Web Technology 2015

Lecture 5. Client- and server-side programming: JavaScript & PHP

Staas de Jong

Notes beforehand...

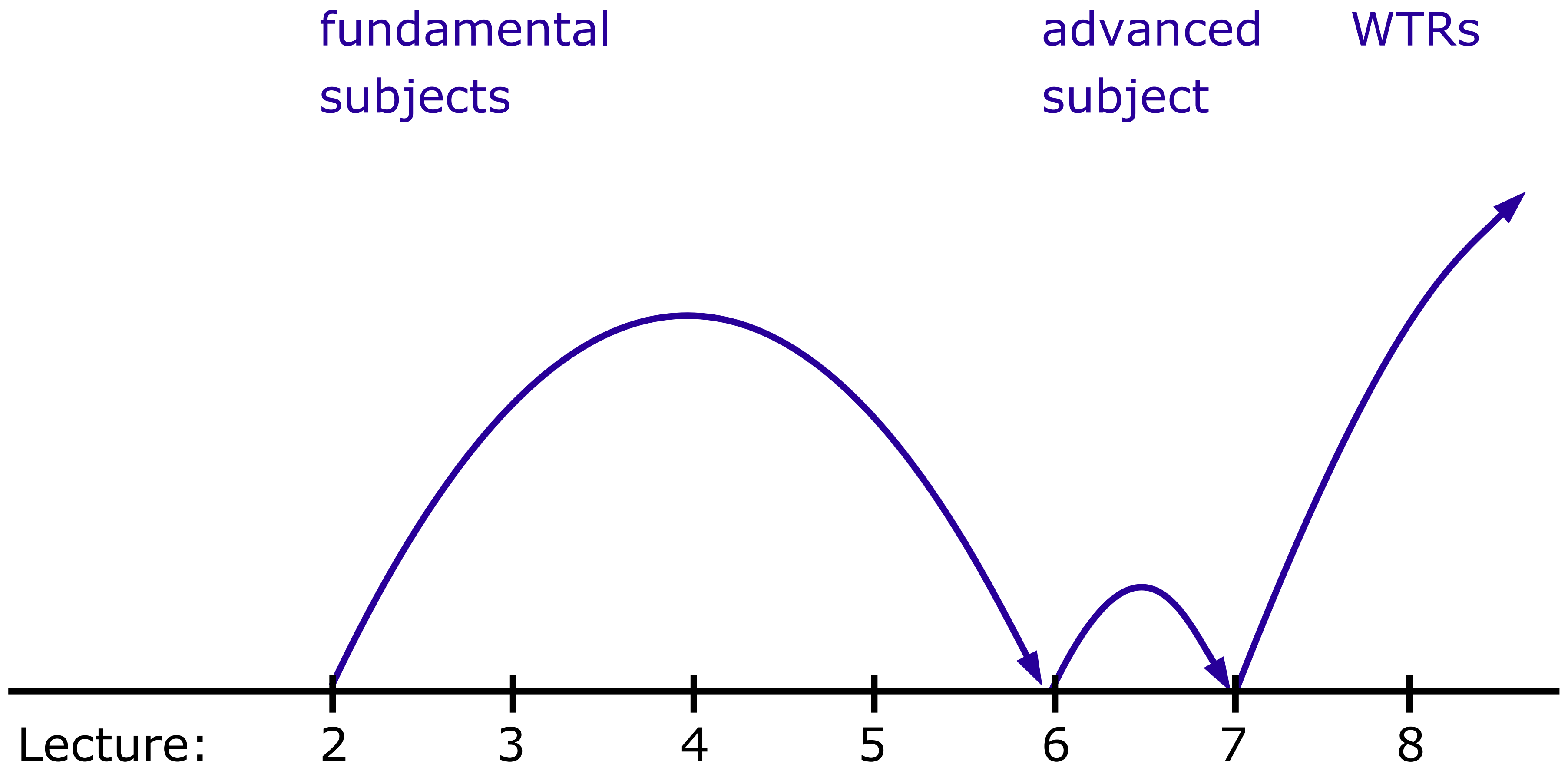
- You can see the TCP/IP layers in action using:
  - Wireshark:
    - Versatile packet sniffer.
    - Runs on Mac OS X, Linux, and Windows.
  - Telnet:
    - Application Layer protocol for *"bidirectional interactive text-oriented communications"*.
    - Also known as a *"virtual terminal"*; exists since 1973.
    - Available even on e.g. low-end Android smartphones.
    - Default: via TCP port 23.

