Scripting & Computer Environments

Web Programming: `JavaScript`

IIIT-H

{Sep 28, Oct 1}, 2013

...Previously & Today...

Thus Far:

- Web Content/Structure → `HTML, XHTML`
- Style/presentation of the Content → `CSS`

Today:

- Core `JavaScript`

Brainstorm

- The whats and whys of client-side + server-side scripting?
- The whats and hows of dynamic web pages?

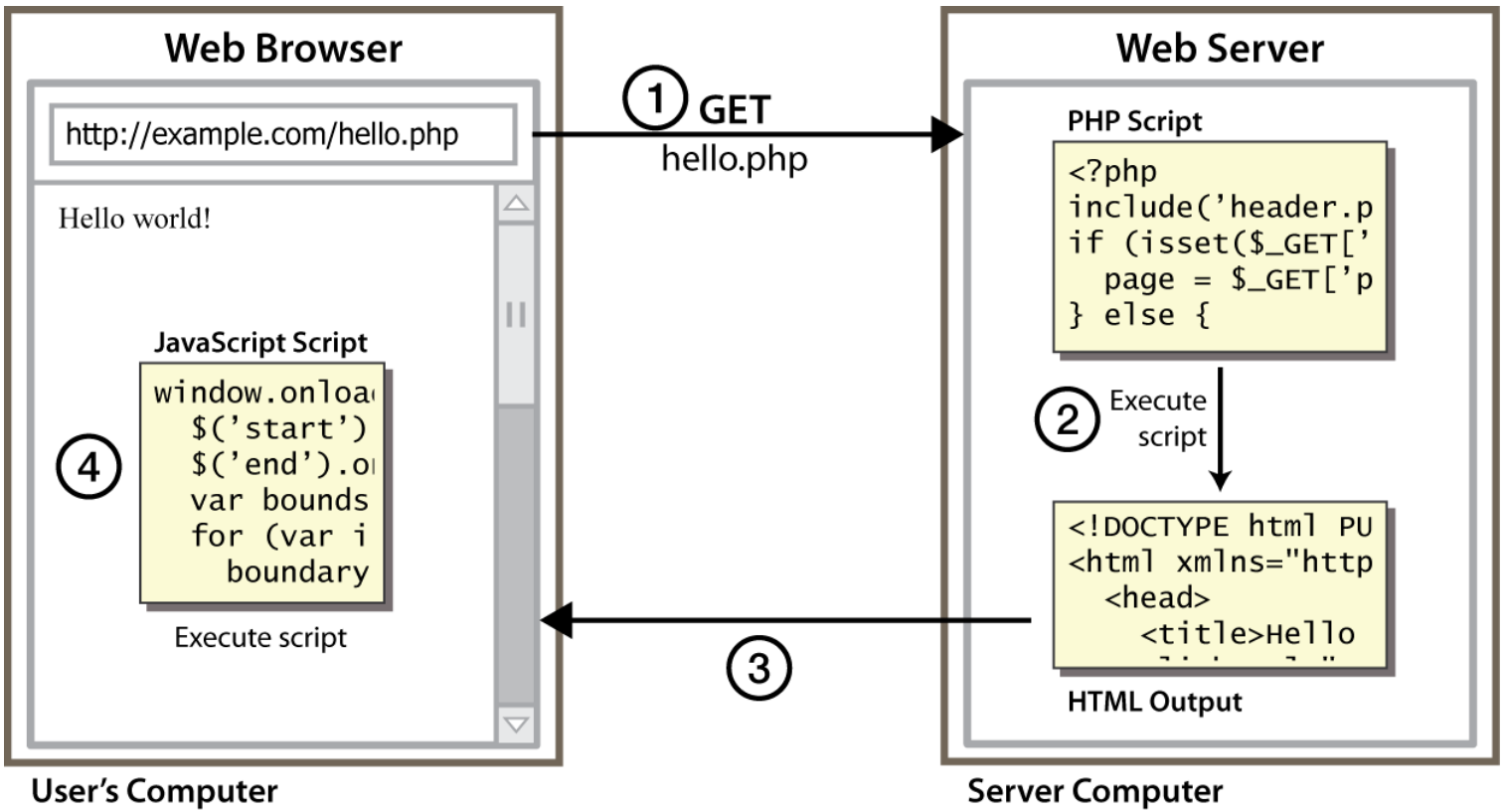
Dynamic HTML (DHTML)

- Not a language or web standard.
- Set of technologies to create dynamic and interactive websites.

  e.g. animated content, client-side form validation, mouse events ...

