Lecture 14

408/508 *Computational Techniques for Linguists*

Today's Topics

- Introduction to Javascript contd.
  - strings, operators, conditionals, loops
  - a quirk with assignment (=)
- Tic-tac-toe example contd. (*step by step*)
  - Files: `sample3.html`, `sample4.html`, `sample5.html`, `sample6.html` and `sample7.html`
  - *on course webpage*
  - Homework 6: *make it playable!*

Javascript: Strings

- Use either **double quotes** or **single quotes** (no difference)
  - escape using backslash, e.g. **\"**, **\n**, ****
- Properties:
  - **"*string*".length**
- Methods (like a function but object-based):
  - **.indexOf(*string*)** returns position (0…length), -1 if not found
  - **.search(regex)** returns position
  - .**replace(regex,string)** replaces with string, returns result
  - **.slice(index,index)** substring from position to position
  - **.substr(index,length)** s ubstring
  - **.toLowerCase()** case conversion
  - **.toUpperCase()**
  - **.trim()** remove whitespace from both ends
  - .**charAt(index)** single character at position
  - .**split(separator)** returns array, single characters if separator=""

Javascript: Operators

Arithmetic Operators:

- +, -, *, /, % (mod), ++, --

Assignment Operators:

- =, +=, -=, *= /=, %=

String Operators:

- + concatenation
- += append to string

```
alert("string" + 5)
produces string5
```

Comparison Operators:

- == can be made equal (type coercion)
- != not equal
- === same type and equal (no type coercion)
- !== not equal (no type coercion)
- >, <, >=, <=
- && logical and
- || logical or
- ! negation

(Javascript: Boolean true/false)

Note: & , !, ~ (not), ^ (xor) are bitwise operators

Javascript: Conditionals

- if-then

```
if (condition) { … }
else if (condition) { … }
else { … }
```

- switch

```
switch (expression) {
   case value: … break;
   …
   default: …
}
```

```
switch(expression) {
    case n:
        code block
        break;
    case n:
        code block
        break;
    default:
        default code block
}
```

Idea: compute *expression* first
2nd stage: compare computed value with each case

Javascript: Loops

- for loop (just like C):
  - **example**:
    - `for (i = 0; i < 100; i++) {  … }`
- for/in loop (object properties):
  - **`for (x in object) { … }`**

- while loop:
  - **example**:
    - `while (true) { … }`

- do/while loop (like traditional repeat/until):
  - **`do { … } while (`*`condition`*`)`**

- loop exit:
  - **`break`** (jump out of loop immediately)
  - **`continue`** (skip rest of current iteration)

Javascript: Miscellaneous

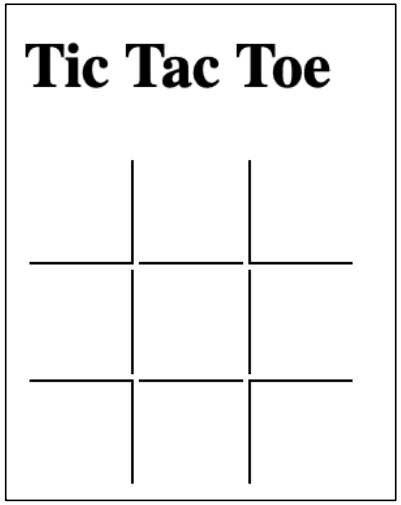
A bit quirky:

- Swapping two variables (*normally requires a third variable*):
  - `var a; var b; var temp;`
  - `temp = a;`
  - `a = b;`
  - `b = temp;`
- Javascript arithmetic):
  - `var a; var b;`
  - `b=a+(a=b)-b;`
  - `a = b + (b=a, 0);`

Arithmetic priority:
Set a to the value of b first,
`(a=b)` evaluates to b,
then evaluate rest of expression

Tic Tac Toe contd.