Notes beforehand...

⇒ For example:

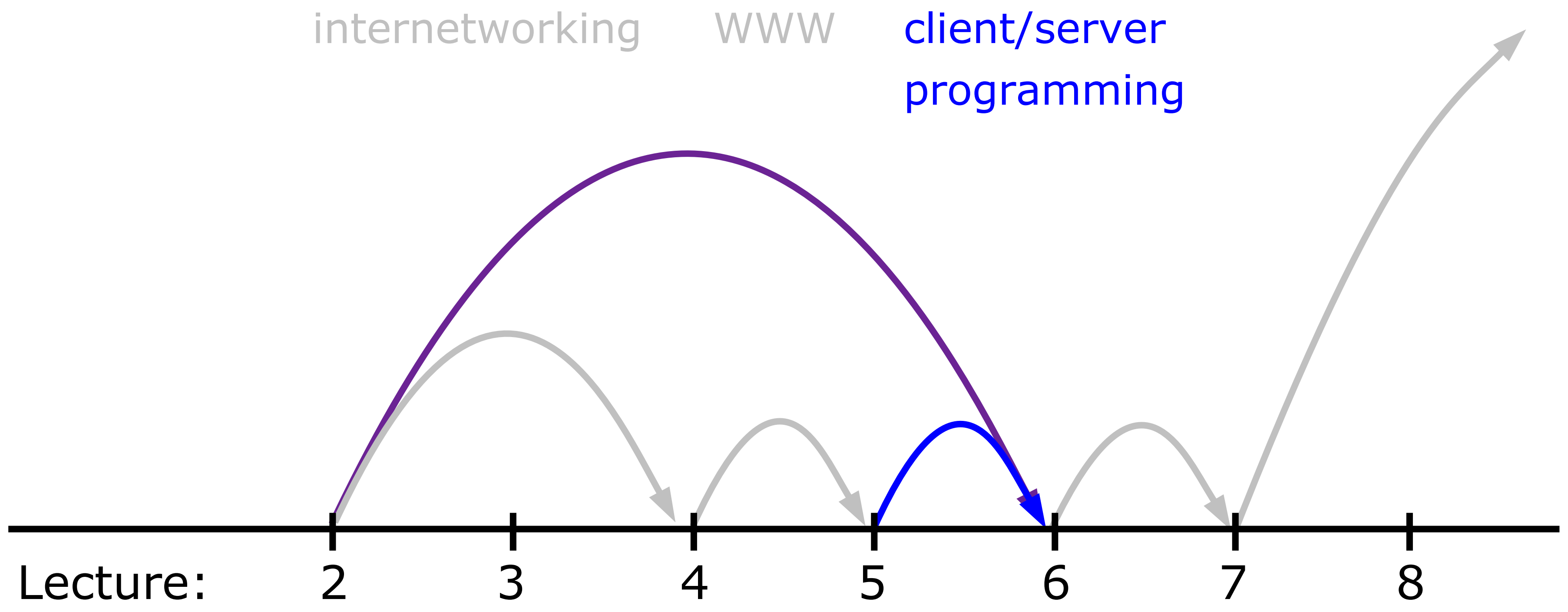
- Physical layer: *laptop with Wi-Fi hardware.*
- Data link layer: *wireshark in monitoring, promiscuous mode: seeing e.g. beacon frames.*
- Network layer: *wireshark in "ordinary" mode: seeing the IP-based traffic to and from your machine.*
- Transport layer: *catching background TCP traffic by a browser.*
- Application layer: *manual HTTP transaction using telnet.*

Topical overview: main arcs



Today: the *client/server programming* arc

- *Sessions 2-5:* from copper wires
  to client/server programming



Context and overview

- *Previously:* HTTP/HTML on client/server model implements interactive hypermedia.

