UI development for the Web

slides by Anastasia Bezerianos

Divide and conquer

A webpage relies on three components:

Content → HTML

text, images, animations, videos, etc

Presentation → CSS

how it will appear through a web browser

Behavior → JavaScript

real time interaction (validation, sorting, d&d)

HTML Documents

Web pages are created using Hypertext Markup Language (HTML)

A markup language is a set of characters or symbols that define a document's logical structure

Basic HTML Syntax

HTML is a text format relying on tags

Tags are enclosed in brackets (< >) and consist of an opening tag and a closing tag

HTML tags

- declare elements, e.g. image, canvas, svg, video, sound, button, checkbox, menu, textfield, etc…
- describe the content, e.g. whether the text should be a title (h1), a paragraph (p), emphasized (em), a quote (quote), etc…
- structure the content

Tutorial for learning HTML

http://www.htmldog.com/

http://www.sitepoint.com/html/

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello HTML</title>
  </head>
  <body>
    <p>Hello World!</p>
  </body>
</html>
```

CSS

Cascading Style Sheets

A single piece of CSS formatting information, (e.g. text alignment), is called style

Cascading refers to the ability for Web pages to use CSS info from more than one source and apply a style rule based on priority rules

CSS

CSS properties:

CSS styles have two parts separated by a colon

- The property refers to a specific CSS style
- The value assigned to it determines the style's visual characteristics
- `color:red`

Together, a CSS property and its value are a declaration or style declaration

Linking CSS and HTML

Inline Styles

- Add style information to a single element in a document, using the **`style`** HTML attribute
- Simplest way, but repetitive across HTML elements

```
<h1 style="color:red;"> My new heading </h1>
```

Linking CSS and HTML

External Style Sheets

A separate text document containing style declarations used by multiple HTML documents

mywebpage.html

```
<head>
   < link type = “text/css” href=”mycss.css” />
<head>
```

mycss.css

```
h1{font-family:Arial}  //all h1 tags
```

General syntax

a selector (where to apply visual characteristics)
and multiple pairs of **property:value**

