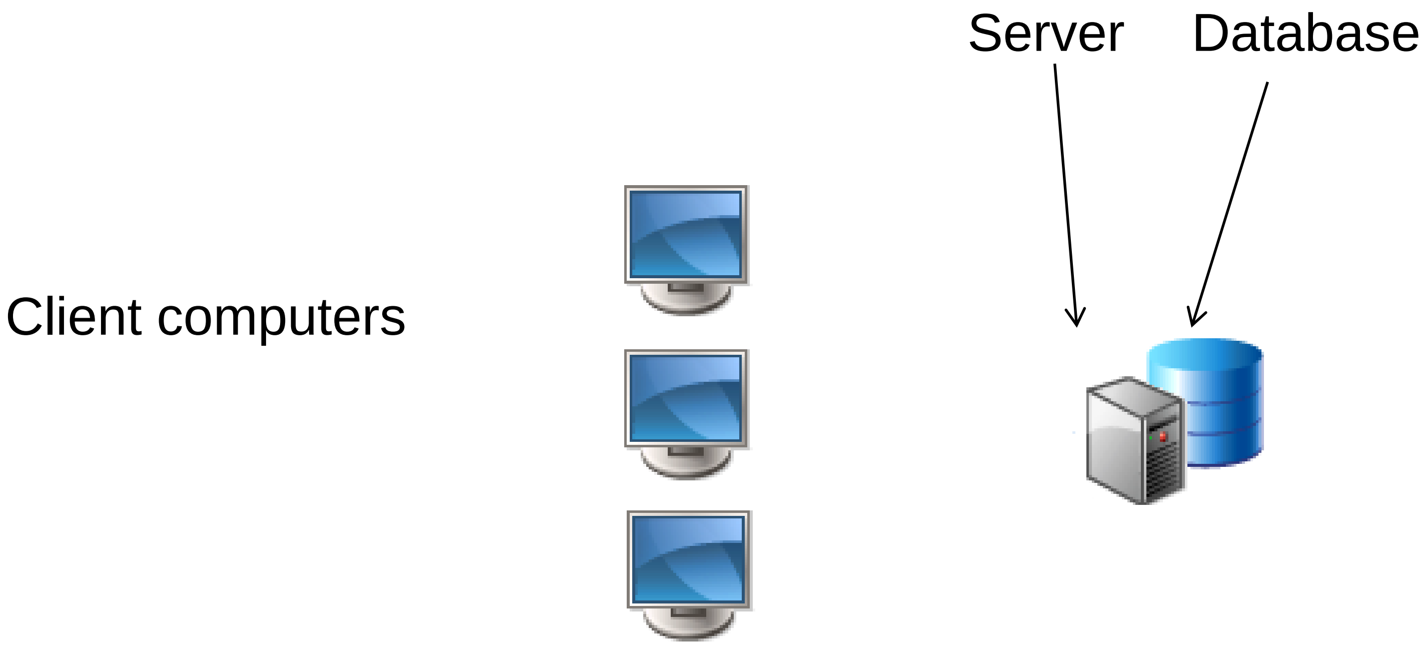
Web Apps

Petar Ivanov
September 09, 2018

PUBLIC

Web Apps



Frontend

- HTML – content
- CSS – appearance
- JavaScript – makes the frontend dynamic

But first:

Frontend
XML

XML is a set of rules for defining self-describing data.

XML documents are both human readable and machine readable.

XML documents are called 'self-describing', because they do not expose only data, but context as well.

Every document consists of 2 ingredients:

- Content – the data itself

- Markup – additional text that represents the context of the data

And those two ingredients are mixed.

Frontend

XML

Jim,

Playing golf with you was a pleasure. I hope we can do that again!

PS: but don’t let me win next time

Eduard

- ❑ Plain text
- ❑ Human readable mostly

```
<letter>
    <from>Eduard</from>
    <to>Jim</to>
    <content>
        Playing golf with you was a pleasure. I hope we can do that again!
    </content>
    <postscriptum>but don’t let me win next time</postscriptum>
</letter>
```

- ❑ XML
- ❑ Human readable
- ❑ Machine readable

Frontend
XML

- Tags – surrounded in angle brackets (< and >)
  1. Starting tag - <from>
  2. Ending tag - </from>
  3. Empty tag - <separator />

- Elements – contain <starting-tag> + content + </ending-tag>
- Attributes – name-value pairs that may be a part of a start or an empty tag. They provide more details about a specific element.
- <letter importance="high"> … </letter>

Frontend
XML

- Elements can be nested
  - <letter> <from> Eduard </from> </letter>
- Pay attention to the order of the opening and closing tags. For every starting tag there got to be a corresponding ending tag:

Good	Very, very bad
<letter> <from> Eduard </from> </letter>	<letter> <from> Eduard </letter> </from>

- Every XML document consists of one single element – the root. Of course, all other element are just nested.
- An XML document is called well-formed if it is compliant with all the rules.

Frontend

XML

Example of a an XML schema:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="letter">
    <xs:complexType>
      <xs:all>
        <xs:element type="xs:string" name="from"/>
        <xs:element type="xs:string" name="to"/>
        <xs:element type="xs:string" name="content"/>
        <xs:element type="xs:string" name="postscriptum" minOccurs="0"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Frontend
XML

There are two errors in the following document. Can you find them?

```
<letter>
    <to>John</too>
    <from>Jim</from>
    <content>See you in an hour!<content>
</letter>
```

Frontend
HTML

```
<!doctype html>
<html>
    <head>
        <title> My first ever HTML page! </title>
    </head>

    <body>
        Hello HTML World!
    </body>
</html>
```

Frontend
HTML

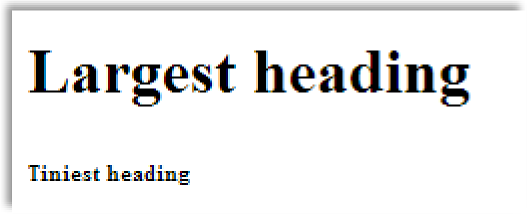
```
<head>
    <title> My first ever HTML page! </title>

    <link rel="stylesheet" type="text/css" href="my-styles.css">

    <style>
        h3 {
          color: blue;
        }
    </style>

    <script>
        function sayHi(sName) {
            alert("Hi, " + sName);
        }
    </script>

    <script type='text/javascript' src='my-script.js'></script>

    <meta name="author" content="Vasil Vasilev">
</head>
```

Frontend
HTML

- Headings

Elements → <h*n*>, where 1 ≤ *n* ≤ 6. The smaller *n*, the bigger the heading

```
<body>
    <h1> Largest heading </h1>
    <h6> Tiniest heading </h6>
</body>
```

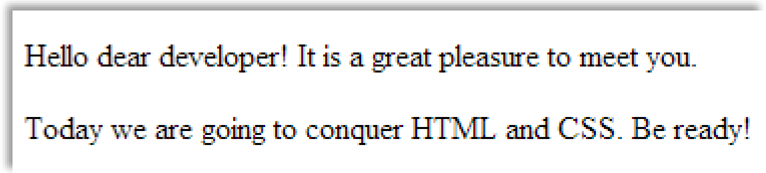

Largest heading

Tiniest heading

- Paragraphs

Element → <p>

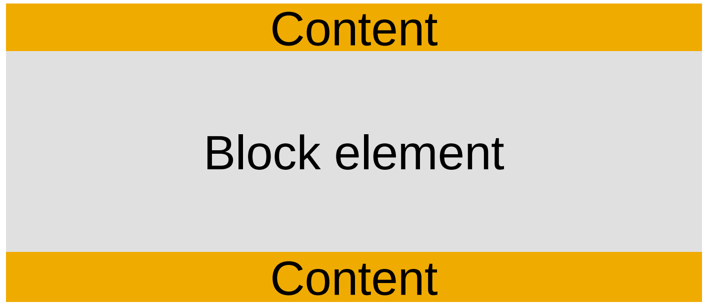
```
<body>
    <p>
    Hello dear developer! It is a great pleasure to meet you.
    </p>
    <p>
    Today we are going to conquer HTML and CSS. Be ready!
    </p>
</body>
```


Hello dear developer! It is a great pleasure to meet you.

