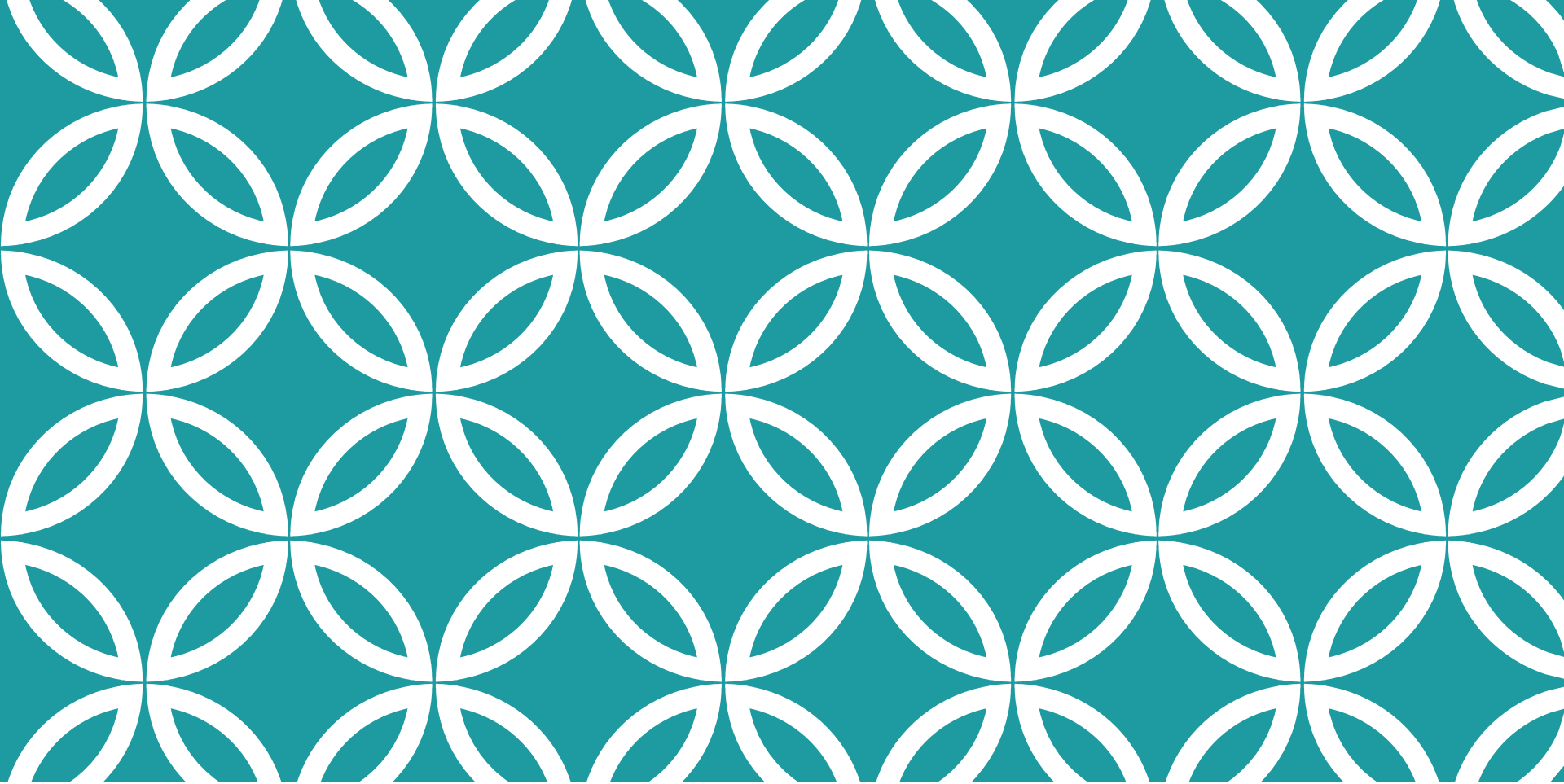
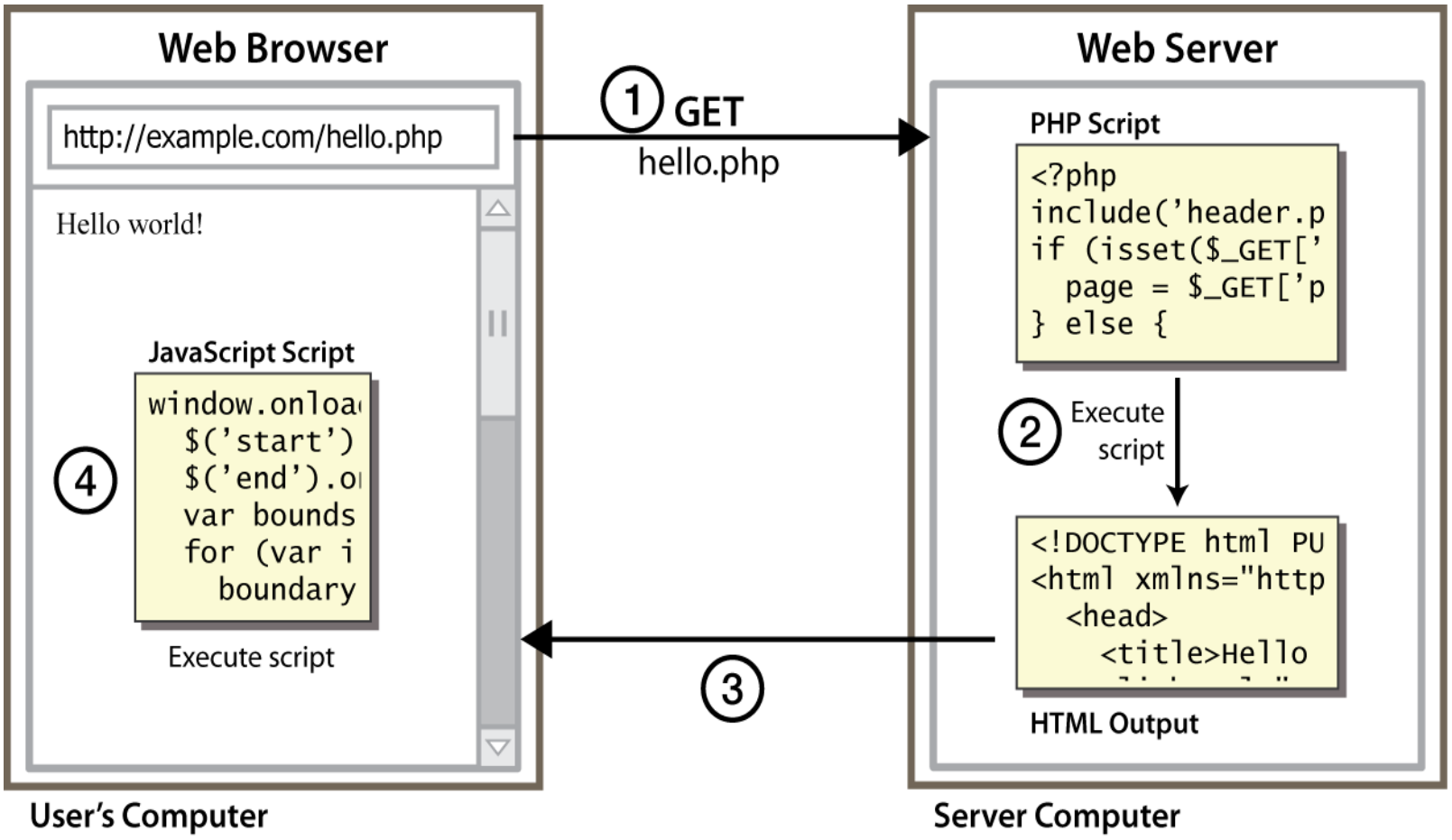


INTRODUCTION TO JAVASCRIPT

CLIENT SIDE SCRIPTING



WHY USE CLIENT-SIDE PROGRAMMING?

