CSC 337

JavaScript

Rick Mercer

What is JavaScript

- It’s a programming language designed for Web pages
- Developed by Netscape while Sun Microsystems was developing Java, 1995
  - Brendan Eich
    - I hacked the JS prototype in ~1 week in May And it showed! Mistakes were frozen early. Rest of year spent embedding in browser
- JS is an OOP scripting language that runs in browsers
  - it is not compiled like Java to run on a virtual machine

What is JavaScript?

- Client-side script: code runs in browser
- Often this code manipulates the page or responds to user actions
- JavaScript is used to make web pages interactive

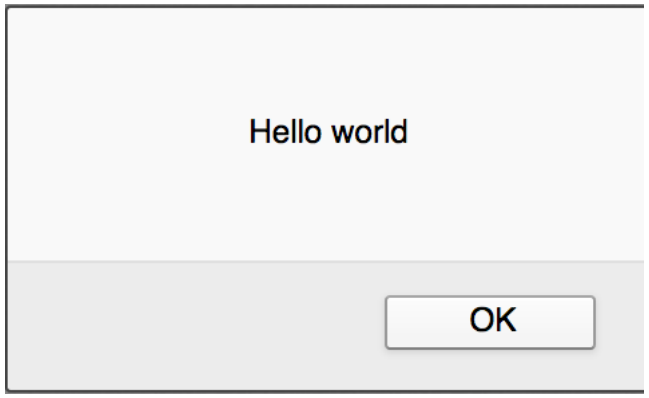
What can JavaScript Do?

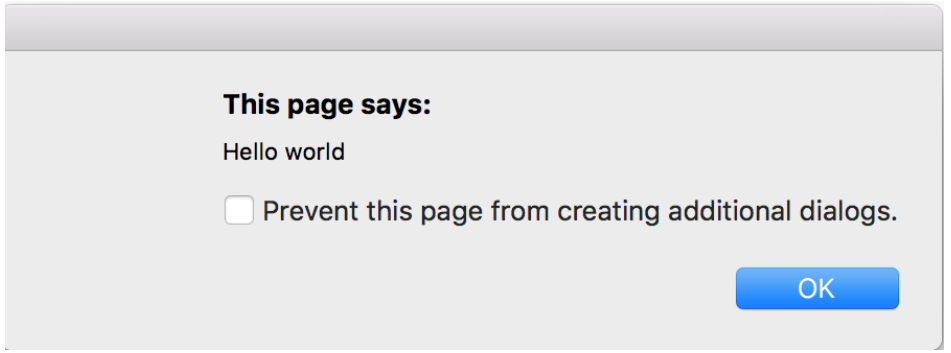
- JavaScript can
  - Implement algorithms like other programming languages
    - has numbers, strings, if..else, loops, functions, arrays
  - React to events like button clicks
  - Validate data
  - Detect the visitor's browser
  - Create cookies
  - Read/write/modify HTML elements

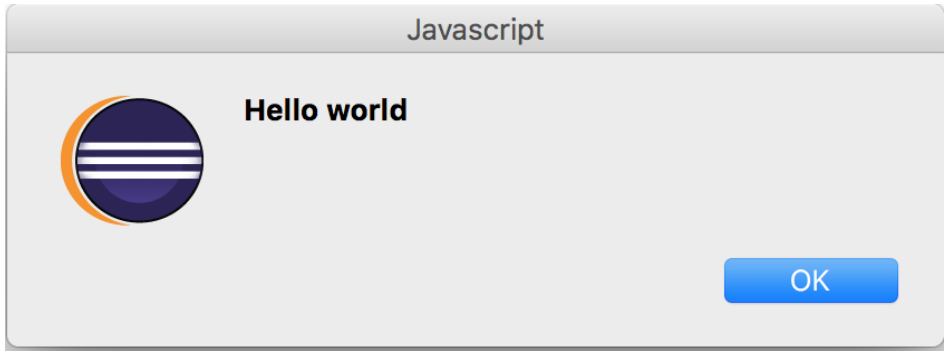
Hello World program

- Can run JavaScript code in a browser (3 browsers shown)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>

