Javascript & DOM

Maher Khan

Recitation 2
Date: 14th September, 2018

Office Hours

- Thursday:
  - 3 PM to 5 PM
- Friday:
  - 6 PM to 8 PM
- By appointment (maherkhan@pitt.edu)

Table of Contents

Javascript

Numbers

- How many number types are there in javascript?
  - One: 64-bit floating point
  - No integers
- Experiment

  x=4

  x+=.33

  x
- How do you get the integer of x?
  - Math.floor(x) // truncates
  - Math.round(x) // if you prefer to round

NaN

NaN is a special number: Not a Number

- Result of undefined or erroneous operations
- Toxic: any arithmetic operation with NaN as an input will have NaN as a result
- NaN is not equal to anything, including NaN
- Experiment

```
var r=12
var t=NaN
x+r
x+t
x+r+t
```

4 ways to declare a variable or a constant

const x = 2;

- const defines a constant within a block scope.
- A block is delineated by { and }
- Assignment is made once and only once

const PITT_ADDR = "5000 Forbes Ave"

let x = 2;

- let defines a variable within a block scope
- A block is delineated by { and }
- This is the best way to define variables

var x = 2;

- var defines a variable in the current execution context
- Function context if in a function
- Global context if outside a function
- var needs not be declared before using a variable

Implicit declaration

- You can just use a variable without declaring it using const, let, and var.
- E.g:

  x = 4;
- In all contexts, the scope will be global
  - If defined in a global scope, then global
  - If defined in a function, still global
- This is a very bad idea because
  - It pollutes the global namespace.
  - Makes debugging harder
    - Where did this variable come from?
    - Where is its value set?
      - It is confusing for other programmers to pick up your code and understand it.
- Watch for implicit declarations in for() statements
  - Best practice: use let in for() statements:

```
for (let x = 1; x <= 5; x++) {      // scope of x will only be the loop
```

Variable Scope

- Experiment:

```
scope1 = "global";
scope2 = "global";
function checkscope() {
    let scope1 = "local";
    let scope3 = "local";
    scope2 = "local";
    scope4 = "local";
    var scope5= "local";
}
checkscope()
```

- What is the resulting value of
  - scope1?
  - scope2?
  - scope3?
  - scope4?
  - scope5?

Defining functions

As a function:

```
function calculator() {…
        OR
var calculator = function() {…
```

As inner functions:

```
function multiplyAbsolute(number, factor){
        function multiply(number){
                return number * factor;
        }
        if(number < 0){
                return multiply(-number);
        }
        else{
                return multiply(number);
        }
}
```

As anonymous functions:

```
(function(){ number = 10; })();
```

Closure

Closure: The scope of an inner function continues even after the parent functions have returned.

Example:

```
    function ping() {
        console.log("Ping");
        let times=0;
        pong = function() {
            console.log("Pong "+ (++times));
        }
}
```

Experiment:

- Try pong() first
- Then ping()
- pong() again
- pong() again

Asynchronous Programming

• Browsers, mobile devices, and servers spend a lot of their time waiting for things to happen.
  – Waiting for a user to click a button
  – Waiting for a server to make a response
  – Waiting for a database to return data

• Two ways to handle delay:
  – **Synchronous**:
    • When initiating something that takes time, "block"
      – Stop and wait until the response is received

  – **Asynchronous**:
    • When initiating something that takes time,
      – start it and provide a "handler" to run when the response is received
      – move on and do other things

Client-side (browser) asynchronousity

– User Events
  - E.g. onclick, onmouseover, onfocus
  - And lots more

– Timer
  - setTimeout - do callback after some given time
    – This is a built-in JavaScript function
    – setTimeout(functionToCall, numMilliseconds)
  - setInterval – do callback after every given time interval
    – Also a built-in JavaScript function
    – setInterval(functinoToCall, numMilliseconds)

– Asynchronous processes
  - E.g. make a request to the server (AJAX) and run a callback function when the reply is completed

Async/Closure/Callback Example

```
function tick() {
    console.log("tick ");
    setTimeout(tock, 1000);
}
function tock() {
    console.log("tock ");
    setTimeout(tick, 1000);
}
```

- Without using a global variable, have tick and tock produce the following:
  tick 0
  tock 1
  tick 2
  tock 3
  …

Will this work?