- Last time:
- `sample3.html`




```
<table>
<tr>
    <td id="1" style="border-right: 1px solid black; border-
bottom: 1px solid black"></td>
    <td id="2" style="border-bottom: 1px solid black"></td>
    <td id="3" style="border-left: 1px solid black; border-
bottom: 1px solid black"></td>
  </tr>
  <tr>
    <td id="4" style="border-right: 1px solid black"></td>
    <td id="5"></td>
    <td id="6" style="border-left: 1px solid black"></td>
  </tr>
  <tr>
    <td id="7" style="border-right: 1px solid black; border-
top: 1px solid black"></td>
    <td id="8" style="border-top: 1px solid black"></td>
    <td id="9" style="border-left: 1px solid black; border-
top: 1px solid black" ></td>
  </tr>
</table>
```

Document Object Model (DOM)

Locating an element:

- `document.getElementById(id)`
  - useful if you have named the document element using the id='Name' property, e.g. `id="5"`.
- `document.getElementsByTagName(tag)`
  - all document elements of type *tag*, e.g. `"td"`
  - returns an array
- *element*`.getElementsByTagName(`*tag*`)`
  - all elements of type tag under element *e*
  - returns an array
- `document.getElementsByName(`*name*`)`
  - useful for elements that support name='Name'
- `(document|`*element*`).getElementsByClassName(class)`
  - returns an array

Document Object Model (DOM)

- `(document|`*`element`*`).querySelector(query)`
  - example query ‘BODY > UL > LI’
  - ‘>’ means immediately dominates
  - returns first matching element
- `(document|element).querySelectorAll(query)`
  - returns an array

Document Object Model (DOM)

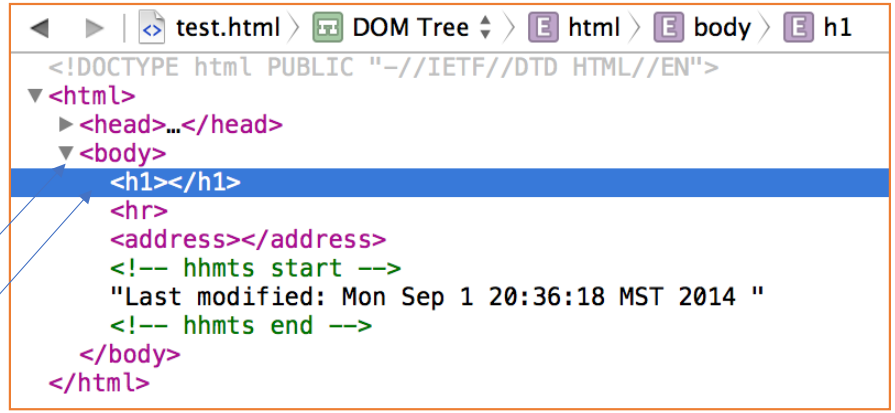
- HTML document

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html> <head>
<title></title>
</head>

<body>
<h1></h1>


<hr>
<address></address>
<!-- hhmts start --><!-- hhmts end -->
</body> </html>
```

- Tree representation




