AN INTRODUCTION TO MARKUP AND ENCODING

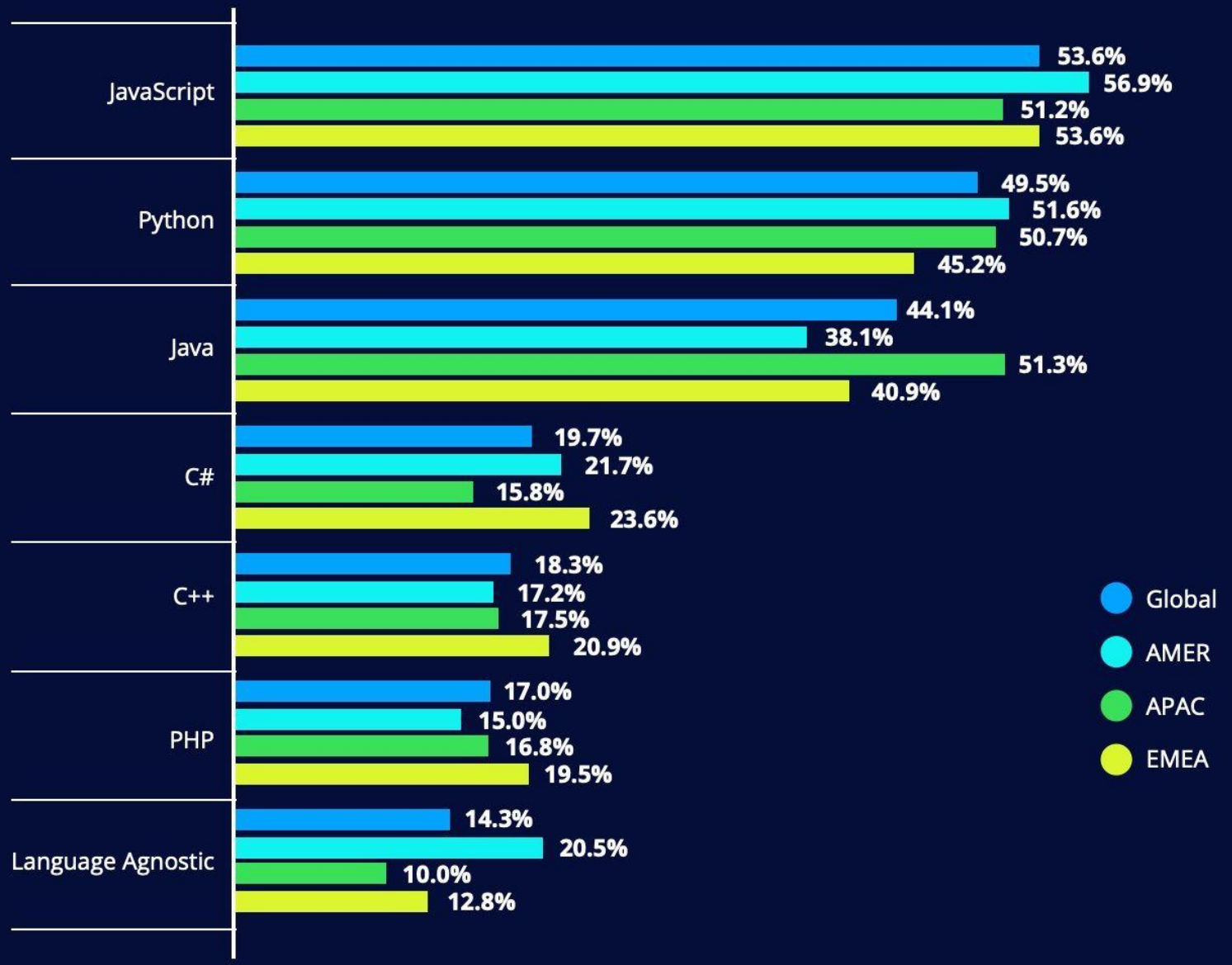
XML, JSON, BASE64

COS216
AVINASH SINGH
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF PRETORIA