- `DHTML = HTML + CSS + client-side scripting (JS) + DOM`
- The DOM is:
  - A standard for how to access/change/add/delete HTML elements.
  - The API for web documents.

Client-Side vs Server-Side Scripting



JavaScript (initially LiveScript)

- Brainchild of Brendan Eich of Netscape corp.
- Design goal: to add *interactivity* to pages.
- A dynamic, weakly-typed, interpreted scripting language
- Typical execution environment → web browser
- Some JavaScript (JS) interpretters/engines:

  `SpiderMonkey, TraceMonkey (Firefox), V8 (Chrome), Rhino...`

- Not Java of Sun Microsystems (now acquired by Oracle).

- Animation of webpage elements
- Interactive content (audio, video, games ...)
- For visual feedback to user actions (warnings, confirmations ...)
- Form data validation
- Computation within HTML
- Client state tracking with cookies

Applications: Google Maps, Google Suggest, Bookmarklets, etc

1. Using the `<script>` tag inside the `<head>` and/or `<body>` tag.

```
<body> or <head>

    <script type="text/javascript">
      Some JS code goes here
    </script>

</body> or </head>
```

- The script inside `<body>` is executed on page loads.
- The script inside `<head>` is executed when an event triggers it (e.g. mouse click, key press).

2. By embedding an external JavaScript file (`.js` extension) into your `(X)HTML`.

```
<body> or <head>

  <script type="text/javascript" src="YourScript.js">
  </script>

</body> or </head>
```

- The link can be placed in the `<head>` or `<body>` tag.
- Its behavior depends on where it is placed.
- The `<noscript>` element

- `JavaScript` is a sequence of statements.
- Are commands to the browser.
- Each semicolon-delimited (optional).
- Extra whitespace ignored.
- Grouping of statements within {} possible (i.e. blocks)

  e.g. functions
- Comments:
  1. Single-line `// your comment`
  2. Multi-line `/* your comment */`

- Hold a value or expression.
- Declaration:
  - Explicit: with the `var` keyword.
  - Implicit

e.g.
```
var course = "SACE";
year = 2013;
```

- Are case-sensitive.
- JavaScript is *loosely-typed*. But does NOT mean no data types.

- Primitives:
  - `Boolean`
  - `Numbers`
  - `Strings`
- Composits:
  - `Arrays`
  - `Objects`
- Specials:
  - `null`
  - `Undefined`
- Other built-in objects:

  `Functions` `Date` `Math` `RegExp` ...

a. Boolean → {true, false}

```
var status = true;
var IsPrime = false;
```

b. Numbers → In JavaScript, one number type only.

```
var n = 100;
var hex = 0xCAFE;
var pi = 3.14;
var avocadro = 6.02e23;
```

c. Strings → data type for representing text

```
var myString = "Cheer up! Everything is gonna be alright" ;
var msg = 'Rejoice always. I will say it again: Rejoice!' ;
```

a. **Boolean** → {true, false}

```
var status = true;
var IsPrime = false;
```

b. **Numbers** → In JavaScript, one number type only.

```
var n = 100;
var hex = 0xCAFE;
var pi = 3.14;
var avocadro = 6.02e23;
```

c. **Strings** → data type for representing text

```
var myString = "Cheer up! Everything is gonna be alright" ;
var msg = 'Rejoice always. I will say it again: Rejoice!' ;
```

a. **Boolean** → {true, false}

```
var status = true;
var IsPrime = false;
```

b. **Numbers** → In JavaScript, one number type only.

```
var n = 100;
var hex = 0xCAFE;
var pi = 3.14;
var avocadro = 6.02e23;
```

c. **Strings** → data type for representing text

```
var myString = "Cheer up! Everything is gonna be alright" ;
var msg = ‘Rejoice always. I will say it again: Rejoice!’ ;
```

d. Arrays

- A collection of data values stored in a variable.
- Index-based access of the elements.
- Created using the `new` keyword.

```
var Arr1=[];                                    //empty array

var Arr2 = new Array(5);                        //array size = 5

var x = new Array(1, "two", 3.0, false);         // + initialization


x[0]                                            //element access

Arr1[0] = x[2];

Arr2[4] = "Hello";
```

e. Objects

- JavaScript is object-based; almost everything is an object.
- Built-in objects: `numbers, strings, array, math, date ...`
- Each object has properties and methods.
- Properties describe the object, methods perform actions.
- The dot operator (.) to access an object's properties/methods.