```
<body>:     document.body
<h1>:       document.body.h1
```

Document Object Model (DOM)

Let *e* below be an element

- (*starting top-down from* `document`)

Properties for traversing the DOM:

- *e*.childNodes
  - children of element el as an array,
  - e.g. childNodes[0]
- *e*.children
  - element nodes only (excludes text nodes)
- *e*.firstChild
- *e*.lastChild
- *e*.parentNode
- *e*.nextSibling
- *e*.previousSibling

Object properties:

- *e*.nodeType
  - 1 = element, 3 = text
- *e.nodeName*
  - uppercase
- *e.innerHTML*
  - for element nodes
  - value is html as text
  - writeable
- *e*.nodeValue
  - for text nodes
    (null: for element nodes)
  - writeable

Document Object Model (DOM)

New content:

- `document.createElement(`*`tag`*`)`
  - *tag* = ‘div’, ‘p’ etc.
  - creates new DOM element
- `document.createTextNode(`*`string`*`)`
  - creates new DOM element of type text

For non-HTML elements:

- `document.createElementNS(NS, `*`tag`*`)`
  - NS = Namespace URL identifier
  - e.g. http://www.w3.org/2000/svg and tag “rect” (rectangle)

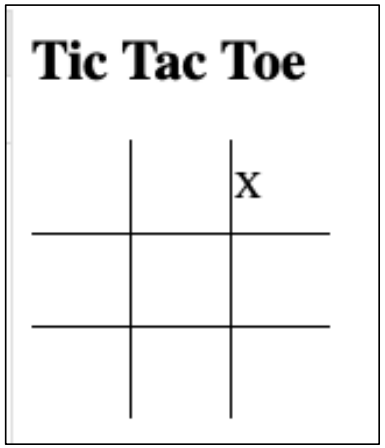
Document Object Model (DOM)

Place *new_el*:

- `e.appendChild(`*new_el*`)`
  - *new_el* is inserted as last child of *el*
- `e.insertBefore(`*new_el*`, `*next_el*`)`
  - *new_el* inserted as previous sibling of *next_el*
  - *el* is common parent
- `e.removeChild(`*child_el*`)`
  - *child_el* is deleted
  - *el* is parent
- `e.replaceChild(`*new_el*`, `*child_el*`)`
  - *new_el* replaces *child_el*
  - *el* is parent

Tic Tac Toe: `sample3.html`

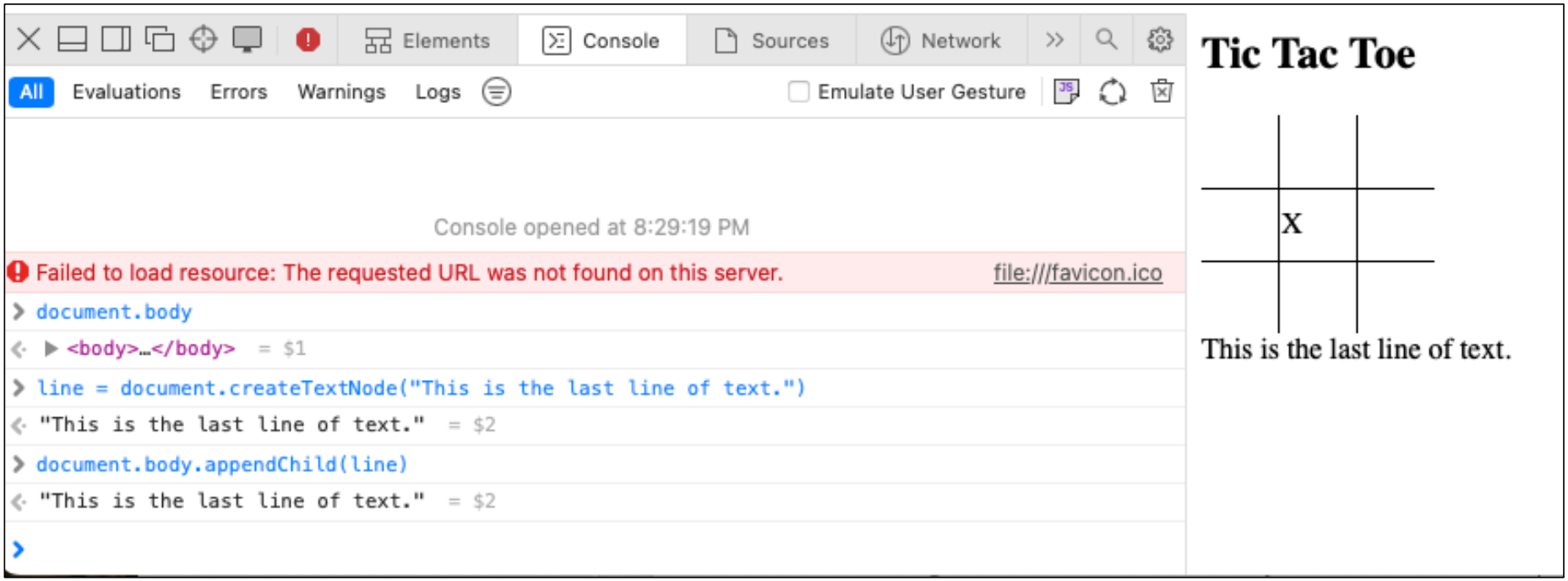
- Let's modify `sample3.html`
- Steps on the console:
  1. get document body
  2. create a text node
  3. append it to the end of the body
- Make it a program:
  - put the code in a <script>



- To add a line break:

```
document.body.appendChild(document.createElement("br"))
```

Tic Tac Toe: `sample3.html`



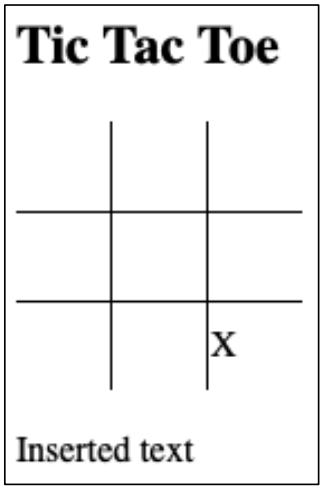
Tic Tac Toe: `sample4.html`

- Make it a program:
  - put the code in a `<script>`
- Function f():