MOST POPULAR PROGRAMMING LANGUAGES

Feb 2023	Feb 2022	Change	Programming Language	Ratings	Change
1	1		Python	15.49%	+0.16%
2	2		C	15.39%	+1.31%
3	4	^	C++	13.94%	+5.93%
4	3	v	Java	13.21%	+1.07%
5	5		C#	6.38%	+1.01%
6	6		Visual Basic	4.14%	-1.09%
7	7		JavaScript	2.52%	+0.70%
8	10	^	SQL	2.12%	+0.58%
9	9		Assembly language	1.38%	-0.21%
10	8	v	PHP	1.29%	-0.49%
11	11		Go	1.11%	-0.12%
12	13	^	R	1.08%	-0.04%
13	14	^	MATLAB	0.99%	-0.04%
14	15	^	Delphi/Object Pascal	0.95%	+0.05%
15	12	v	Swift	0.93%	-0.25%

What language skills do you look for when hiring developers?



https://www.youtube.com/watch?v=RH0o-QjnwDg

EXTENSIBLE MARKUP LANGUAGE

- Extensible Markup Language (XML)
- Markup language that defines syntax rules and format for documents
- Machine-readable and human-readable
- Allows for almost all Unicode characters

- File extension: .xml
- Mime type: application/xml or text/xml

EXTENSIBLE MARKUP LANGUAGE

- XML emphasize simplicity, generality, and usability across the Internet
- APIs mainly used XML in the past to exchange data

- XHTML is an application of XML
  - HTML uses the same rules and syntax as XML
  - HTML started in 1993
  - XML started in 1996

- Extended from: SGML (1986)
- Extended to: XHTML, RSS, KML, Atom

XML - TAGS

- Data is enclosed by opening and closing tags
- Developers can choose any names for the tags

- Opening tag: `<element>`
- Closing tag: `</element>`
- Example:

```
<element>Some data here</element>
```

- Elements without data (empty elements) can be abbreviated
  - This `<element></element>` is equivalent to this `<element/>`

XML - ELEMENTS

- Elements are logical documents
- Elements have an opening and closing tag
- Elements can have other nested elements
- Each XML document can only contain a single **root** element

- Example element with text:

```
<element>Some data here</element>
```

XML - ELEMENTS

- Example element with nested elements:

```
<element>
        <element2>Some data here</element2>
        <element3>Some other data here</element3>
</element>
```

- Elements cannot contain text and nested elements at the same time
- This is **not** allowed:

```
<element>
        Some data here
        <element2>Some other data here</element2>
</element>
```

XML - ATTRIBUTES

- Parameters can be added as attributes to elements
- All attributes must be enclosed in double-quotes, even numbers

- Attribute format:
- Example:

```
attributeName="attributeValue"
```

```
<element attr="1.2.3">Some data here</element>
```

XML – ELEMENTS VS ATTRIBUTES

- No specification when to use elements and when to use attributes
- Decision left up to the developer
- Both of these are valid XML representations:

```
<bitcoin price="8102"></bitcoin>
```

and

```
<bitcoin>
        <price>8102</price>
</bitcoin>
```

XML – DECLARATION

- A XML declaration is needed
- Must be placed at the start of the document
- Provides information to parsers and interpreters
  - That the document is XML
  - Which version of XML is used
  - Which character encoding is used

- Example:

```
<?xml version="1.0" encoding="UTF-8"?>
```

XML – COMMENTS

- Comments can be added which are ignored by the parser

- Opening tag: `<!--`
- Closing tag: `-->`
- Example: `<!-- This is my random comment -->`

XML – ESCAPING

- XML uses some characters in its syntax (< > " ' &)
- If you want to use these characters in your test, they have to be escaped
- Can also use Unicode hexadecimal
- Starts with an ampersand (&) and ends with a semicolon (;)

Character	Escaped
>	>
<	<
&	&
'	'
"	"
€	€ or €

XML – EXAMPLE

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- A summary of crypto currencies. -->
<currencies>
	<bitcoin symbol="BTC">
		<price>8102</price>
		<change duration="12">2.1</change>
		<change duration="24">4.2</change>
	</bitcoin>
	<ethereum symbol="ETH">
		<price>904</price>
		<change duration="12">3.2</change>
		<change duration="24">6.3</change>
	</ethereum>
</currencies>
```

```
<!ELEMENT element-name category>
or
<!ELEMENT element-name (element-content)>
```

XML – DTD

- Document Type Definition (DTD)
- Defines the structure and the legal elements and attributes of an XML document
- Declared inside XML wrapped in the DOCTYPE (inline mode) otherwise a separate file ".dtd" file for external declaration

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend</body>
</note>
```