<script>
  alert('Hello world');
</script>

</body>
</html>
```

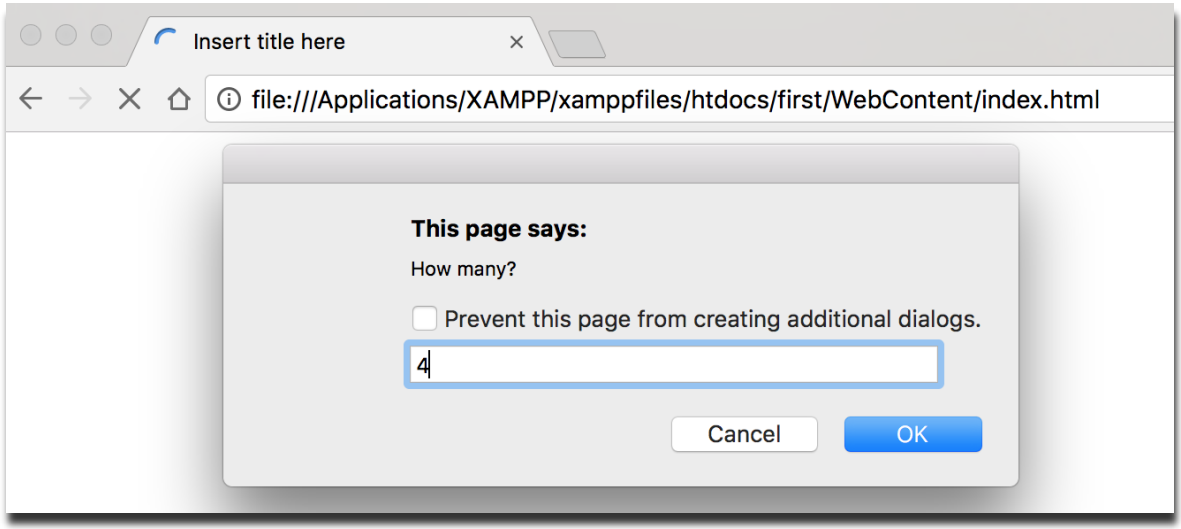


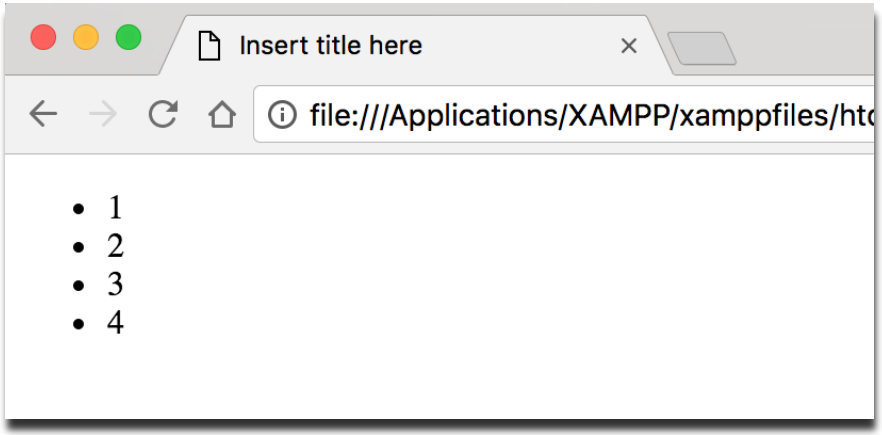




Write to the HTML document (in `<body>`)

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>JS</title>
</head>
<body>
<script>
  var n = prompt("How many?");
  document.write("<ul>");
  for (i = 1; i <= n; i++) {
    document.write("<li>" + i);
  }
  document.write("</ul>");
</script>
</body>
</html>
```





