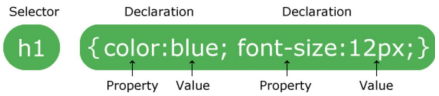
HTML: **attributes**: additional information in tag

Head: <title>, <style>, <link>, <meta>, <script>; Body: Text, Image, Hyperlinks, Lists, Tables, … ; Block & inline elements: (span, td, a, img); List: ol, ul, li; description list: dl, dt, dd; Tables: table, trm tdm border, col/rowspan

- **Forms**: <input type = "text/radio/submit">
  <input type = "radio" name = "gender" value = "male"> Male
  <input type = "radio" name = "gender" value = "female"> Female

CSS: Benefit: customization, reusability, inheritance, attributes tie HTML + CSS



external css files <link rel = "stylesheet" type = "text/css" href = "mystyle.css">; Inline css style <h1 style = "color : blue; margin-left : 30px;">This is a heading</h1>

- Selectors: element, #id, .class, [attribute], *, (div,p), (div p), (div>p), (div+p), pseudo-classes(:), pseudo=elements(::)
- Properties: color, background, font, text, opacity, list, inheritance
- !important
- **Specificity**: style > id > class, pseudo-class, attribute > elements, pseudo-element
- Box model: margin, border, padding, content
- Float, clear, overflow, positioning (static, relative, fixed, absolute, sticky), z-index (+ in front), display & visibility
- Nav bar: link/visiter, hover/active
- Box-sizing: border-box

flex-direction: column, row, column-reverse, row-reverse; flex-wrap: wrap, nowrap, wrap-reverse

justify-content: align items horizontally: center, flex-start/end, space-around/between

align-items: align items vertically: center, flex-start/end, stretch, baseline

align-content: align flex lines: center, flex-start/end, stretch, space-between/around

order (>=1), flex-grow (>=0), flex-shrink (>=1), flex-basis (initial length); Align-self: alignment for selected item, override "align-items"

- Responsive: <meta name = "viewport" content = "width = device-width, initial-scale = 1.0">
  - @media only screen and (max/min-width: ..px)

JS: Interpreted language, loosely typed, fail "silently", object-oriented (prototype-based)

- Type: dynamic (same variable can hold different types)
  - Primitive: string, number, boolean, undefined
  - Special: typeof(undefined) = undefined, typeof(null) = object
  - Complex: function, object (array is object)
- Hoisting: declaration yes, initialization no
- Objects: name : "value"
  - Access property: person.lastName; or person["lastName"];
  - Access method: person.fullName(); or person["fullName"](); person["fullname"] gives raw code
  - Primitives can be objects, complex types are always objects
  - Modify property: Person.prototype.nationality = "English";
  - Add method to property: Person.prototype.name = function();
  - String methods: slice(start, end), substring(start, end), substr(start, length), replace(before, after), char(Code)At(num)
  - String to array: split(",")/""/)
  - Array method: join(" * "), pop(), push("newLast"), shift(), unshift("newFirst"), splice(start, #remove, "add", "add"), slice(), sort(), reverse()
- = = = and != =
- Closures: a function having access to the parent scope, even after the parent function has closed

```
var add = (function () {
        var counter = 0;
        return function () { return counter += 1; }
});
```

DOM: document object model; interface for HTML, a tree of objects

getElementById("intro"); getElementsByTagName("p"); getElementsByClassName("intro"); querySelectorAll("input[type=button]");

- Change HTML attributes: document.getElementById("myImage").src = "landscape.jpg";
- Change CSS style: document.getElementById("p2").style.color = "blue"
- Event: on(un)load, onchange, onmouseover/out, onkeydown
  - <h1 = onclick = "this.innerHTML = 'Ooops!'">Click on this text!</h1>
  - element.addEventListener("click", function() { alert("hello"); });
  - Bubbling (in → out), capturing [parameter "useCapture" = "false" by default]
- Nodes: appendChild(node); insertBefore(para, node); removeChild(child);

```
funnction getResourceByURI(num)
    return resources.concat("sample", num, ".php");

function generate(response) {
    var info = JSON.parse(response);
    if(info.hasOwnProperty("friends") && Array.isArray(info.friends)) {
        var list = $("<ul></ul>");
        for(friend of info.friends)
            list.append("<li>" + friend.name + "</li>");
        $("#sample3").append(list);
    }
}

for(var i = 1; i <= 2; i ++) {
    var resourceURI = getResourceByURI(i);
    $("#sample".concat(i)).load(resourceURI);
}

$.get(getResourceByURI(3), generate);
```

```
exports.handleTodoList = function(req, res, session) {
    switch(req.method) {
        case "GET":
            nodeTodoDB.getTodos(session.id, (err, data) => {
                let dbList = null;
                if(!err) dbList = data.map(row => row.description);
                const todoList = dbList ? session.todoList = dbList : session.todoList || [];
                res.writeHead(200, {'Content-Type': 'application/json'});
                res.end(JSON.stringify(todoList.map(todo, index) => ({
                    id: index,
                    description: todo
                })));
            });
            break;
        case "POST":
            if(!session.todoList) session.todoList = [];
            convertRequest(req, data => {
                session.todoList.push(data.todo);
                nodeTodoDb.addTodo(session.id, data.todo, err => {
                    if(err) {
                        console.log(err);
                        rew.writeHead(500, {'Content-Type': 'text/html'});
                        return res.end("500 Internal Server Error");
                    }
                    res.writeHead(200, 'OK');
                    return res.end();
                });
            });
            break;
        default:
            res.writeHead(405, {'Allow': 'GET, POST'});
            res.end("Not Allowed");
    }
};
```

```
exports.addTodo = function(sessionId, todo, callback) {
    const con = mysql.createConnection(config);
    con.connect(function(err) {
        if(err) return callback(err);
        console.log("Connected");

        var sql = "INSERT INTO Todos (description, sessionId) VALUES (?, ?)";
        con.query(sql, [todo, sessionId], function(err, result) {
            if(err) throw err;
            console.log(result);
        });
        con.end();
    });
};

exports.getTodos = function(sessionId, callback) {
    const con = mysql.createConnection(config);
    con.connect(function(err) {
        if(err) return callback(err);
        console.log("connected");

        var sql = "SELECT * FROM Todos WHERE sessionId = ?";
        con.query(sql, [sessionId], function(err, result) {
            console.log(result);
            callback(err, result);
        });
        con.end();
    });
};
```

