JavaScript -- Objects

Object-Oriented Concept

An *object* is a custom data type that data with functions to act upon it. The data items in an object are its *types* or *properties*, and the functions are its *methods*.

Defining JavaScript Object

```
function Card(name,address,work,home)
{
  this.name = name;
  this.address = address;
  this.work_phone = work;
  this.home_phone = home;
}
```

Defining Methods

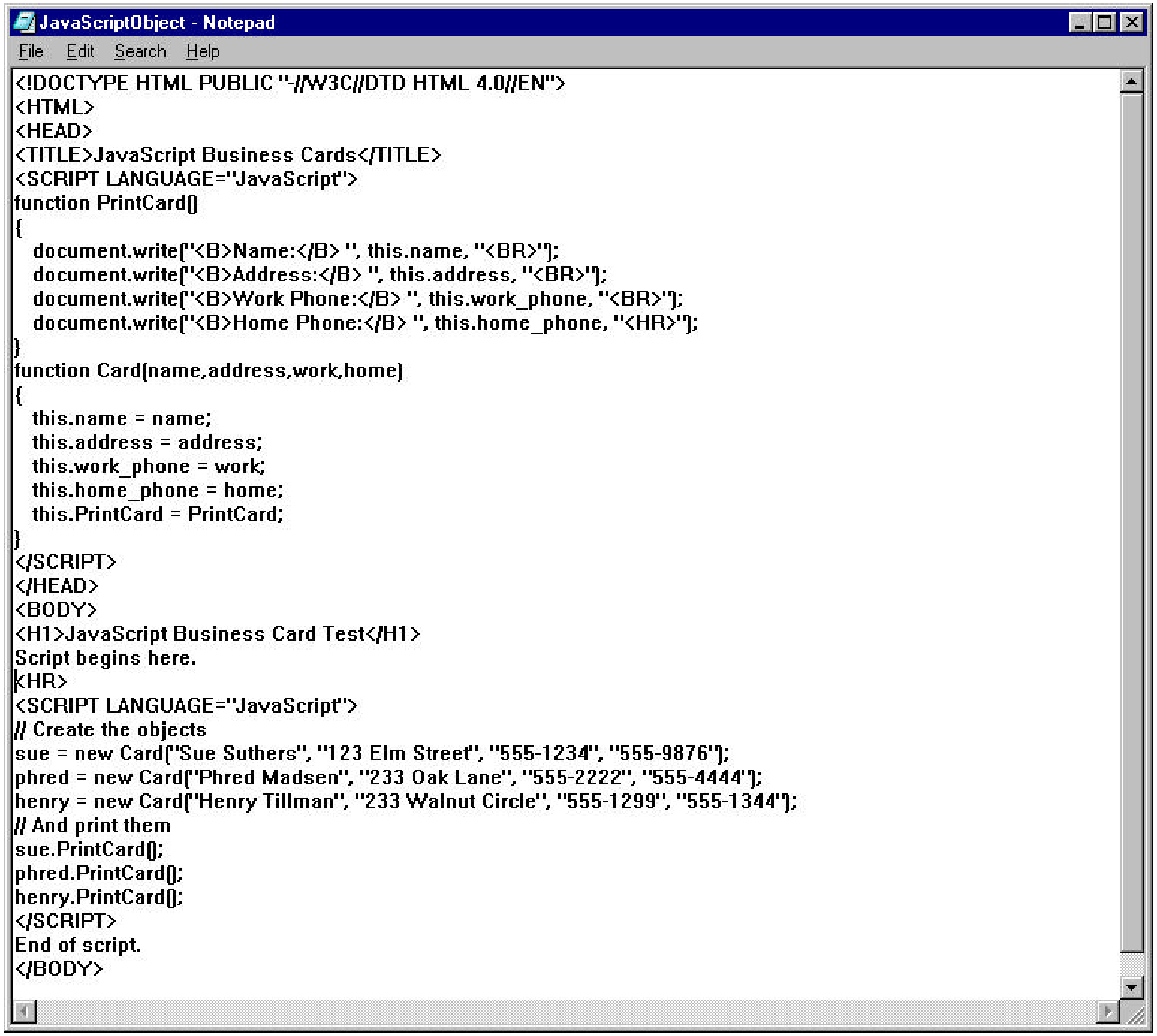
```
function PrintCard()
{
  document.write("Name: ", this.name, "\n");
  document.write("Address: ", this.address, "\n");
  document.write("Work Phone: ", this.work_phone, "\n");
  document.write("Home Phone: ", this.home_phone, "\n");
}
```