- *Today:* As always, we need Turing-complete computation...
  - *Consider:* The Church-Turing conjecture.

⇒ **Q:** How to extend use of the client/server model to general computation ?

- HTML is for document markup
  - *is not* a programming language.

⇒ *Approach discussed today:* HTML used as the substrate for

- client-side programming languages *as well as*
- server-side programming languages.

Interactive hypertext, so far:

- HTML forms enable interactivity between web user and web application.
- Using a GET or POST method HTTP request, data entered into an HTML form is sent to a web server.
- After processing the data, the server responds with a resulting document.
- This cycle is repeated during an HTTP session, but still:

  HTML lacks instant interactivity/feedback:
  processing of data happens on the server side.

Adding dynamics: JavaScript

- JavaScript *is* a programming language.
- Routinely used within browsers to enhance HTML documents
  with dynamic content,
  instant feedback,
  user interaction.
- JavaScript code can be embedded within an HTML document and is interpreted within the web browser.

Languages: *interpretation* versus *compilation*

- "Code written in an **interpreted** programming language (often called "script") may be executed from source form, by an **interpreter**. Any language may, in theory, be compiled or interpreted; therefore, it refers to languages' implementations rather than designs."

- "An interpreted program can not be as efficient as a **compiled** program, which has been processed by a language **compiler**. A language compiler converts source statements into something close to the strings of 0's and 1's that a processor ultimately is given to work on. Because this work is already done before a compiled program is run, it runs much more quickly."

(Sources: wikipedia.org, whatis.com.)

JavaScript

- *“[...] interpreted within the web browser [...]”*

⇒ therefore, this is client-side technology.

- Reduces server computation & network traffic overheads.
  - A new HTML page no longer has to be requested from the server for every small change in appearance or user action.

- The JavaScript language is not limited to web browsers:
  - may run on servers;
  - may run from command-line interpreters;
  - may run everywhere – in principle.

JavaScript

- *History:* introduced in 1995 by Netscape and Sun corporations.
- *Reasons for popularity:*
  - its code can be embedded into HTML;
  - it can change or add content to HTML documents;
  - it can control the web browser;
  - it can react to and implement interaction with the user;
  - it is built into (*understood by, interpreted by*) common browsers.

JavaScript

- *Some “classical” uses in HTML documents:*
  - open pop-up windows with a specific size, location and other settings;
  - change images when the user's pointer goes over them;
  - validate the content that a user typed into an HTML form, e.g. · checking required fields
    - · acceptable values
    - · e-mail address format
    - · etc.
- But much more is possible.

Avoiding confusion: What about *Java*?

- Java also is a programming language.
- It enabled the development of web programs called applets.
- Applets could be included in web pages; are now outdated.

- Java remains very relevant however
  - e.g. when developing Android apps.

JavaScript versus Java

- JavaScript != Java.
- Their syntax is similar: both based on C programming language.
  - (As is the syntax of e.g. the Processing language.)
- Java is typically *compiled* (to machine-independent bytecode).
- JavaScript is typically *interpreted* (hence the term *script*).
- Roles as web technologies:
  - In Java, you can write apps, programs running on mobile platforms for consumer computing; possibly web-related.
  - JavaScript code is tied into an HTML document, and can control the document and browser.

JavaScript: study material

Let's have a look at the required reading.

JavaScript: The Document Object Model (DOM)

- DOM: An *object-oriented* description of an HTML document.