Server side scripting (PHP or Python) already allows us to create dynamic web pages. Why also use client-side scripting?

client-side scripting (JavaScript) benefits:

- **usability**: can modify a page without having to post back to the server (faster UI)
- **efficiency**: can make small, quick changes to page without waiting for server
- **event-driven**: can respond to user actions like clicks and key presses

WHY USE CLIENT-SIDE PROGRAMMING?

server-side programming benefits:

- **security**: has access to server's private data; client can't see source code
- **compatibility**: not subject to browser compatibility issues
- **power**: can write files, open connections to servers, connect to databases

WHAT IS JAVASCRIPT?

a lightweight programming language ("scripting language")

- used to make web pages interactive
- insert dynamic text into HTML (ex: user name)
- **react to events** (ex: page load user click)
- get information about a user's computer (ex: browser type)
- perform calculations on user's computer (ex: form validation)

WHAT IS JAVASCRIPT?

a web standard (but not supported identically by all browsers)

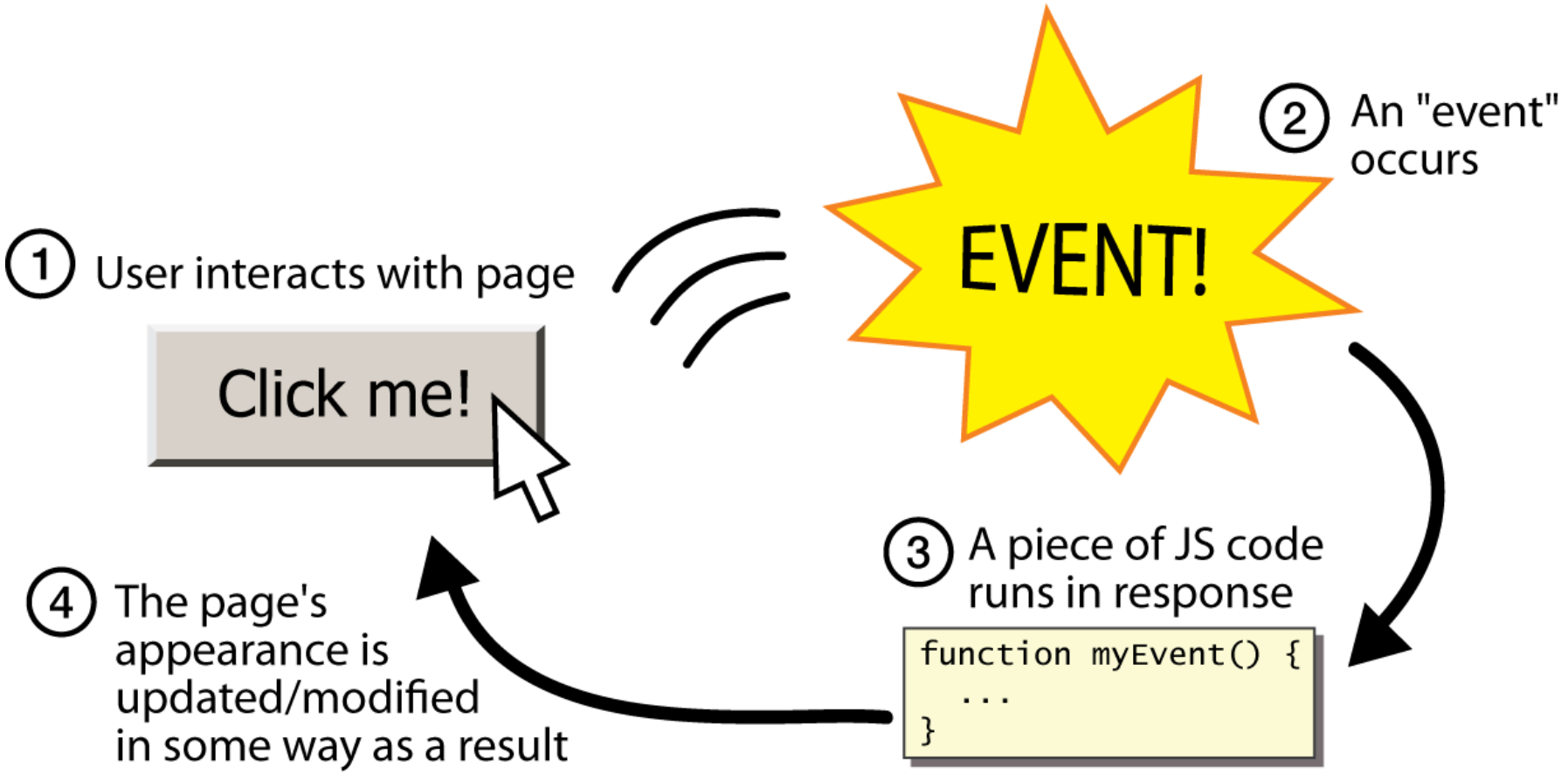
NOT related to Java other than by name and some syntactic similarities

JAVASCRIPT VS. SERVER SIDE SCRIPT

similarities:

- both are interpreted, not compiled
- both are relaxed about syntax, rules, and types
- both are case-sensitive
- both have built-in regular expressions for powerful text processing

EVENT-DRIVEN PROGRAMMING



A JAVASCRIPT STATEMENT: ALERT

```
alert("Hello! I am an alert box!");
```
JS

a JS command that pops up a dialog box with a message

EVENT-DRIVEN PROGRAMMING

- JavaScript programs wait for user actions called *events* and respond to them
- event-driven programming: writing programs driven by user events
- Let's write a page with a clickable button that pops up a "Hello, World" window...

BUTTONS

```
<button>Click me!</button>                    HTML
```

button's text appears inside tag; can also contain images

To make a responsive button or other UI control:

1. choose the control (e.g. button) and event (e.g. mouse 1. click) of interest
2. write a JavaScript function to run when the event occurs
3. attach the function to the event on the control

JAVASCRIPT FUNCTIONS

```
function name() {
statement ;
statement ;
...
statement ;
}
```
JS

```
function myFunction() {
    alert("Hello!");
    alert("How are you?");
}
```
JS

- the above could be the contents of example.js linked to our HTML page
- statements placed into functions can be evaluated in response to user events

EVENT HANDLERS

```
<element attributes onclick="function();">...
```
HTML

```
<button onclick="myFunction();">Click me!</button>
```
HTML

JavaScript functions can be set as event handlers

- when you interact with the element, the function will execute

onclick is just one of many event HTML attributes we'll use

but popping up an alert window is disruptive and annoying

- A better user experience would be to have the message appear on the page...

