Debugging

CSE 190 M (Web Programming), Spring 2007
University of Washington

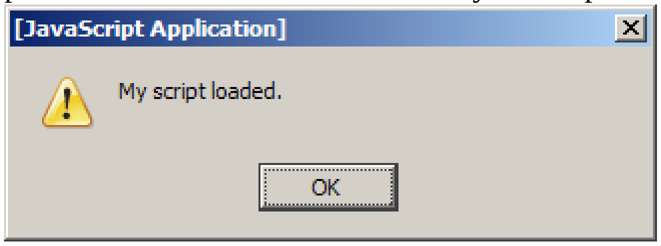
References: ***

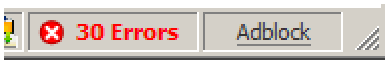
"My program does nothing"

Since Javascript has no compiler, many errors will cause your Javascript program to just "do nothing." Some questions you should ask when this happens:

- Is the browser even loading my script file?
- If so, is it reaching the part of the file that I want it to reach?
- If so, what is it doing once it gets there?

Is my JS file loading?

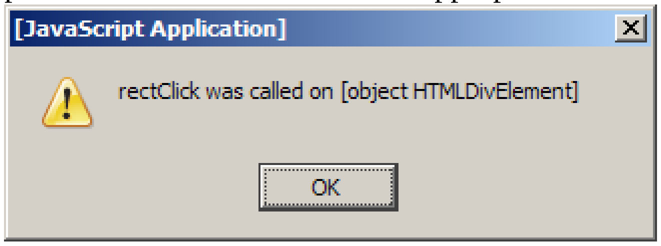
- put an `alert` at the VERY TOP of your script:

[JavaScript Application]
My script loaded.

- if it shows up, good!
- if it doesn't show up:
  - maybe your HTML file isn't linking to the script properly
    - double-check file names and folders
  - maybe your script has a syntax error
    - check bottom-right for Firebug error text 

    - comment out the rest of your script and try it again
    - run your script through JSLint to find some syntax problems

Is it reaching the code I want it to reach?

- put an `alert` at the start of the appropriate function:



rectClick was called on [object HTMLDivElement]

  - write a descriptive message, not just `"hello"` or `"here"`
- if it shows up, good!
- if it doesn't show up:
  - if it's an event handler, maybe you didn't attach it properly; check the code to attach the handler
  - maybe your script has a syntax error; run JSLint

COMMON BUG: spelling error

```
window.onload = initalizeBody;    // spelled wrong
...
function initializeBody() {
    var theDiv = document.getElementById("puzzlearea");
    ...
}
```

- if you misspell an identifier, the value `undefined` is used
- if you set `undefined` as an event handler, nothing happens (fails silently)
- Manifestation of bug: function doesn't get called, or a value is unexpectedly `undefined`
- Fix: JSLint warns you if you use an undeclared identifier

COMMON BUG: bracket mismatches

```
function foo() {
    ...

function bar() {
    ...
}
```

- JS unfortunately doesn't always tell us when we have too many / too few brackets in our JS code
- Manifestation of bug: script often becomes (fully or partially) non-functional
- Fix: bracket matching in TextPad (highlight bracket, press Ctrl-M)
- Fix: JSLint sometimes catches this
- **Detection**: use bracket match in TextPad: Ctrl-M

COMMON BUG: () in event handler

```
myObject.onclick = handleClick();   // BAD!
myObject.onclick = setSpeed(200);   // BAD!
myObject.onclick = handleClick;     // better
```

- when you attach an event handler, write only the name of the function
- this attaches the function to the event, so it can be called later
- if you write `()` after the function name, the function is called immediately rather than attached as an event handler
- Manifestation of bug: event handler code runs immediately, rather than waiting until you trigger the event
  - if your event code runs too soon, you may likely have this bug
- Fix: JSLint now often catches this bug

`window.onload` variation

```
window.onload = initializeBody();   // BAD!
...
function initializeBody() {
    var theDiv = document.getElementById("puzzlearea");
    ...
}
```

- `()` bug in a `window.onload` handler has different symptoms
- common message: `theDiv has no properties`
  - occurs because `initializeBody` is being called immediately (as the script is loading in the document's `head`)
  - `document`'s elements haven't been loaded yet at that time (they are in the `body`, below the `head`)

"But I need parameters in my handler!"