Creating Instances of Objects

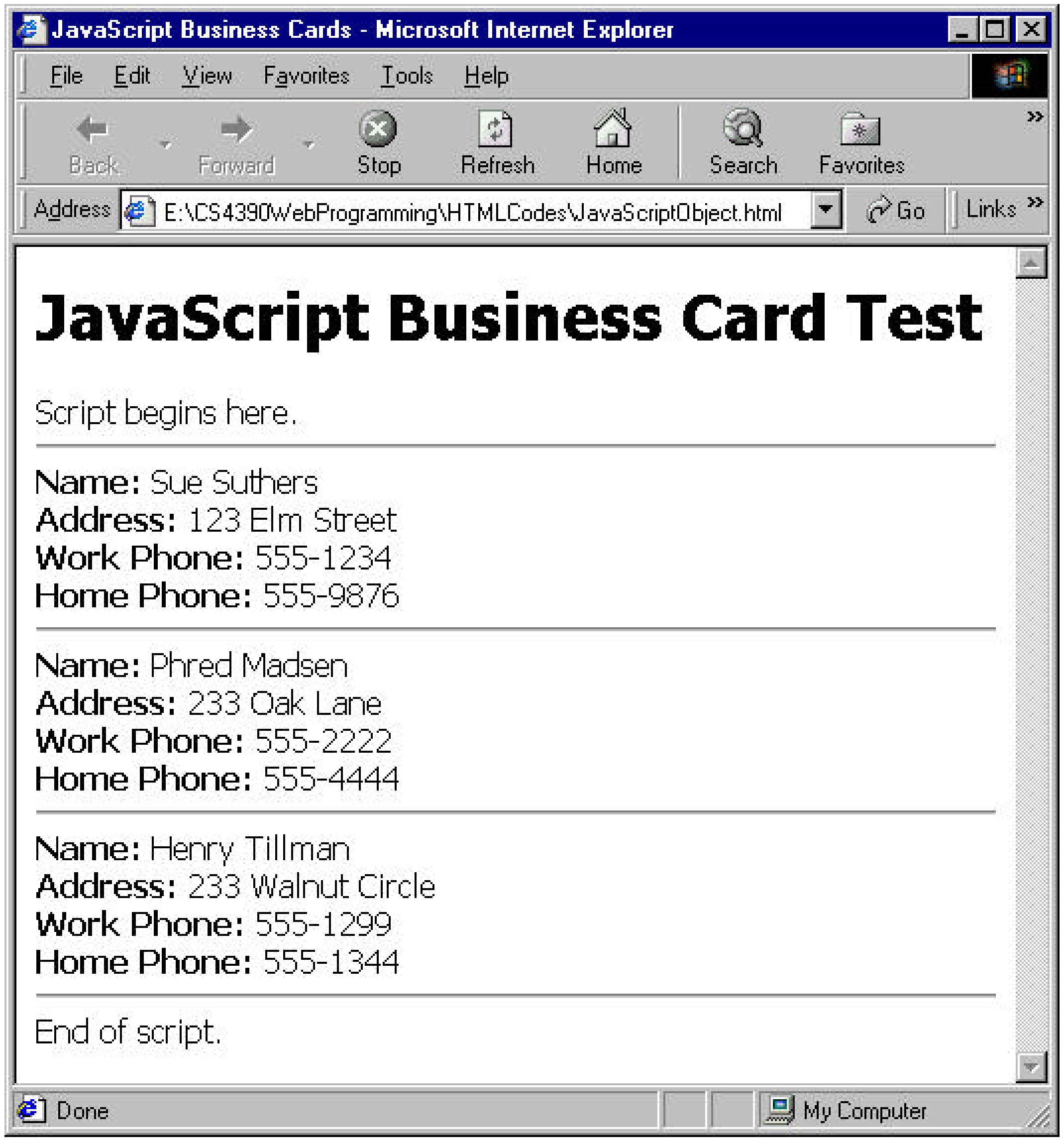
```
tom = new Card("Tom Jones", "123 Elm Street", "555-1234", "555-9876");
```

