INTRO TO HTML, CSS, AND JAVASCRIPT

S. Rinzivillo – rinzivillo@isti.cnr.it
OUTLINE

• Web Application Architecture
• Crash courses on:
  • HTML
  • CSS
  • Javascript
• Web Server
  • Node.js and NPM
WEB APPLICATIONS ARCHITECTURE
STATIC WEBSITES

- The content of each page is sent AS IS from the server to the client.
DYNAMIC WEBSITE

• Web page content is composed on demand
• Content is stored in different forms: databases, external resources, other static web pages
SERVER SIDE VS CLIENT SIDE

• Client-side coding includes HTML, CSS, and Javascript
• This code is transmitted AS IS and executed in the browser
WEB SERVER

• Implements HTTP protocol
  • The web server handles a folder, called Document Root
  • For security reasons, only the files within the DocRoot are visible for the web server

• A web server is reachable via a URL

• A URL consists of 4 parts:
  • A selector of the protocol (http or https)
  • The domain name of the server (www.nytimes.com)
  • The port number (by default it is 80)
  • A path to localize additional information
  • Ex: http://www.nytimes.com:80/sport/baseball
HTTP

• 3. Send
• HTTP Request
  • Methods to tell server what the client need
• HTTP Methods:
  • GET; POST; PUT; DELETE; OPTIONS;...
4. Wait and 5. Load

HTTP Response
- Read Response Codes
- Read data

HTTP Response Codes
- 1xx – Informational
- 2xx – Success
- 3xx – Redirection
- 4xx – Client Error
- 5xx – Server Error
EXAMPLE – REQUESTING PAGE.HTML

Client
• GET ‘/page.html’
• ... wait ...
• Download and parse file
• GET ‘/css/style.css’
• ... wait ...
• Download and parse file
• GET /page.html
• ... wait ...
• Page not found error

Web Server
• Search for the file page.html
• Send response 200
• Send the content of file
• Look in folder css for file
• Send response 200
• Send content of file
• Search for file page.html
• Send response 404
HTML, CSS, AND JAVASCRIPT

- HTML
- CSS
- Javascript

- Structure
- Presentation
- Behavior
HTML 101
HYPER TEXT MARKUP LANGUAGE (HTML)

• Hyper Text

• Markup Language
  • Composed of markup tags
  • Tags group and describe page content
MARKUP LANGUAGE

• HTML tags give structure
• They also provide semantics
  • Headings for headers
  • UL for unordered list
  • …
• Browser applies built-in styles to each tag

• Even with default style, web pages should be readable and its hierarchy clear
RELEVANT TAGS: DOCTYPE

• It is not a common tag
  • No closing tag
  • Opening with “!”

• It is a declaration

• Select the correct dialect of HTML the page is using

• E.g.: <!DOCTYPE html> selects HTML5
RELEVANT TAGS: HTML

• This tag enclose the whole document
• <html></html>
RELEVANT TAGS: HEAD

• It provides information to browser to retrieve additional information for the page
  • Javascript, styles, information, meta, etc.
• <head></head>
RELEVANT TAGS: BODY

• Contains the document content
• The enclosed tags are showed in the browser window
• <body></body>
MINIMAL STRUCTURE

• This is a basic structure for a web page
• HTML uses nesting to code hierarchies
• For readability, enclosed tags are indented w.r.t. container
**DOCUMENT HIERARCHY**

- Each tag has a parent
- A tag may have children or siblings
- Examples:
  - `h1` is a child of `body`
  - `body` has two children
  - `p` is sibling of `h1`
HTML ELEMENT

• An element is the union of two corresponding tags and their content

• Tags are usually present in pairs:
  • Start tag
  • End tag

<tag>Content</tag>
NAMED TAGS

• HTML has a set of predefined tag names, associated with special structures

\(<h1>My Title</h1>\)
ESSENTIAL TAGS

• Primary Structure
  • html
  • head
  • body

• Head Elements
  • title
  • meta
  • link

• Formatting elements (inline)
  • em, i
  • strong, b
  • q, blockquote
  • Span

• Structural Elements (blocks)
  • p
  • h1-h6
  • ul, ol
  • a
  • img
  • div
CSS – CASCADING STYLE SHEET

• A stylesheet specifies a set of rules to define how html elements are presented on the browser
• Each rule applies to a specific set of elements of the page
• Rules have a cascading behaviour
  • Conflicts between multiple rules are resolved by priorities
  • Elements not covered by explicit rules inherit presentation of ancestors
RULE PRIORITIES

• Browser stylesheet
• Linked external stylesheet
• Embedded stylesheet (tag style)
• Inline style (attribute style)
INHERITANCE

body
make font 16px, Verdana, red

\[
p \quad \text{make font blue}
\]

h1 is red
p is blue
ANATOMY OF A CSS RULE

• Every rule is composed of a selector and a declaration
• Declaration contains at least one pair property/value

selector {property: value;}

[ ] selector [ ] declaration
BASIC CSS SELECTORS

• Type selectors
  • Target an element by name
    • body {font-family: Verdana }
    • h1 {color: red}

