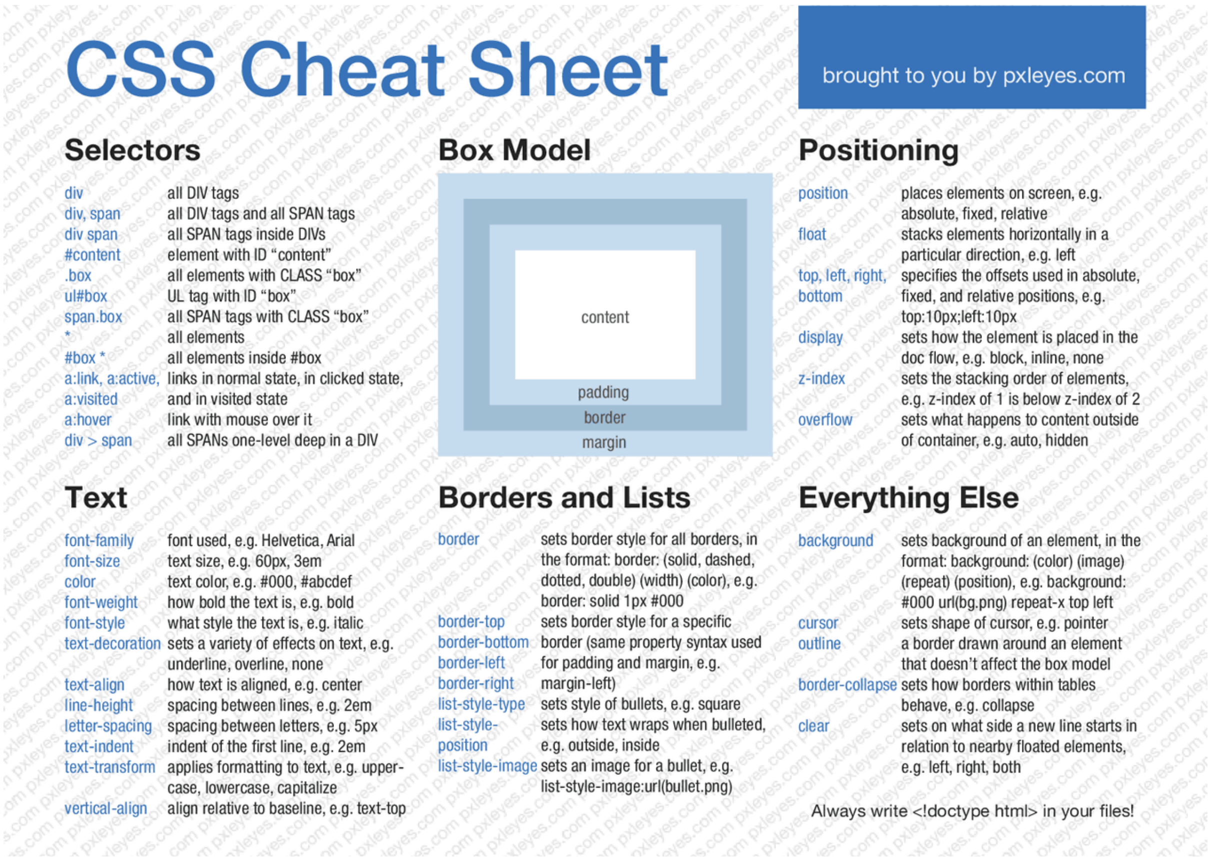


CSS Cheat Sheet

brought to you by pxleyes.com

Selectors

div	all DIV tags
div, span	all DIV tags and all SPAN tags
div span	all SPAN tags inside DIVs
#content	element with ID "content"
.box	all elements with CLASS "box"
ul#box	UL tag with ID "box"
span.box	all SPAN tags with CLASS "box"
*	all elements
#box *	all elements inside #box
a:link, a:active, a:visited	links in normal state, in clicked state, and in visited state
a:hover	link with mouse over it
div > span	all SPANs one-level deep in a DIV

Box Model

Positioning

position	places elements on screen, e.g. absolute, fixed, relative
float	stacks elements horizontally in a particular direction, e.g. left
top, left, right, bottom	specifies the offsets used in absolute, fixed, and relative positions, e.g. top:10px;left:10px
display	sets how the element is placed in the doc flow, e.g. block, inline, none
z-index	sets the stacking order of elements, e.g. z-index of 1 is below z-index of 2
overflow	sets what happens to content outside of container, e.g. auto, hidden