- many event handler parameters are not needed
- recall: event handler function knows the object it was attached to
  - can access the object using `this`
- shouldn't attach handler like this:

  ```
  myObject.onclick = setSpeed(this.value);   // BAD!
  ```

- Fix: instead attach it like this:

  ```
  myObject.onclick = setSpeed;   // better
  ...
  function setSpeed() {
      var speed = this.value;
      ...
  }
  ```

COMMON BUG: misuse of `.style`

```
var theDiv = document.getElementById("puzzlearea");
theDiv.left = "100px";                    // BAD!
theDiv.style.onclick = myClickFunction;   // BAD!
```

- DOM objects have internal `style` object that represents CSS styles
  - setting styles: `object.style.property = value;`
- the DOM objects themselves also have properties of their own
  - setting DOM properties: `object.property = value;`
- Manifestation of bug: "I set the property, but it didn't do anything."
- Fix: JSLint now tries to catch this and shows an error
- Avoidance: if you're setting something that you would have set in the CSS file, use `.style`. If you would have set it in the HTML file, don't.

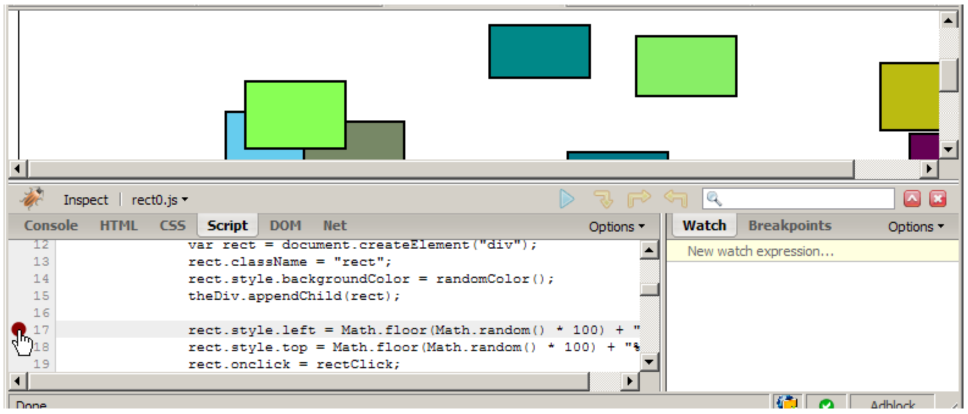
COMMON BUG: incorrect units on styles

```
theDiv.style.left = 100;   // BAD! should be "100px"
theDiv.style.top = 200;    // BAD! should be "200px"
```

- all CSS property values must be Strings, and many require units and/or a specific format
- Manifestation of bug: code fails silently; style is not set
- Detection: use Firebug debugger, step through code and look at `style`
- Detection: use an `alert` immediately after style property is set

```
theDiv.style.left = 100;   // BAD!
alert("div left is " + theDiv.style.left);
```

Debugging in Firebug

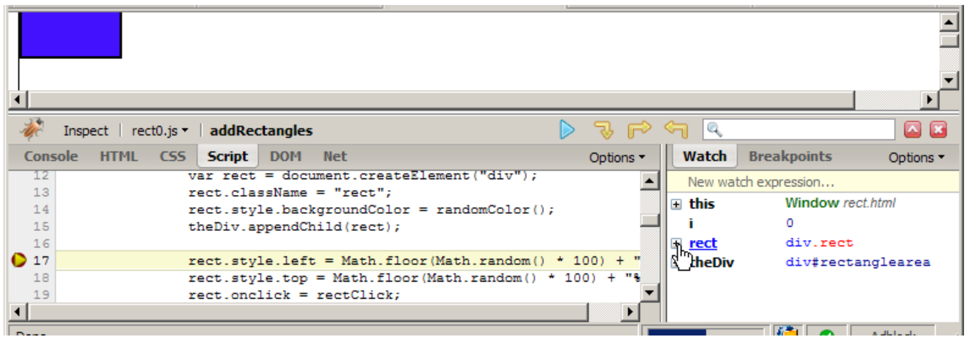


```
var rect = document.createElement("div");
rect.className = "rect";
rect.style.backgroundColor = randomColor();
theDiv.appendChild(rect);

rect.style.left = Math.floor(Math.random() * 100) + "
rect.style.top = Math.floor(Math.random() * 100) + "%
rect.onclick = rectClick;
```

- open Firebug, click **Script** tab
- click to the left of a line to set a **breakpoint**
- refresh page
- when page runs, if it gets to that line in the JS code, program will halt