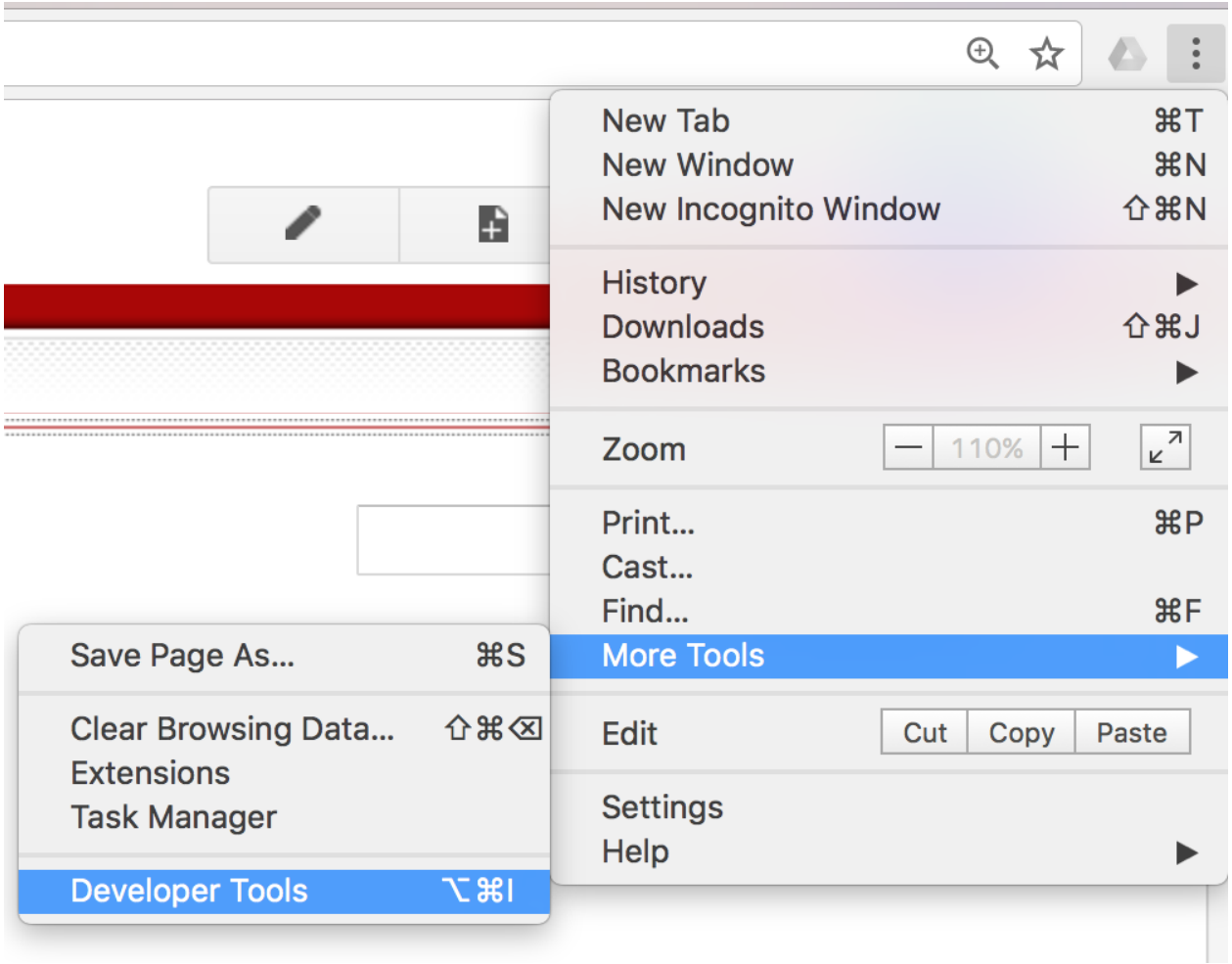
JavaScript Output

- JavaScript can "display" data in different ways:
  - Writing into an alert box, using window.alert()
  - Writing into the HTML output using document.write()
  - Changing an HTML element using innerHTML
  - Writing into the browser console, using console.log()
    - Need Chrome's > Developer Tools > Console
      - Ctrl+Shift+I (Windows)
      - Option+Command+I (Mac)
    - All browsers (except Eclipse) have developer tools to test and debug JavaScript code

Output

- The browser console is very important
  - With JS, we get silent errors: syntax and runtime
  - To see them, open the browser's console window for syntax errors
  - Also can add debugging output
  - You often see no output, but are not told why
- All browsers have developer tools
  - Let us stick with Chrome