- The method `PrintCard` can be called –

```
tom.PrintCard();
```



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<TITLE>JavaScript Business Cards</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function PrintCard()
{
  document.write("<B>Name:</B> ", this.name, "<BR>");
  document.write("<B>Address:</B> ", this.address, "<BR>");
  document.write("<B>Work Phone:</B> ", this.work_phone, "<BR>");
  document.write("<B>Home Phone:</B> ", this.home_phone, "<HR>");
}
function Card(name,address,work,home)
{
  this.name = name;
  this.address = address;
  this.work_phone = work;
  this.home_phone = home;
  this.PrintCard = PrintCard;
}
</SCRIPT>
</HEAD>
<BODY>
<H1>JavaScript Business Card Test</H1>
Script begins here.
<HR>
<SCRIPT LANGUAGE="JavaScript">
// Create the objects
sue = new Card("Sue Suthers", "123 Elm Street", "555-1234", "555-9876");
phred = new Card("Phred Madsen", "233 Oak Lane", "555-2222", "555-4444");
henry = new Card("Henry Tillman", "233 Walnut Circle", "555-1299", "555-1344");
// And print them
sue.PrintCard();
phred.PrintCard();
henry.PrintCard();
</SCRIPT>
End of script.
</BODY>
```


JavaScript Business Cards - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites
Address E:\CS4390WebProgramming\HTMLCodes\JavaScriptObject.html
Go Links
Done
My Computer

JavaScript Business Card Test

Script begins here.

Name: Sue Suthers
Address: 123 Elm Street
Work Phone: 555-1234
Home Phone: 555-9876

Name: Phred Madsen
Address: 233 Oak Lane
Work Phone: 555-2222
Home Phone: 555-4444

Name: Henry Tillman
Address: 233 Walnut Circle
Work Phone: 555-1299
Home Phone: 555-1344

End of script.

Defining Object within Object

```
function Address(street1, street2, city, state, zip)
{
    this.street1 = street1;
    this.street2 = street2;
    this.city = city;
    this.state = state;
    this.zip = zip;
}
```

Create an `Address` object --

```
   tomaddr = new Address("123 Elm Street", "Apartment 312",
      "Ogden", "UT", "84404");
```

Create an `Address` object --

```
   tom = new Card("Tom Smith", tomaddr, "555-1239",
       "555-2394");
```

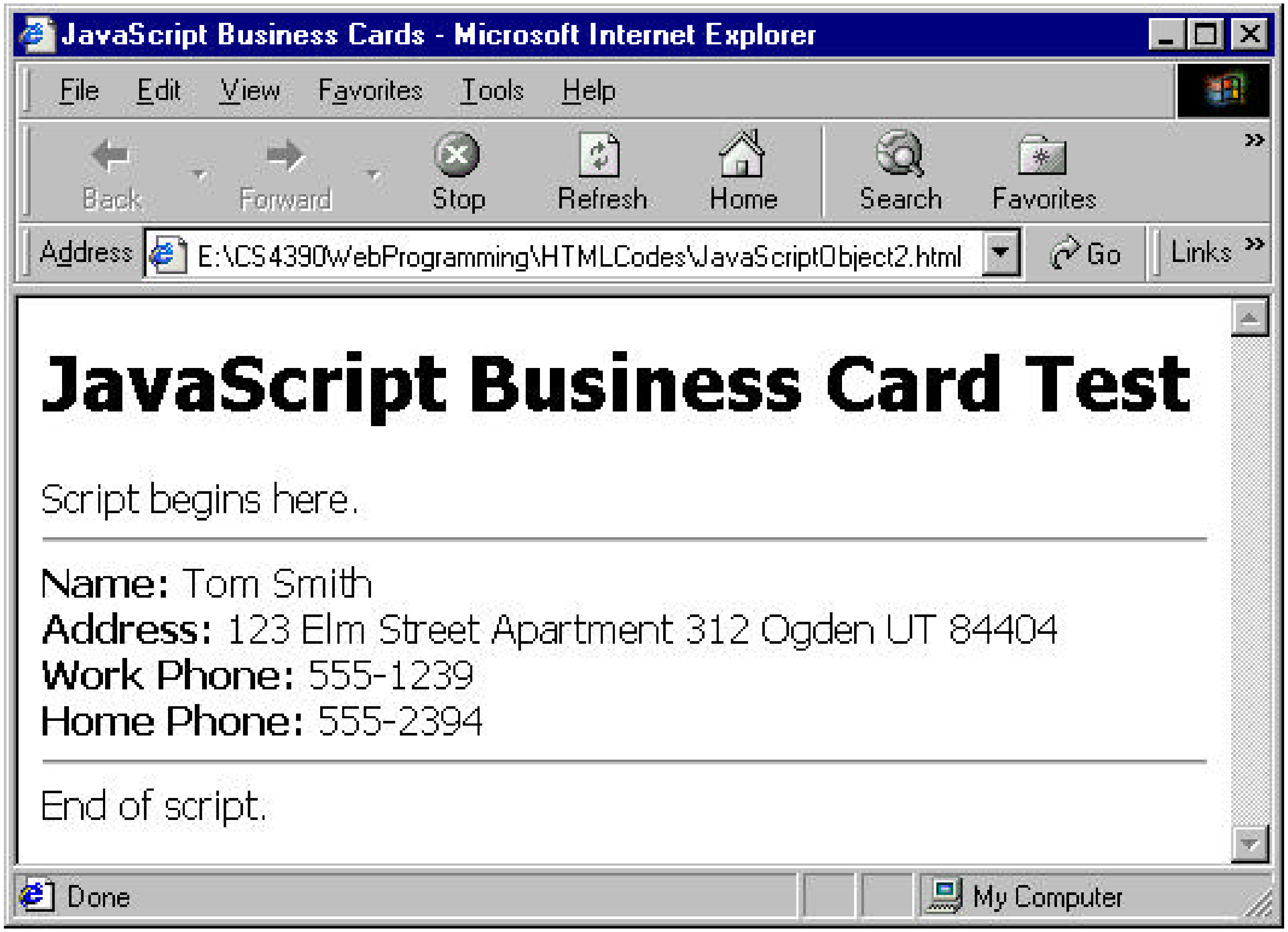
Defining Method within Method