EVENT HANDLERS

```
<p>Click the button to display an alert box.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  alert("Hello! I am an alert box!");
}
</script>
```

HTML

DOCUMENT OBJECT MODEL (DOM)

most JS code manipulates elements on an HTML page

we can examine elements' state

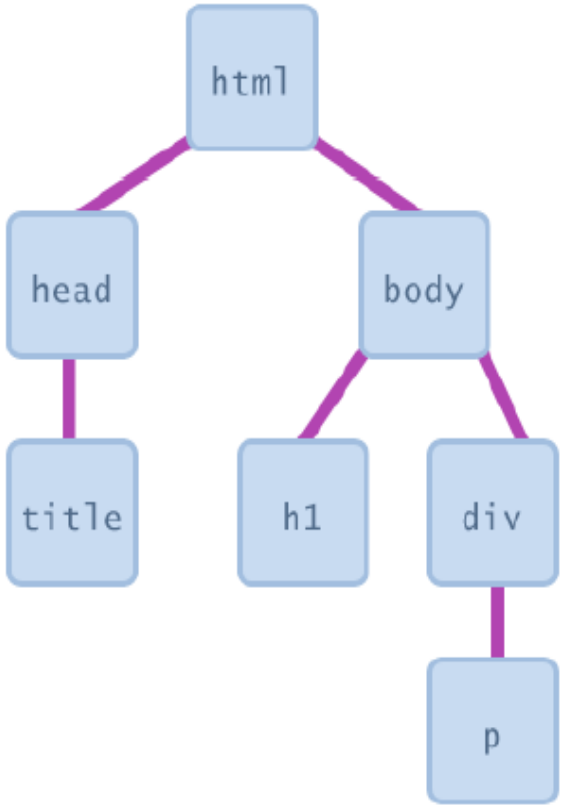
- e.g. see whether a box is checked

we can change state

- e.g. insert some new text into a div

we can change styles

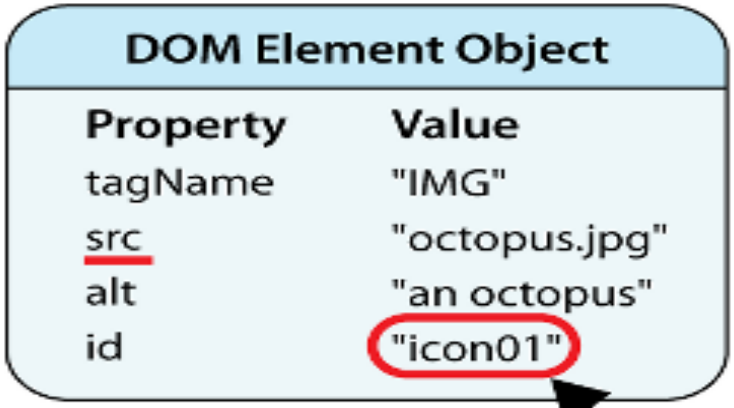
- e.g. make a paragraph red



DOM ELEMENT OBJECTS

HTML

```
<p>
  Look at this octopus:
  <img src="octopus.jpg" alt="an octopus" id="icon01" />
  Cute, huh?
</p>
```



JavaScript

```
var icon = document.getElementById("icon01");
icon.src = "kitty.gif";
```

ACCESSING ELEMENTS: DOCUMENT.GETELEMENTBYID

```
var name = document.getElementById("id");
```
JS

```
<button onclick="changeText();">Click me!</button>
<span id="output">replace me</span>
<input id="textbox" type="text" />
```
HTML

```
function changeText() {
    var span = document.getElementById("output");
    var textBox = document.getElementById("textbox");

     textbox.style.color = "red";

}
```
JS

ACCESSING ELEMENTS: DOCUMENT.GETELEMENTBYID

- document.getElementById returns the DOM object for an element with a given id
- can change the text inside most elements by setting the innerHTML property
- can change the text in form controls by setting the value property

CHANGING ELEMENT STYLE: ELEMENT.STYLE

Attribute	**Property or style object**
color	color
padding	padding
background-color	backgroundColor
border-top-width	borderTopWidth
Font size	fontSize
Font famiy	fontFamily

PREETIFY

```
function changeText() {
    //grab or initialize text here

    // font styles added by JS:
    text.style.fontSize = "13pt";
    text.style.fontFamily = "Comic Sans MS";
    text.style.color = "red";
}
```
JS

PREETIFY

```
<body>

<p id="p1">
This is a text.
This is a text.
This is a text.
</p>

<input type="button" value="Hide text"
onclick="document.getElementById('p1').style.visibility
='hidden'">

<input type="button" value="Show text"
onclick="document.getElementById('p1').style.visibility
='visible'">

</body>
```

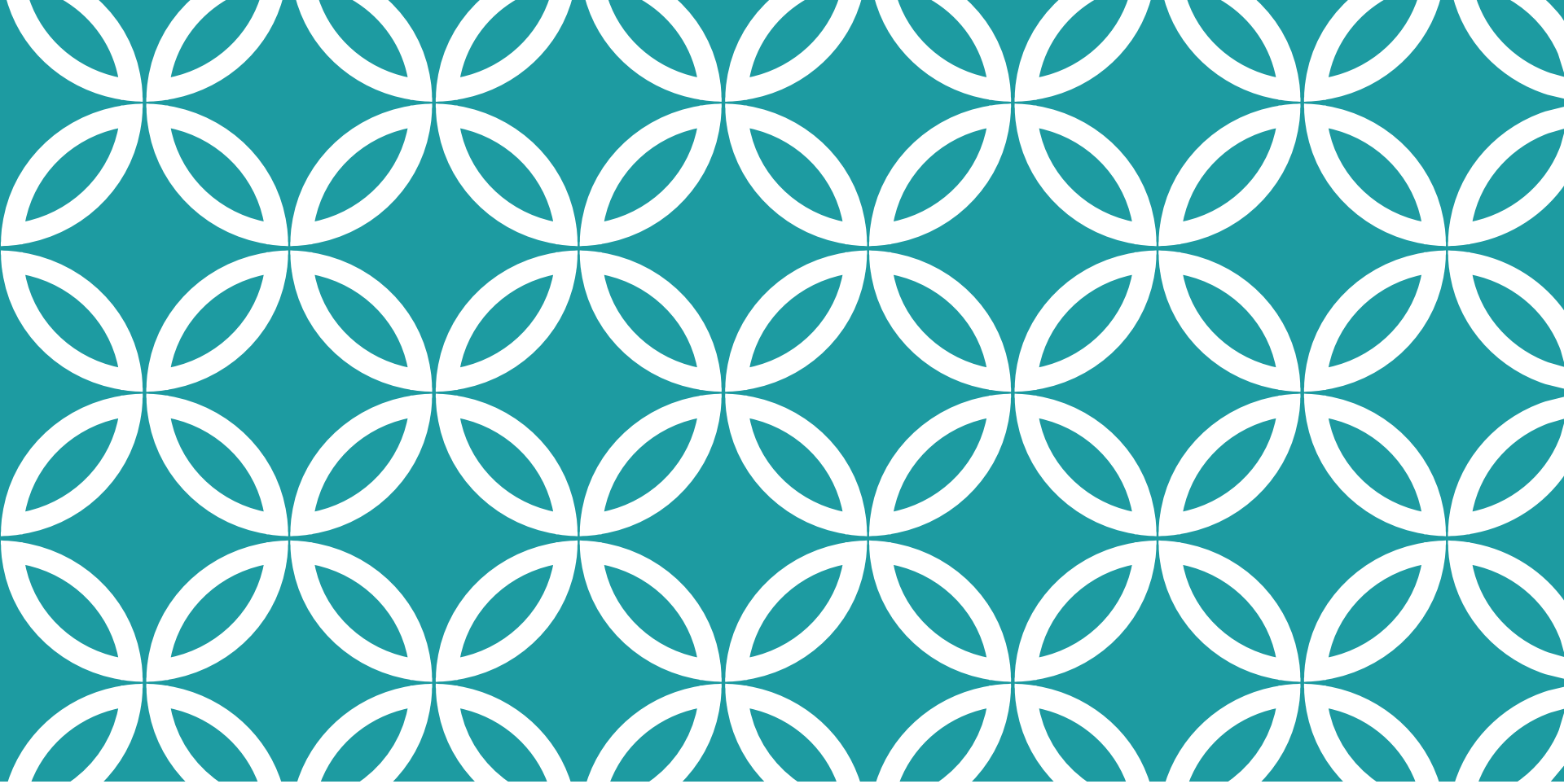
JS

PREETIFY

```
<h2>JavaScript Functions</h2>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML =
"The temperature is " + toCelsius(77) + " Celsius";

function toCelsius(fahrenheit) {
  return (5/9) * (fahrenheit-32);
}
</script>
```