  e.g. The `document` object, the `Date` object

```
document.title = "This is a title";          // property
document.write("Hello");                     // write() method
document.writeln("Hello");                   // writeln() method
```

The Date Object

- Created with the Date constructor.
- Can be set to different values.
- Some methods of the Date object:

```
getDate(), getDay(), getMonth(), getFullYear(),

getHours(), getMinutes(), getTime(), setDate() ...
```

e.g.

```
var now = new Date();

var bdate = new Date("September 28, 2013");

now.getMonth()
now.getSeconds()
bdate.getTime()                    // No of ms since 1/1/1970
```

Arithmetic Operators

`+` `-` `*` `/` `%` `++` `--`

Assignment Operators

`=` `+=` `-=` `*=` `/=` `%=`

Relational Operators

`==` `!=` `===` `!==` `<` `>` `<=` `>=`

(=== is for strict equality)

Arithmetic Operators

`+` `-` `*` `/` `%` `++` `--`

Assignment Operators

`=` `+=` `-=` `*=` `/=` `%=`

Relational Operators

`==` `!=` `===` `!==` `<` `>` `<=` `>=`

(=== is for strict equality)

Arithmetic Operators

`+` `-` `*` `/` `%` `++` `--`

Assignment Operators

`=` `+=` `-=` `*=` `/=` `%=`

Relational Operators

`==` `!=` `===` `!==` `<` `>` `<=` `>=`

(=== is for strict equality)

Logical Operators

`&&` (and) `||` (or) `!` (not)

String Operator

`str1 + str 2` (concatenation)

In addition, defined are various string methods and properties:

`toLowerCase()`, `toUpperCase()`, `substring()`, `indexOf()`, `match()`, `replace()`, `split()`, `length` ...

Logical Operators

`&& (and)` `|| (or)` `! (not)`

String Operator

`str1 + str 2` (concatenation)

In addition, defined are various string methods and properties:

`toLowerCase()`, `toUpperCase()`, `substring()`, `indexOf()`, `match()`, `replace()`, `split()`, `length` ...

- Via the `Math` object.
- Properties for math constants *&* methods for math functions.
- *Properties*:

  `E, PI, SQRT2, LN2, LN10` ...
- *Methods*:

  `sqrt(x), sin(x), pow(x,y), exp(x), floor(x), ceil(x), log(n), cos(x), random(), log(x)` ...

```
var base = Math.E;

var r = Math.floor(Math.PI);

var area = Math.PI * Math.pow(r,2);
```

- Block of code that performs an action.
- Function vs Method
- *May* return a value.
- *Often* writen inside the `<head>` element.
- Some built-ins:

`alert()` `prompt()` `confirm()`

`eval()` `parseInt()` `parseFloat()` `etc`

- User-defined functions

Function

```
function  FunctionName ([arguments])
{
statements;
[return value]
}
```

```
function area (r)
{
var r = prompt ("Please enter the radius:");
return (Math.PI * r * r );
}
```

The if/else Construct

```
if (condition)
{
 statements I;
}
else                    //optional
{
statements II;
}
```

The Ternary Operator (?:)

```
variable = condition ? value1 : value2
```

- Multi-way selection using `if...else if...else` or `switch`

The switch Construct

```
switch (expression)
{
case value1: statement 1;
             break;
case value2: statement 2;
             break;
.
.
.
default:     statement ;
}
```

Examples

1.

2.

- Supports the for, while and do-while looping constructs.

The for Loop

```
for (initialization; condition; update)
{
statements;
}
```

```
for (var i=1; i<=20; i++)
{
document.write(i * i + "<br/>" );
}
```

The while Loop

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

The do-while Loop

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

Examples

1.

2.