```
function Address(street1, street2, city, state, zip)
{
    this.street1 = street1;
    this.street2 = street2;
    this.city = city;
    this.state = state;
    this.zip = zip;
    this.PrintAddress = PrintAddress;
}

function PrintAddress()
{
   document.write(this.street1, "\n");
   document.write(this.street2, "\n");
   document.write(this.city, "\n");
   document.write(this.state, "\n");
   document.write(this.zip, "\n");
}
```

Defining Method within Method

```
function PrintCard()
{
   document.write("<B>Name:</B> ", this.name, "<BR>");;
   document.write("<B>Address:</B> ");
   this.address.PrintAddress();
   document.write( "<BR>");
   document.write("<B>Work Phone:</B> ", this.work_phone, "<BR>");
   document.write("<B>Home Phone:</B> ", this.home_phone, "<HR>");
}
.
.
.
tom.PrintCard();
```


JavaScript Business Cards - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites
Address E:\CS4390WebProgramming\HTMLCodes\JavaScriptObject2.html
Go Links
Done
My Computer

JavaScript Business Card Test

Script begins here.

Name: Tom Smith
Address: 123 Elm Street Apartment 312 Ogden UT 84404
Work Phone: 555-1239
Home Phone: 555-2394

End of script.

Built-in Objects

- `Array` objects -- store numbered variables.
- String objects -- manipulate strings of characters.
- Date objects -- to store and work with dates.
- Math object -- methods and properties for mathematical functions.
- navigator object -- store information about the user's browser and its capabilities.

Array Object

- Create an array object --

  `employees = new Array(30);`

- Array object has a single property –

  `length`

  `Employees.lenghth;`

Array Object -- Methods

- **`join()`** -- joins all the array's elements, resulting in a string. The elements are separated by commas.
- **`reverse()`** -- returns a reversed version of the array: the last element becomes the first, and the first element becomes the last.
- **`sort()`** returns a sorted version of the array. Normally, this is an alphabetical sort; however, you can use a custom sort method by specifying a comparison routine.

Example

```
employees = new Array(4);
employees[0]="tom";
employees[1]="bob";
employees[2]="bill";
employees[3]="dave";
document.writeln(employees.join());
document.writeln(employees.reverse().join());
document.writeln(employees.sort().join());
```

String Object

- Create an array object --
  ```
  name = new String(25);
  ```
- Array object has a single property –
  ```
  length
  name.lenghth;
  ```

String Object -- Methods

- **String Conversion**
  - `toUpperCase()` -- converts all characters in the string to uppercase.
  - `toLowerCase()` -- converts all characters in the string to lowercase.
- **Substring**
  - `toString()` -- converts to string.
  - `subString(*,*)` -- returns a string consisting of a portion of the original string between two index values.

Example

```
alpha = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
```

`alpha.substring(0,4);` **returns "ABCD".**

`alpha.substring(10,12);` **returns "KL".**

`alpha.substring(12,10);` **also returns "KL". Because it's smaller, 10 is used as the first index.**

`alpha.substring(6,7);` **returns "G".**

`alpha.substring(24,26);` **returns "YZ".**

`alpha.substring(0,26);` **returns the entire alphabet.**

`alpha.substring(6,6);` **returns the null value, an empty string. This is true whenever the two index values are the same.**

String Object -- Methods

- **Substring**
  - `charAt(*)` -- returns a single character.
  - `split()` -- splits a string into an array of strings, based on a separator you specify.
- **Search string**
  - `indexOf("string")` -- searches for a string within another string. (option: starting index)
  - `lastIndexOf(*,*)` -- finds the *last* occurrence of the string. It searches the string backwards, starting with the last character.

Example