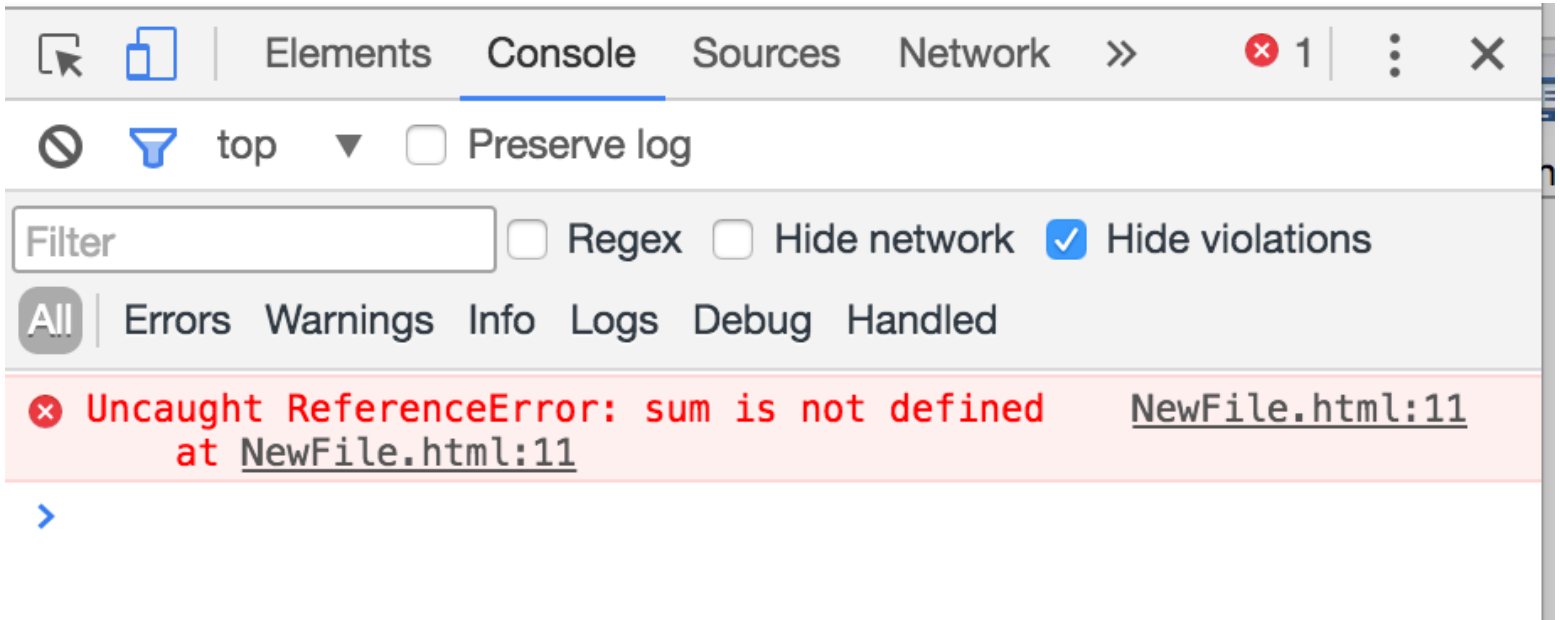
Developer Tools in Chrome



Example JS code and the Console

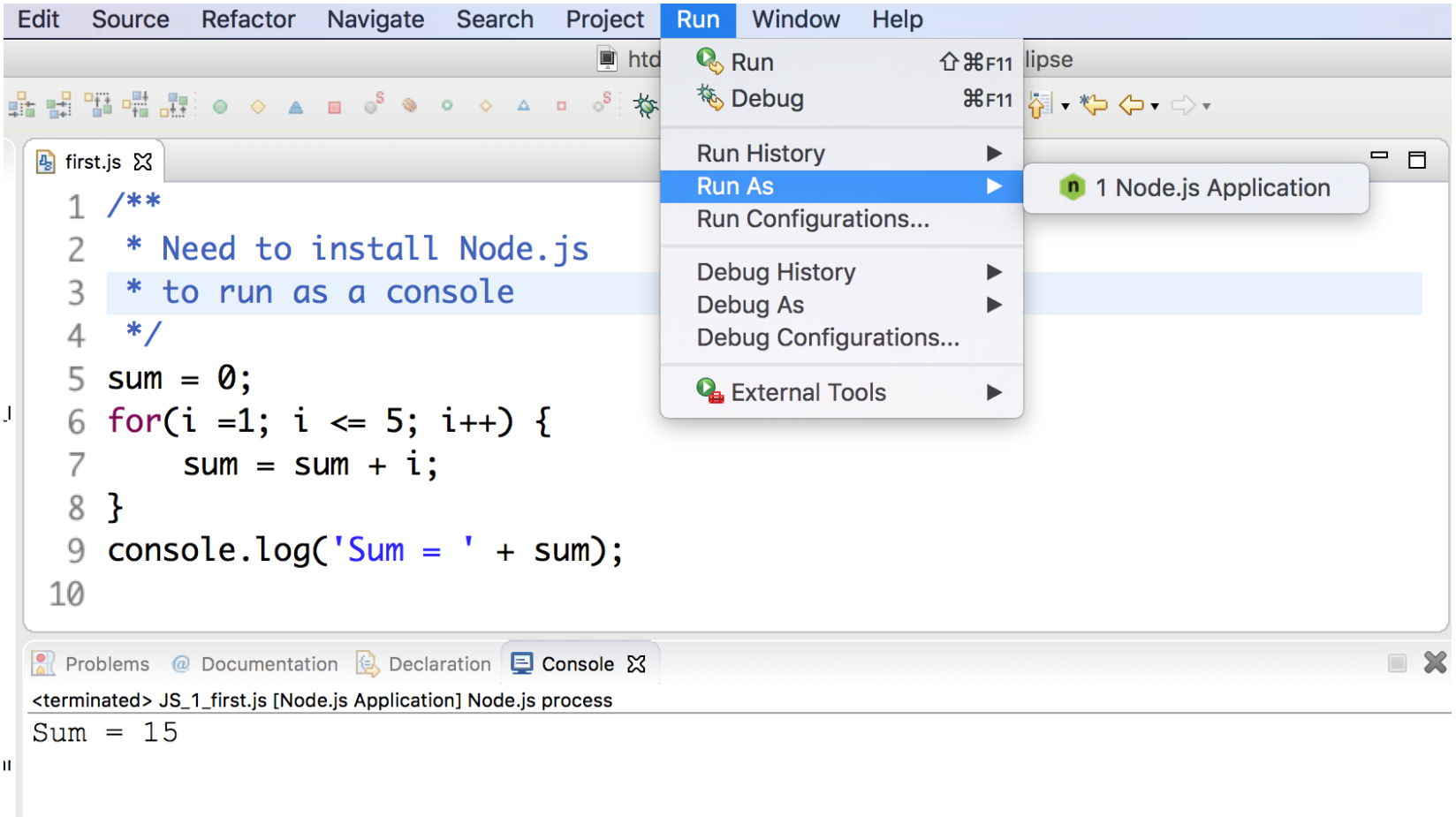
```
<body> <!-- file name loaded NewFile.html has no output -->
<script>
for(i = 1; i <= 5; i++) {
      sum = sum + i;
}
document.write('Sum = ' + sum);
alert('Sum = ' + sum);
</script>
</body>
```