```
body {font-family:Arial ; font-size:9pt}
```

case insensitive, whitespace and line-breaks ignored
selectors can be complex (unions, intersections, etc)
comments: /* this is a comment */

many online css tutorials

e.g. http://developer.mozilla.org
http://css-tricks.com/

Why CSS?

Easy to maintain

change once apply everywhere

CSS caching = less bandwidth + fast loading

Flexible

can load different CSS under different situations

e.g. devices (more later)

css layout and fun

block vs inline

HTML block-level elements

contain inline or other blocks and begin on new lines

e.g. `<h1>...<h6>, <p>, <ul>, <ol>, <li>, <table>, <tr>, <td>, <div> ...`

HTML inline (text) level elements

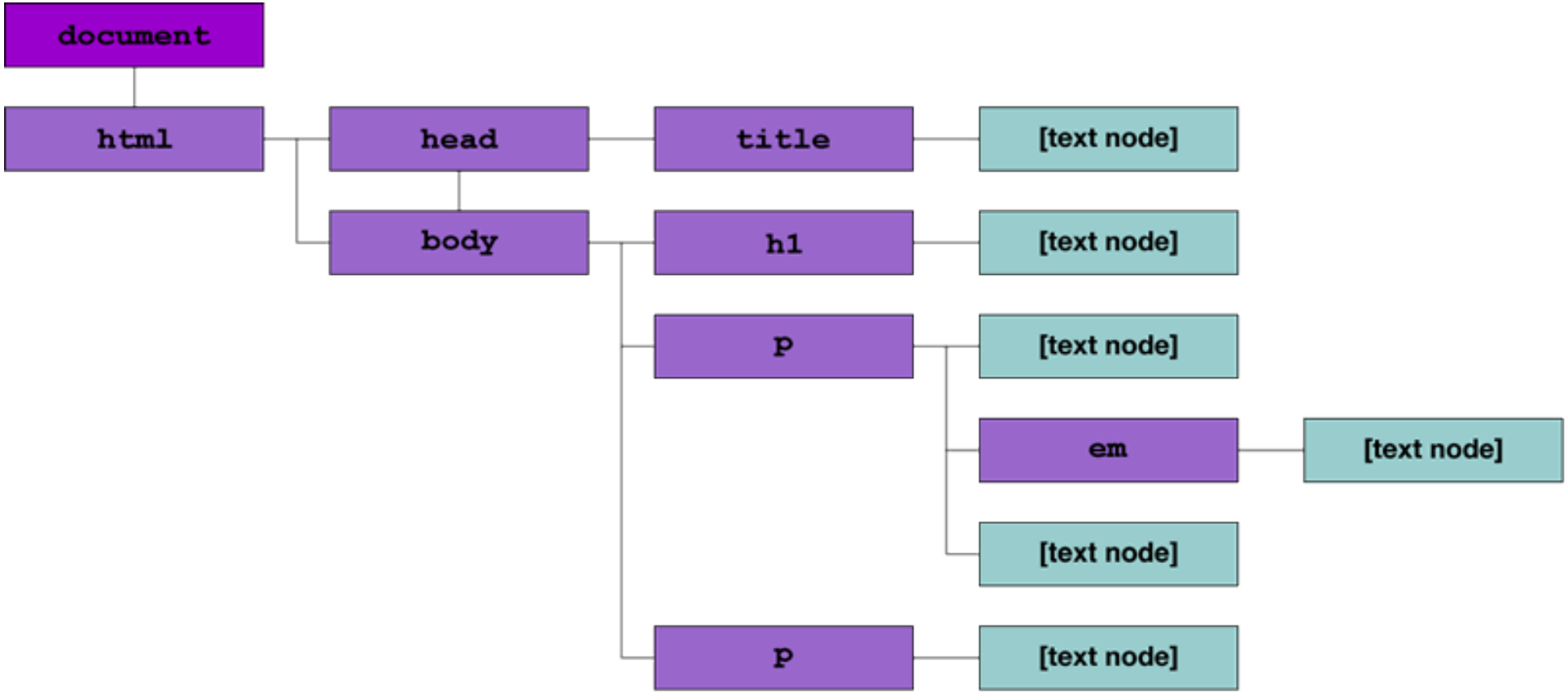
must be nested in blocks, may contain text or other inline elements, don't begin on new lines

e.g. `<em>, <strong>, <a>, <img>, <abbr>, <span> ...`

CSS helps define their visual properties

DOM

The browser builds a document object model (DOM), or tree of nodes



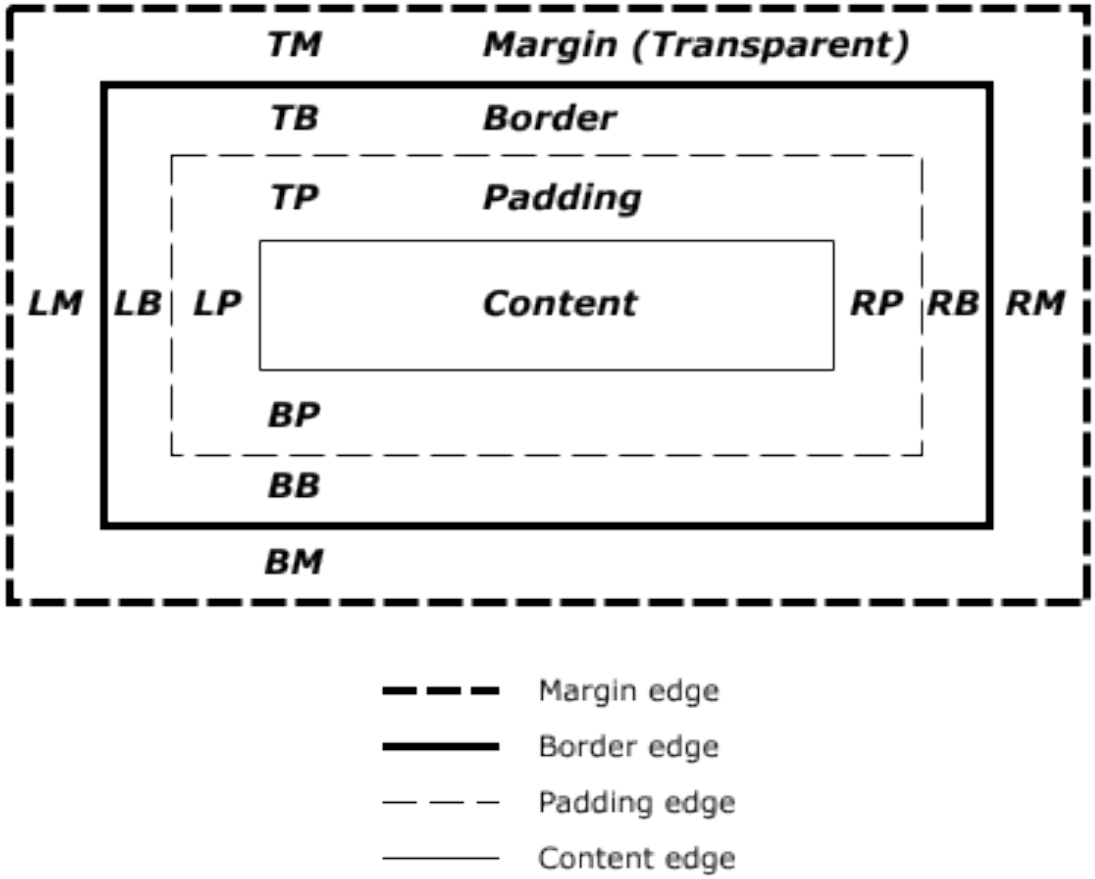
DOM

The browser builds a document object model (DOM), or tree of nodes

Each node is rendered as 0 or more boxes:

- inline elements generate inline boxes
- block elements block boxes
- using css you can edit their visual properties
- can fix the size of a box (`width, hight`)
- and go crazy inside (or outside)...

Properties: the box model



Properties: margin

Margin: space around an element

Properties:

`margin-top, margin-bottom, margin-left, margin-right`

Values: auto (up to browser), length (px,pt,cm,...),% of containing element, inherit (from parent)

Shorthand property:

