Web
Bugs and Debugging
Types of Bugs

• Syntactic
  – No semicolon at the end of a line
  – Capital letter for function or variable instead of lower case

• Semantic or Logical
  – The program works but does not do what you intended
Say you’ve written some code

• It’s not *running*
  – won’t run in the browser

• It’s not doing what it should
  – won’t output the correct value
  – won’t display correctly
  – Will display some output correctly and some not

• *It’s taking too long*
HTML BUGS
HTML is very forgiving but..

• Syntactic bugs
  – <body> without a </body>
  – <head> without a </head>
  – <div missing a ‘>’ or <div> missing a </div>

• But it will not break if you miss a tag!
  – Formatting issues (CSS)
Write clean HTML code

• Close a Tag as soon as you open it!
  – <p></p>, then <p>blabla</p>
• Indent indent and indent
  – Use 4 white spaces or tab

• Use meaningful names for IDs and classes (e.g., “form-1″,”tab-1” rather than “f” or “t”)
Validating HTML code

- W3C html validator
  – http://validator.w3.org/

- Check browser compatibility
  – http://browsershots.org/
Output from W3C’s validator

Validation Output: 2 Errors

**Line 133, Column 18:** An attribute specification must start with a name or name token

```
alt="" width="10"" _ height="1""/></a>
```

An attribute name (and some attribute values) must start with one of a restricted set of characters. This error usually indicates that you have failed to add a closing quotation mark on a previous attribute value (so the attribute value looks like the start of a new attribute) or have used an attribute that is not defined (usually a typo in a common attribute name).

**Line 133, Column 19:** End tag for "img" omitted, but OMITTAG NO was specified

```
alt="" width="10""_ height="1""/></a>
```

You may have neglected to close an element, or perhaps you meant to "self-close" an element, that is, ending it with "/>" instead of ">".

**Line 132, Column 37:** Start tag was here

```
<a href="http://www.pricemycab.com"><img src="images/spacer.gif"
```

CSS BUGS
Typical CSS mistakes

• Selector bug
  – confusing ‘id’ for ‘class’
    • E.g., #para1 (id selector) instead of .para1 (class selector)

• Property bug
  – Does not exist
  – E.g., ‘align’ instead of ‘text-align’
Debugging CSS

• Use a debugging tool like FireBug
  – Inspect DOM
  – Turn on/off css elements
  – Edit css element values (E.g., width, padding)

• Check if there are no overriding CSS elements
  – The one which is more specialised takes precedence (see past lecture on CSS).
JAVASCRIPT BUGS
JS Syntactic Bugs

• Case sensitivity: Variable names, properties and methods are all case sensitive
• Spelling mistakes in variable and function names
• Mismatching quotes, parenthesis or curly braces will throw a syntax error
• Line breaks and forgetting semi-colons: Always end statements in semi-colons to avoid common line break issues
JS Semantic Bugs

• Conditional Statements:
  – if (x=y) returns true as x is assigned y and is greater than 0
  – Type error: ('2' > '12') evaluates to true because the string ‘2’ is alphabetically after ‘12’, whereas (2 > 12) results in false

Q: What does (2 > '12') result in?

• Forgetting to use quotes. What does the following do?
  var myName = Enrico;

• Variable Scope: global versus local scope
  – Variable/function declared without ‘var’ becomes a global variable

• Forgetting to use code blocks. Example:
  if (x==y)
    document.write('yippee');
    alert('yippee');

Other bugs

• See many more here:
Debugging Principles

• Breakpoint
  – A line of code
  – Right before or after a major piece of code
  – Stops execution and retains state of all variables

• Watch
  – Select an expression/variable and track it
  – Set flags for things you want to see
    ```javascript
    if (x!=y){
      alert('X is not set right!');
    }
    ```
  – Output stuff to the console:
    ```javascript
    console.log();
    ```
Typical Debugger Steps

• **Set a breakpoint in your code (usually click on a line number)**

• **Step Into** - Executes the current statement and then stops at the next statement. If the current statement is a function or script call, then the debugger steps into that function or script, otherwise it stops at the next statement.

• **Step Over** - Executes the current statement and then stops at the next statement. If the current statement is a function or script call then the debugger executes the whole function or script, and it stops at the next statement after the function call.

• **Step Out** - Steps out of the current function and up one level if the function is nested. If in the main body, the script is executed to the end, or to the next breakpoint. The skipped statements are executed, but not stepped through.

• **Continue** - Continues execution to the end, or to the next breakpoint. The skipped functions and invocations are executed, but not stepped through.
Resources

• YouTube videos on debugging websites
  – Firebug
  – Asp.net
  – Safari (Mac)
  – Chrome
• W3Schools
• StackOverflow
• Doug Crockford’s website & videos