JS

```
<script>
function bgChange(bg) {
  document.body.style.background = bg;}
</script>
</head>
<body>
<h2>Change background color</h2>
<p>Mouse over the squares!</p>
<table style="width:300px;height:100px">
  <tr>
    <td onmouseover="bgChange(this.style.backgroundColor)"
    onmouseout="bgChange('transparent')"
    style="background-color:Khaki">
    </td>
    <td onmouseover="bgChange(this.style.backgroundColor)"
    onmouseout="bgChange('transparent')"
    style="background-color:PaleGreen">
    </td>
    <td onmouseover="bgChange(this.style.backgroundColor)"
    onmouseout="bgChange('transparent')"
    style="background-color:Silver">
    </td>
  </tr>
</table>
```



MORE JAVASCRIPT SYNTAX

VARIABLES

```
var name = expression;                    JS
```

```
var clientName = "Connie Client";
var age = 32;
var weight = 127.4;                       JS
```

variables are declared with the var keyword (case sensitive)

types are not specified, but JS does have types ("loosely typed")

- `Number`, `Boolean`, `String`, `Array`, `Object`, `Function`, `Null`, `Undefined`
- can find out a variable's type by calling `typeof`

NUMBER TYPE

```
var enrollment = 99;
var medianGrade = 2.8;
var credits = 5 + 4 + (2 * 3);
```
JS

integers and real numbers are the same type (no int vs. double)

same operators: + - * / % ++ -- = += -= *= /= %=

similar precedence to Java

many operators auto-convert types: "2" * 3 is 6

COMMENTS (SAME AS JAVA)

```
// single-line comment
/* multi-line comment */
```
JS

identical to Java's comment syntax

recall: 4 comment syntaxes

- HTML: <!-- comment -->
- CSS/JS/PHP: /* comment */
- Java/JS/PHP: // comment
- PHP: # comment

MATH OBJECT

```
var rand1to10 = Math.floor(Math.random() * 10 + 1);
var three = Math.floor(Math.PI);
```
JS

- **methods:** abs, ceil, cos, floor, log, max, min, pow, random, round, sin, sqrt, tan
- **properties:** E, PI

SPECIAL VALUES: NULL AND UNDEFINED

```
var ned = null;
var benson = 9;
// at this point in the code,
// ned is null
// benson's 9
// caroline is undefined
```
JS

- `undefined` : has not been declared, does not exist
- `null` : exists, but was specifically assigned an empty or null value
- Why does JavaScript have both of these?

LOGICAL OPERATORS

- > < >= <= && || ! == != === !==
- most logical operators automatically convert types:
  - 5 < "7" is true
  - 42 == 42.0 is true
  - "5.0" == 5 is true
- === and !== are strict equality tests; checks both type and value
  - "5.0" === 5 is false

OUTPUT IN JAVASCRIPT

```
<body>
<script>
document.write(5 + 6);
</script>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>

<script>
window.alert(5 + 6);
</script>

<script>
console.log(5 + 6);
</script>
</body>
```

JS

EXAMPLE

```
<body>

<form>
1st Number : <input type="text" id="firstNumber" /><br>
2nd Number: <input type="text" id="secondNumber" /><br>
<input type="button" onClick="multiplyBy()"
Value="Multiply" />
<input type="button" onClick="divideBy()"
Value="Divide" />
</form>

<p>The Result is : <br>
<span id = "result"></span>
</p>
</body>
```

JS

OUTPUT IN JAVASCRIPT

```
<script>
function multiplyBy()
{
num1 = document.getElementById("firstNumber").value;
num2 = document.getElementById("secondNumber").value;
document.getElementById("result").innerHTML = num1 * num2;
}

function divideBy()
{
num1 = document.getElementById("firstNumber").value;
num2 = document.getElementById("secondNumber").value;
document.getElementById("result").innerHTML = (num1 /
num2).toPrecision(3);
}
</script>                                                    JS
```

IF/ELSE STATEMENT (SAME AS JAVA)

```
if (condition) {
    statements;
} else if (condition) {
    statements;
} else {
    statements;
}
```

JS

- identical structure to Java's if/else statement
- JavaScript allows almost anything as a condition

BOOLEAN TYPE

```
var iLike190M = true;
var ieIsGood = "IE6" > 0; // false
if ("web devevelopment is great") { /* true */ }
if (0) { /* false */ }
```
JS

- any value can be used as a Boolean
  - "falsey" values: 0, 0.0, NaN, "", null, and undefined
  - "truthy" values: anything else
- converting a value into a Boolean explicitly:
  - `var boolValue = Boolean(otherValue);`
  - `var boolValue = !!(otherValue);`

FOR LOOP (SAME AS JAVA)

```
var sum = 0;
for (var i = 0; i < 100; i++) {
    sum = sum + i;
}
```
JS

```
var s1 = "hello";
var s2 = "";
for (var i = 0; i < s1.length; i++) {
    s2 += s1.charAt(i) + s1.charAt(i);
}
// s2 stores "hheellllloo"
```
JS

WHILE LOOPS (SAME AS JAVA)

```
while (condition) {
    statements;
}
```
JS

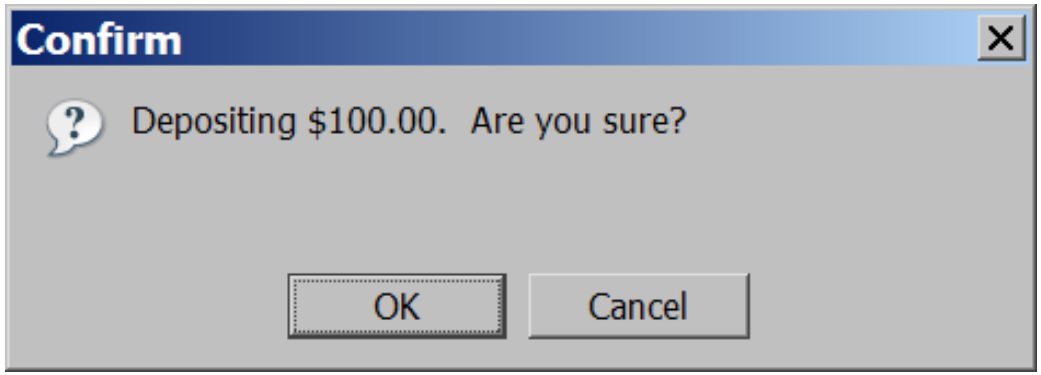
```
do {
    statements;
} while (condition);
```
JS

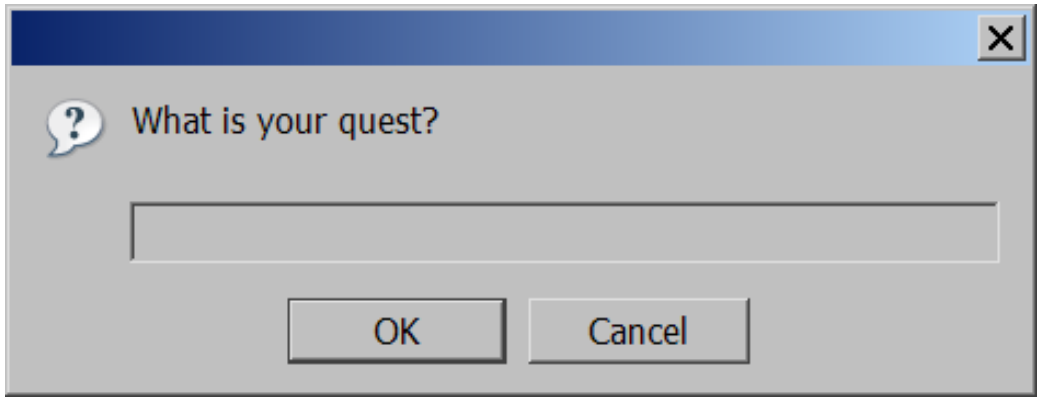
- break and continue keywords also behave as in Java