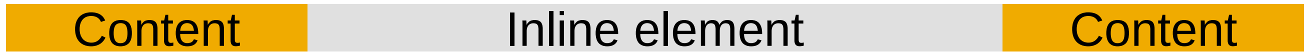
Today we are going to conquer HTML and CSS. Be ready!

Frontend
HTML

- Block elements – occupy a whole row (or several rows) on the screen. Start on a new row, and the element after a block element starts on a new row as well:



- Inline elements – when such elements are rendered, they are displayed on the very same row on the screen as the elements before and after that one.

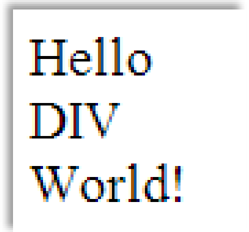


Frontend

HTML

- <div> - block element

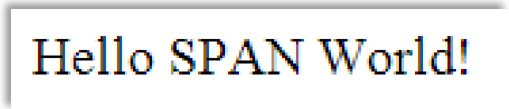
```
<body>
  <div>Hello</div>   <div>DIV</div>   <div>World!</div>
</body>
```



Hello
DIV
World!

- <span> - inline element

```
<body>
  <span>Hello</span>   <span>SPAN</span>   <span>World!</span>
</body>
```



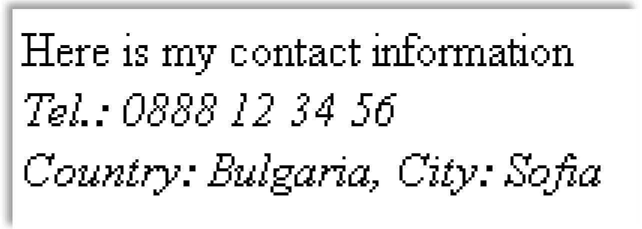
Hello SPAN World!

Frontend
HTML

- <address> is an example of a container with special semantics – it is used for packing contact information together.

```
<body>
  Here is my contact information
  <address>
  Tel.: 0888 12 34 56
  <br/>
  Country: Bulgaria,
  City: Sofia
  </address>
</body>
```

$\longrightarrow$


Here is my contact information
Tel.: 0888 12 34 56
Country: Bulgaria, City: Sofia

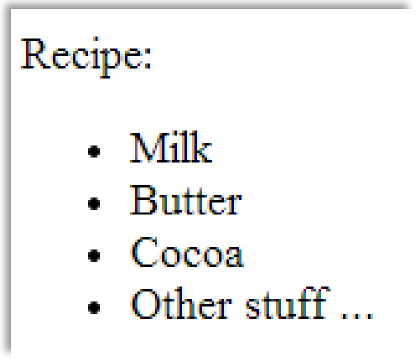
Frontend
HTML

- Unordered list - <ul>

(3 types of bullet shapes – circle, square, disc (default))

```
<body>
  <p>
    Recipe:
    <ul type="disc">
      <li>Milk</li>
      <li>Butter</li>
      <li>Cocoa</li>
      <li>Other stuff ...</li>
    </ul>
  </p>
</body>
```

$\longrightarrow$



Recipe:

- Milk
- Butter
- Cocoa
- Other stuff ...

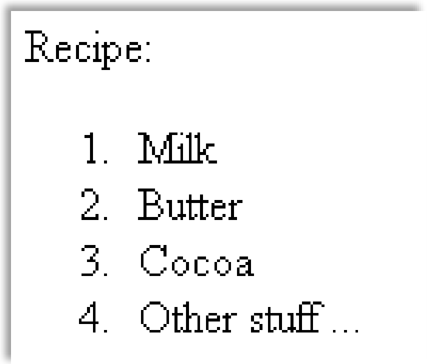
- Ordered list - <ol>

types –

“1” (default), “a”, “A”, “i”, “I

```
<body>
    <p>
        Recipe:
        <ol>
            <li>Milk</li>
            <li>Butter</li>
            <li>Cocoa</li>
            <li>Other stuff ...</li>
        </ol>
    </p>
</body>
```

$\longrightarrow$



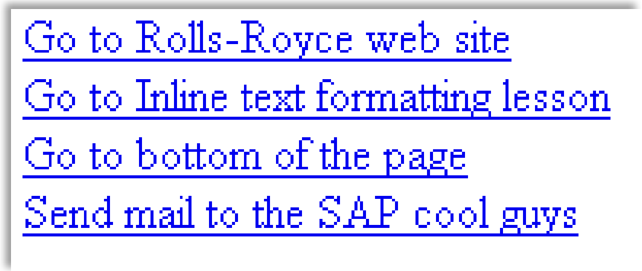
Recipe:

1. Milk
2. Butter
3. Cocoa
4. Other stuff ...

Frontend
HTML

- <a> A link points to a target. This target may be another web page, another place in the same page or an e-mail. Let's have a look at some examples:

```
<body>
   <a href="http://www.rolls-roycemotorcars.com/" target="_blank">Go to Rolls-Royce web site</a>
   <br/>
   <a href="05_inline_text_formatting.html" target="_self">Go to Inline text formatting lesson</a>
   <br/>
   <a href="#bottom" target="_self">Go to bottom of the page</a>
   <br/>
   <a href="mailto:cool.guys@sap.com?subject=HTML lesson">Send mail to the SAP cool guys</a>

   <br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
   <h4 id="bottom">This is the bottom of the page</h4>
</body>
```

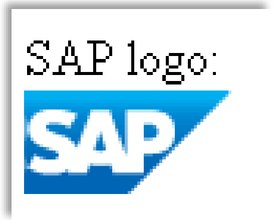

Go to Rolls-Royce web site
Go to Inline text formatting lesson
Go to bottom of the page
Send mail to the SAP cool guys

Frontend

HTML

- <img>

```
<body>
  SAP Logo: <br/>
  <img src="/sap.png" alt="SAP Logo" />
</body>
```



The alt attribute is highly recommended. It represents the text to be displayed if no image is found at the src path and it is used by screen readers.

Frontend
HTML

- <table>

```
<body>
    <table border="1">
        <caption> Table Structure Demo</caption>

        <tr>
            <th>Column #1 heading</th>
            <th>Column #2 heading</th>
        </tr>

        <tr>
            <td>Cell 1-1</td>
            <td>Cell 1-2</td>
        </tr>

        <tr>
            <td>Cell 2-1</td>
            <td>Cell 2-2</td>
        </tr>

    </table>
</body>
```

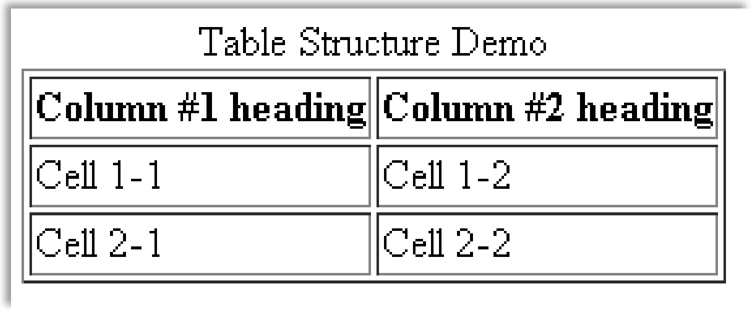


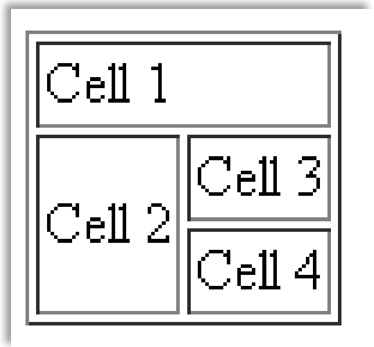
Table Structure Demo

Column #1 heading	Column #2 heading
Cell 1-1	Cell 1-2
Cell 2-1	Cell 2-2

Frontend
HTML

- <table>, rowspan, colspan. Both are attributes of the <td> tag.

```
<body>
<table border="1">

    <tr>
        <td colspan="2">Cell 1</td>
    </tr>

    <tr>
        <td rowspan="2">Cell 2</td>
        <td>Cell 3</td>
    </tr>

    <tr>
        <td>Cell 4</td>
    </tr>

</table>
</body>
```