Demo this



Viewing JavaScript output: To Console

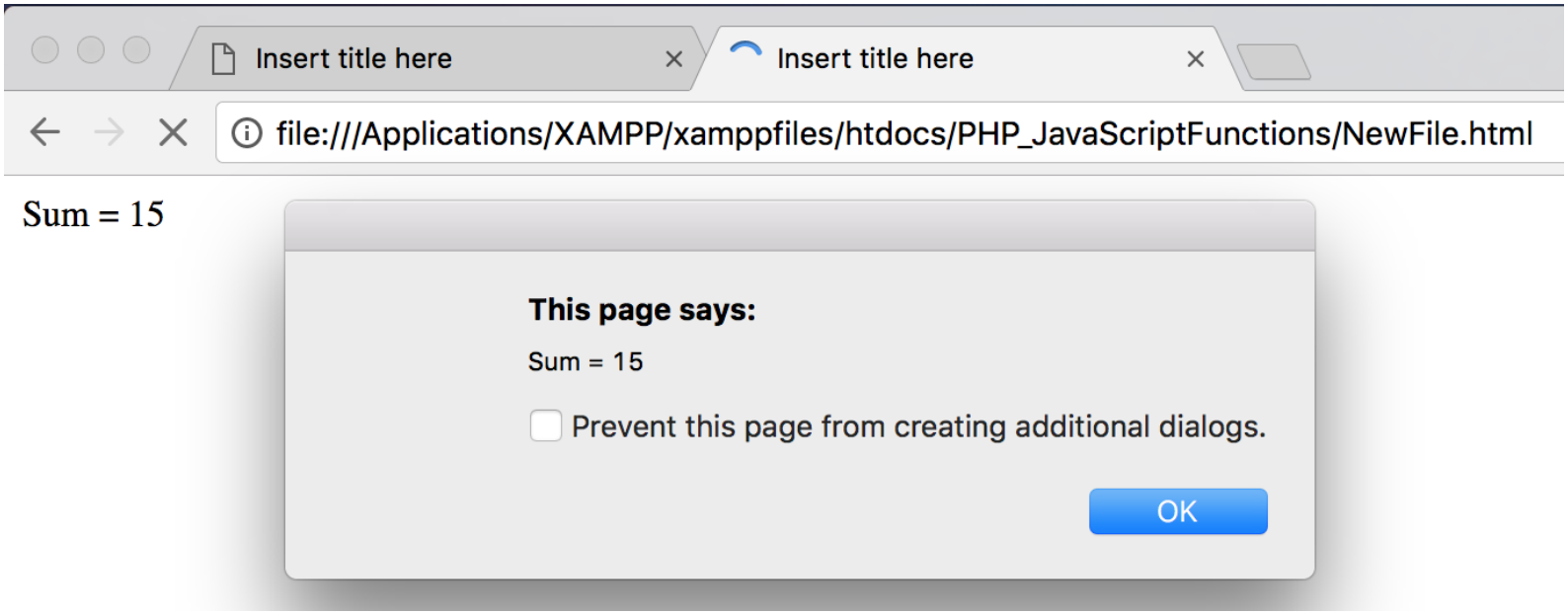
- We could install Node.js and run js files in Eclipse



Viewing JavaScript output: In a Web Page

- We will put JS code in <script></script> an HTML file and load it in a browser

```
<script>
var sum = 0;
for(i =1; i <= 5; i++) {
      sum = sum + i;
}
document.write('Sum = ' + sum);
alert('Sum = ' + sum);
</script>
```



