Exception Handling

TRY CATCH

```
try {
   // Statements 1
} catch (ex) {
   // Statements 2
}
```

- Code in "Statements 1" is executed until error occurs ("exception")
- Code does not top on error, but continues with "Statements 2"
- "Throwing" of own exceptions: `throw("My Exception");`
- Important Concepts here:
  - "Design by Contract"
  - "Graceful Degradation"

Exercises 4.3 - 4.6

Loops

Check condition first

WHILE

```
while (Condition) {
    Statements;
}
```

Check condition after first iteration

DO WHILE

```
do {
    Statements;
} while (Condition);
```

Self-managed iteration

FOR

```
for ([Start Expression]; [Condition];
     [Iteration Step]) {
    Statements;
}
```

Iteration through collection (array) of object

FOR IN

```
for (Property_Name in Object) {
    Statements;
}
```

Examples: Loops

```
var i = 0;
while (i < 10) {
   document.write(i);
   i++;
}
```

```
var i = 0;
do {
   document.write(i);
   i++;
} while (i < 10);
```

```
for (var i=0, s=""; s.length<=10; i++, s += i) {
   document.write(s + "<br />");
}
```

Examples: Loops

```
var course;
var myCourses = new Array();
myCourses[0] = "Web Engineering I";
myCourses[1] = "Web Engineering II";
myCourses[2] = "Java Programming";

for (course in myCourses)
{
document.write(myCourses[course] + "<br
   />");
}
```

Functions

function

```
function myFunction (Param1, Param2, …) {
    Statements;
    return Value;
}
```

- Function parameters are untyped
- Must include return statement
- Support grouping and reuse of code
- Code simplification
- Improves code readability

```
function add(val1, val2) {
 return val1 + val2;
};
```

Eval Function

- Evaluates the provided text as a command
- "Statements" or command can be composed of strings

Example

```
var car1 ="BMW";
var car2 ="VW";
var car3="Volvo";

for(i=0;i<3;i++)
  document.write( eval("car"+(i+1)) + "<br/>" );
```

Objects in JavaScript

- JavaScript is object-oriented
- Objects can be
  - language elements (e.g., String),
  - browser objects (e.g., navigator), or
  - self-defined objects
- Are created using the **`new()`** function
- Have properties (data) and methods (functionality or "behavior")

JavaScript Objects

String Object

- Provides methods such as
  - bold() - Displays a string in bold
  - charAt() - Returns the character at a specified position
  - link() - Displays a string as a hyperlink
  - replace() - Replaces some characters with some other characters in a string
  - slice() - Extracts a part of a string and returns the extracted part in a new string
  - toLowerCase() - Displays a string in lowercase letters
- Property (selected)
  - length - Returns the number of characters in a string

JavaScript Objects

Array Object

- Provides methods such as
  - concat() - Joins two or more arrays and returns the result
  - pop() - Removes and returns the last element of an array
  - reverse() - Reverses the order of the elements in an array
  - sort() - Sorts the elements of an array
  - toString() - Converts an array to a string and returns the result
- Property (selected)
  - length - Sets or returns the number of elements in an array

