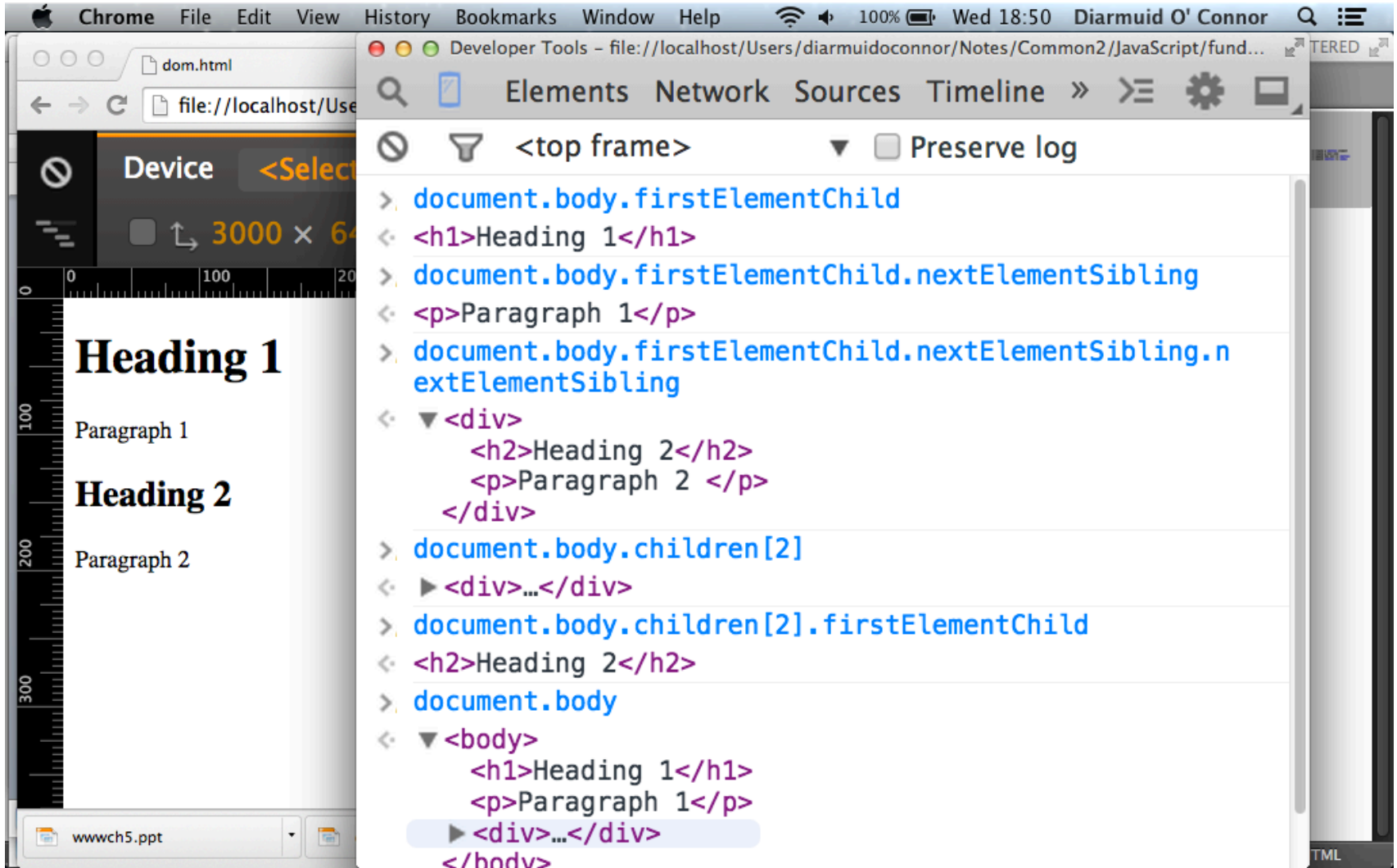
```
1  <!DOCTYPE html>
2  <html>
3  <head>
4
5  </head>
6  <body>
7      <h1>Heading 1</h1>
8      <p>Paragraph 1</p>
9      <div>
10         <h2>Heading 2</h2>
11         <p>Paragraph 2 </p>
12     </div>
13 </body>
```

Two red speech bubble annotations are present:

- A speech bubble pointing to the `<h1>Heading 1</h1>` tag on line 7, containing the text "A child of the body".
- A speech bubble pointing to the `<h2>Heading 2</h2>` tag on line 10, containing the text "A child of the div".