POPUP BOXES

```
alert("message"); // message
confirm("message"); // returns true or false
prompt("message"); // returns user input string
```

JS





ARRAYS

```
var name = []; // empty array
var name = [value, value, ..., value]; // pre-filled
name[index] = value; // store element
```
JS

```
var ducks = ["Huey", "Dewey", "Louie"];
var stooges = []; // stooges.length is 0
stooges[0] = "Larry"; // stooges.length is 1
stooges[1] = "Moe"; // stooges.length is 2
stooges[2] = "Curly"; // stooges.length is 3
stooges[3] = "Shemp"; // stooges.length is 4
```
JS

```
<h2>JavaScript For Loop</h2>

<p id="demo"></p>

<script>
var cars = ["BMW", "Volvo", "Saab", "Ford", "Fiat",
"Audi"];
var text = "";

for (var i = 0; i < cars.length; i++) {
  text += cars[i] + "<br>";
}
document.getElementById("demo").innerHTML = text;

var txt = 'JavaScript';
var x;

for (x of txt) {
  document.write(x + "<br >");
}

</script>
```

JS

```
 <title>Multiplication Table</title>
<body>
  <script type="text/javascript">
    var rows = prompt("How many rows for your multiplication table?");
    var cols = prompt("How many columns for your multiplication table?");
    if(rows == "" || rows == null)
         rows = 10;
    if(cols== "" || cols== null)
         cols = 10;
    createTable(rows, cols);
    function createTable(rows, cols)
    {
      var j=1;
      var output = "<table border='1' width='500'
cellspacing='0'cellpadding='5'>";
      for(i=1;i<=rows;i++)
      {
    output = output + "<tr>";
        while(j<=cols)
        {
           output = output + "<td>" + i*j + "</td>";
           j = j+1;
         }
          output = output + "</tr>";
          j = 1;
    }
    output = output + "</table>";
    document.write(output);
    }
  </script>
</body>
```

ARRAY METHODS

```
var a = ["Stef", "Jason"]; // Stef, Jason
a.push("Brian"); // Stef, Jason, Brian
a.unshift("Kelly"); // Kelly, Stef, Jason, Brian
a.pop(); // Kelly, Stef, Jason
a.shift(); // Stef, Jason
a.sort(); // Jason, Stef
```

JS

- array serves as many data structures: list, queue, stack, ...
- methods: `concat, join, pop, push, reverse, shift, slice, sort, splice, toString, unshift`
  - push and pop add / remove from back
  - unshift and shift add / remove from front
  - shift and pop return the element that is removed

```
<h2>JavaScript Objects</h2>

<p id="demo"></p>

<script>
var person = {
  firstName : "John",
  lastName  : "Doe",
  age       : 50,
  eyeColor  : "blue"
};

document.getElementById("demo").innerHTML =
person.firstName + " is " + person.age + " years old.";
</script>
```

JS

STRING TYPE

```
var s = "Connie Client";
var fName = s.substring(0, s.indexOf(" ")); // "Connie"
var len = s.length; // 13
var s2 = 'Melvin Merchant';
```
JS

methods: `charAt, charCodeAt, fromCharCode, indexOf, lastIndexOf, replace, split, substring, toLowerCase, toUpperCase`

- charAt returns a one-letter String (there is no char type)

length property (not a method as in Java)

Strings can be specified with "" or ''

concatenation with + :

- 1 + 1 is 2, but "1" + 1 is "11"

MORE ABOUT STRING

- escape sequences behave as in Java: \' \" \& \n \t \\
- converting between numbers and Strings:

```
var count = 10;
var s1 = "" + count; // "10"
var s2 = count + " , ah ah ah!"; // "10 bananas, ah ah ah!"
var n1 = parseInt("42 is the answer"); // 42
var n2 = parseFloat("booyah"); // NaN
```
JS

accessing the letters of a String:

```
var firstLetter = s[0]; // fails in IE
var firstLetter = s.charAt(0); // does work in IE
var lastLetter = s.charAt(s.length - 1);
```
JS

SPLITTING STRINGS: SPLIT AND JOIN

```
var s = "the quick brown fox";
var a = s.split(" "); // ["the", "quick", "brown", "fox"]
a.reverse(); // ["fox", "brown", "quick", "the"]
s = a.join("!"); // "fox!brown!quick!the"
```
JS

- split breaks apart a string into an array using a delimiter
  - can also be used with regular expressions (seen later)
- join merges an array into a single string, placing a delimiter between them

STRINGS EXAMPLES

```
<body>
    <script>
    var str = "Hello World!";
    document.write(str[0] + "<br>"); // Prints: H
    document.write(str[6] + "<br>"); // Prints: W
    document.write(str[str.length - 1] + "<br>"); // Prints: !
    document.write(str[30]); // Prints: undefined
    </script>
</body>
```