```
function f(txt) {
document.body.appendChild(document.createElement("br"))
document.body.appendChild(document.createTextNode(txt))
}
```

- Call the function (*put this somewhere after the function definition*):

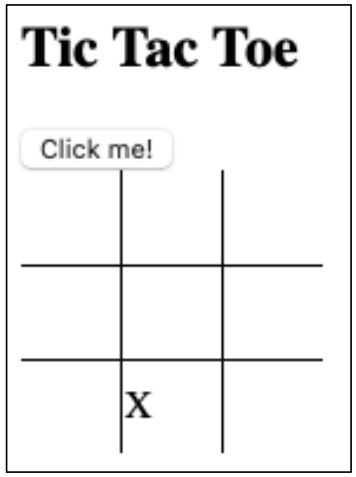
```
<script>
f("inserted text.")
<script>
```



Tic Tac Toe: `sample5.html`

- Make it a program:
  - put the code in a function in a `<script>`
- Trigger the function with a button:

```
<button onclick="f('Inserting new line of text.')">Click
me!</button>
```

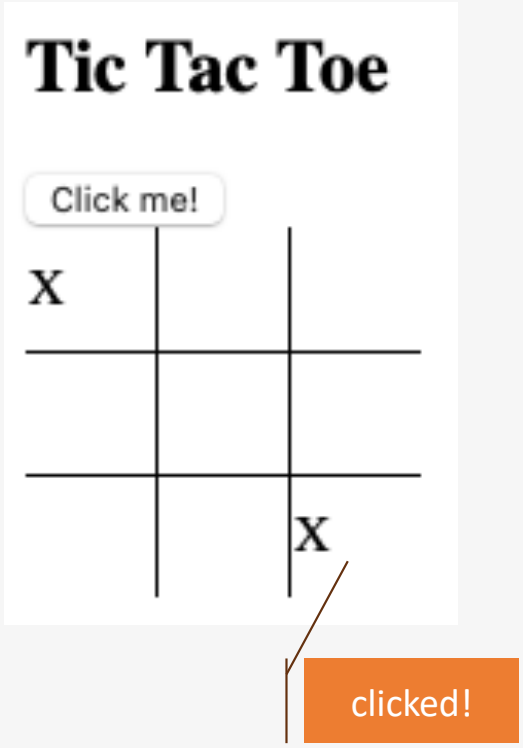


Tic Tac Toe: `sample6.html`

- let's make a table data cell clickable using the `onclick` attribute
  1. **table cell:**
  - `<td onclick="mark(this)">..</td>`
  - `this`: a special identifier/variable containing the td element
  2. **function definition:**