Cell 1	
Cell 2	Cell 3
	Cell 4

Frontend
HTML

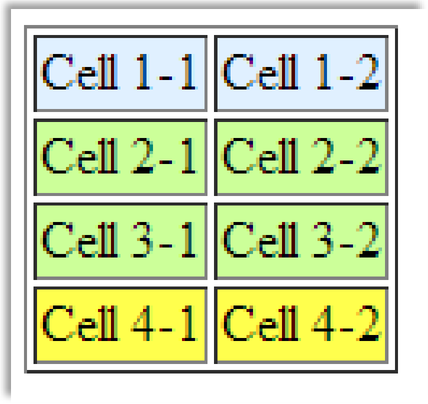
- <table>, vertical grouping

<thead>, <tbody>, <tfoot>,

```
<style>
  thead {
      background-color: #E0F0FF;
  }

  tbody {
      background-color: #CCFF99;
  }

  tfoot {
      background-color: #FFFF4D;
  }
</style>
```

This is CSS – more on that later on ☺

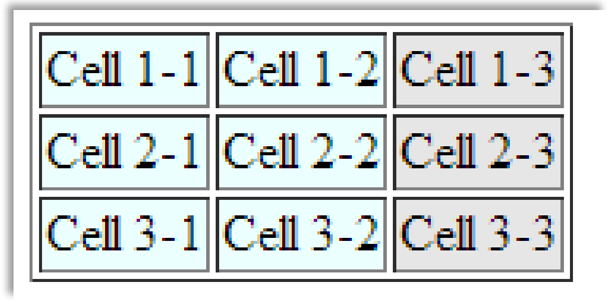
```
<body>
    <table border="1">
        <thead>
            <tr>
                <td>Cell 1-1</td>
                <td>Cell 1-2</td>
            </tr>
        </thead>

        <tbody>
            <tr>
                <td>Cell 2-1</td>
                <td>Cell 2-2</td>
            </tr>

            <tr>
                <td>Cell 3-1</td>
                <td>Cell 3-2</td>
            </tr>
        </tbody>

        <tfoot>
            <tr>
                <td>Cell 4-1</td>
                <td>Cell 4-2</td>
            </tr>
        </tfoot>
    </table>
</body>
```



Cell 1-1	Cell 1-2
Cell 2-1	Cell 2-2
Cell 3-1	Cell 3-2
Cell 4-1	Cell 4-2

Frontend
HTML

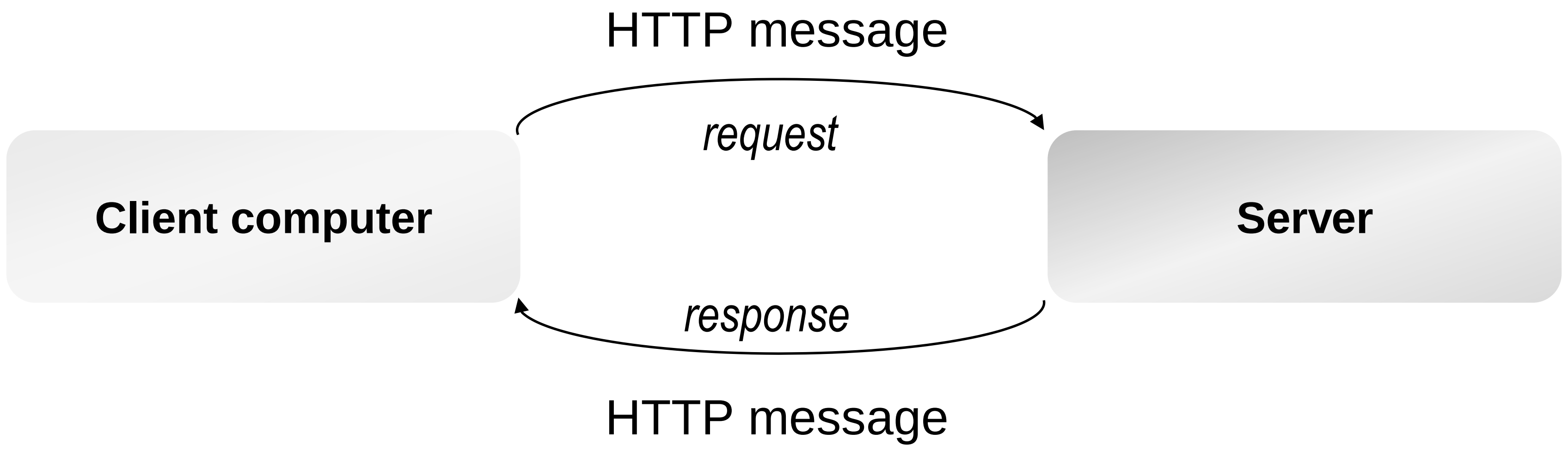
- <table>, vertical grouping <colgroup>



Cell 1-1	Cell 1-2	Cell 1-3
Cell 2-1	Cell 2-2	Cell 2-3
Cell 3-1	Cell 3-2	Cell 3-3

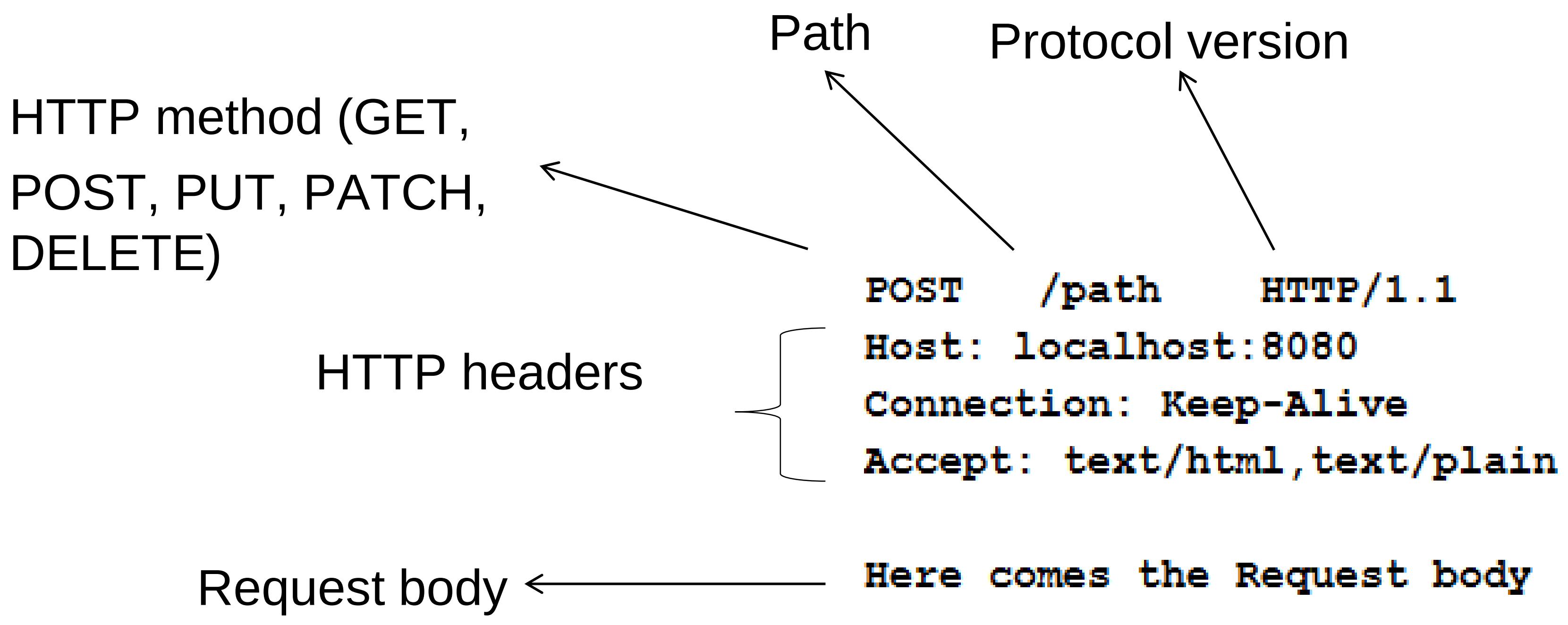
```
<body>
    <table border="1">
        <colgroup span="2" style="background-color: #ECFFFF;">
        <colgroup span="1" style="background-color: #E6E6E6;">

        <tr>
            <td>Cell 1-1</td>
            <td>Cell 1-2</td>
            <td>Cell 1-3</td>
        </tr>

        <tr>
            <td>Cell 2-1</td>
            <td>Cell 2-2</td>
            <td>Cell 2-3</td>
        </tr>

        <tr>
            <td>Cell 3-1</td>
            <td>Cell 3-2</td>
            <td>Cell 3-3</td>
        </tr>

    </table>
</body>
```