JS

STRINGS EXAMPLES

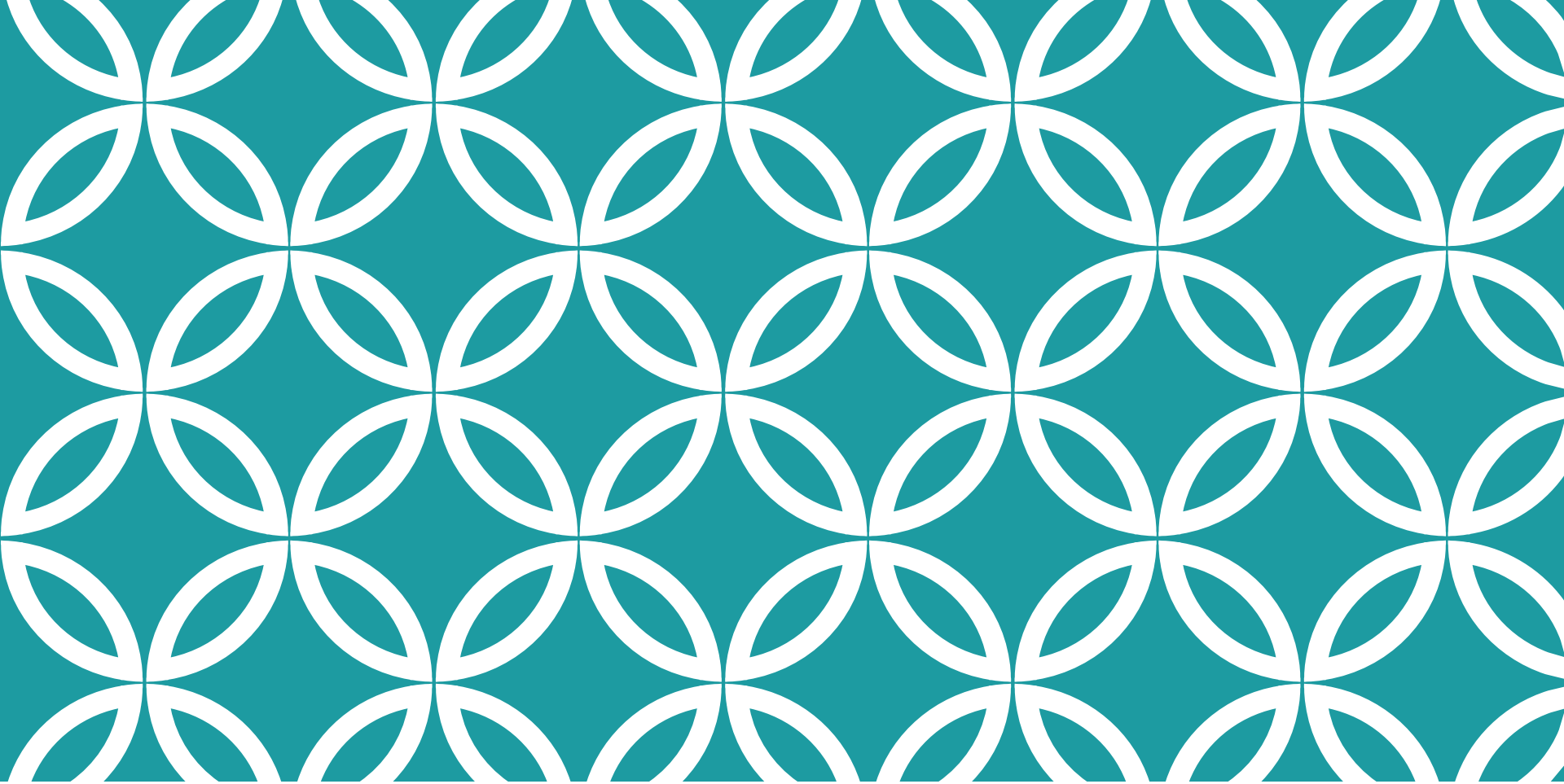
```
<body>
    <script>
    var str = "INTERSTELLAR";
    var strArr = str.split("");
    document.write(strArr[0] + "<br>"); // Prints: I
    document.write(strArr[1] + "<br>"); // Prints: N
    document.write(strArr[strArr.length - 1]); // Prints: R
    document.write("<hr>"); // Prints: N

    // Loop through all the elements of the characters array and
print them
    for(var i in strArr) {
        document.write("<br>" + strArr[i]);
    }
    </script>
</body>
```

JS

STRINGS EXAMPLES

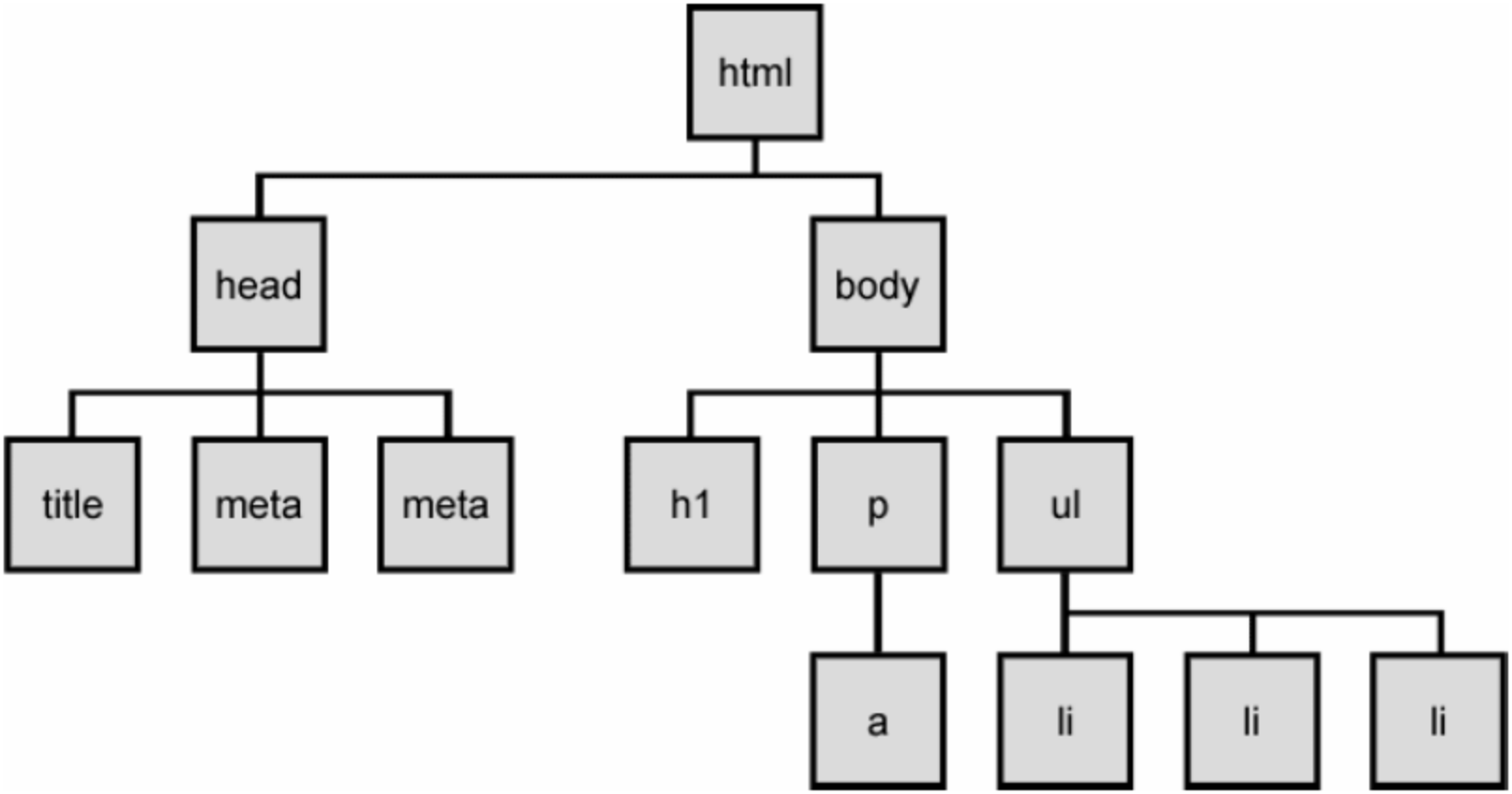
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>JavaScript Split a String into an Array</title>
</head>
<body>
    <script>
    var fruitsStr = "Apple, Banana, Mango, Orange, Papaya";
    var fruitsArr = fruitsStr.split(", ");
    document.write(fruitsArr[0] + "<br>"); // Prints: Apple
    document.write(fruitsArr[2] + "<br>"); // Prints: Mango
    document.write(fruitsArr[fruitsArr.length - 1]); // Prints:
Papaya
    document.write("<hr>");

    // Loop through all the elements of the fruits array
    for(var i in fruitsArr) {
        document.write("<p>" + fruitsArr[i] + "</p>");
    }
    </script>
</body>
</html>
```

JS



THE DOM TREE

THE DOM TREE



TYPES OF DOM NODES