Text

font-family	font used, e.g. Helvetica, Arial
font-size	text size, e.g. 60px, 3em
color	text color, e.g. #000, #abcdef
font-weight	how bold the text is, e.g. bold
font-style	what style the text is, e.g. italic
text-decoration	sets a variety of effects on text, e.g. underline, overline, none
text-align	how text is aligned, e.g. center
line-height	spacing between lines, e.g. 2em
letter-spacing	spacing between letters, e.g. 5px
text-indent	indent of the first line, e.g. 2em
text-transform	applies formatting to text, e.g. uppercase, lowercase, capitalize
vertical-align	align relative to baseline, e.g. text-top

Borders and Lists

border	sets border style for all borders, in the format: border: (solid, dashed, dotted, double) (width) (color), e.g. border: solid 1px #000
border-top border-bottom border-left border-right	sets border style for a specific border (same property syntax used for padding and margin, e.g. margin-left)
list-style-type	sets style of bullets, e.g. square
list-style-position	sets how text wraps when bulleted, e.g. outside, inside
list-style-image	sets an image for a bullet, e.g. list-style-image:url(bullet.png)

Everything Else

background	sets background of an element, in the format: background: (color) (image) (repeat) (position), e.g. background: #000 url(bg.png) repeat-x top left
cursor	sets shape of cursor, e.g. pointer
outline	a border drawn around an element that doesn't affect the box model
border-collapse	sets how borders within tables behave, e.g. collapse
clear	sets on what side a new line starts in relation to nearby floated elements, e.g. left, right, both

Always write <!doctype html> in your files!

Properties of the XMLHTTPRequest

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

Pembahasan UTS

No. 1

```
function validasi_form() {
  let nama = document.getElementById("nama").value;
  let email = document.getElementById("email").value;
  let bidang = document.getElementsByName("bidang");
  let sub_bidang = document.getElementById("sub_bidang").value;
  let password = document.getElementById("password").value;
  let konf_password = document.getElementById("konf_password").value;
  let jadwal = document.getElementsByName("jadwal[]");

  // Untuk menghitung total jadwal yang dipilih
  count_jadwal = 0;
  for (let i = 0; i < jadwal.length; i++) {
    if (jadwal[i].checked) {
      count_jadwal++;
    }
  }

  // Untuk mengecek apakah bidang sudah dipilih
  let bidang_checked = false;
  for (let i = 0; i < bidang.length; i++) {
    if (bidang[i].checked) {
      bidang_checked = true;
    }
  }

  // Pengecekan apakah semua input sudah diisi
  if (
    nama != "" &&
    email != "" &&
    bidang_checked &&
    sub_bidang != "" &&
    password != "" &&
    konf_password != "" &&
    count_jadwal > 0
  ) {
    if (count_jadwal != 2) {
      alert("Jadwal harus 2");
      return false;
    } else {
      // Pengecekan apakah password dan konfirmasi password sama
      if (password == konf_password) {
        return true;
      } else {
        alert("Password dan konfirmasi password tidak sama");
        return false;
      }
    }
  } else {
    alert("Anda harus mengisi data dengan lengkap !");
    return false;
  }
}

function get_sub_bidang() {
  let bidang = document.getElementsByName("bidang");
  let sub_bidang = document.getElementById("sub_bidang");

  for (let i = 0; i < bidang.length; i++) {
    if (bidang[i].checked) {
      // Untuk mendapatkan nilai bidang yang dipilih
      let bidang_value = bidang[i].value;
      sub_bidang.innerHTML = "";

      // Untuk mengecek nilai bidang yang dipilih
      if (bidang_value == "design") {
        sub_bidang.innerHTML += `
          <option value="html">HTML</option>
          <option value="css">CSS</option>
            <option value="js">Javascript</option>
        `;
      } else {
        sub_bidang.innerHTML += `
          <option value="php">PHP</option>
          <option value="ajax">AJAX</option>
          <option value="webservice">Web Service</option>
        `;
      }
    }
  }
}
```

No. 2

db.php

```
<?php

$db_host = 'localhost';
$db_database = 'utspbp';
$db_username = 'root';
$db_password = '';

$db = new mysqli($db_host, $db_username, $db_password, $db_database);

if ($db->connect_errno) {
    die("Could not connect to the database: <br/>" . $db->connect_error);
}
```

get_detail.php

```
<?php

require_once 'db.php';

$kategori = $_GET['kategori'];
if ($kategori != '') {
    $query = "SELECT * FROM produk WHERE idkategori = $kategori";
    $result = mysqli_query($db, $query);


    // Create a table to display the results
    echo "<table border='1' cellpadding='10'>";
    echo "<tr>
            <th>Subkategori</th>
            <th>ID</th>
            <th>Nama</th>
            <th>Harga</th>
        </tr>";
    echo "<tr>";
    while ($row = mysqli_fetch_array($result)) {
        $sub_kategori = mysqli_query($db, "SELECT nama FROM sub_kategori
WHERE idsub_kategori = " . $row['idsub_kategori']);
        $sub_kategori = mysqli_fetch_array($sub_kategori);
        echo "<td>" . $sub_kategori['nama'] . "</td>";
        echo "<td>" . $row['nama'] . "</td>";
        echo "<td>" . $row['harga'] . "</td>";
        echo "<td>" . $row['idkategori'] . "</td>";
        echo "</tr>";
    }
    echo "</table>";
}
```