HTTP


HTTP message
request
Client computer
Server
response
HTTP message

HTTP

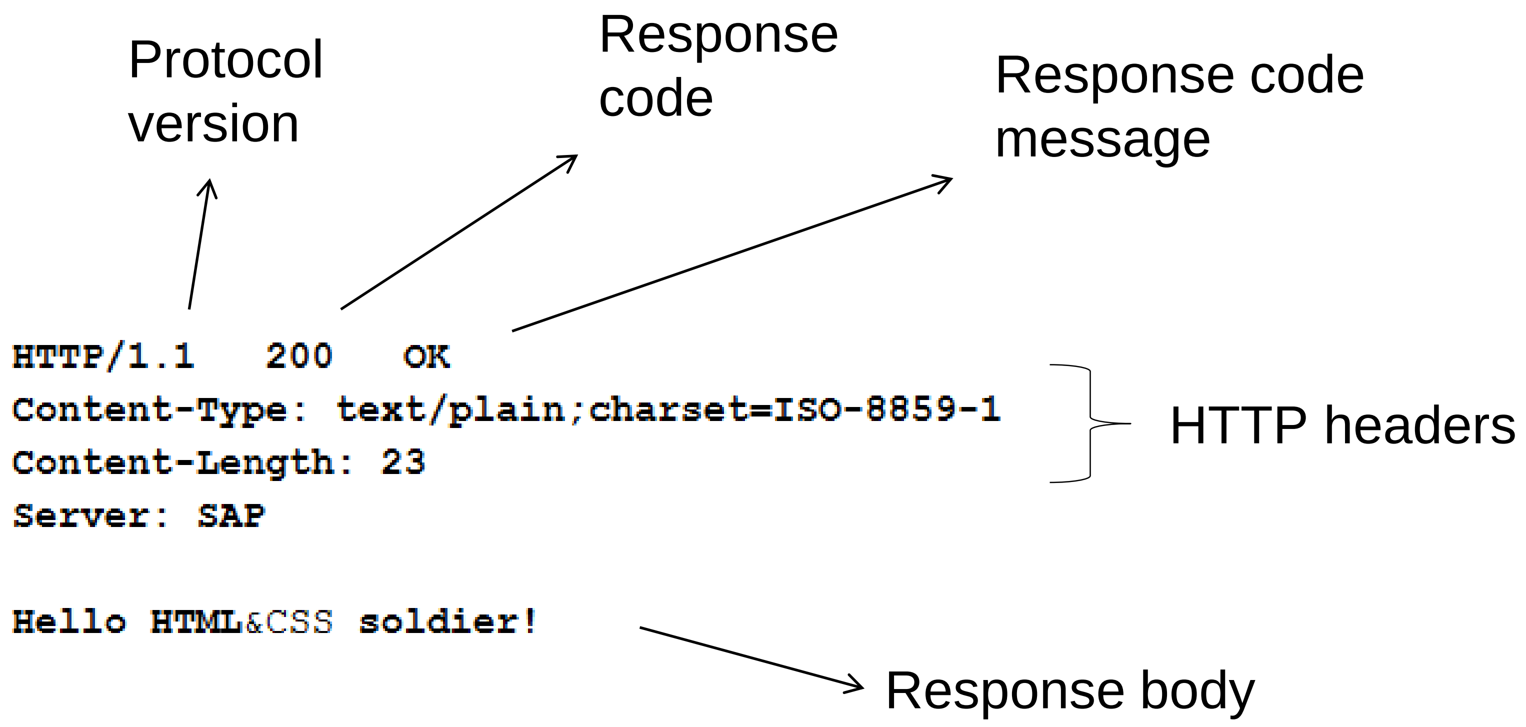
HTTP request format



```
POST   /path    HTTP/1.1
Host: localhost:8080
Connection: Keep-Alive
Accept: text/html,text/plain

Here comes the Request body
```

HTTP

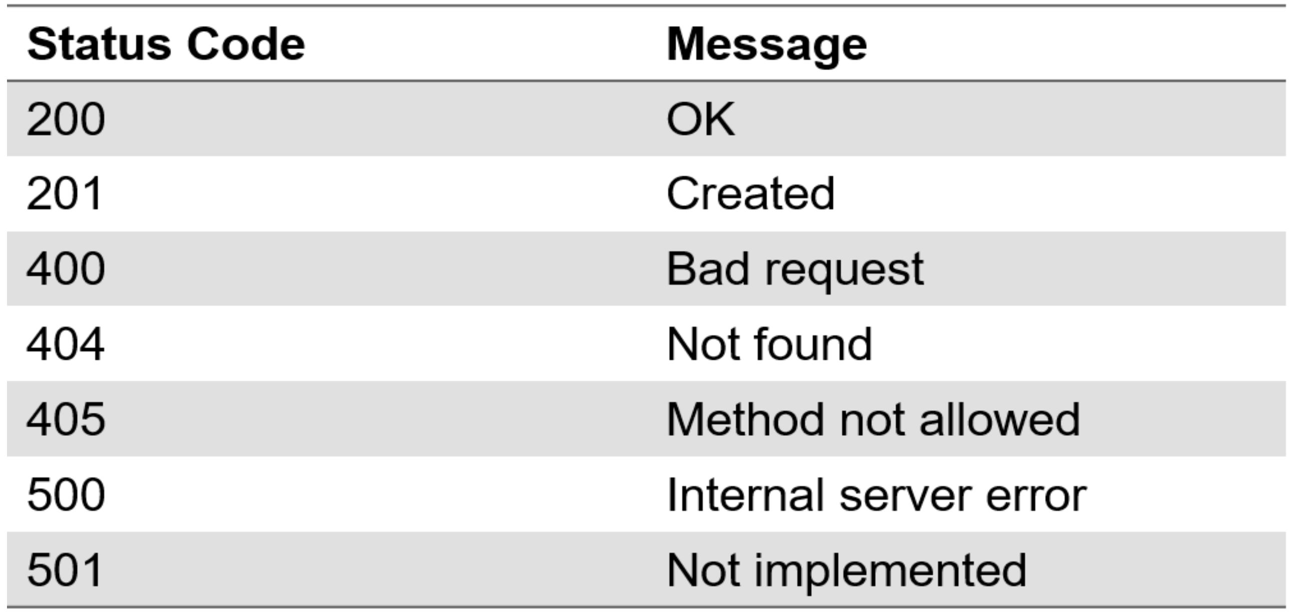
HTTP response format



```
HTTP/1.1   200   OK
Content-Type: text/plain;charset=ISO-8859-1
Content-Length: 23
Server: SAP

Hello HTML&CSS soldier!
```

HTTP

Common response codes



Status Code	Message
200	OK
201	Created
400	Bad request
404	Not found
405	Method not allowed
500	Internal server error
501	Not implemented