```
margin: 25px 50px 75px 40px;
```

Shorthand values (1 to 4)

- 1: all the same, 2:top/bottom, left/right
- 3: top right/left bottom 4: top right bottom left

Properties: padding

Padding: space between element border and content

Properties:

```
padding-top, padding-bottom, padding-left, padding-right
```

Values: length (px,pt,cm),% of element

Shorthand property:

```
    pudding: 25px 50px 75px 40px;
```

Shorthand values (1 to 4), as with Margin

Properties: borders

Borders (always need border-style)

```
body{
    border-style:solid; //none,dashed,groove...
    border-width:5px; //pixels,thin/medium/thick
    border-color:rgb(255,0,0);
    border-top-style:dotted;
          // top/left/right/bottom border can be
          // different in style, width,color
}

body{border: 5px thin green;}
```

fun with css

- css3 menus
- css3 animations
- css3 transforms

CSS and easy menus

Easy navigation is important

Navigation bar = a (pretty) list of links

```
<nav>
<ul>
  <li><a href="default.asp">Home</a></li>
      <ul>
        <li><a href="about.asp">About</a></li>
      </ul>
  <li><a href="news.asp">News</a></li>
  <li><a href="contact.asp">Contact</a></li>
</ul>
</nav>
```

CSS and easy menus

in css remove the (default) bullets and padding

```
nav ul {
  list-style-type:none;
  margin:0;
  padding:0;
}
```

hide submenus and on hover drop down menu

```
nav ul ul { display: none; }
nav ul li:hover > ul { display: block;}
```

other “event” selectors: link, visited, hover, active, focus, selection, checked, etc.

CSS and easy menus

a vertical bar

```
nav a {
  display:block;
  width:60px;
}
```

a horizontal bar

```
nav li { float:left; }
nav a {
  display:block;
  width:60px;
}
```

CSS3 animations

Animations are transitions between style configurations

- style describing the CSS animation
- keyframes for start and end states of style (and possible intermediate points along the way)

```
h1 {
  animation-duration: 3s;
  animation-name: slidein;
  animation-iteration-count: infinite;
}
@keyframes slidein {
  from { margin-left: 100%; width: 300%}
  to   { margin-left: 0%; width: 100%; }
}
```

more at https://developer.mozilla.org/en-US/docs/CSS/Using_CSS_animations

CSS3 transforms

With the **transform** we change the coordinate space of elements (translated, rotated, scaled, and skewed)

stacking context (applied one after the other)

```
transform: none
transform: matrix(1.0, 2.0, 3.0, 4.0, 5.0, 6.0)
transform: translate(12px, 50%)  // translateX,translateY
transform: scaleX(2)             // scale, scaleY
transform: rotate(0.5turn)       // degrees/rad rotateX ...
transform: skewX(30deg)          // skewY
transform-origin ....
```