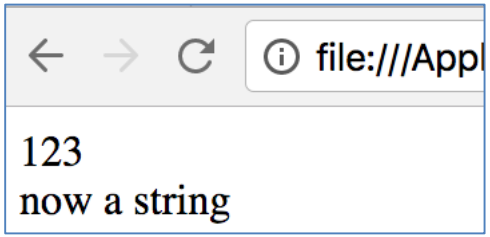
Output from <script> tags in an HTML page

```
<title>JavaScript output</title>
<style>
.simple {
  color: blue;
  border: solid;
}
</style>
</head>
<body>
<script>
document.write('<h2>Hello world</h2>');
document.write('<h4>Smaller text</h4>');
document.write('<p class ="simple">A paragraph</p>');
</script>
</body>
</html>
```

A few JavaScript details

- Case sensitive
- Variable can start with var (or not)
- Statements end with **;** (or not)
- Concatenation with **+** (Like Java)
- Loosely typed, more than Python
- JS has Java-like comments

```
<body>
<script>
/* Multi-line comments
 */
var loose = 123
document.write(loose + '<br>');
//       No semicolon ;
loose = 'now a string'
document.write(loose);
</script>
</body>
```



JavaScript Operators

- Arithmetic operators

  + - * / % ++ --

- Assignment operators

  = += -= *= /= %=

- Concatenation

```
document.write("abc" + "xyz")    // abcxyz

document.write("string and number " + 123 + " --- "
        + 4.56)  // string and number 123 --- 4.56
```

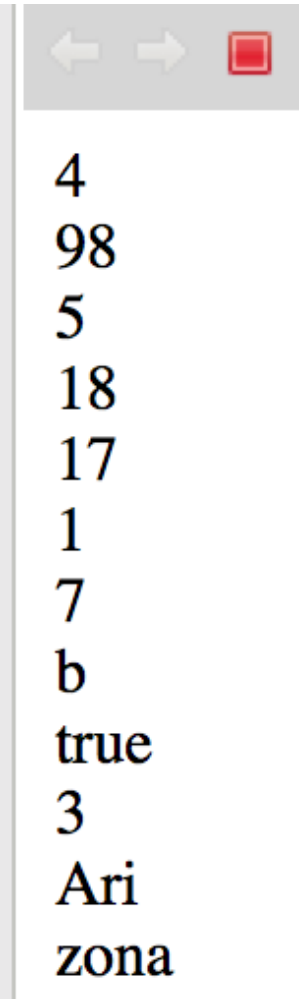
JS Primitive Types using typeof operator

```
document.write(typeof "");   // string

document.write('<br>');
document.write(typeof 123);  // number

document.write('<br>');
document.write(typeof 4.56); // number

document.write('<br>');
document.write(typeof true); // boolean

document.write('<br>');
document.write(typeof { id:"Li", balance:123 } ); // object
```

```
string
number
number
boolean
object
```

A Few Built-in Functions

```
// Math functions
document.write(Math.sqrt(16) + '<br>');                 // 4
document.write(Math.abs(1-99) + '<br>');                // 98
document.write(Math.round(5.4444) + '<br>');            // 5
document.write(Math.ceil(17.000001) + '<br>');          // 18
document.write(Math.floor(17.999999) + '<br>');         // 17
document.write(Math.min(3, 5, 1, 7, 4) + '<br>');       // 1
document.write(Math.max(3, 5, 1, 7, 4) + '<br>');       // 7

// String functions
document.write('abcd'.charAt(1) + '<br>');              // b
document.write('Arizona'.endsWith('zona') + '<br>');    // true
document.write('Arizona'.indexOf('zona') + '<br>');     // 3
document.write('Arizona'.substring(0, 3) + '<br>');     // Ari
document.write('Arizona'.substring(3, 99) + '<br>');    // zona
```



If statements

```
var num = 124;
if(num < 0 || num > 100)
   document.write('Out of range');
else
   document.write('In range');
```

```
var grade = 72;
if (grade >= 90)
      document.write("A");
else if (grade >= 80)
      document.write("B");
else if (grade >= 70)
      document.write("C");
else if (grade >= 60)
      document.write("D");
```

Loops

```
for(var count = 1; count <= 3; count ++) {
  document.write('pow(count, 2) = '+ Math.pow(count, 2 ) + '<br>');
}
```

```
pow(count, 2) = 1
pow(count, 2) = 4
pow(count, 2) = 9
```

```
var str = 'Arizona';
var i = 0;
while ( str.charAt(i) != 'z' ) {
  document.write(str.substring(i) + "<br>");
  i++;
}
```

```
Arizona
rizona
izona
```

