

HACK YALE

< WEB DEVELOPMENT / >

WWW.HACKYALE.COM

HACK YALE

< WEB DEVELOPMENT / >

week_2

MASTERING BASIC CSS

CODING RIGHT ALONG...

The Agenda

- Oscars / Showcase
- Common problems / questions
- Browser defaults
- The DOM and CSS Selectors
- CSS positioning and display
- Coding!



HACKYALE OSCARS

WEEK 3





COMMON SLIP UPS

AND SOLUTIONS



COMMON MISTAKES

- Don't forget to use the `<html>` tag
- Make sure you close your body tag with `</body>`
- Don't put any content outside the body!
- Use the `.html` extension when saving files

SUGGESTIONS

- Name your main page index.html
- Keep your images in an images folder and your stylesheets in a styles folder
- Include a link back to the home page

The box model

THE BOX MODEL



Each element exists as a nested box.

PROPERTIES THAT AFFECT THE BOX MODEL

- width
- height
- padding
- border
- margin
- All can have -top, -right, -left, and -bottom
 - e.g. `margin-top: 40px;`
- (Note: Position and Float also play a role)

STYLESHEETS

BROWSER DEFAULTS

DEFAULT STYLESHEETS

Each browser has its own default stylesheet

- This is why links are blue and underlined even when you don't style them
 - Default stylesheets vary a bit from browser to browser
 - Many developers use CSS resets to standardize their styles
-

CSS RESETS

Resets override the browser's default stylesheet, allowing for cross-browser consistency

- There are many resets on the market. *Normalize.css* is nice.
 - Normalize isn't quite a reset, because it keeps some useful styles, saving the developer time and effort of reimplementing
 - Supported in Chrome, Firefox 3+, Safari 4+, Opera 10+, Internet Explorer 6+
 - Used by Twitter Bootstrap, HTML 5 Boilerplate, and css-tricks.com
 - <http://necolas.github.com/normalize.css/>



CSS SELECTORS

FOR SOME FINESSE



CSS SELECTORS

- We already know about # and .
- Some of you are using :hover, :visited, :active, etc.
 - Anything with a colon is a *pseudoselector*
 - Allows us to select elements in a *particular state*
- *Example:*
 - `a:hover {color: green;}`



DISPLAY PROPERTY

DISPLAY: SICKNASTY;



DISPLAY VALUES: BLOCK

`display: block;`

- Element takes up the full available width
- Begins on a new line, forces following content onto a new line

Some text I wrote here.
<div>My div is here!</div>
And more text continues here.

Some text I wrote here
My div is here!
And more text continues here.

DISPLAY VALUES: INLINE

`display: inline;`

- Element takes up only the width of its content
- Remains in the flow of the document (does not start a new line)

```
<p>I've got a paragraph of  
text here. In the middle, I'd  
like to put some <span>frilly  
text</span> so that my  
students think I'm cute.</p>
```

I've got a paragraph of text
here. In the middle, I'd like to
put some rilly tex so that my
students think I'm cute. *frilly text*

DISPLAY VALUES: NONE

`display: none;`

- Element is not rendered in the browser
- Removed from the flow of the document
 - i.e. it does not affect the positioning of other elements

```
<p>I've got a paragraph of  
text here. In the middle, I'd  
like to put some <span  
style="display:none;">frilly  
text</span> so that my  
students think I'm cute.</p>
```

I've got a paragraph of text
here. In the middle, I'd like to
put some so that my students
think I'm cute.

DISPLAY VALUES: INLINE-BLOCK

`display: inline-block;`

- Rendered like an inline element (only takes up needed width, doesn't disrupt document flow)
- Allows us to set block-display properties like width, height, and top and bottom padding / margin

DISPLAY VALUES: PRACTICE

```
<ul id="my-boxes">
  <li>First box</li>
  <li>Second box</li>
  <li>Third box</li>
</ul>
```

```
#my-boxes li {
  display: block; /* default */
  width: 175px;
  height: 75px;
  background: gray;
  border: 1px solid black;
}
```

• First box
• Second box
• Third box

DISPLAY VALUES: PRACTICE

```
<ul id="my-boxes">  
  <li>First box</li>  
  <li>Second box</li>  
  <li>Third box</li>  
</ul>
```

• First box • Second box • Third box

```
#my-boxes li {  
  display: inline;  
  width: 175px;  
  height: 75px;  
  background: gray;  
  border: 1px solid black;  
}
```

DISPLAY VALUES: PRACTICE

```
<ul id="my-boxes">
  <li>First box</li>
  <li>Second box</li>
  <li>Third box</li>
</ul>
```

• First box

• Second box

• Third box

```
#my-boxes li {
  display: inline-block;
  width: 175px;
  height: 75px;
  background: gray;
  border: 1px solid black;
}
```

EACH ELEMENT TYPE HAS A DEFAULT DISPLAY PROPERTY

Defaults

- usually '*inline*' or '*block*'
 - inline
 - span, a, em, img, most text modifiers
 - block
 - div, p, ul, form, etc.
-

CSS POSITIONING

The position property

THE POSITION PROPERTY SPECIFIES A WAY
FOR AN ELEMENT TO POSITION ITSELF
WITH REGARD TO THE BROWSER WINDOW,
THE PARENT ELEMENT, OR RELATIVE TO
SIBLING ELEMENTS.