```
<script>
function mark(element) {
  .. code ..
}
</script>
```

- Let's place an X in the cell:
  - in `.. code ..` write `element.innerHTML = "X"`

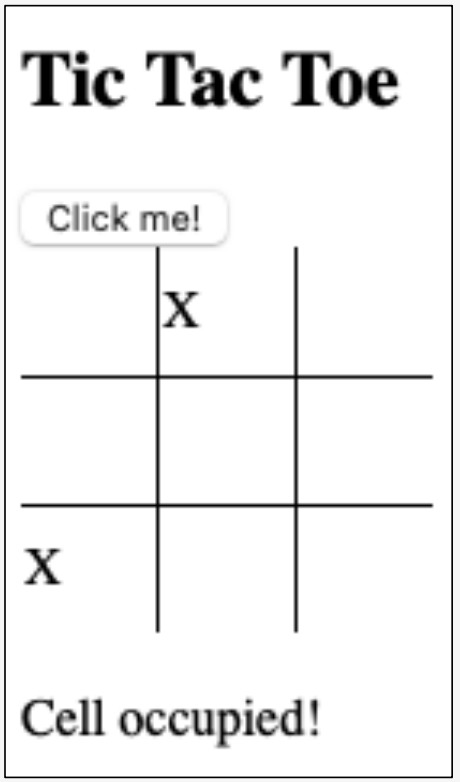


Tic Tac Toe: `sample7.html`

- Let it only write in an empty table data <td> cell

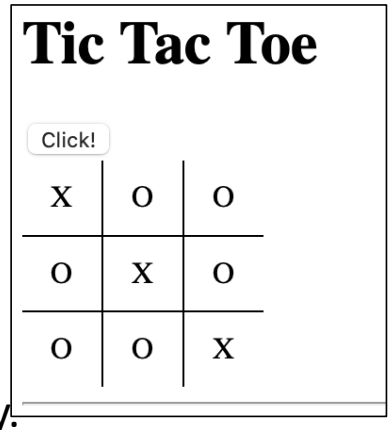
```
<script>
function mark(element) {
  if (condition) {
      element.innerHTML = "X"
  } else {
     f("Cell occupied!")
  }
}
</script>
```

- What's the empty condition?
  - `element.innerHTML == ""`
  - recall == is equality



Homework 6

- Question 1: modify sample7.html to make an interactive tic tac toe player
- Using the random number generator described in the previous lecture, add a button "Click!" that inserts a O in an empty cell.
  - `var o = Math.floor(Math.random()*9)+1;`
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random
- Note A:
  - use `onclick` with a new function, e.g. `insert_o()` that you write
- Note B:
  - it should not overwrite an existing cell, i.e. one that has X or O already.
- Note C:
  - you can click again if it fails to insert, or you can repeatedly generate a random number until it happens upon an empty cell



Homework 6

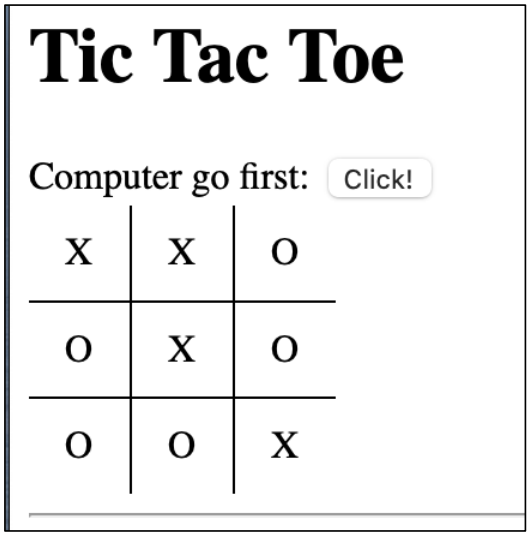
- Question 2: attempt only after you successfully complete Q1!
  - add a variable called `total` that counts the number of cells already filled. Start with:
  - `var total = 0`
- Note:
  - function `mark()`, which inserts an X, should increment `total` by 1 (*assuming you click on an empty cell correctly*)
  - function `insert_o()`, which inserts an O, should also increment `total`
  - `total += 1`

Homework 6

- Question 3: check the value of `total` after each insertion of an X or O.
  - If `total` is 9, print a message saying *Game Over*! using `alert()`.

Homework 6

- Question 4: modify `mark()` to call `insert_o()` directly after you click on an empty cell.
  - Idea: you no longer have to press Click! for the computer to take its turn.
- **Note**:
  - To restart the game, reload the page in the browser.
  - Then the Click! button means the computer gets to go first.



Homework 6

- Extra Credit Question 5:
  - modify the program to spot when the game has been won and print a message
  - Note: to solve this you need a big if-condition

    e.g. when:
    - cells: 1 2 3 have the same non-zero value (X or O)
    - cells: 4 5 6 have the same non-zero value (X or O)
    - cells: 7 8 9 have the same non-zero value (X or O)
    - cells: 1 4 7 have the same non-zero value (X or O)
    - cells: 2 5 8 have the same non-zero value (X or O)
    - cells: 3 6 9 have the same non-zero value (X or O)
    - cells: 1 5 9 have the same non-zero value (X or O)
    - cells: 3 5 7 have the same non-zero value (X or O)
    - && is conjunction, || is disjunction, != is not equals

Homework 6

- Do as many questions as you can.
- Due date Sunday midnight
- Subject: 408/508 Homework 6 *YOUR NAME*
- Submit screenshots in your PDF
- If you have errors, you can screenshot the Javascript console
- Attach a copy of your .html file separately to your email (*so I can play tic-tac-toe*)