Frontend

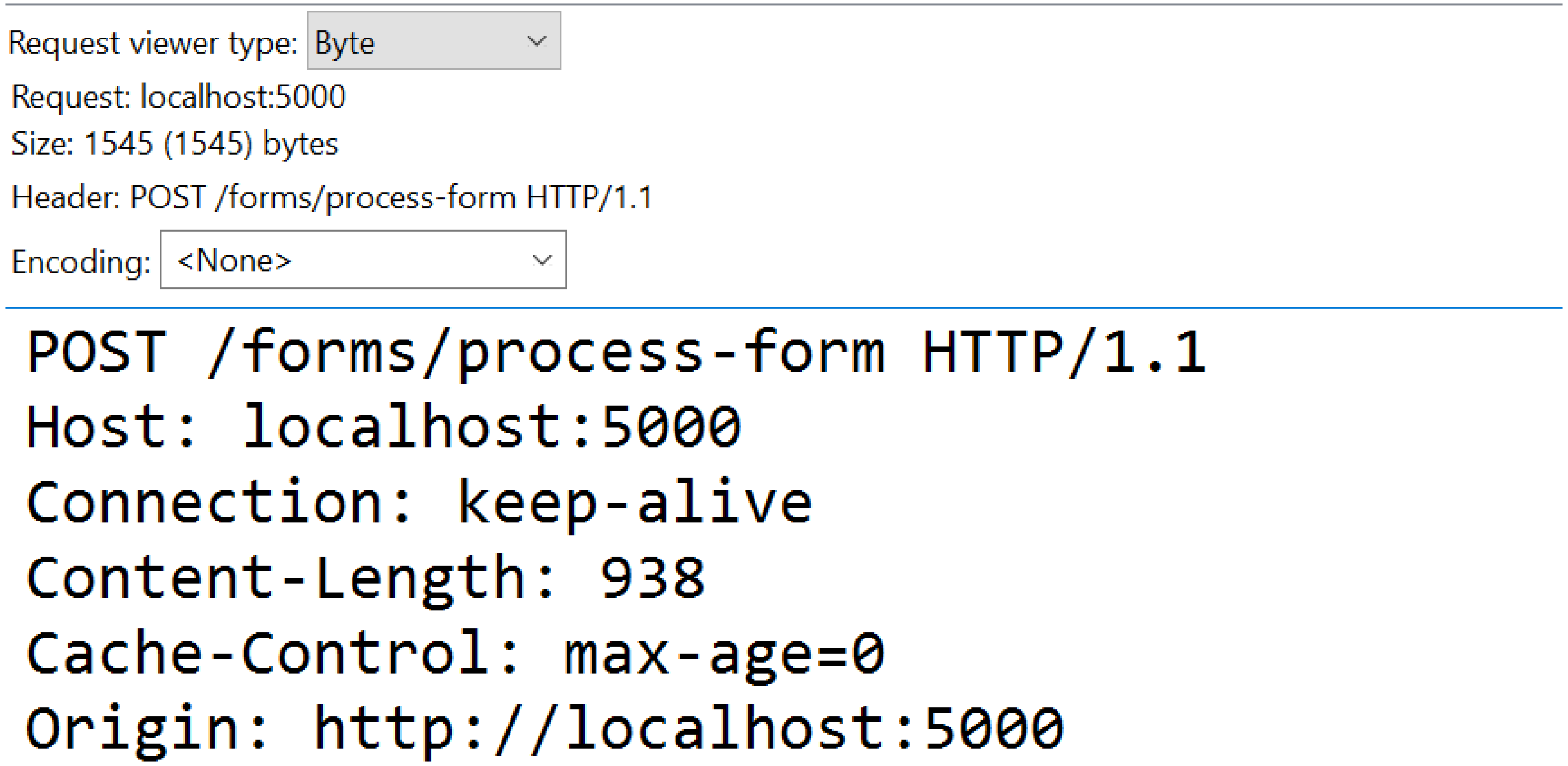
HTML forms

```
<body>
  <form method="post" action="/forms/process-form" enctype="multipart/form-data">


  </form>
</body>
```

Frontend
HTML forms

```
<input type="text" name="name" size="30" />

<input type="password" name="password" size="30" />

<textarea name="address" cols="30" rows="4">
</textarea>

<select name="select-box">
  <option value="opt-1"> Option 1 </option>
  <option value="opt-2"> Option 2 </option>
</select>

<input type="radio" name="choose-1" value="opt-1" /> Option 1
<input type="radio" name="choose-1" value="opt-2" /> Option 2
<input type="radio" name="choose-1" value="opt-3" /> Option 3

<input type="checkbox" name="choose-as-many-as-you-want" value="opt-1" /> Option 1
<input type="checkbox" name="choose-as-many-as-you-want" value="opt-2" /> Option 2
<input type="checkbox" name="choose-as-many-as-you-want" value="opt-3" /> Option 3

<input type="file" name="upload" />

<input type="submit" value="Submit" />
```

Frontend
HTML forms, Eclipse’s TCP/IP Monitor



Request: localhost:5000
Size: 1545 (1545) bytes
Header: POST /forms/process-form HTTP/1.1

```
POST /forms/process-form HTTP/1.1
Host: localhost:5000
Connection: keep-alive
Content-Length: 938
Cache-Control: max-age=0
Origin: http://localhost:5000
```

Frontend

CSS

which-element {

what-property: *to-look-how*

}

Frontend
CSS

```css
selector {
  property: value;
  property: value;
  property: value;
}
```

```css
#my-div {
    color: #FF0000;
}
```

```html
<div id="my-div">
  Lorem ipsum
</div>
```

```css
.nice-div {
    color: #FF0000;
}
```

```html
<div class="nice-div">
  Lorem ipsum
</div>
```

Lorem ipsum
Lorem ipsum

Frontend

CSS

```
<head>
  <style>
    body {
      color: red;
    }

    div {
      color: blue;
    }
  </style>
</head>
<body>
  Text - 1
  <br/>
  <span> Text - 2 </span>
  <div> Text - 3 </div>
</body>
```

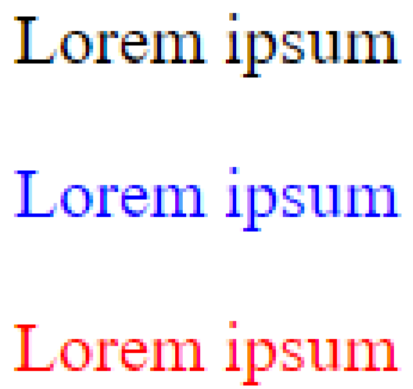
```
Text - 1
Text - 2
Text - 3
```

Frontend

CSS

```
p {
    color: #0000FF;
}

div p {
    color: #FF0000;
}
```

```
<div>
    Lorem ipsum
</div>

<p>
    Lorem ipsum
</p>

<div>
    <p>
        Lorem ipsum
    </p>
</div>
```