POSITION: STATIC

Render the element relative to its neighbor with no offset

- The default setting (so you never need to declare this in CSS)
- You can imagine this as “Tetris”, where the gravity is up instead of down,
 - And you’re holding the left key

POSITION: FIXED

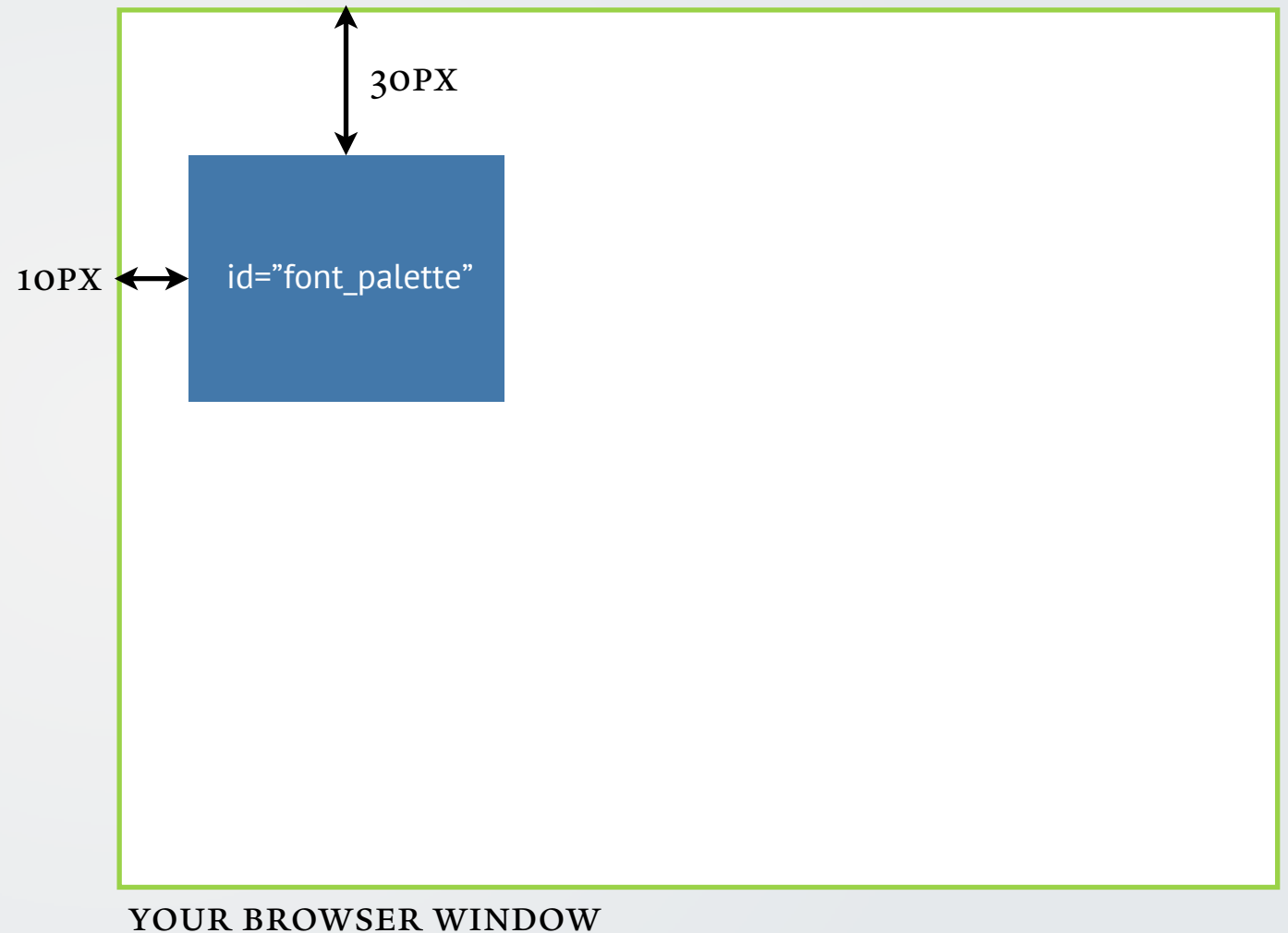
Sets the element to be rendered at a fixed location in the browser window, regardless of page scrolling