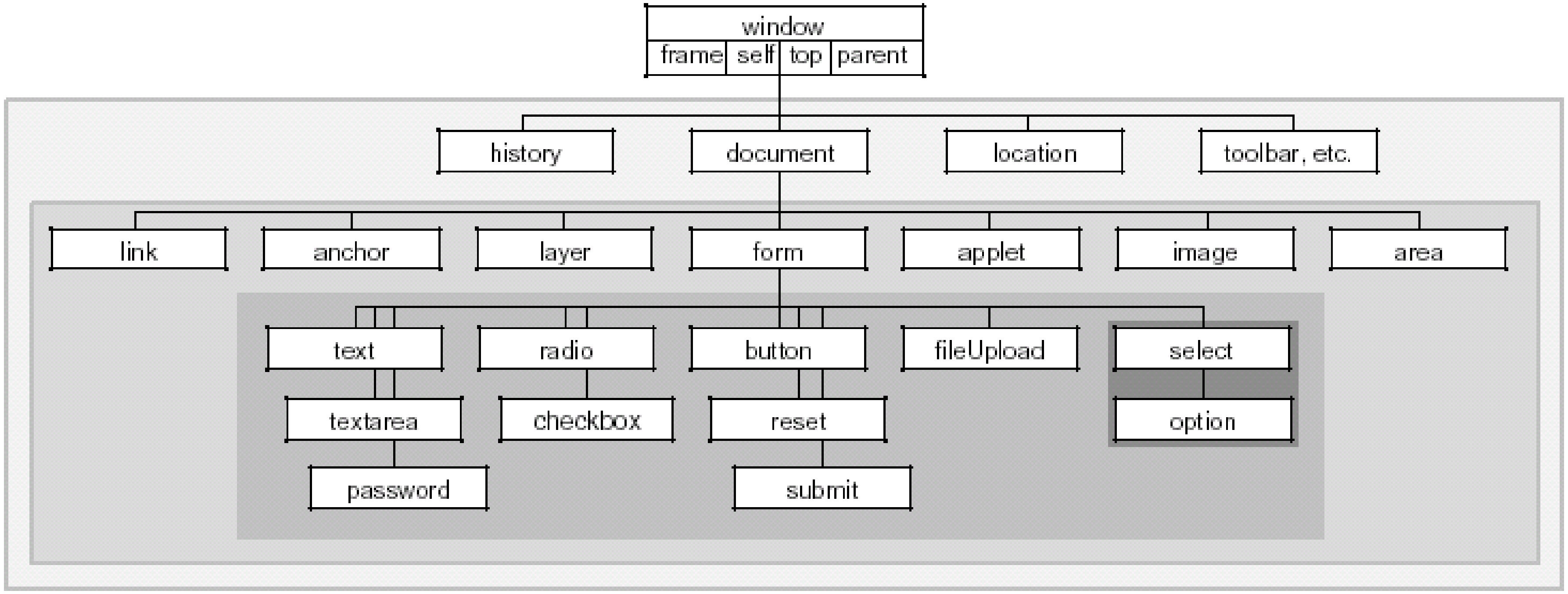
- Used by programs to *access*, *change*, and even *newly instantiate* a hypertext document's content, structure, and style.

- For programming: HTML document is composed *hierarchically* of many *objects*, which can be accessed and changed.

- E.g. a browser window object (`window`)
  ...with a page object (`document`)
  ...with an image object (`image`)
  ...and a button object (`button`)
  ...etc.

- JavaScript adds dynamics and interactivity to HTML documents via the DOM.

DOM: an example object tree



DOM: properties, methods, events

- *Key concepts:* object properties & methods; and events.

- Properties are things that an object may have, or be:

```
document.title
document.lastModified
image.border
```

- Methods are actions that can be executed for an object:

```
document.write(“Eyjafjallajökul”)
string.toUpperCase( )
```

- Events are actions that may happen to an object:

```
<img onLoad="alert(‘April 1 is a dangerous day.‘);">
<a href=“vla.html“ onClick=“alert(‘Vanille vla!‘);“>
```