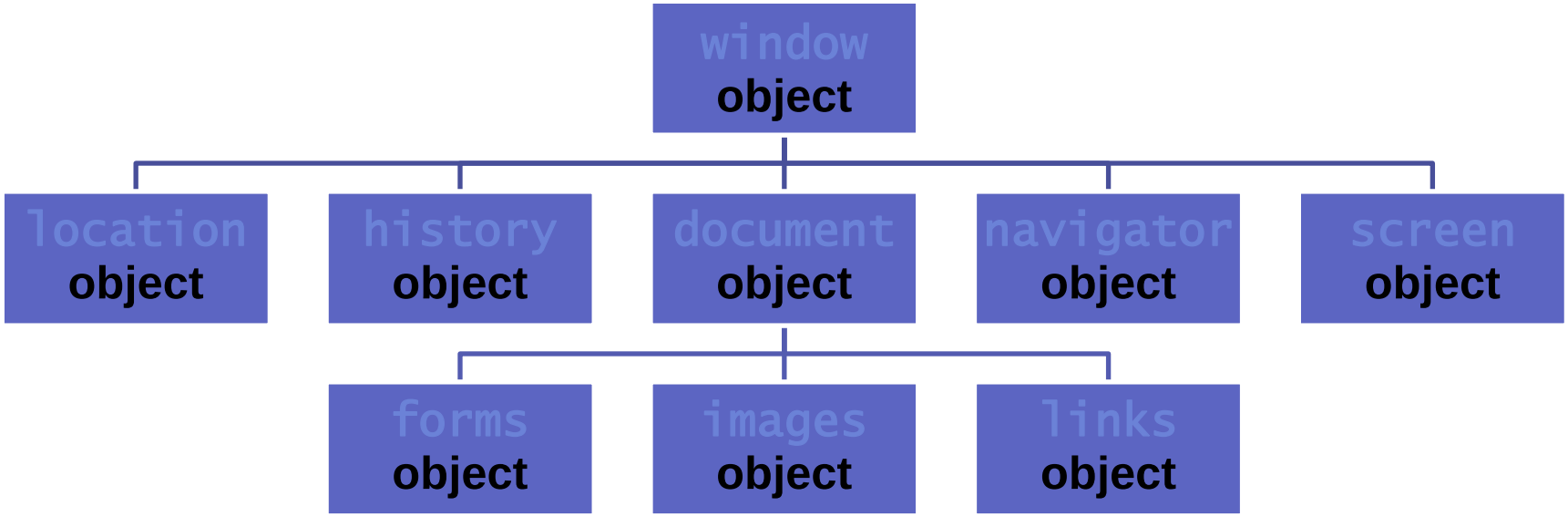
Other JavaScript Objects

- Date – representation of data and time
- Boolean – conversion of non-boolean values to booleans
- Math – mathematical functions
- RegExp – text search by regular expressions
- HTML DOM – dynamic access to and manipulation of HTML (will be covered later)

Browser Objects

- Window – represents the browser window
- Navigator – information of the client's browser
- Screen – information about the client's display
- History – visited links in the browser window
- Location – information about the current URL

Standard Browser Object Hierarchy



will be discussed in detail later…

Self-defined Objects

- 2 ways to create objects:
  - Creation of direct instance

Example

```
myObject = new Object();
myObject.name = "Peter Parker";
myObject.ZIP  = "12345";
```

  - Creation of a template ("class")

Example

```
function ba_student(firstname,lastname,employer)
{
this.firstname=firstname;
this.lastname=lastname;
this.employer=employer;
}
```

Object Orientation

- Inheritance by using the prototype property

Example

```
function A() {
this.prop1 = "123";}

function B() {
this.prop2 = "456";
}
B.prototype = new A;
```

Obtaining Browser Information

- Browser (and client) information is stored in navigator object
- Navigator object exposes an array of properties (attributes) providing information
- Identifying a browser allows for a provision of browser-specific code

Example

```
document.write( navigator.appName);
// returns, e.g., "Microsoft Internet Explorer"
```

The Window Object

- Supports showing modal dialog boxes
- [window.]alert(<text>) – alert information
- [window.]confirm(<text>) – user choice YES/NO
- [window.]prompt(<text>, <default_value>) – user is prompted for a value

Example

```
var filename;
  filename = prompt("Which file should be deleted?",
  "<nothing>");
  var confirmed;
  confirmed = confirm("Delete file?");
  if (confirmed == true)
   {
     //delete file
     window.alert("File " + filename + " deleted.");
   }
```

Opening a new Window

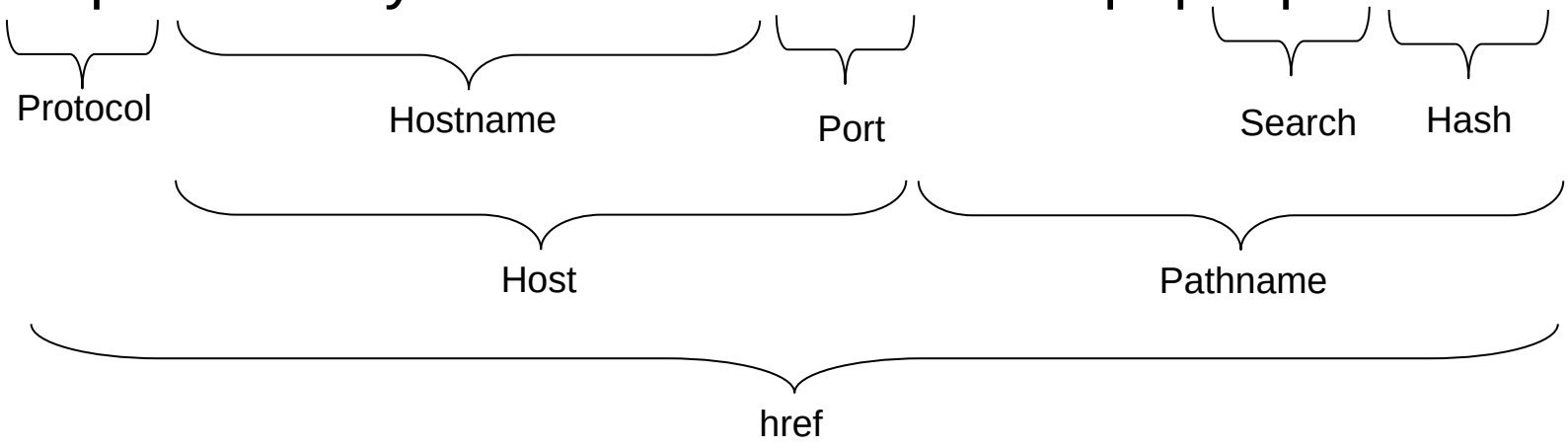
- Opens a URL in a new browser window (blocked by pop-up filters)
- Behavior differs between browsers
- Can set properties (e.g., alwaysRaised or fullscreen) for new window

Example

```
var mywin = window.open("http://www.google.de",
 "Google", "width=400,status=1");
```

The JavaScript location Object

- http://www.mydomain.com:8080/index.php?open#Test



Example

```
location.href = "http://www.google.de";
```

Exercises 4.7 – 4.9