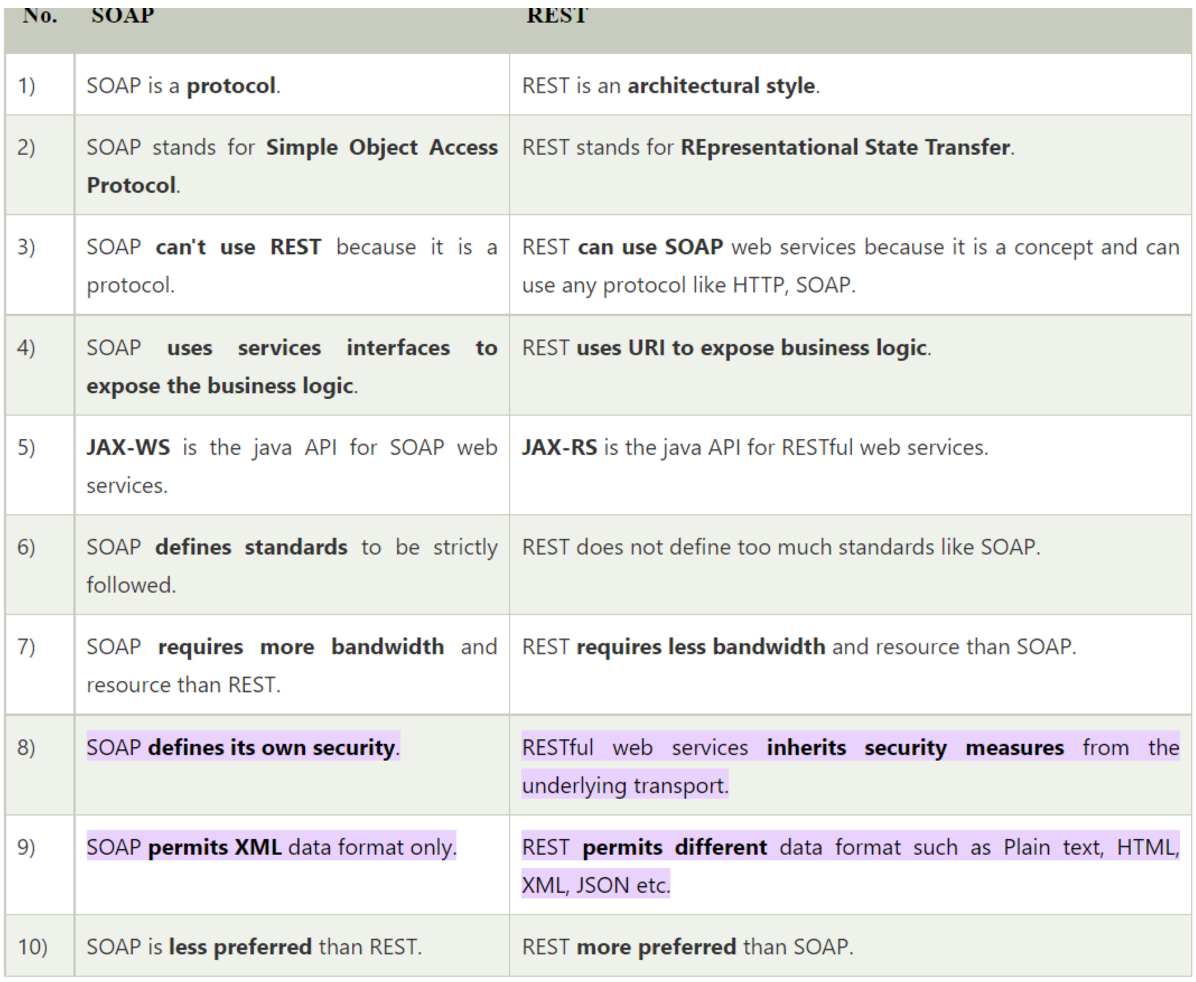
get_kategori.php

```
<?php
require_once 'db.php';
?>

<!DOCTYPE HTML>
<html>

<head>
   <script defer src="js/ajax.js" type="text/javascript"></script>
</head>

<body>
   Kategori :
   <select id="kategori">
      <option value="">Pilih Kategori</option>
      <?php
      $sql = "SELECT * FROM kategori";
      $result = mysqli_query($db, $sql);
      while ($row = mysqli_fetch_array($result)) {
         echo "<option value='" . $row['idkategori'] . "'>" . $row['nama'] . "</option>";
      }
      ?>
   </select>
   <button onclick="get_detail()">Lihat</button>
   <div id="detail"></div>
</body>

</html>
```