```
<p>
This is a paragraph of text with a
<a href="/path/page.html">link in it</a>.
</p>
```
HTML

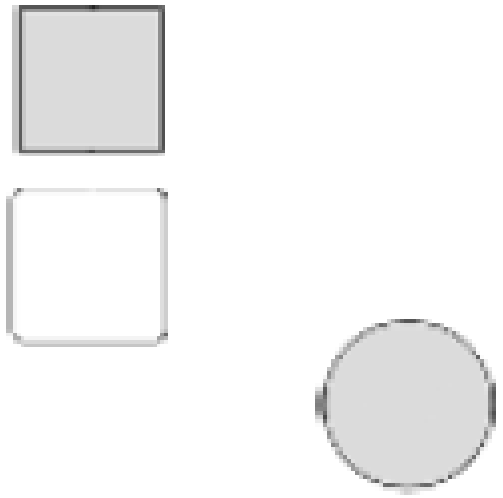
element nodes (HTML tag)

- can have children and/or attributes

text nodes (text in a block element)

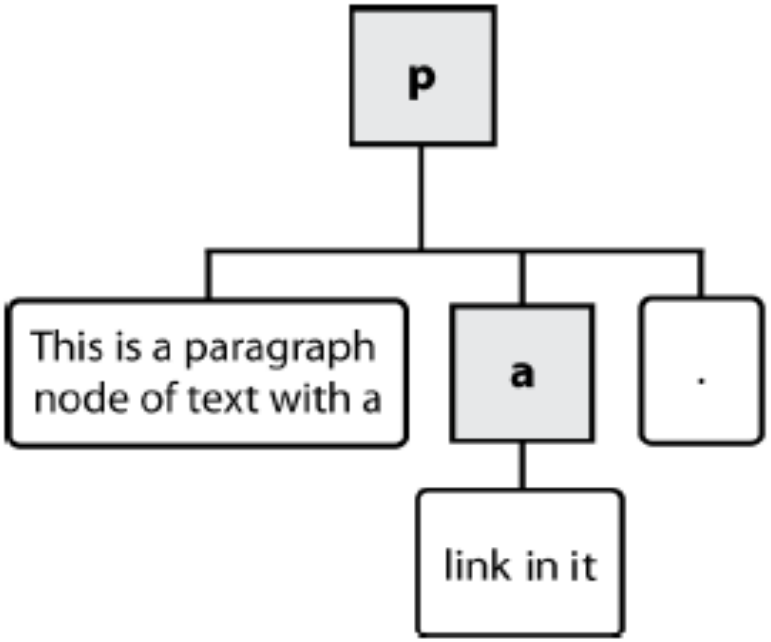
attribute nodes (attribute/value pair)

- text/attributes are children in an element node
- cannot have children or attributes
- not usually shown when drawing the DOM tree



TYPES OF DOM NODES

```
<p>
This is a paragraph of text with a
<a href="/path/page.html">link in it</a>.
</p>
```
HTML



TRAVERSING THE DOM TREE

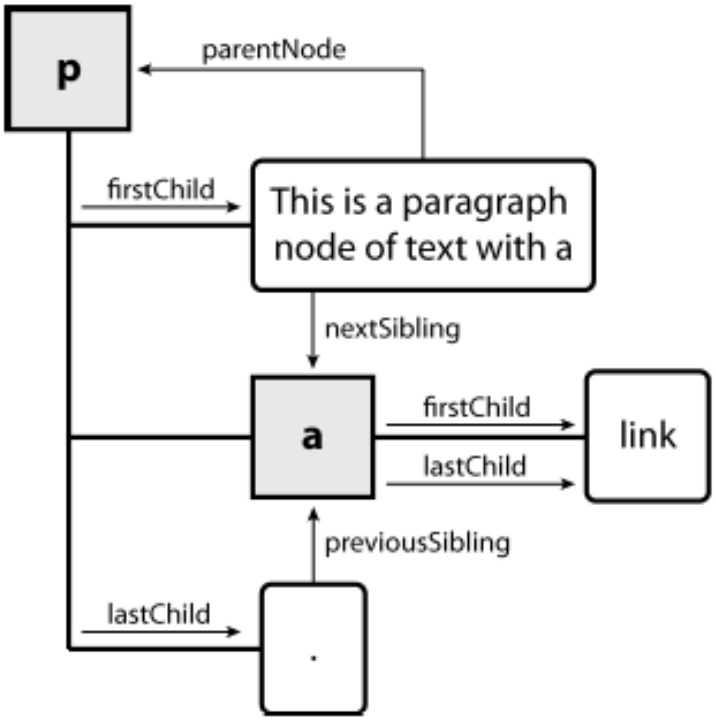
name(s)	description
firstChild, lastChild	start/end of this node's list of children
childNodes	array of all this node's children
nextSibling, previousSibling	neighboring nodes with the same parent
parentNode	the element that contains this node

- complete list of DOM node properties
- browser incompatiblity information

DOM TREE TRAVERSAL EXAMPLE

```
<p id="foo">This is a paragraph of text with a
<a href="/path/to/another/page.html">link</a>.</p>
```

HTML



ELEMENTS VS TEXT NODES

```
<div>
    <p>
        This is a paragraph of text with a
        <a href="page.html">link</a>.
    </p>
</div>
```
HTML

Q: How many children does the div above have?

A: 3

- an element node representing the <p>
- two text nodes representing "\n\t" (before/after the paragraph)

Q: How many children does the paragraph have? The a tag?

PROTOTYPE'S DOM ELEMENT METHODS

absolutize	addClassName	classNames	cleanWhitespace	clonePosition
cumulativeOffset	cumulativeScrollOffset	empty	extend	firstDescendant
getDimensions	getHeight	getOffsetParent	getStyle	getWidth
hasClassName	hide	identify	insert	inspect
makeClipping	makePositioned	match	positionedOffset	readAttribute
recursivelyCollect	relativize	remove	removeClassName	replace
scrollTo	select	setOpacity	setStyle	show
toggle	toggleClassName	undoClipping	undoPositioned	update
viewportOffset	visible	wrap	writeAttribute	

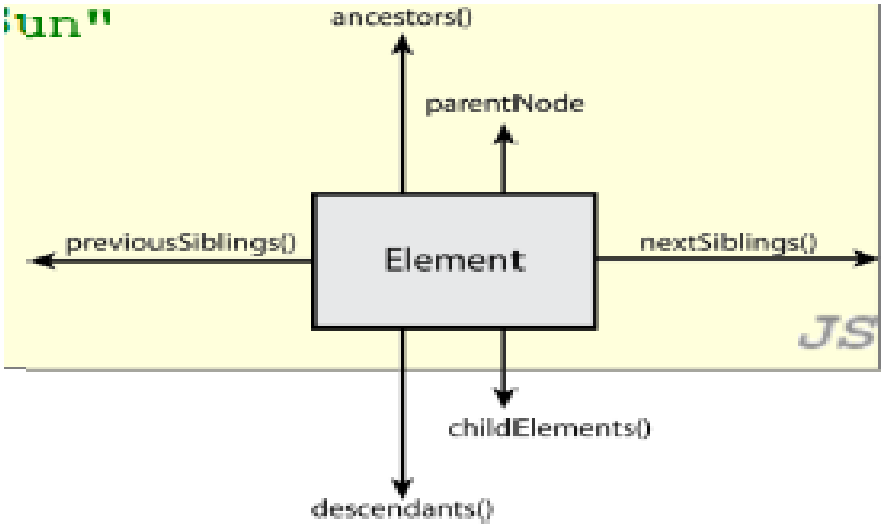
PROTOTYPE'S DOM TREE TRAVERSAL METHODS

method(s)	description
ancestors, up	elements above this one
childElements, descendants, down	elements below this one (not text nodes)
siblings, next, nextSiblings, previous, previousSiblings, adjacent	elements with same parent as this one (not text nodes)

PROTOTYPE'S DOM TREE TRAVERSAL METHODS

```
// alter siblings of "main" that do not contain "Sun"
var sibs = $("main").siblings();
for (var i = 0; i < sibs.length; i++) {
    if (sibs[i].innerHTML.indexOf("Sun") < 0) {
        sibs[i].innerHTML += " Sunshine";
    }
}
```

JS



SELECTING GROUPS OF DOM OBJECTS

methods in document and other DOM objects for accessing descendants:

name	description
getElementsByTagName	returns array of descendents with the given tag, such as "div"
getElementsByName	returns array of descendants with the given name attribute (mostly useful for accessing form controls)

GETTING ALL ELEMENTS OF A CERTAIN TYPE