• ID selectors
  • An ID is an attribute added to an HTML element
    • #logo {declaration}
    • <img id="logo" src="img/imga.jpg" alt="description"/>

• Class selectors
  • An identifier attribute added to a set of HTML elements
    • .ingredients {declaration}
    • <ul class="ingredients">
**ID OR CLASS**

- There can be only one element with a given ID
- ID is more specific than a class
- An element can have both ID and classes
DESCENDANT SELECTORS

• Descendant selectors are composed of two basic selectors separated by a space
• The rule targets the elements of the second selectors that are descendent of the element of the first selector
• Example
  • #sidebar .author {declaration}
  • <div id="sidebar">
      <p class="author"></p>
    </div>
    <p class="author"></p>
JAVASCRIPT 101
Eloquent Javascript – Second Edition
Marijn Haverbeke
Licensed under CC license.
Available here: http://eloquentjavascript.net/
DEVELOPER TOOLS (SAFARI, CHROME, FIREFOX)
JAVASCRIPT CONSOLE (SAFARI, CHROME, FIREFOX)
VARIABLES

• Containers for data
  let number = 5;
  let address = “Largo Bruno Pontecorvo 5”;
OBSERVABLE HQ

• Collection of notebooks for fast prototyping Javascript solutions
  • https://observablehq.com/collection/@rinziv/va602aa
Arrays

• Store sequences of values with a single name
  
  ```javascript
  let numberA = 5;
  let numberB = 10;
  let numberC = 15;
  let numberD = 20;
  let numberE = 25;
  let numbers = [ 5, 10, 15, 20, 25 ];
  numbers[0]  //Returns 5
  numbers[1]  //Returns 10
  numbers[2]  //Returns 15
  ```
# Objects

- A sort of custom data structures
- Object is declared with curly brackets
- A sequence of property value pairs are separated by commas

```javascript
let fruit = {
    kind: "grape",
    color: "red",
    quantity: 12,
    tasty: true
};

fruit.kind  // Returns "grape"
fruit.color  // Returns "red"
fruit.quantity  // Returns 12
fruit.tasty  // Returns true
```
let fruits = [
  {
    kind: "grape",
    color: "red",
    quantity: 12,
    tasty: true
  },
  {
    kind: "kiwi",
    color: "brown",
    quantity: 98,
    tasty: true
  },
  {
    kind: "banana",
    color: "yellow",
    quantity: 0,
    tasty: true
  }
];
CONTROL STRUCTURES

• If statement
  
  if (test) {
      //Code to run if true
  }

• Example
  
  if (3 < 5) {
      console.log("Eureka! Three is less than five!");
  }

• for statement
  
  for (initialization; test; update) {
      //Code to run each time through the loop
  }

• Example
  
  for (let i = 0; i < 5; i++) {
      console.log(i);  //Prints value to console
  }
FUNCTIONS

• Declaration
  ```javascript
  let functionName = function(arg1, arg2){
    return something;
  }
  ```

• Example
  ```javascript
  let calculateGratuity = function(bill) {
    return bill * 0.2;
  };
  ```

• Call of a function
  ```javascript
  functionName(arg1, arg2)
  var tip = calculateGratuity(38);
  console.log(tip);  //Prints 7.6 to the console
  ```
LINK TO JAVASCRIPT FROM A WEB PAGE

• Embedded within body element

```html
<body>
  <script type="text/javascript">
    alert("Hello, world!");
  </script>
</body>
```

• Linked from the head section

```html
<head>
  <title>Page Title</title>
  <script type="text/javascript" src="myscript.js"></script>
</head>
```
DEVELOPMENT CHECKLIST
TOOLS

• A modern browser (Chrome, Firefox, etc)
• An integrated IDE, like WebStorm for example
• Node.js and NPM installed
TOOLS (OLD SCHOOL)

• A modern browser (Chrome, Firefox, etc)
• A modern text editor (TextMate, Sublime, Atom, ...)
• A terminal (Command prompt) to run an http-server [Terminal A]
• A terminal to handle code versioning [Terminal B]
• Node.js and NPM installed
FAST PROTOTYPING WITH OBSERVABLES

Welcome to Observable! Get started with these four tutorials:

- **Five-Minute Introduction**
  - Learn how Observable notebooks work with simple examples.

- **Tutorial 1: Lunch Calculator**
  - Build a series of simple notebooks to learn the fundamentals.

- **Introduction to Data**
  - Databases, files, inline, spreadsheets, and data of all kinds.

- **Introduction to require**
  - How to use open source modules with Observable.

https://observablehq.com/collection/@rinziv/va602aa