More JS

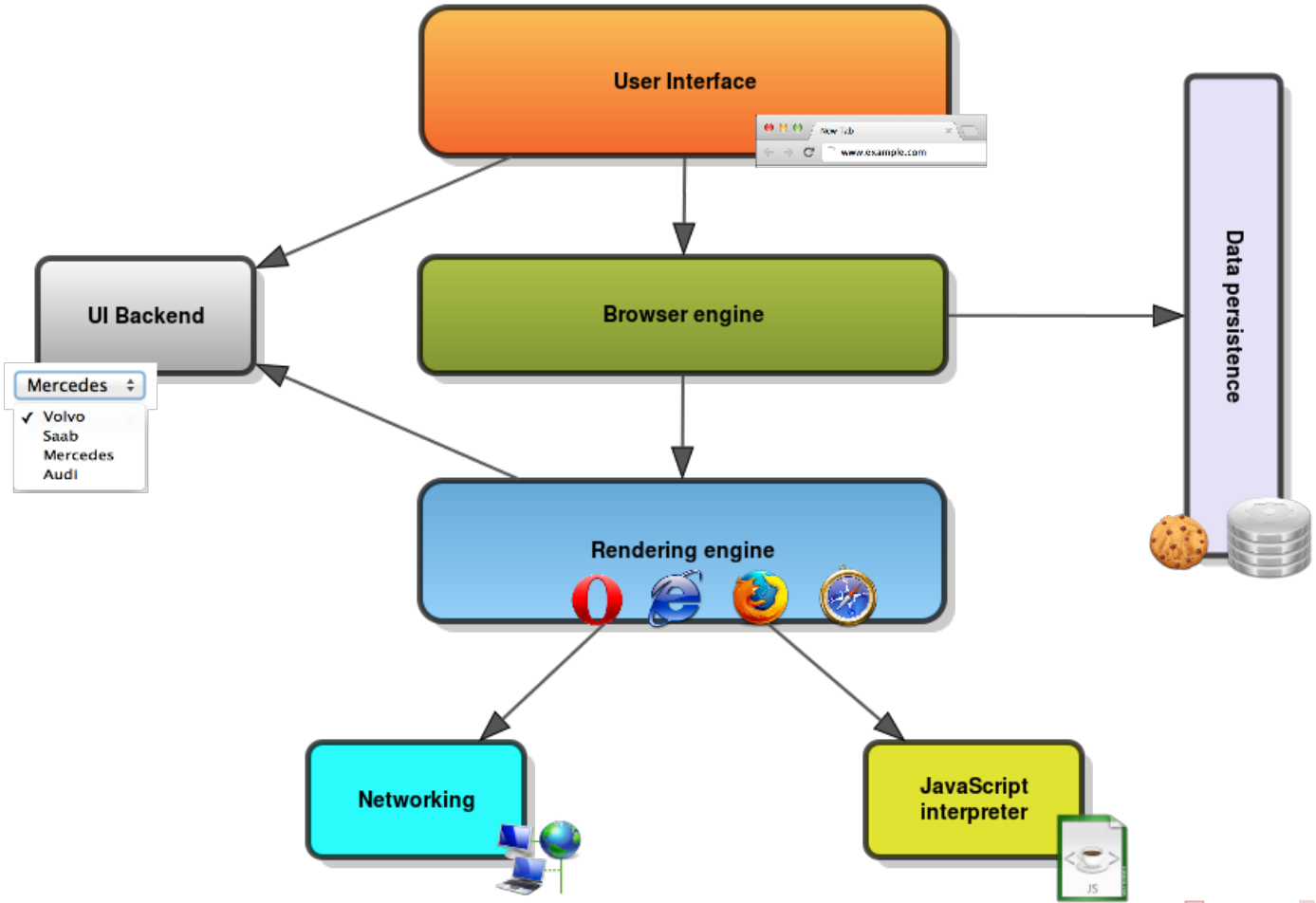
Oct 1, 2013

Today...

- The Document Object Model (DOM)
- Event Handling
- Regular Expressions
- Practical Application (`JS + HTML5`)

Can you make sense out of these browser components?

(Fig source: Figurepool)



Browser Objects (1)

- An HTML page consists of various elements:

  `head, body, headings, buttons, links, images, forms ...`

- To browsers, it is a hierarchical collection of objects.

- Object = properties + methods.

- Each object can be accessed & manipulated independently.

(Check out the `Tilt 3D` Firefox add-on)

The Document Object Model (DOM)

- Standard (not a language) for accessing & manipulating HTML elements (e.g. get/change/add/delete).
- API for web docs (i.e. an abstract model defining the interface b/n HTML docs and apps).
- Can be accessed with JS or any programming language.
- Some HTML DOM methods and properties:
  - `getElementById()`
  - `getElementsByTagName()`
  - `getElementsByClassName()`
  - The `innerHTML` property
  - The `nodeValue` property
  - The `attributes` property

The DOM views HTML documents as a tree structure/ Node Tree.