At the bottom of the browser window, there is a taskbar with three application icons labeled `wwwch5.ppt`, `dom.ppt`, and `hackday.ppt`, along with a "Show All" button.

Navigating the DOM



Amending the DOM

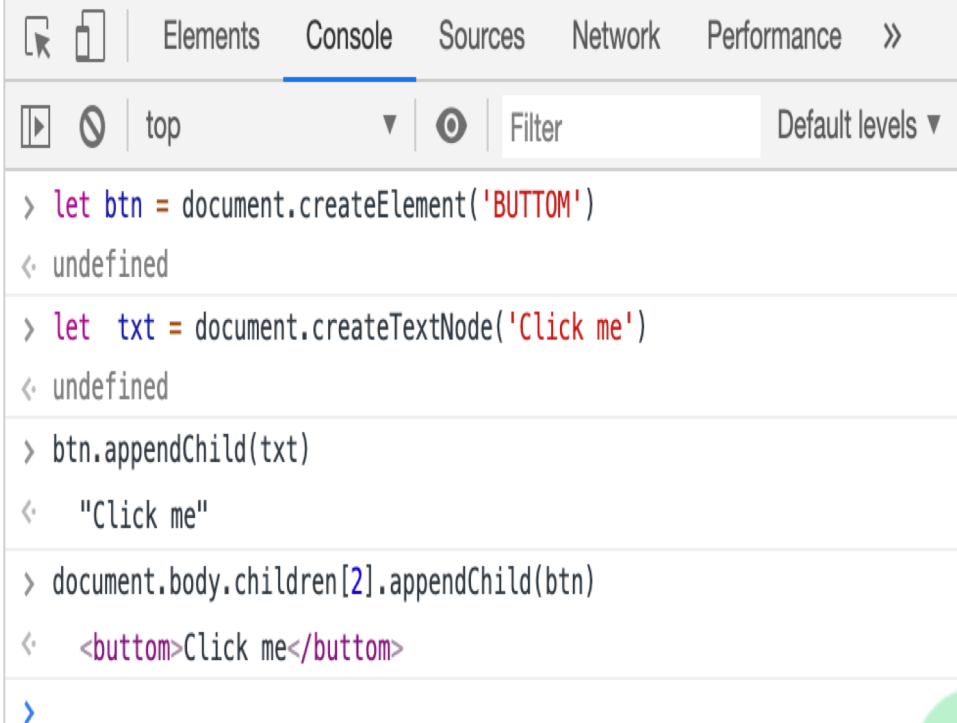
Heading 1

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 interface includes tabs for Elements, Console, Sources, Network, and Performance. The Console tab shows a filter input and a 'Default levels' dropdown. The output of the final command is a string representation of the created button element: `<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 event handlers (JS function) to a DOM element.**
 - Browser manages handler execution in 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