Breakpoints



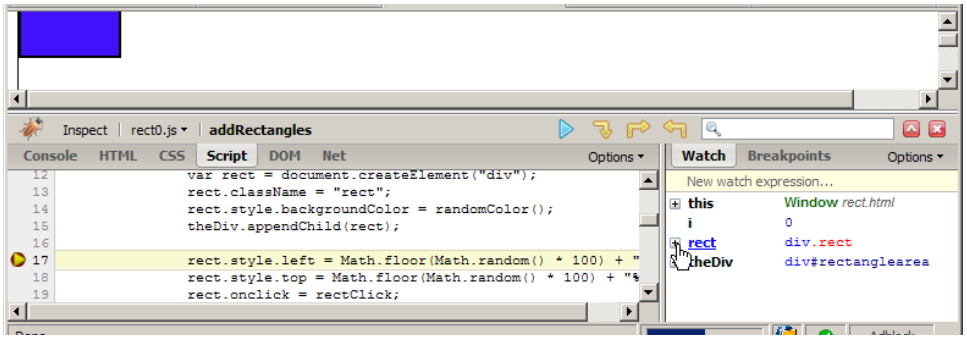
```
                    var rect = document.createElement("div");
                    rect.className = "rect";
                    rect.style.backgroundColor = randomColor();
                    theDiv.appendChild(rect);

                    rect.style.left = Math.floor(Math.random() * 100) + "
                    rect.style.top = Math.floor(Math.random() * 100) + "%
                    rect.onclick = rectClick;
```

this	Window *rect.html*
i	0
rect	div.rect
theDiv	div#rectanglearea

- **data**: once you've stopped at a breakpoint, you can examine any variables in the **Wat**
    - can click $\boxed{+}$ to see properties/methods inside any object
    - **this** variable holds data about current object, or global data
    - make sure Options → Show DOM Properties is checked, so you can see any DOM-re

Stepping through code



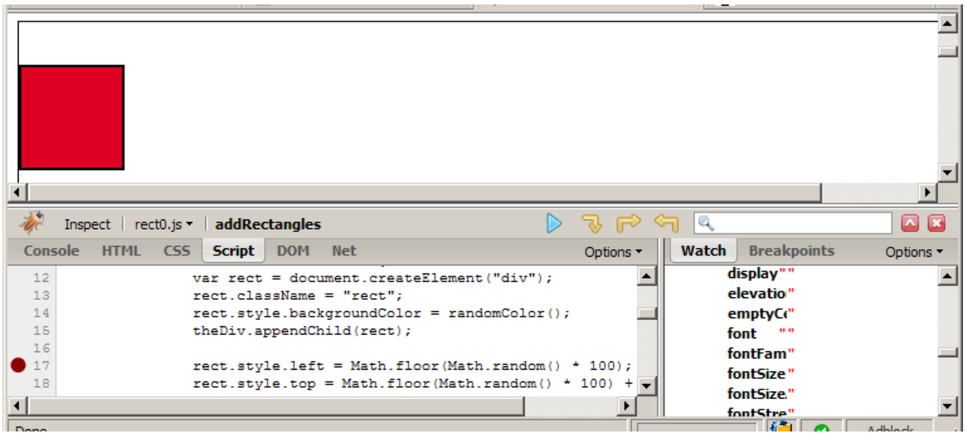
```
var rect = document.createElement("div");
rect.className = "rect";
rect.style.backgroundColor = randomColor();
theDiv.appendChild(rect);

rect.style.left = Math.floor(Math.random() * 100) + "
rect.style.top = Math.floor(Math.random() * 100) + "%
rect.onclick = rectClick;
```

this	Window rect.html
i	0
rect	div.rect
theDiv	div#rectanglearea

- **`code`**`: once stopped at a breakpoint, you can continue execution:`
  - **`continue`**` (F8): start the program running again`
  - **`step over`**`: run the current line of code completely, then stop again`
  - **`step into`**` (F10): run the current line of code, but if it contains any call`
  - **`step out`**` (F11): run the current function to completion and return, then st`

Debugging CSS property code

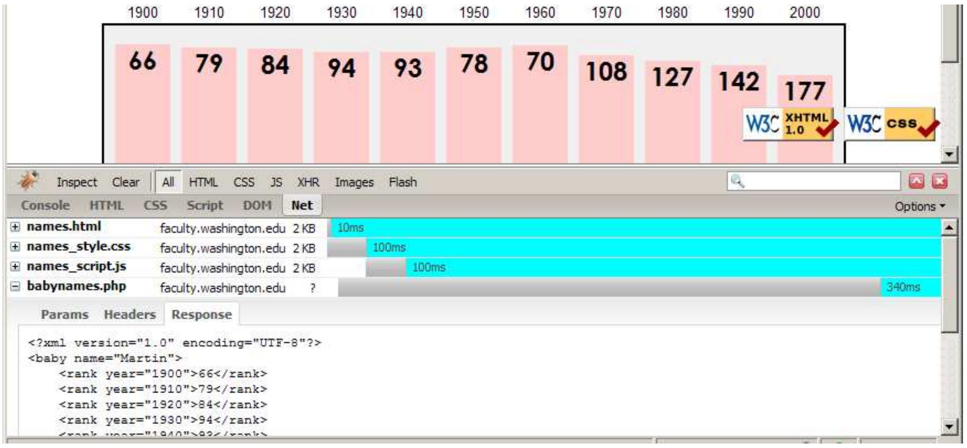


```
var rect = document.createElement("div");
rect.className = "rect";
rect.style.backgroundColor = randomColor();
theDiv.appendChild(rect);

rect.style.left = Math.floor(Math.random() * 100);
rect.style.top = Math.floor(Math.random() * 100) +
```