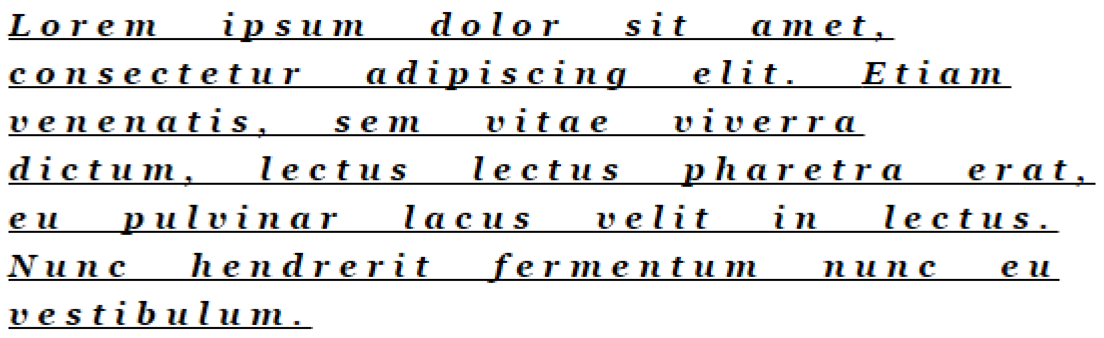


Lorem ipsum

Lorem ipsum

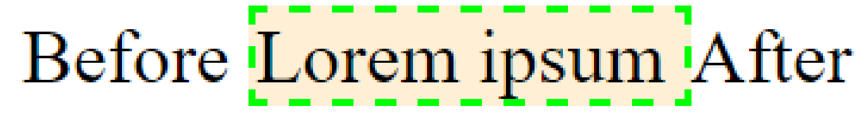
Lorem ipsum

Frontend
CSS

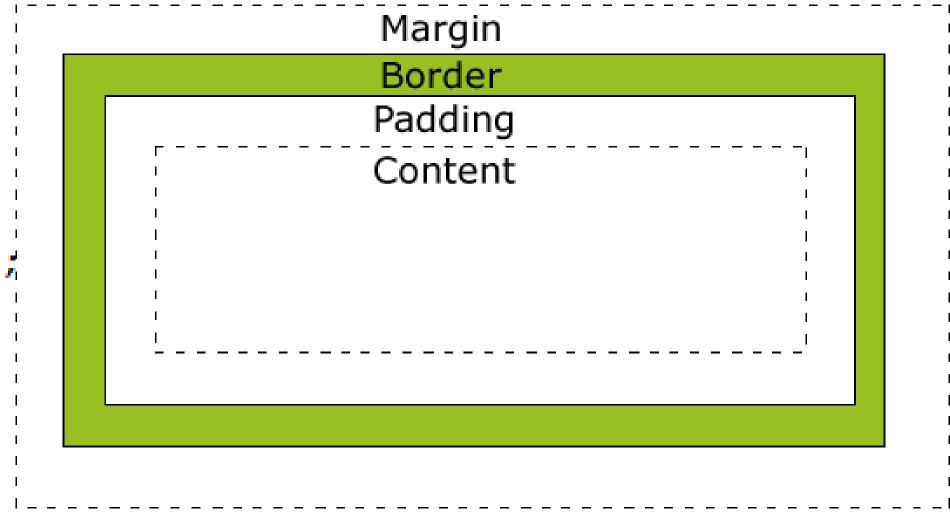

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam venenatis, sem vitae viverra dictum, lectus lectus pharetra erat, eu pulvinar lacus velit in lectus. Nunc hendrerit fermentum nunc eu vestibulum.

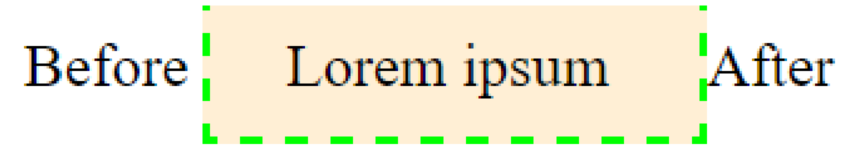
```
font-family: Georgia;
font-weight: bold;
font-style: italic;
text-decoration: underline;
letter-spacing: 0.3em;
word-spacing: 1.2em;
line-height: 1.5;
text-align: justified;
```

Frontend
CSS

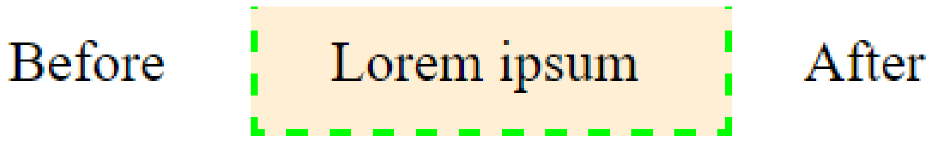


```
border-style: dashed;
border-color: #00FF00;
border-size: 1px;
background-color: #FFEFD5;
```





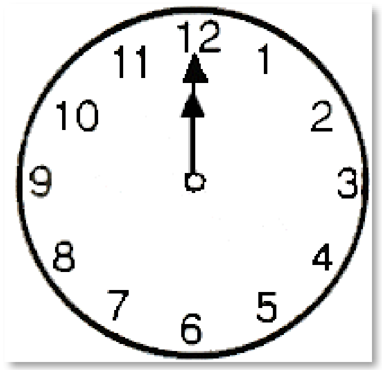
```
border-style: dashed;
border-color: #00FF00;
border-size: 1px;
background-color: #FFEFD5;
padding: 10px 20px 10px 20px;
```



```
border-style: dashed;
border-color: #00FF00;
border-size: 1px;
background-color: #FFEFD5;
padding: 10px 20px 10px 20px;
margin: 10px 20px 10px 20px;
```

Frontend
CSS

```
border-top-style: solid;
border-right-style: dotted;
border-bottom-style: dashed;
border-left-style: inset;
```



```
border-style: solid dotted dashed inset;
```

Frontend
CSS

```
p {
    font-style: italic;
}
```

```
<p>
  Lorem ipsum
</p>
<p>
  Lorem ipsum
</p>
<p>
  Lorem ipsum
</p>
```

Lorem ipsum

Lorem ipsum

Lorem ipsum

```
p, span {
    color: rgb(65%, 80%, 70%);
}
```

```
<p>
  Lorem ipsum
</p>
<span>
  Lorem ipsum
</span>
<p>
  Lorem ipsum
</p>
```

Lorem ipsum

Lorem ipsum

Lorem ipsum

Frontend
CSS

```css
div p {
    color: #FF0000;
}
```

```html
<p>
  Lorem ipsum
</p>

<div>
  <p>
    Lorem ipsum
  </p>
</div>

<div>
  <span>
    <p>
      Lorem ipsum
    </p>
  </span>
</div>
```

```css
.that-red p {
    color: #FF0000;
}
```

```html
<p class="that-red">
  Lorem ipsum
</p>

<div class="that-red">
  <p>
    Lorem ipsum
  </p>
</div>

<div class="that-red">
  <span>
    <p>
      Lorem ipsum
    </p>
  </span>
</div>
```

Lorem ipsum

Lorem ipsum

Lorem ipsum

Frontend
CSS

```
div > p {
    color: #FF0000;
}
```

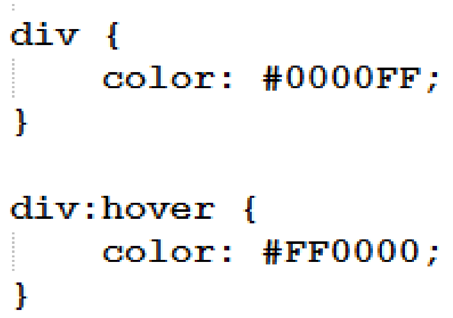
```
.parent > p {
    color: #FF0000;
}
```

```
<div>
    Lorem ipsum
</div>

<p>
    Lorem ipsum
</p>

<div>
  <p>
    Lorem ipsum
  </p>
  <p>
    Lorem ipsum
  </p>
</div>

<div>
  <span>
    <p>
      Lorem ipsum
    </p>
  </span>
</div>
```

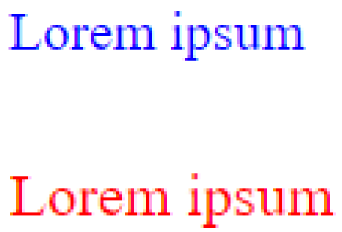
```
<div class="parent">
    Lorem ipsum
</div>

<p class="parent">
    Lorem ipsum
</p>

<div class="parent">
  <p>
    Lorem ipsum
  </p>
  <p>
    Lorem ipsum
  </p>
</div>

<div class="parent">
  <span>
    <p>
      Lorem ipsum
    </p>
  </span>
</div>
```

```
Lorem ipsum

Lorem ipsum

Lorem ipsum

Lorem ipsum

Lorem ipsum
```

Frontend
CSS

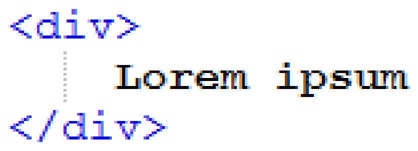


```
div {
    color: #0000FF;
}

div:hover {
    color: #FF0000;
}
```



Lorem ipsum

Lorem ipsum



```
<div>
    Lorem ipsum
</div>
```

Frontend
CSS

```
<p style="color: #0000FF">
    Lorem ipsum
</p>
```

Inline

```
p {
  color: #00FF00;
}
```

my-styles.css

External

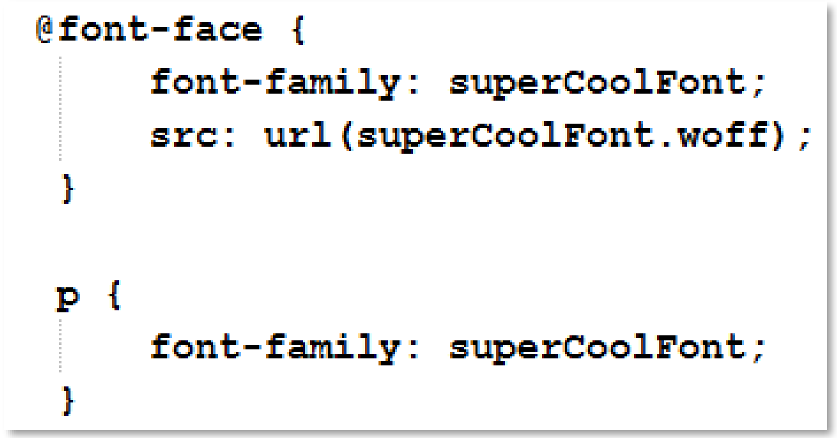
```
<html>
  <head>
    <style>
        p {
          color: #FF0000;
        }
    </style>
  </head>

  <body>
    <p>
      Lorem ipsum
    </p>
  </body>
</html>
```