```
h1 {-webkit-transform:rotate(45deg)}
```

seen these in Computer Graphics?

- https://developer.mozilla.org/en-US/docs/CSS/transform

javascript

What is JavaScript (JS)?

dynamic and functional language (like java, C#)

- syntax influenced by C
- names and naming conventions from Java, O-O
- input treated with listeners

can be interpreted by web browsers

can be used for web client programming

can be used for server programming (e.g., Node.js)

JavaScript (JS)

importance to us

- used to provide interactivity to Web sites and apps

allows us to change

- the document's list of stylesheets
- the rules of a stylesheet
- the individual elements of the DOM, independent of stylesheet used

JavaScript (JS)

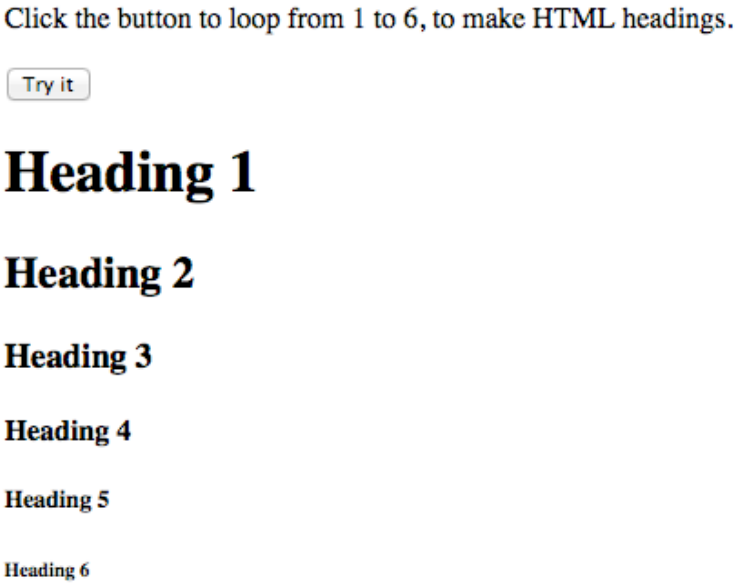
classic programming structures

- statements, functions, comments, `IF ... THEN`, `FOR`, `WHILE`,...
- events (onmouseover, onclick, onkeyup, etc)
- access to the html DOM

- Examples and tutorials
  https://developer.mozilla.org/

JavaScript (JS)

```
<!DOCTYPE html>
<html>
<body>

<p>Click the button to loop from 1 to 6, to make HTML headings.</p>
<button onclick="myFunction()">Try it</button>
<div id="demo"></div>

<script>
function myFunction()
{
   var x="",i;
   for (i=1; i<=6; i++)
   {
      x=x + "<h" + i + ">Heading " + i + "</h" + i + ">";
   }
  document.getElementById("demo").innerHTML=x;
}
</script>

</body>
</html>
```



JQuery library for JS

JQuery is a library for JS

It provides a cross-browser API for

- HTML/DOM manipulation
- DOM event handling
- CSS manipulation
- Effects and animations
- AJAX (client-server communication)
- Other utilities

JQuery Syntax

selecting HTML elements and perform action on them

Basic syntax: **$(selector).action()**

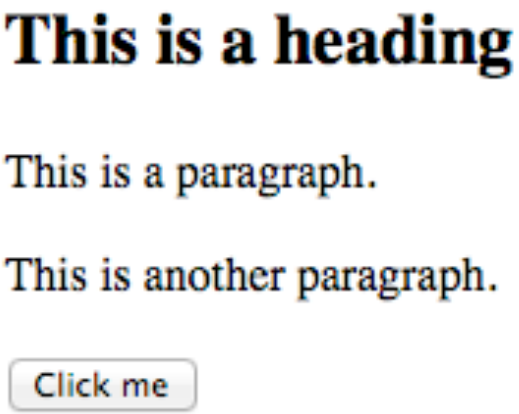
- A $ sign defines/accesses jQuery
- A (selector) finds HTML elements
- A jQuery action() is performed on the element(s)

Examples:

- $(this).hide() - hides the current element.
- $("p").hide() - hides all <p> elements.

JQuery library for JS

```
<!DOCTYPE html>
<html>
<head>

<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>

<script>

$(document).ready( function(){
    $("button").click( function(){
       $("p").hide();
    });
});

</script>

</head>

<body>
<h2>This is a heading</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Click me</button>
</body>
</html>
```



Cooler stuff in the TA !!

More on JQuery

Functions can have “callbacks” to order events

```
$("button").click(function(){
  $("p").hide("slow",function(){
    alert("The paragraph is now hidden");
  });
});
```

and can be chained