XML – DTD (ATTRIBUTES)

- Attributes are declared with an ATTLIST declaration.

```
<!ATTLIST element-name attribute-name attribute-type
attribute-value>

DTD example:
<!ELEMENT payment EMPTY>
<!ATTLIST payment type CDATA "check">

XML example:

<payment type="check" />
```

XML – SCHEMA

- Describes the structure of the XML document
- XML Schema is also referred too XML Schema Definition (XSD)
- Purpose is to define the legal building blocks:
  - the elements and attributes that can appear in a document
  - the number of (and order of) child elements
  - data types for elements and attributes
  - default and fixed values for elements and attributes

XML – SCHEMA

```xml
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="https://www.w3schools.com"
xmlns="https://www.w3schools.com"
elementFormDefault="qualified">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

```xml
<?xml version="1.0"?>

<note
xmlns="https://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://www.w3schools.com/xml note.xsd">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

XML – MORE INFO

https://www.w3schools.com/xml/default.asp

https://www.w3schools.com/xml/xml_dtd_intro.asp

https://www.w3schools.com/xml/schema_intro.asp

XML – VALIDATION TOOLS

- You can prettify and validate your XML:
  - https://www.w3schools.com/xml/xml_validator.asp
  - https://codebeautify.org/xmlvalidator

XML – PROBLEMS

- XML has long opening and closing tags
  - Unnecessary data consumption
- XML does not support native arrays
- XML requires more code to create a parser

JAVASCRIPT OBJECT NOTATION

- JavaScript Object Notation (JSON)
- Simple structure to represent data
- Easy to interpret by humans and machine
- Released in early 2000s
- Replaces XML for most online websites and APIs
- Based on the object structure of JavaScript, but is language-independent

- File extension: .json
- Mime type: application/json

JAVASCRIPT OBJECT NOTATION

- Is build on top of three structures:
  - Standalone variables: booleans, integers, floats, strings, null
  - Arrays: list of variables, objects, or other arrays
  - Objects: contains other variables, objects, or arrays

- Advantages over XML:
  - Concise and therefore saves bandwidth
  - Supports arrays
  - Easier to understand
  - Faster to implement parser/interpreter code
  - Supported by most programming languages (although XML is also widely supported)

JSON SYNTAX

- Names must be enclosed in double quotes (")
  - Many parsers except single quotes, but this is not valid JSON
- Name-value pairs are separated by a colon
- Variables are separated by a comma
- JSON root must be either an object or an array.
- JavaScript comments are allowed
  - However, most parsers cannot interpret comments and fail/crash
  - Avoid comments

JSON VARIABLES

- Null: `"degree" : null`
- Boolean: `"valid" : false`
- Integer: `"timestamp" : 1518181140`
- Float: `"price" : 12.3584`
- String: `"name" : "Satoshi Nakamoto"`

- Booleans are send as words which wastes bandwidth
  - Rather use integers 0 or 1
  - Almost all programming languages see 0 and 1 as Boolean synonyms

JSON OBJECTS

- An object can contain other variables, objects, arrays, or be empty
- Enclosed in curly opening and closing brackets ({ })
- Root objects and objects in arrays are unnamed (without a name)
- Any other objects must be named

```
{
        "timestamp" : 1518181140,
        "name" : "Satoshi Nakamoto"
}
```

```
{
        "timestamp" : 1518181140,
        "name" :
        {
                "firstname" : "Satoshi",
                "lastname" : "Nakamoto"
        }
}
```

JSON ARRAYS

- A list with multiple (or no) values
- Array elements can be of different type
- Enclosed in square opening and closing brackets ([ ])

```
["value1", "value2", "value3"]
```

```
["value1", true, 1456, {"name" : null}]
```

JSON EXAMPLE

```
{
        "bitcoin" :
        {
                "symbol" : "BTC",
                "price" : 8102,
                "change" : {"12" : 2.1, "24" : 4.2}
        },
        "ethereum" :
        {
                "symbol" : "ETH",
                "price" : 904,
                "change" : {"12" : 3.2, "24" : 6.3}
        }
}
```