ajax.js

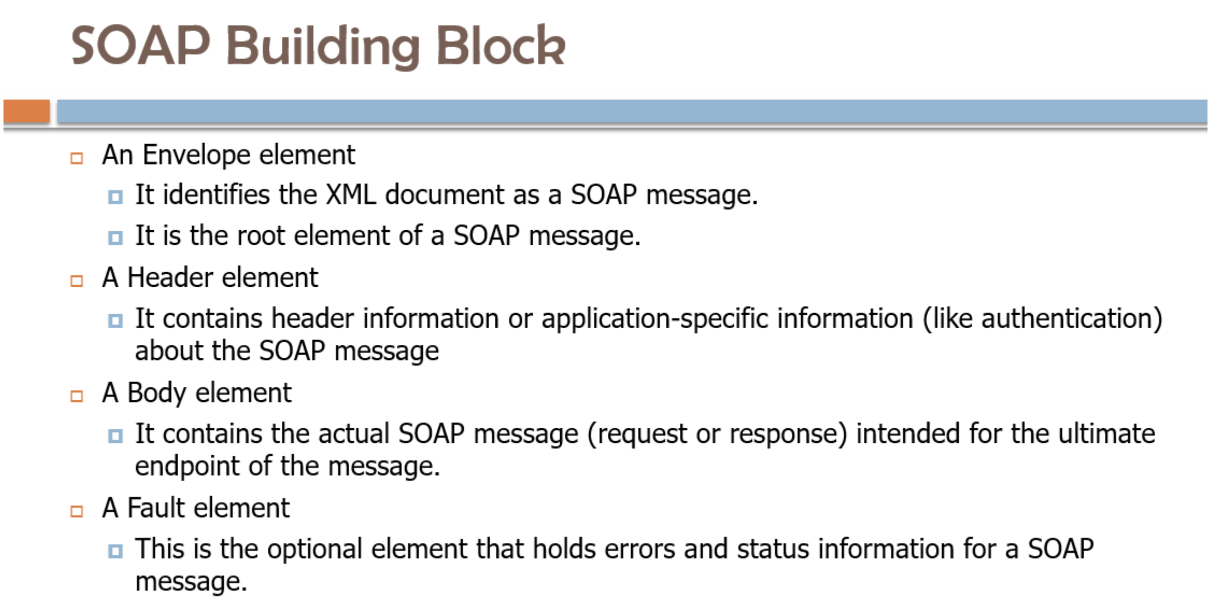
```
function get_detail() {
  var xmlhttp = new XMLHttpRequest(); //untuk membuat object XMLHttpRequest
  //   Get kategori value
  var kategori = document.getElementById("kategori").value;

  xmlhttp.onreadystatechange = function () {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("detail").innerHTML = this.responseText;
    }
  };
  xmlhttp.open("GET", "get_detail.php?kategori=" + kategori, true);
  xmlhttp.send();
}
```

No. 3



No.	SOAP	REST
1)	SOAP is a **protocol**.	REST is an **architectural style**.
2)	SOAP stands for **Simple Object Access Protocol**.	REST stands for **REpresentational State Transfer**.
3)	SOAP **can't use REST** because it is a protocol.	REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP.
4)	SOAP **uses services interfaces to expose the business logic**.	REST **uses URI to expose business logic**.
5)	**JAX-WS** is the java API for SOAP web services.	**JAX-RS** is the java API for RESTful web services.
6)	SOAP **defines standards** to be strictly followed.	REST does not define too much standards like SOAP.
7)	SOAP **requires more bandwidth** and resource than REST.	REST **requires less bandwidth** and resource than SOAP.
8)	SOAP **defines its own security**.	RESTful web services **inherits security measures** from the underlying transport.
9)	SOAP **permits XML** data format only.	REST **permits different** data format such as Plain text, HTML, XML, JSON etc.
10)	SOAP is **less preferred** than REST.	REST **more preferred** than SOAP.

Elemen



SOAP Building Block

- An Envelope element
  - It identifies the XML document as a SOAP message.
  - It is the root element of a SOAP message.
- A Header element
  - It contains header information or application-specific information (like authentication) about the SOAP message
- A Body element
  - It contains the actual SOAP message (request or response) intended for the ultimate endpoint of the message.
- A Fault element
  - This is the optional element that holds errors and status information for a SOAP message.