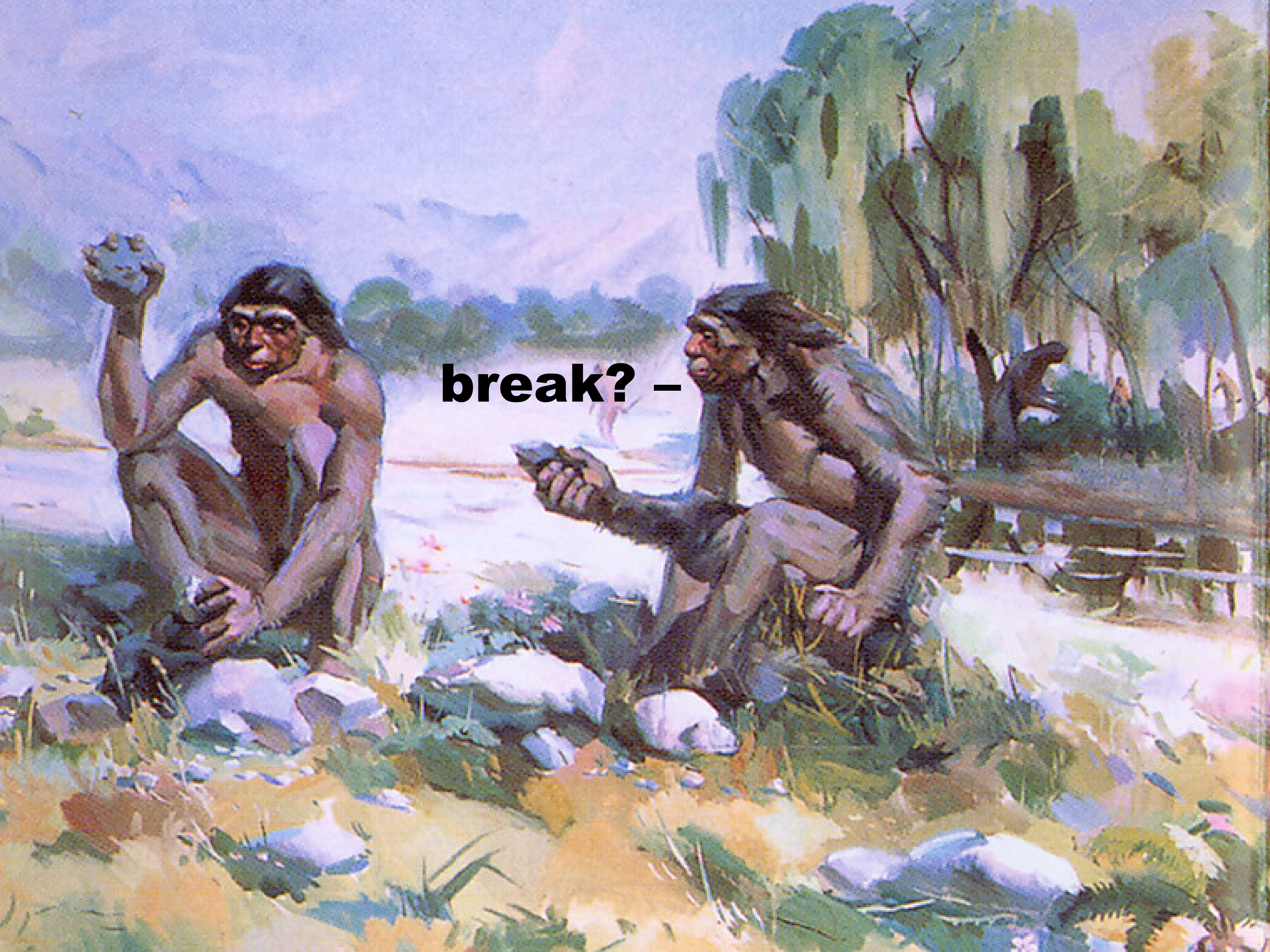
DOM: object tree example

- Imagine the browser has loaded the following HTML document:

```
<html>
 <body>
  <form name="icecream">
   <input type="text" name="scoops">
   <input type="text" name="flavor">
  </form>
 </body>
</html>
```

- Your code can now access the value of the text-input objects as:

```
window.document.icecream.scoops.value;
window.document.icecream.flavor.value;
```


break? –