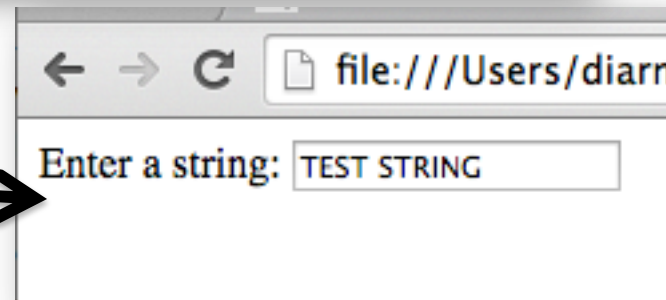
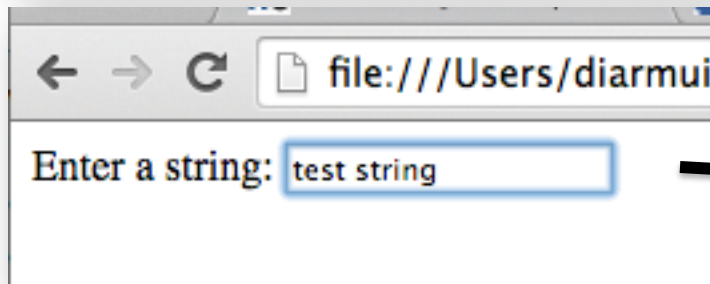
- `<tagName on{type}= 'func'>`

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



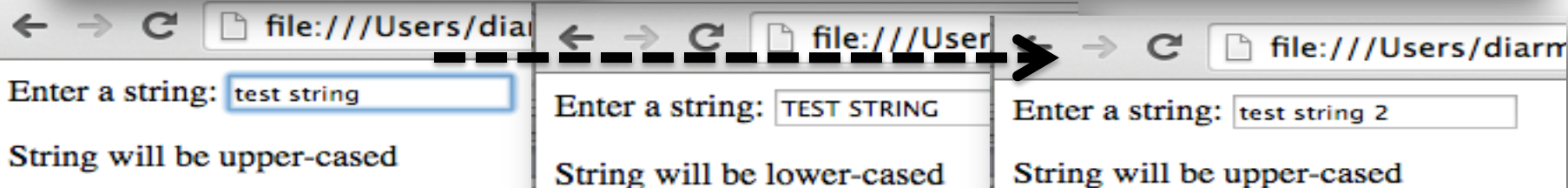

```

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.

JQuery.

- JQuery promotes an imperative programming model.

Ref. 03_1_jq-change.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <script src="./jquery.min.js"></script>
5 <script>
6 $(document).ready(function(){
7     $("#string").change(function(){
8         var input = $(this).val().toUpperCase()
9         $(this).val(input)
10    })
11 })
12 </script>
13 </head>
14 <body>
15     Enter a string: <input type="text" id="string">
16 </body>
17
```