```
alpha = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
```

`alpha.charAt(0);` **returns "A".**

`alpha.charAt(12);` **returns "M".**

`alpha.charAt(25);` **returns "Z".**

`alpha.charAt(27);` **returns nothing.**

`name = "John Q. Public".split(".");` **returns "John Q,Public".**

`name = "John Q. Public".split(" ");` **returns "John,Q.,Public".**

`alpha.indexOf("GHI");` **returns 6.**

String Object -- Methods

- **Changing string appearance**
  - **`String.big()`**-- displays big text, using the `<BIG>` tag in HTML 3.0.
  - **`String.small()`**-- displays small letters, using the `<SMALL>` tag in HTML 3.0.
  - **`String.blink()`**-- displays blinking text, using the `<BLINK>` tag in Netscape.
  - **`String.bold()`**-- displays bold text, using the `<B>` tag.
  - **`String.italics()`**-- displays the string in italics, using the `<I>` tag.

Changing string appearance

- **`String.fixed()`**-- displays fixed-font text, using the `<TT>` tag.
- **`String.fontcolor()`**-- displays the string in a colored font, equivalent to the `<FONTCOLOR>` tag in Netscape.
- **`String.fontsize()`**-- changes the font size, using the `<FONTSIZE>` tag in Netscape.
- **`String.strike()`**-- displays the string in a strike-through font, using the `<STRIKE>` tag.
- **`String.sub()`**-- displays subscript text, equivalent to the `<SUB>` tag in HTML 3.0.
- **`String.sup()`**-- displays superscript text, equivalent to the `<SUP>` tag in HTML 3.0.

String Object -- Methods

- **Links and Anchors**
  - **`String.link()`**-- produce HTML for links

    ```
    "This is a Test".anchor("test");
    ```
  - **`String.anchor()`**-- produce HTML for anchors

    ```
    "Click Here".link("#test");
    ```

Date Object

- Create an array object --

```
birthday = new Date();  
```
set the current date

```
birthday = new Date("June 20, 1996
  08:00:00");
birthday = new Date(6, 20, 96);
birthday = new Date(6, 20, 96, 8, 0, 0);
```

- Array object has a single property – `length`

```
name.lenghth;
```

Date Object -- Methods

- **Getting Date Values**
  - **`getDate()`** -- gets the day of the month.
  - **`getMonth()`** -- gets the month.
  - **`getYear()`** -- gets the year.
  - **`getTime()`** -- gets the time (and the date) as the number of milliseconds since January 1st, 1970.
  - **`getHours(), getMinutes(), getSeconds()`** -- gets the time.

Date Object -- Methods

- **Working with Time Zones**
  - **`getTimeZoneOffset()`** -- gives you the local time zone's offset from GMT (Greenwich Mean Time).
  - **`toGMTString()`** -- converts the date object's time value to text, using GMT.
  - **`toLocalString()`** -- converts the date object's time value to text, using local time.

Date Object -- Methods

- **Converting Between Date Formats**
  - **`Date.parse()`** -- converts a date string, such as "June 20, 1996" to a Date object (number of milliseconds since 1/1/1970).
  - **`Date.UTC()`** -- it converts a Date object value (number of milliseconds) to a UTC (GMT) time.

Math Object

- The Math object's properties represent mathematical constants, and its methods are mathematical functions.

Math Object -- Methods

- **Rounding and Truncating**
  - `Math.ceil()` -- rounds a number up to the next integer.
  - `Math.floor()` -- rounds a number down to the next integer.
  - `Math.round()` -- rounds a number to the nearest integer.

navigator Object

- The navigator object is a special object that stores information about the version of the browser.
- This is a Netscape-specific tag, and it may or may not be implemented in other browsers.
  - **`navigator.appCodeName`** is the browser's code name, usually "Mozilla".
  - **`navigator.appName`** is the browser's name, usually "Netscape".
  - **`navigator.appVersion`** is the version of Netscape being used-for example, "2.0(Win95;I)".
  - **`navigator.userAgent`** is the user-agent header, which is sent to the host when requesting a Web page. It includes the entire version information, "Mozilla/2.0(Win95;I)".

navigator Object

- `navigator.javaEnabled` indicates whether Java is enabled in the browser.
- `navigator.plugIns` is a list of the currently available plug-ins.
- `navigator.mimeTypes` is a list of the currently available MIME types.