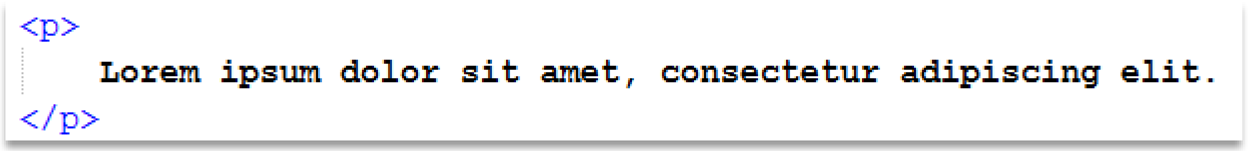
Embedded

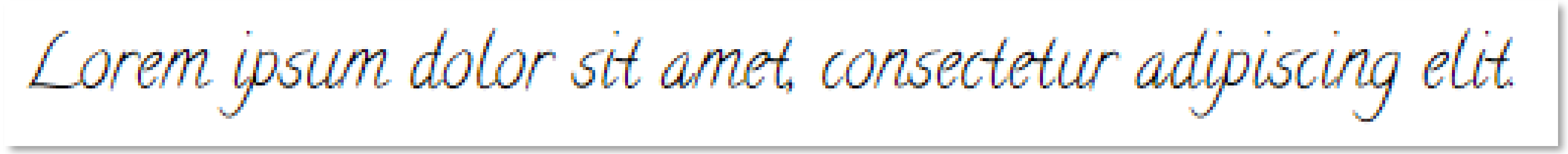
```
<html>
  <head>
    <title>CSS Example</title>
    <link rel="stylesheet" href="my-styles.css">
  </head>
...
```

my-page.html

Frontend
CSS


```
@font-face {
    font-family: superCoolFont;
    src: url(superCoolFont.woff);
}

p {
    font-family: superCoolFont;
}
```


```
<p>
    Lorem ipsum dolor sit amet, consectetur adipiscing elit.
</p>
```


Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Frontend

CSS

Prefix	Browsers
-webkit	Chrome, Safari
-moz	Firefox
-o	Opera
-ms	Internet Explorer

Keep in mind: not all browsers support everything. Before you start implementing something nicely looking, check which browsers support the required features and whether they need a prefix.

Frontend

CSS

```
<style>
  div {
    width: 150px;
    background-color: aqua;
    transition: width 2s;
  }

  div:hover {
    width: 300px;
  }
</style>
```

```
<div>
  Content
</div>
```

```
@keyframes cool-animation {
    0%   {background-color: aqua;}
    10%  {background-color: red;}
    80%  {background-color: green;}
    100% {background-color: blue;}
}
```

```
<style>
  .animated {
    width: 150px;
    background-color: aqua;
  }

  .animated:hover {
    width: 300px;
    background-color: blue;
    animation: cool-animation 5s;
  }
</style>
```

```
<div class="animated">
  Content
</div>
```

Frontend
Bootstrap

- Very popular front-end framework.
- Based on HTML, CSS and JavaScript.
- Mobile first framework – the aim is to write the UI of your application once and to look good on small screens as well as it does on big monitors.
- Relatively easy to use – developing with Bootstrap mainly requires using special CSS classes. HTML elements have default styling as well.
- You can recognize a Bootstrap application – the look and feel of all of them is much similar (which is good for the users).
- There are themes for Bootstrap (here and here) with which you can customize your app.

Frontend
Bootstrap

https://getbootstrap.com/docs/4.1/layout/grid/#grid-options

Frontend
JavaScript

- Dynamically typed
- Basic types:
  undefined, string, number, boolean, array, object, function

- Familiar syntax

```
var name; // undefined
name = "Martin"; // string
name = 5; // number
name = true; // boolean
name = ["one", "two", 3]; // object array
name = {}; // object
name = function() {return "something";}; // function
```

```
for (var i=0; i<5; i++) {
    // do something
}
```

```
function max(a, b) {
    return (a > b) ? a:b;
}
```

```
switch(myInt) {

case 1: // ...
    break;

case 2: // ...
    break;

default:
    // ...
}
```

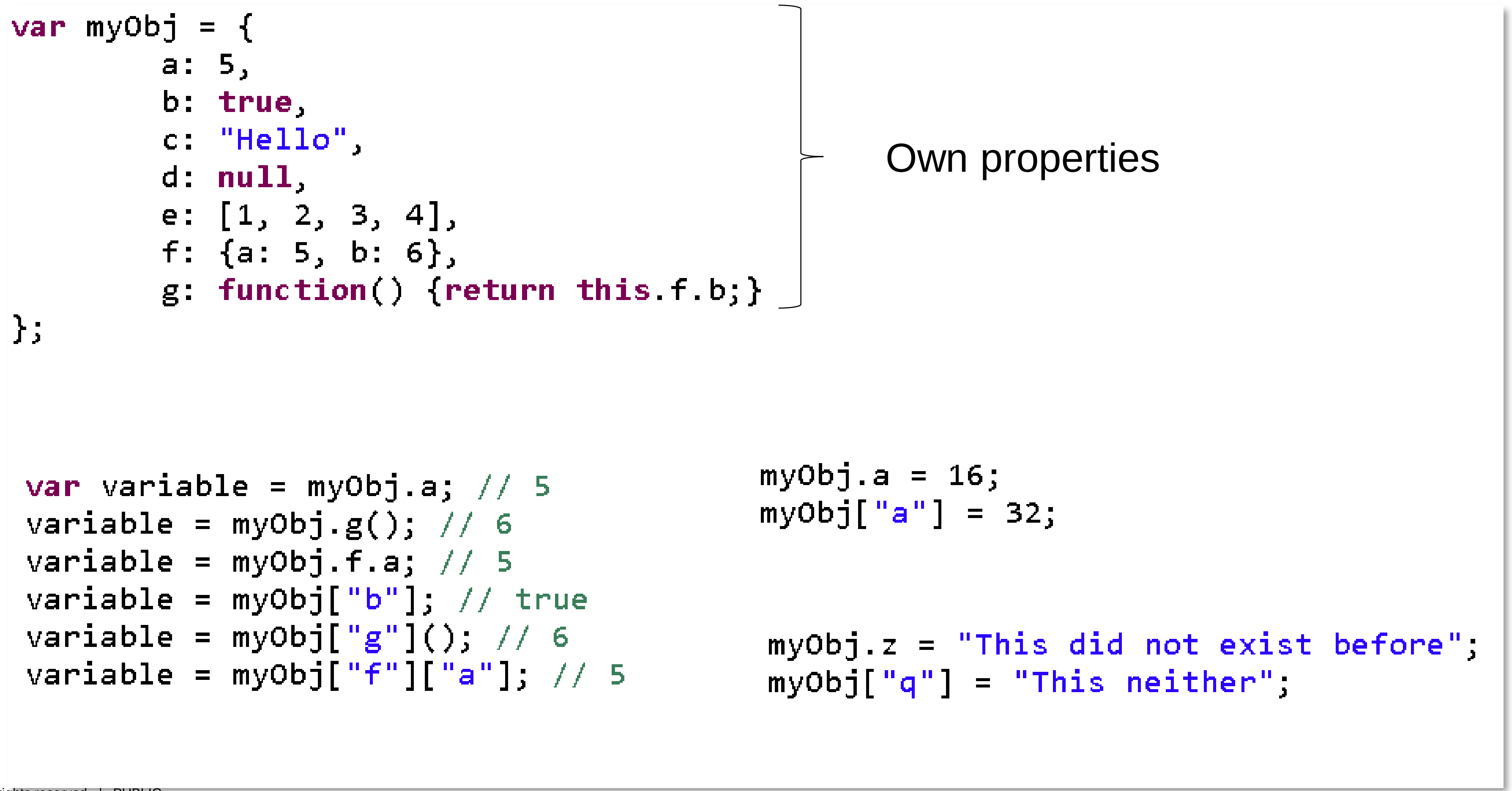
```
var i = 0;
while(i != 5) {
    // do something
    i++;
}
```

```
var i = 0;
do
{
    // do something
    i++;
} while(i != 5);
```