JavaScript Functions

```
function name(parameterName, ..., parameterName) {
  statements;
}
```

```
document.write('BMI at 160 pounds, 70 inches: ' + bmi(160, 70) + "<br>");
function bmi(weight, height) {
  result = 703 * weight / height / height;
  return result;
}
document.write( 'BMI at 160 pounds, 68 inches: ' + bmi(160, 68));
```

```
BMI at 160 pounds, 70 inches: 22.955102040816328
BMI at 160 pounds, 68 inches: 24.325259515570934
```

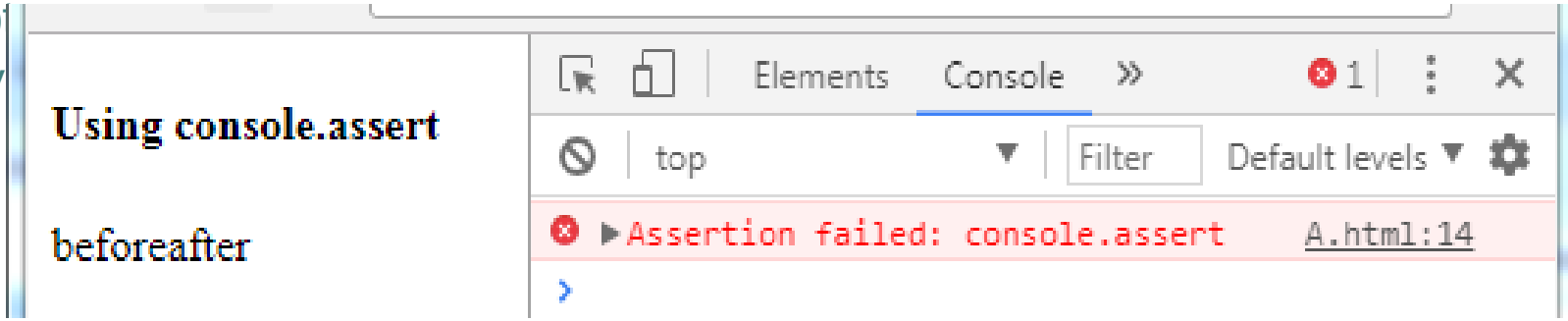
- Parameter types and return types are not needed
- A function with no return statements is "void"
- Declared in any PHP block, start/end/middle

More JavaScript

console.assert

- We could print results and examine the output
  - Not consistent, can take a long time to read them
- Or test with console.assert(Boolean)
  - If the Boolean expression is false, console output appears because either the test is wrong or the function is wrong

```
 9 <body>
10 <h4>Using console.assert</h4>
11 <script>
12 document.write('before');
13 console.assert(5 == 1+4);
14 console.assert(6 == 1+4);
15 document.write('after');
16 </script
17 </body
```