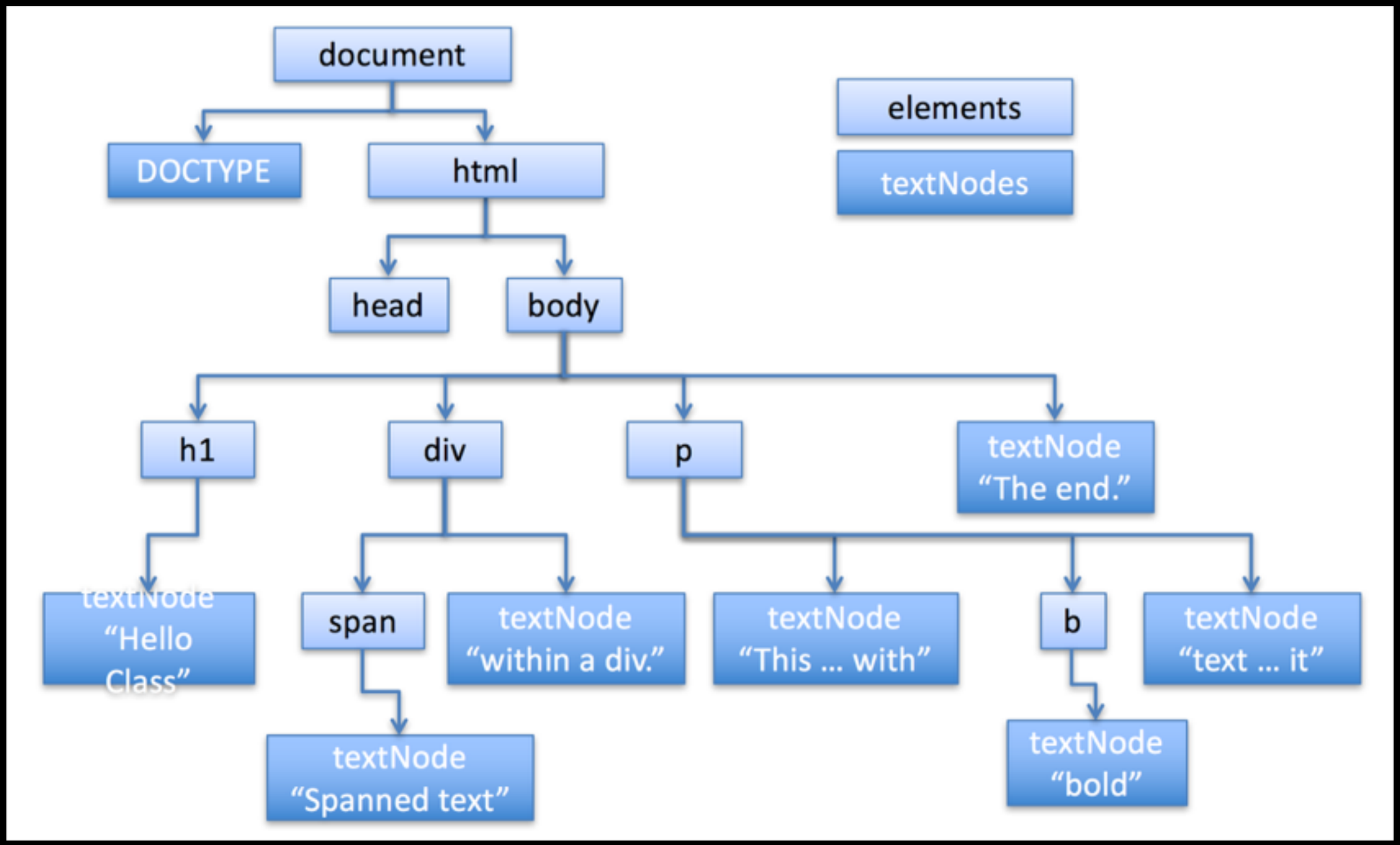
```
function tick() {
    let tcount = 0;
    console.log("tick "+ tcount++);
    setTimeout(function(){
        console.log("tock "+ tcount++);
        setTimeout(tick, 1000);
    }, 1000);
}
```

Solution with zero global variable

```
(function() {
    var tcount = 0;
    function tick() {
        console.log("tick "+ tcount++);
        setTimeout(tock, 1000);
    }
    function tock() {
        console.log("tock "+ tcount++);
        setTimeout(tick, 1000);
    }
    return tick;
})()();
```

DOM

Document Object Model



Walking the DOM

Experiment with…

- document.body.childNodes[1]
- document.body.firstChild
- document.body.firstChild.nextSibling
- document.body.firstElementChild
- document.body.firstElementChild.innerHTML
- document.body.firstElementChild.nodeName
- document.body.firstElementChild.id
- hw.nodeName
- hw.nextElementSibling

Searching the DOM

There are 3 ways to search the DOM:

- document.getElementById("pp1").innerHTML
  - "This is a paragraph with bold and <span id=“span1">spanned text within it”

- document.getElementsByName("2nd")
  - [spanned]
  - Notice the plural Elements and that an array is returned (Even if only for one element)

- document.getElementsByTagName("span")
  - [Spanned text, spanned]

Creating Elements

- Create a new element:
  - de=document.createElement(“p”);
- Create a textNode
  - tn=document.createTextNode(“It is 11:30am.”);
- Add the textNode to the <p> element
  - de.appendChild(tn);
- Insert a node before another
  - tn=document.createTextNode(“What time is it? ”);
  - de.insertBefore(tn, de.firstChild);
- Add the whole thing to the document
  - var dv=document.getElementsByTagName("div");
  - document.body.insertBefore(de,dv[0]);