```
var allParas = document.getElementsByTagName("p");
for (var i = 0; i < allParas.length; i++) {
    allParas[i].style.backgroundColor = "yellow";
}
```

JS

```
<body>
    <p>This is the first paragraph</p>
    <p>This is the second paragraph</p>
    <p>You get the idea...</p>
</body>
```

HTML

COMBINING WITH GETELEMENTBYID

```
var addrParas = $("address").getElementsByTagName("p");
for (var i = 0; i < addrParas.length; i++) {
    addrParas[i].style.backgroundColor = "yellow";
}
```

JS

```
<p>This won't be returned!</p>
<div id="address">
    <p>1234 Street</p>
    <p>Atlanta, GA</p>
</div>
```

HTML

PROTOTYPE'S METHODS FOR SELECTING ELEMENTS

```
var gameButtons = $("game").select("button.control");
for (var i = 0; i < gameButtons.length; i++) {
    gameButtons[i].style.color = "yellow";
}
```
JS

Prototype adds methods to the document object
(and all DOM element objects) for selecting groups of elements:

getElementsByClassName	array of elements that use given class attribute
select	array of descendants that match given CSS selector, such as "div#sidebar ul.news > li"

PROTOTYPE'S METHODS FOR SELECTING ELEMENTS

```
<style>
#container {
  width: 400px;
  height: 400px;
  position: relative;
  background: yellow;
}
#animate {
  width: 50px;
  height: 50px;
  position: absolute;
  background-color: red;
}
</style>
```

```
<body>
<p><button onclick="myMove()">Click Me</button></p>
<div id ="container">
  <div id ="animate"></div>
</div>
<script>
function myMove() {
  var elem = document.getElementById("animate");
  var pos = 0;
  var id = setInterval(frame, 5);
  function frame() {
    if (pos == 350) {
      clearInterval(id);
    } else {
      pos++;
      elem.style.top = pos + "px";
      elem.style.left = pos + "px";
    }
  }
}
</script>
</body>
```

THE $$ FUNCTION

```
var arrayName = $$("CSS selector");
```
JS

```
// hide all "announcement" paragraphs in the "news"
//section
var paragraphs = $$("div#news p.announcement");
for (var i = 0; i < paragraphs.length; i++) {
    paragraphs[i].hide();
}
```
JS

$$ returns an array of DOM elements that match the given CSS selector

- like $ but returns an array instead of a single DOM object
- a shorthand for document.select

useful for applying an operation each one of a set of elements

COMMON ISSUES WITH $$

```
// get all buttons with a class of "control"
var gameButtons = $$("control");
var gameButtons = $$(".control");
```
JS

```
// set all buttons with a class of "control" to have red
text
$$(".control").style.color = "red";
var gameButtons = $$(".control");
for (var I = 0; i < gameButtons.length; i++) {
    gameButtons[i].style.color = "red";
}
```
JS

Q: Can I still select a group of elements using $$ even if my CSS file doesn't have any style rule for that same group? (A: Yes!)

CREATING NEW NODES

name	description
document.createElement("tag")	creates and returns a new empty DOM node representing an element of that type
document.createTextNode("text")	creates and returns a text node containing given text

```
// create a new <h2> node
var newHeading = document.createElement("h2");
newHeading.innerHTML = "This is a heading";
newHeading.style.color = "green";
```
JS

merely creating a node does not add it to the page

you must add the new node as a child of an existing element on the page...

MODIFYING THE DOM TREE

name	description
appendChild(node)	places given node at end of this node's child list
insertBefore(new, old)	places the given new node in this node's child list just before old child
removeChild(node)	removes given node from this node's child list
replaceChild(new, old)	replaces given child with new node

```
var p = document.createElement("p");
p.innerHTML = "A paragraph!";
$("main").appendChild(p);
```
JS

REMOVING A NODE FROM THE PAGE

```
function slideClick() {
    var bullets = document.getElementsByTagName("li");
    for (var i = 0; i < bullets.length; i++) {
        if (bullets[i].innerHTML.indexOf("children") >= 0)
{
            bullets[i].remove();
        }
    }
}
```

JS

each DOM object has a `removeChild` method to remove its children from the page

Prototype adds a `remove` method for a node to remove itself

DOM VERSUS INNERHTML HACKING

Why not just code the previous example this way?

```
function slideClick() {
    $("thisslide").innerHTML += "<p>A paragraph!</p>";
}
```
JS

Imagine that the new node is more complex:

- ugly: bad style on many levels (e.g. JS code embedded within HTML)
- error-prone: must carefully distinguish " and '
- can only add at beginning or end, not in middle of child list

```
function slideClick() {
    this.innerHTML += "<p style='color: red; " +
    "margin-left: 50px;' " +
    "onclick='myOnClick();'>" +
    "A paragraph!</p>";
}
```
JS

PROBLEMS WITH READING/CHANGING STYLES

```
window.onload = function() {
    $("clickme").onclick = biggerFont;
};
function biggerFont() {
    var size = parseInt($("clickme").style.fontSize);
    size += 4;
    $("clickMe").style.fontSize = size + "pt";
}
```
JS

style property lets you set any CSS style for an element

problem: you cannot (usually) read existing styles with it

ACCESSING STYLES IN PROTOTYPE

```
function biggerFont() {
    // turn text yellow and make it bigger
    var size = parseInt($("clickme").getStyle("font-
size"));
    $("clickme").style.fontSize = (size + 4) + "pt";
}
```
JS

getStyle function added to DOM object allows accessing existing styles

addClassName, removeClassName, hasClassName manipulate CSS classes

COMMON BUG: INCORRECT USAGE OF EXISTING STYLES

```
this.style.top = this.getStyle("top") + 100 + "px";
// bad!
```
JS

the above example computes e.g. "200px" + 100 + "px" , which would evaluate to "200px100px"

a corrected version:

```
this.style.top = parseInt(this.getStyle("top")) + 100 +
"px"; // correct
```
JS

SETTING CSS CLASSES IN PROTOTYPE

```
function highlightField() {
    // turn text yellow and make it bigger
    if (!$("text").hasClassName("invalid")) {
        $("text").addClassName("highlight");
    }
}
```

JS

addClassName, removeClassName, hasClassName manipulate CSS classes

similar to existing className DOM property, but don't have to manually split by spaces

EXAMPLE: CREATEELEMENTS

```
<html>
<head>
<script src="
https://ajax.googleapis.com/ajax/libs/prototype/1.7.0.0/pr
ototype.js " type="text/javascript"></script>
<script src="paragraph.js "
type="text/javascript"></script>
</head>

<body>
<div id="paragrapharea">
    <button id="add">Add a paragraph</button>
</div>
</body>
</html>
```

EXAMPLE: CREATEELEMENTS

```
window.onload = function(){
    var button = $("add");
    button.onclick = addParagraphClick;
}

function addParagraphClick(){
    var paragraph = document.createElement("p");
    paragraph.innerHTML = "All work and no play makes Jack
a dull boy";
    var area = $("paragrapharea");
    area.appendChild(paragraph);
}

function addListClick(){
    }
```

JS

JAVASCRIPT EXERCISES

Create a webpage with an of Homer Simpson image at the center of the page. Develop a script that prints an alert: “Duh, you are hovering!!” every time the mouse crosses over the image.

Add 5 buttons to your webpage: red, yellow, green, black, and silver. Every time you click on one of these buttons the background should take the corresponding color.