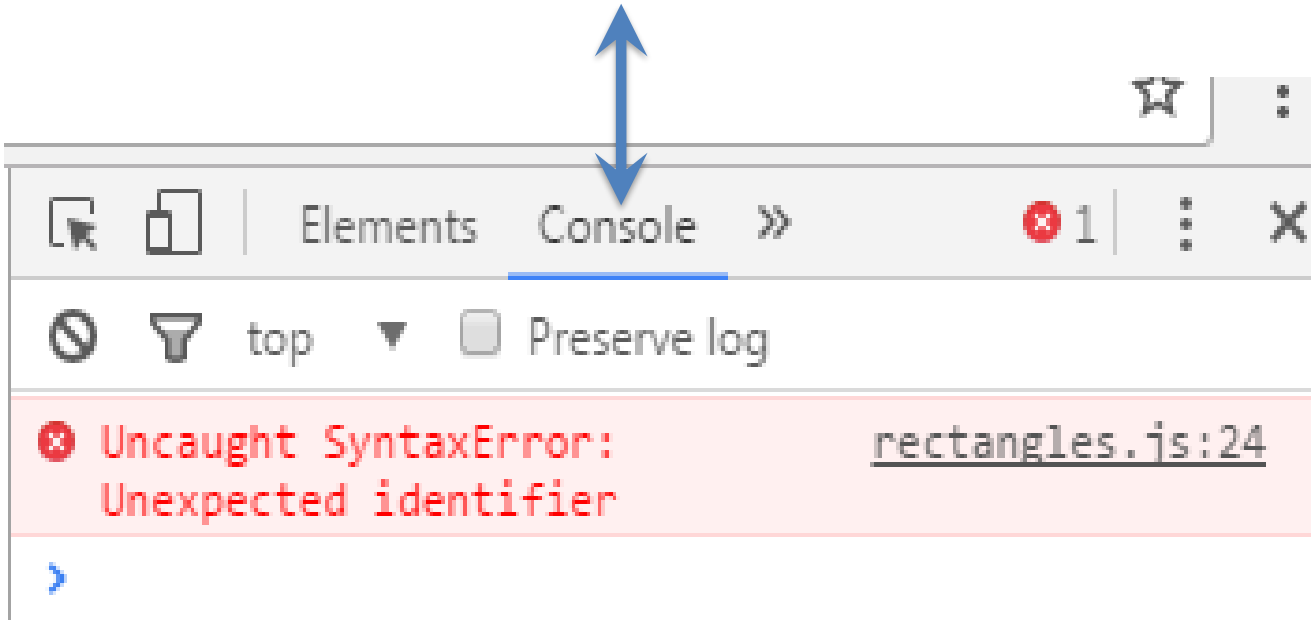


Testing JavaScript Functions

```
<script>
function numberOf(ch, str) {
  var result = 0;
  for(var i = 0; i < str.length; i++) {
     if(ch == str.charAt(i))
     result ++;
  }
  return result;
}
// Look at the console, there should be no errors.
// Use clear() in the console to clear console between changes.
console.assert(0 == numberOf('x', 'abcde'));
console.assert(1 == numberOf('x', 'abcdx'));
console.assert(1 == numberOf('x', 'xbcde'));
console.assert(2 == numberOf('x', 'xbcdx'));
console.assert(3 == numberOf('x', 'xbxdx'));
console.assert(5 == numberOf('x', 'xxxxx'));
console.assert(0 == numberOf('', ''));

</script>
```

How do you comfort a JavaScript bug?

You console it ☺



JavaScript Arrays

- Can create arrays on the fly
  - Not with { } as in Java. JavaScript uses [ ]

```
var array = [ 'abc', 'def', 'ghi' ];
console.log(array[0]);
console.log(array[1]);
console.log(array[2]);
console.log();  // Blank line
for(index = array.length - 1; index >=0; index--)
    console.log(array[index]);
```

Output

```
abc
def
ghi

ghi
def
abc
```

```
> var x = [1, 2, 3];
<· undefined
> for(i = 0; i < x.length; i++)
      console.log(x[i]);
  1
  2
  3
<· undefined
>
```

JS Arrays

- Use the length property for the number of elements

```
var array = [ 123, 'def', true];
console.log(array.length); // 3
```

- No subscript array checking

```
var array = [ 123, 'def', true];
array[6] = ['Out there with holes to the left']
for(var i = 0; i < array.length; i++) {
  document.write(array[i] + "<br>");
}
```

123
def
true
undefined
undefined
undefined
Out there with holes to the left

- Can use pop and push to add and remove elements

JS Arrays: push and pop

- Can use pop and push to add and remove elements like a stack

```
<body>
<script>
var array = [ ];
array.push(1);
array.push(2);
array.push(3);
var top = array.pop();
array.push(4);
for(var i = 0; i < array.length; i++) {
  document.write(array[i] + "<br>");
}
document.write('Popped ' + top);
</script>
</body>
</html>
```

Output?

Pass by value when the value is a reference

- A change to an array in a function changes the argument
  - This means you can change the parameter instead of returning a modified or new array

```
function changeMe(param) {
  for (i = 0; i < param.length; i++)
      param[i] += 77;
}

var arg = [ 1, 2, 3 ];
console.log(arg);
changeMe(arg);
console.log(arg);
```

Output

```
[ 1, 2, 3 ]
[ 78, 79, 80 ]
```

Parameters

- Like Java: pass by value always

```
function changeMe(param) {  // or try to change argument
     param = 456;
}

var argument = 123;
console.log('argument before: ' + argument);
changeMe(argument);
console.log(' argument after: ' + argument);
```

Output:

```
argument before: 123
 argument after: 123
```

Parameters

- Just like java, if the argument is an object,
  - a change to the parameter can change the Object argument

```
function changeMe(param) {
    param.balance += 222.22;
}

// An object referenced by bankAccount
var bankAccount = { id:"Chris", balance:123.45 };
console.log('before: ' + bankAccount.balance);
changeMe(bankAccount);
console.log(' after: ' + bankAccount.balance);
```

Output

```
before: 123.45
 after: 345.67
```

Strings are immutable

- Like java, string objects can not be changed
  - A replace message ***returns*** a new string

```
function changeMe(param) {
   var str = param.replace('org', 'new');
   return str;
}

var string = 'orginal' ;
console.log('before: ' + string);
changeMe(string);
console.log(' after: ' + string);

var newStr = changeMe(string);
console.log(newStr);
```

Output

```
before: orginal
 after: orginal
newinal
```