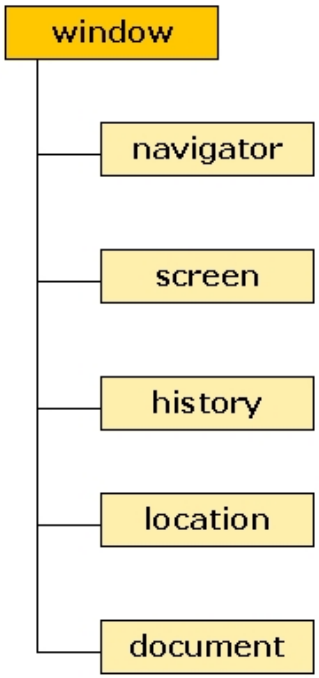
Example

```
<html>
<body>
<p> <b> Paragraph 1 </b> </p>
   <table>
      <tr>
         <td>cell 1</td>
         <td>cell 2</td>
      </tr>
      <tr>
         <td>cell 3</td>
         <td>cell 4</td>
      </tr>
   </table>
</body>
</html>
```

DOM representation = ? root, parents, children, siblings?

Browser Objects (2)

- JS is object-based!
- Offers several browser-based objects that allow interaction with the browser, the document, etc.



- `window` object - represents the browser's window.
- `navigator` object - info about the browser itself.
- `screen` object - info about the client screen.
- `history` object - the browser's history.
- `location` object - info about the current URL.
- `document` object - the active document.

The Document Object

- A loaded document in a browser becomes a document object.
- Allows you to access the (X)HTML elements of a document.
- Some of its properties:

`title`	`URL`	`cookie`	`body`	
`links`	`referrer`	`images`	`forms`	`...`

- Some of its methods:

`write()`	`getElementsByName()`	
`writeln()`	`getElementById()`	`...`

Accessing Elements/Nodes

- Good idea to name/identify your HTML elements.
- Generally, to reference a particular object:

```
ObjectName.propertyName/methodName
```

Example

```
navigator.userAgent

document.write("JS Rocks");

window.document.getElementById("elementID")

document.getElementsByName("orderForm").submit()

window.status = "Hola, I appear on the status bar"      (on Opera)
```

Accessing Elements/Nodes

- Good idea to name/identify your HTML elements.
- Generally, to reference a particular object:

```
ObjectName.propertyName/methodName
```

Example

```
navigator.userAgent

document.write("JS Rocks");

window.document.getElementById("elementID")

document.getElementsByName("orderForm").submit()

window.status = "Hola, I appear on the status bar"      (on Opera)
```