- expand DOM object with $\boxed{+}$, and expand its style property to see all styles
- also look at HTML (left) tab, Style (right) tab to see styles

Debugging Ajax code

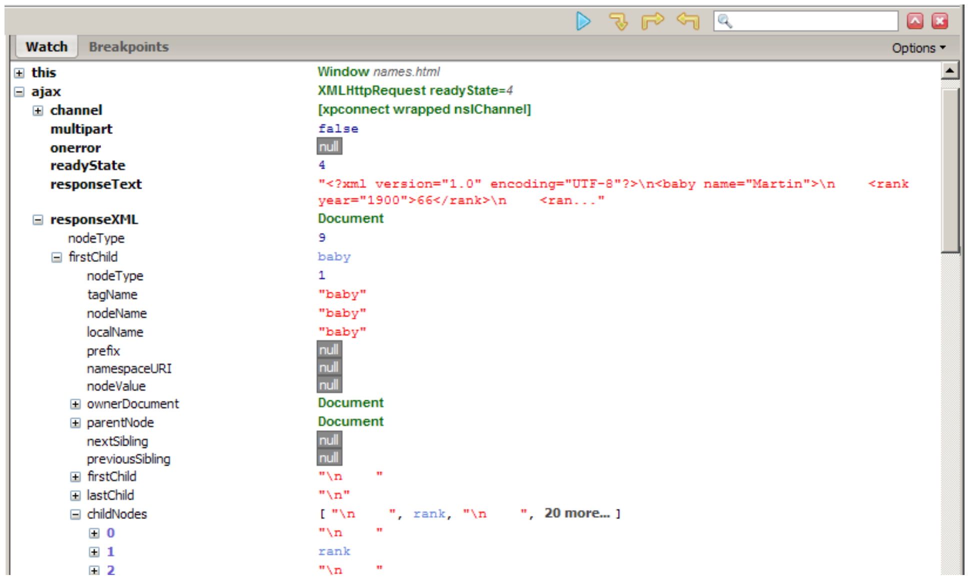


```
names.html          faculty.washington.edu  2 KB  10ms
names_style.css     faculty.washington.edu  2 KB  100ms
names_script.js     faculty.washington.edu  2 KB  100ms
babynames.php       faculty.washington.edu  ?     340ms
```

```
<?xml version="1.0" encoding="UTF-8"?>
<baby name="Martin">
    <rank year="1900">66</rank>
    <rank year="1910">79</rank>
    <rank year="1920">84</rank>
    <rank year="1930">94</rank>
```

- **Net** tab shows each request, its parameters, response, any errors
- expand a request with $\boxed{+}$ and look at Response tab to see Ajax result

Debugging responseXML


Watch Breakpoints Options

```
+ this                        Window names.html
- ajax                        XMLHttpRequest readyState=4
  + channel                   [xpconnect wrapped nsIChannel]
    multipart                 false
    onerror                   null
    readyState                4
    responseText              "<?xml version="1.0" encoding="UTF-8"?>\n<baby name="Martin">\n    <rank
                              year="1900">66</rank>\n    <ran..."
  - responseXML               Document
      nodeType                9
    - firstChild              baby
        nodeType              1
        tagName               "baby"
        nodeName              "baby"
        localName             "baby"
        prefix                null
        namespaceURI          null
        nodeValue             null
      + ownerDocument         Document
      + parentNode            Document
        nextSibling           null
        previousSibling       null
      + firstChild            "\n    "
      + lastChild             "\n"
      - childNodes            [ "\n    ", rank, "\n    ", 20 more... ]
        + 0                   "\n    "
        + 1                   rank
        + 2                   "\n    "
```

- can examine the entire XML document, its node/tree structure

General good coding practices

- ALWAYS code with Firebug installed
- write a little, test a little
- follow good general coding principles
    - remove redundant code

  - make each line short and simple
  - always use { } even when not needed on if, for, etc.

- use lines and variables liberally
  - it's good to save parts of a complex computation as variables
  - helps see what part of a big expression was bad/undefined/etc.
  - blank lines and profuse whitespace make code easier to read