```
$("#p1").css("color","red").slideUp(2000).slideDown(2000);
```

Even more on JQuery

Lots of JQuery extensions for

- widget creation and manipulation
- interaction extensions e.g. http://jqueryui.com/
- Note: jQueryUI is a library for JavaScript in the same manner that Swing is a library for Java

How to draw/interact

- option 1: <canvas>
  - <canvas> is an HTML5 element
  - used to draw graphics using scripting (e.g. JS)
  - good for graphs, photo compositions or animations

```
<canvas id="canvas" width="300" height="300"></canvas>

//get a reference to the canvas
var ctx = $('#canvas')[0].getContext("2d");

var x,y;
//draw a circle at x, y
ctx.beginPath();
ctx.arc(x, y, 10, 0, Math.PI*2, true);
ctx.closePath();
ctx.fill();
```

```
$(document).mousemove(onMouseMove);

function onMouseMove(evt) {
  if (evt.pageX > x+10 && evt.pageX < x-10 &&
      evt.pageY > y+10 && evt.pageY < y-10) {
        alert (Little ball clicked);
  }
}
```

How to draw/interact

- option 2: <svg> (Scalable Vector Graphics)
  - XML-based language for creating graphics
  - used for static images, animations and UI
  - supports CSS stylesheets

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <circle cx="100" cy="50" r="40" stroke="black"
    stroke-width="2" fill="red" />
</svg>
```

... media queries ...
(responsive web design)

Why CSS?

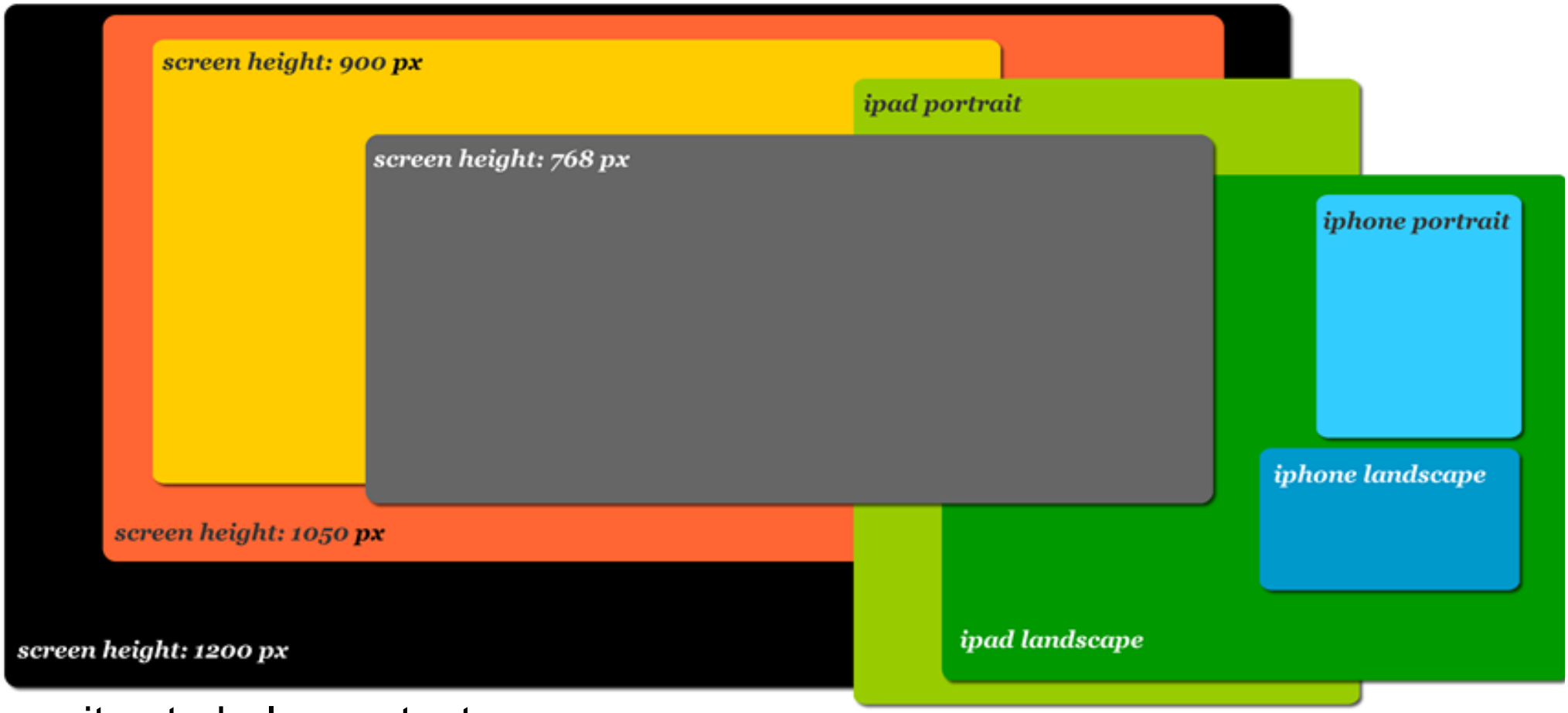
Flexible (can load different CSS under different situations, e.g. devices)

How do we know what device/resolution?