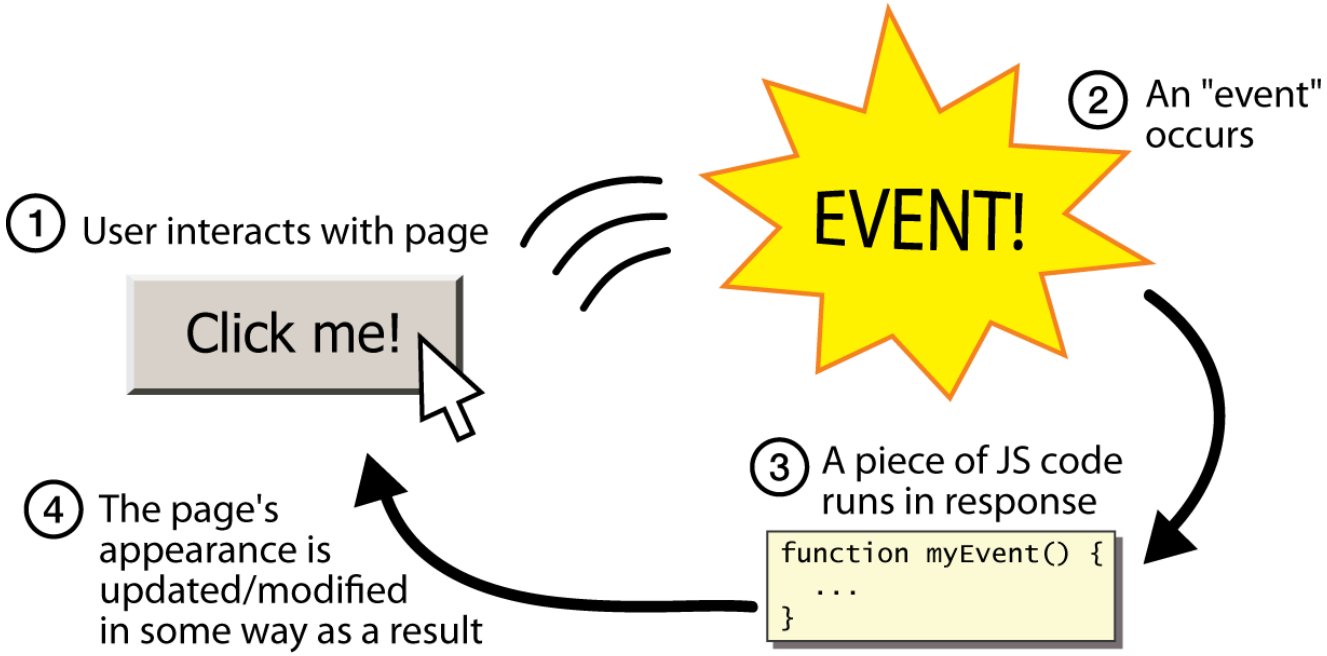
Demo

1.

2.

JavaScript Events

- Events are actions JavaScript can detect and react to.



JavaScript Events (2)

onError	onKeyDown	onMouseMove
onLoad	onKeyUp	onMouseDown
onclick	onKeyPress	onMouseUp
onMouseOver	onFocus	onScroll
onMouseOut	onSubmit	onResize
onSelect	onReset	...

- Built-in support for Regex.
- Via the `RegExp` object

Syntax

```
var re = new RegExp(pattern, flags);        or

var re = /pattern/flags;
```

- `pattern` - a string regular expression
- `flags` - `g` (global), `i` (case-insensitive matching)

```
var msg = "Wisdom is better than rubies";

var pat = /is/i;

var pat = /is/g;
```

- Regex special characters:
  - `.` (dot) any character except '\n'
  - `^` beginning of a line, `$` end of a line
  - `<` beginning of a word, `>` end of a word
  - `[...]` Character set, `[^...]` Negation, Ranges
  - `|` Alternation
  - `(...)` Grouping
  - Quantifiers

`*`	0 or more	`+`	1 or more
`?`	0 or 1	`{n}`, `{n,}`, `{m,n}`	

- Common shortcuts:

```
d (digit)                D (non-digit)          s (space)

S (nonwhitespace)        w (alphabnumeric & _ )
```

- The `test()` and `exec()` `RegExp` methods.

- Useful methods of the `String` object

```
string.match(regex)

string.replace(regex, 'newtext')
```

Demo

1. Form Data Validation (using `JavaScript`)

Demo

1. Form Data Validation (using `JavaScript`)
2. Form Data Validation (using `HTML5`)


HTML
5
FORMS

- Forms pass data to a server (`GET` vs `POST` methods).
- Form input types:

`<input type="X" >` where X can be:

`text`	`button`
`password`	`submit`
`radio`	`reset`
`checkbox`	`hidden`
`file`	

- New + expanded form elements and attributes
- New input types (12+)
- Native browser support for form validation ✓ (The implications?)
- No 100% browser support but promising (visit here and here too).
- How to deal with legacy browsers?

HTML5 Form Attributes

required	readonly
placeholder	multiple
pattern	autocomplete
disabled	list (+ the <datalist> elment)

`<input type="X" >` where X can be:

HTML5 Form Input Types *New!*

`search`	search boxes
`color`	color pickers
`tel`	telephone numbers
`email`	email addresses
`url`	web addresses
`date`	calendar date pickers
`datetime`	date and time (with time zone)
`datetime-local`	local date and time
`month`	to select month + year
`week`	to select week + year
`time`	timestamps
`range`	range of numbers
`number`	numeric values

Other New Form Controls

<output> <progress>

<keygen> <meter>

Other New Form Controls

`<output>`	`<progress>`
`<keygen>`	`<meter>`

Demo

1. Putting it all together ...

- JS is the de facto client-side scripting language. ✓
- `HTML5` comes to the aid. Get to know it more! ✓
- Be mindful of cross-browser compatibility issues. ✓
- Write your scripts with fallback mechanisms. ✓