Frontend

JavaScript

- Objects consist of properties (key-value pairs)



```
var myObj = {
        a: 5,
        b: true,
        c: "Hello",
        d: null,
        e: [1, 2, 3, 4],
        f: {a: 5, b: 6},
        g: function() {return this.f.b;}
};
```

Own properties

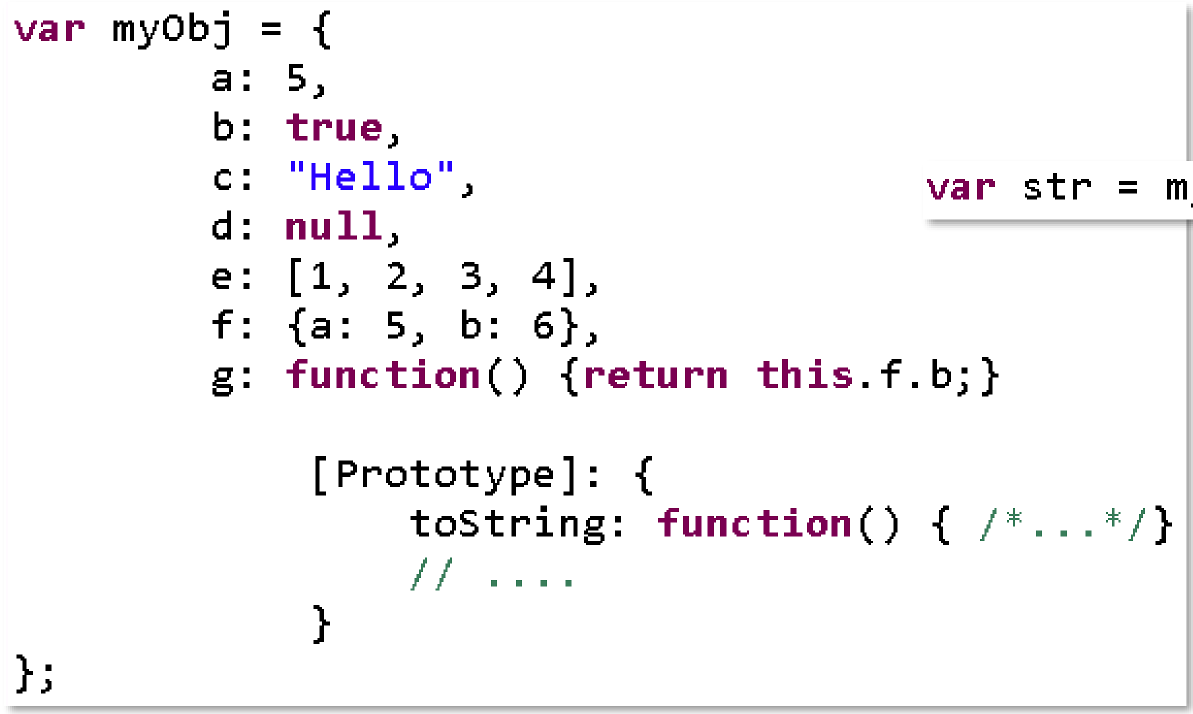
```
var variable = myObj.a; // 5
variable = myObj.g(); // 6
variable = myObj.f.a; // 5
variable = myObj["b"]; // true
variable = myObj["g"](); // 6
variable = myObj["f"]["a"]; // 5
```

```
myObj.a = 16;
myObj["a"] = 32;
```

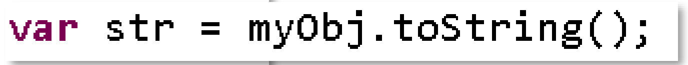
```
myObj.z = "This did not exist before";
myObj["q"] = "This neither";
```

Frontend
JavaScript

- Beside the own properties, there are also other pairs that can be accessed. They are located in another object that is linked by the current one. This other object is called a prototype.



```
var myObj = {
        a: 5,
        b: true,
        c: "Hello",
        d: null,
        e: [1, 2, 3, 4],
        f: {a: 5, b: 6},
        g: function() {return this.f.b;}

            [Prototype]: {
                toString: function() { /*...*/}
                // ....
            }
};
```



```
var str = myObj.toString();
```

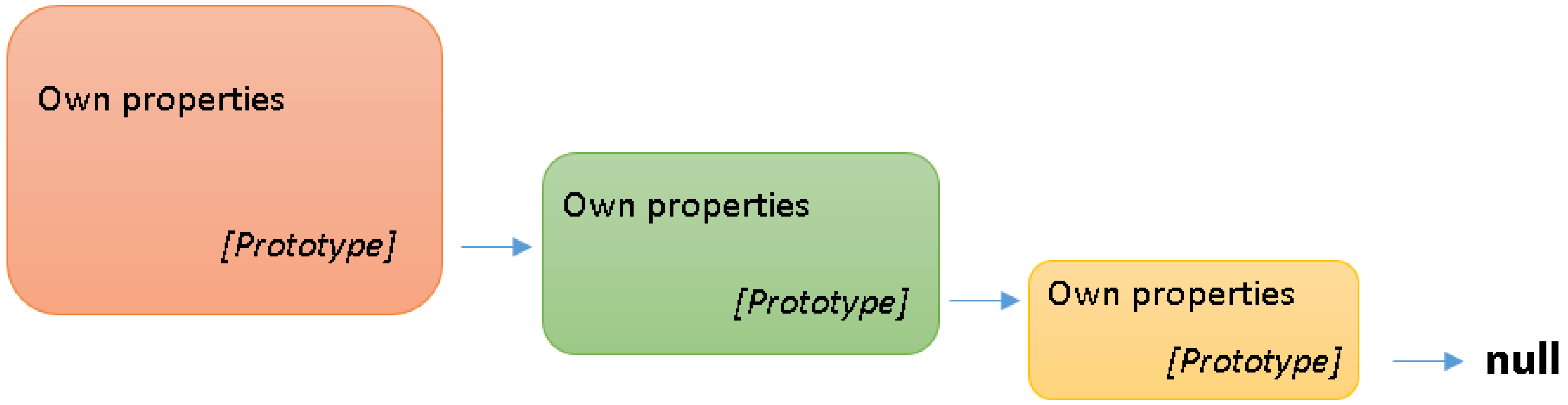
Frontend

JavaScript

- Every single JavaScript object links to a prototype
- The prototype is a JavaScript object as well

Thus a prototype also links to a prototype.

This is called a *prototype chain*.



Frontend
React

- An application consists of Components

```
class MyComponent extends React.Component {
  constructor() {
    this.state = {}
  }
  render() {
    // Returns a React.Element
    // which maps to a DOM element.
    // React.Elements are part of a Virual DOM
  }
}
```

Use this.setState to update the state and refresh the view.

Frontend
React

- Creating React.Elements:
  - Via JS:

```
var element = React.createElement(‘h1’, null, ‘Hello World’);
```

  - Via JSX:

```
var element = <h1>Hello World</h1>
```

This is achieved via transpiling (Babel)

Backend
REST

- REST is an architectural style
- Fits well into HTTP
- Everything is a resource
- Every resource has got an address (URL path)
- Every resource has a representation(s) – JSON, XML, text
- The action over a resource is determined by the HTTP method being used
- The result of an operation (whether successful or not) is determined by the response code

Backend
REST

```
<dependency>
    <groupId>com.sun.jersey</groupId>
    <artifactId>jersey-servlet</artifactId>
    <version>1.19.2</version>
</dependency>
```

```
@ApplicationPath("/")
@Path("/rest")
public class RestServices extends Application {

    @GET
    @Path("/hello")
    @Produces(MediaType.TEXT_PLAIN)
    public String helloRest() {
        return "Hello REST";
    }

}
```