JSON VS XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<currencies>
            <bitcoin symbol="BTC">
                        <price>8102</price>
                        <change duration="12">2.1</change>
                        <change duration="24">4.2</change>
            </bitcoin>
            <ethereum symbol="ETH">
                        <price>904</price>
                        <change duration="12">3.2</change>
                        <change duration="24">6.3</change>
            </ethereum>
</currencies>
```

```json
{
            "bitcoin" :
            {
                        "symbol" : "BTC",
                        "price" : 8102,
                        "change" : {"12" : 2.1, "24" : 4.2}
            },
            "ethereum" :
            {
                        "symbol" : "ETH",
                        "price" : 904,
                        "change" : {"12" : 3.2, "24" : 6.3}
            }
}
```

- Minified without comments.
- XML: 302 characters
- JSON: 139 characters
- JSON in this case is 46% smaller than XML

JSON TOOLS

- Whitespaces (spaces, tabs, newlines) are not essential
- Remove all spaces before transferring JSON to save bandwidth
  - This process is called minification/minify
  - Use for various web documents (JSON, XML, HTML, JS, CSS)

- Validate and prettify your JSON:
  - https://jsonformatter.curiousconcept.com
- Minify your JSON:
  - https://codebeautify.org/jsonminifier

BASE64

- Transferring data in a unified format is difficult
  - Foreign languages with Unicode characters
  - Binary data like images
- Base64 is binary-to-text encoding scheme
  - Any data can be presented in widely supported ASCII notation
- Uses 64 characters: a-z, A-Z, 0-9, and the + and / characters
- Base64 is about 30% larger than the original data
  - However, it can easily be encoded and transferred, even through JSON APIs
  - Avoid for very large files such as videos, use for smaller files such as images, PDFs, text files, etc

BASE64

Hello World!

Base64 Encode

SGVsbG8gV29ybGQh

BASE64

Base64 Encode Image

iVBORw0KGgoAAAANSUhEUgAAAfQAAAH0CAYAAADL1t+KAAAABHNCSVQICAg
IfAhkiAAAAAlwSFlzAACtAwAArQMBsABLVAAAAB10RVh0U29mdHdhcmUAd3d3Lm
lua3NjYXBlLm9yZ5vuPBoAADi8SURB .....

BASE64

- Can be directly added to HTML images:

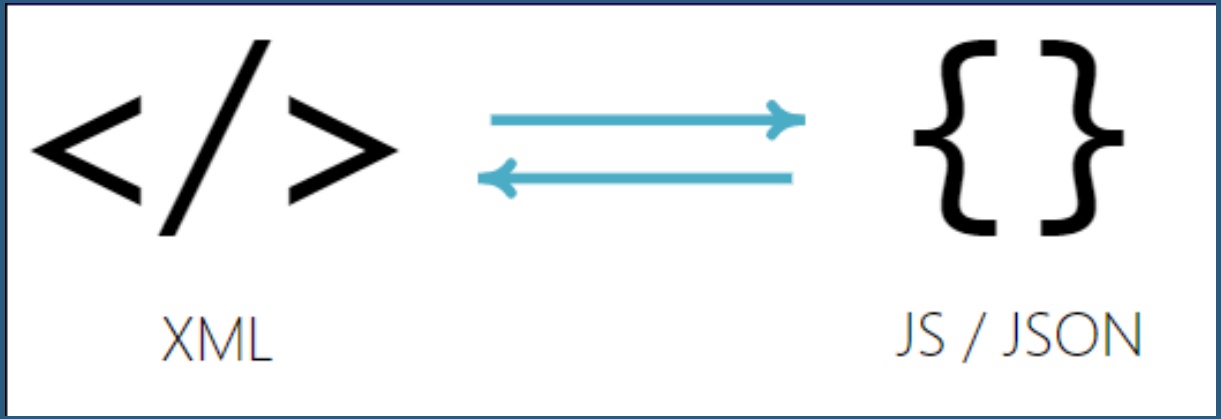
```
<img alt="Embedded Image" src="..." />
```

- Can be directly added to CSS images:

```
.image
{
        width:100px;
        height:100px;
        background-image:url(...);
}
```

BASE64 - TOOLS

- Base64 Text:
  - Encode: https://www.base64encode.org
  - Decode: https://www.base64decode.org
- Base64 Images:
  - Encode: https://www.base64encode.net/base64-image-encoder
  - Decode: https://www.base64decode.net/base64-image-decoder


XML
JS / JSON