But also, how do we design for it (later ...)

Display Sizes

some display resolutions for iphones (http://www.websitedimensions.com/)



sites to help you test your page:

e.g. http://quirktools.com/screenfly/, http://www.viewlike.us/index.php

Media queries

Since CSS2 **media types** (device)

```
screen, braille, speech, ...
```

In CSS3 added **media queries** (device capabilities)

```
width & height of browser window,

device-width, device-height or device-aspect-ration

orientation (landscape or portrait in phone)

resolution (dpi) ....
```

http://www.w3.org/TR/css3-mediaqueries/

Media queries

we query our media type and capabilities

```
@media screen and (min-device-width:481px) and ..
```

and we

- create style blocks for this query
- or call a different style sheet

http://www.w3.org/TR/css3-mediaqueries/

http://css-tricks.com/resolution-specific-stylesheets/

Media queries

e.g., blocks for different media. On your CSS file

```
/* all screens */

#mypar{ font-size:12px;backround-color:#9F0;}


/* large screen (1440px or more)  */

@media screen and {min-width:1440px} {

      #mypar{ font-size:18px;backround-color:#F90;}

  }

...
```

Media queries

e.g., of calling a different css file.

On your HTML, first link a default CSS sheet

```
< link type = “text/css” href=”my_default_css.css” />
```

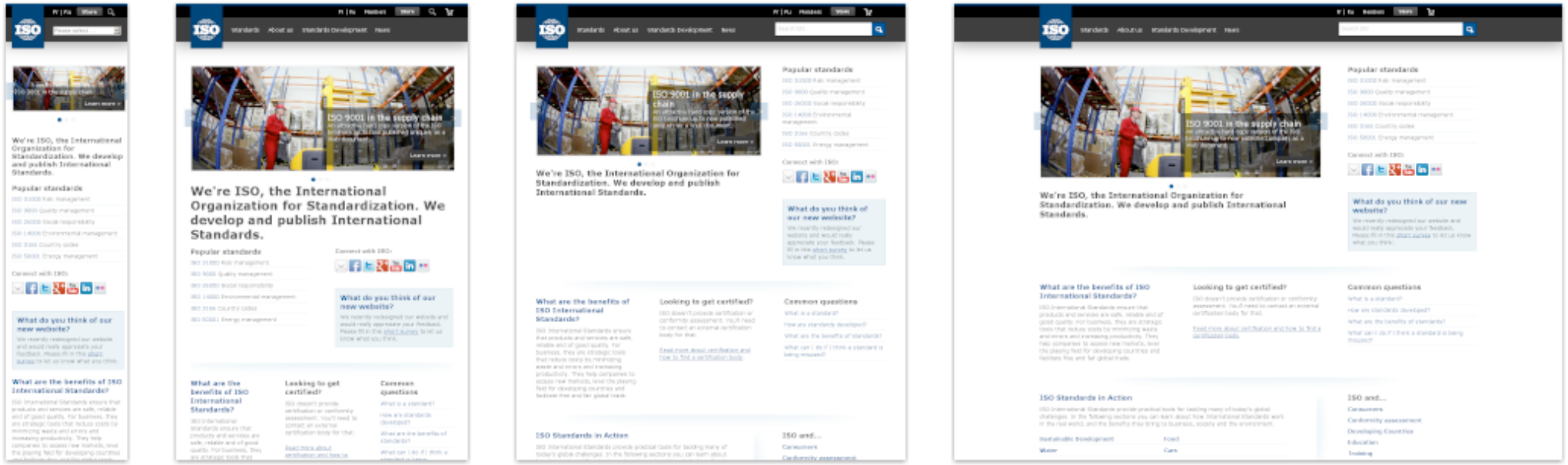
then override it under specific conditions

```
< link rel="stylesheet" type="text/css" media="only screen
and (max-device-width:480px)"href="small-device.css" />

...
```

Media queries

ISO

6



from mediaqueri.es