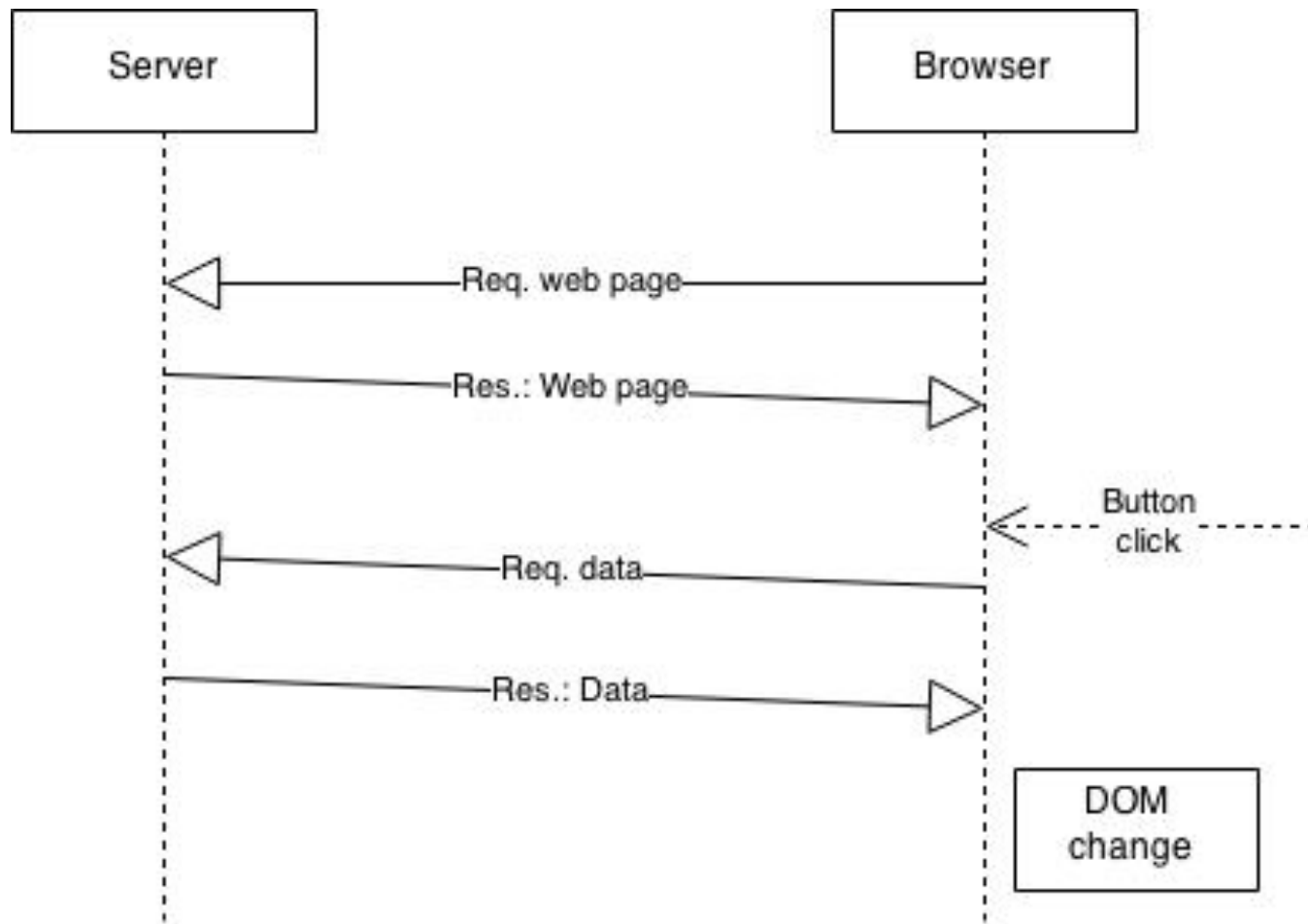
Helpful this binding

JQuery.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <script src="jquery.min.js"></script>
5  <script>
6  $(document).ready(function(){
7      $("#lower").hide();
8      $("#string").change(upperCase);
9  })
10
11  function upperCase() {
12      $(this).val($(this).val().toUpperCase());
13      $(this).change(lowerCase);
14      $("#lower").show();
15      $("#upper").hide();
16  }
17
18  function lowerCase() {
19      $(this).val($(this).val().toLowerCase());
20      $(this).change(upperCase);
21      $("#upper").show();
22      $("#lower").hide();
23  }
24 </script>
25 </head>
26 <body>
27     Enter a string: <input type="text" id="string">
28     <p id="upper">String will be upper-cased</p>
29     <p id="lower">String will be lower-cased</p>
30 </body>
```

Ref. 03_2_jq-change.html

Simple AJAX example (using JQuery) (1/2)



Simple AJAX example (using JQuery) (2/2)

- \$.get(URL,callback) – Send HTTP request to URL; Execute callback function when response arrives.

Ref. 04_ajax-jquery.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.
min.js"></script>
5  <script>
6  $(document).ready(function(){
7      $("button").click(function(){
8          // AJAX request
9          $.get("http://localhost:8080/sample.txt",function(data,status){
10             $('h2').text(data)
11         })
12     })
13 })
14 </script>
15 </head>
16 <body>
17     <h2>Let jQuery AJAX Change This Text</h2></div>
18     <button>Get External Content</button>
19 </body>
20 </html>
```

Summary

- The browser stores the ‘current’ web page as a tree of nodes (JS objects)
- An API is available to navigate the tree.
 - DOM API (Default) ; JQuery (Developer-friendly)
- The browser provides an event-driven environment.
- Event handlers can be linked to nodes for specific events.
 - Result: A web page can be dynamic!!