- Window: property, location, history, popups (alert, confirm, prompt)
- setTimeout, setInterval, clearInterval

jQuery: a JS library, easy to learn & use, cross-browser compliant, $(selector).action()
- Event: $("p").on("click", function() {}); res.write('<td> ${i} * ${j} = ${i * j}</td>');
- CSS: $("#div1").add/removeClass("important blue"); $("p").css("color", "yellow");
- DOM manipulation: $("#test").text/html/val()
  - Callback function: $("#test1").text/html/val(function(i, orig){});
  - append(), prepend(), after(), before(), remove(element & children), empty(children)
- Traverse: parent(), parents(), parentsUntil(), children(), find(), siblings(), next/prev[All/Until]()
- Effect: toggle, hide/show, fadeIn/Out/Toggle/To, slideUp/Down/Toggle, animate
- Chaining: $("#p1").css("color", "red").slideUp(2000).slideDown(2000);
- AJAX: asynchronous JS & XML, $(selector).load(URL, data, callback); $.get(URL, callback); $.post(URL, data, callback);
- HTTP: browser open a connection → browser sends request → browser reads response from server → browser closes connection
  - var xhttp = new XMLHTTPRequest(); xhttp.onreadystatechange = function() {}; xhttp.open/send()

ES6: the scripting language that forms the basis of JS
- Hosting: var yes, let no, const no
- You can change the properties of a const, but you can NOT reassign a const
- Arrow functions: const x = (x, y) => { return x * y }; no **this**, no **hoist**
- Class: no **hoist**

```
let Rec = class Rectangle {
        constructor() {}
        calcArea() {} // prototype method
};
```

```
class Form extends Rectangle {
    constructor(name) {
        super(name);    // call the super class constructor
    } speak() {} }
```

Node.js: open source server environment, send tasks → ready to handle next request → return to client (eliminate waiting)
- var http = require('http'); http.createServer(function(req, res) {}); [HTTP is a built-in module, allows data transfer]
- Fs: readF(f, fc), appendF(f, .., fc), open(f, 'w', fc), writeF(f, .., fc), unlink(f, fc), rename(old, new, fc)
- NPM: package manager, or module; var eventEmitter = new events.EventEmitter(); eventEmitter.emit('scream');

State management: HTTP statelemess
- Cookie: small piece of data, sent back on next request, used to check if request comes from same browser, stored on user computer
- Purpose: session management, personalization, tracking; Set-Cookie: cookieName = cookieValue
- Session cookie: deleted when browser is closed; permanent cookies: Expires, Max-Age
- Secure cookie: HttpOnly to prevent XSS (cookie can't be accessed by JS); CSRF: sanitize input, short cookie lifetime
- PHP cookie: setcookie(name, ..) **[before <html>]**; count($_COOKIE) > 0: cookies enabled
- PHP session: store info and use across pages; last until close browser; session_start() **[before <html>]**; $_SESSION[ ]
- localStorage (no expiration), sessionStorage (data lost when close browser); set/get/removeItem()

Database: structured collection of data

CREATE DATABASE db; DROP DATABASE db; SHOW DATABASE;
CREATE TABLE tb (c1 type, c2 type); DROP/TRUNCATE TABLE tb; ALTER TABLE tb ADD/DROP/(MODIFY COLUMN) c type NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, DEFAULT, AUTO_INCREMENT
SELECT c FROM tb WHERE; ORDER BY ASC/DESC; INSERT INTO tb (c1, c2) VALUES (v1, v2); UPDATE tb SET c=v WHERE; DELETE FROM tb WHERE; COUNT(c); LIKE (%, _); mysql.escape(adr)

```
var con = mysql.createConnection({});
var sql = "INSERT INTO customers (name, address) VALUES ?";
var values = [ ['John', 'a'], ['Mary', 'b']];
con.query(sql, [values], function(err, result) {});
```

Express.js: fast, unopinionated, minimalist web framework for Node.js
- Routing: app.METHOD(PATH, HANDLER); app.get/post/put/delete('/', function(req, res) { res.send('hello'); });
- Middleware: req, res, next middleware func[next()]