- top, left, bottom, and right properties tell the element where to position itself in the browser window
- often used for navigation panes

POSITION: FIXED

```
2 #font_palette {  
3   position: fixed;  
4   top: 30px;  
5   left: 10px;  
6 }
```

Take the element with
`id="font_palette"` and render it
30px from the top of the window
and 10px from the left of the window



POSITION: ABSOLUTE

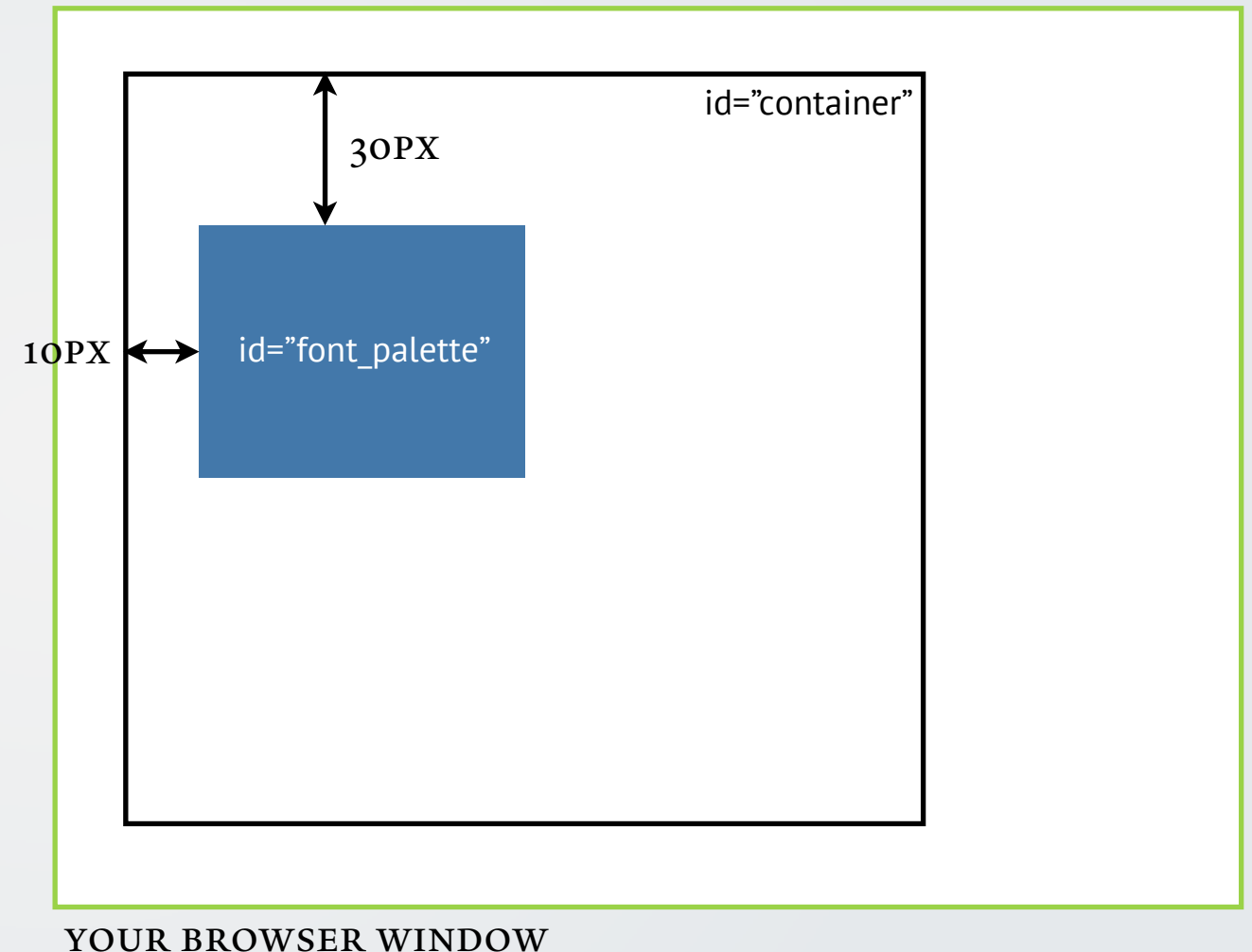
Sets the element to be rendered at a specific location in the parent element

- top, left, bottom, and right properties tell the element where to position itself in the parent element
- The parent element must not be position static!!
 - Quick fix: make it position: relative;

POSITION: ABSOLUTE

```
2 #container #font_palette {  
3     position: absolute;  
4     top: 30px;  
5     left: 10px;  
6 }
```

Take the element with
`id="font_palette"` and render it
30px from the top and 10px from the
left of its parent (`id="container"`)



POSITION: RELATIVE

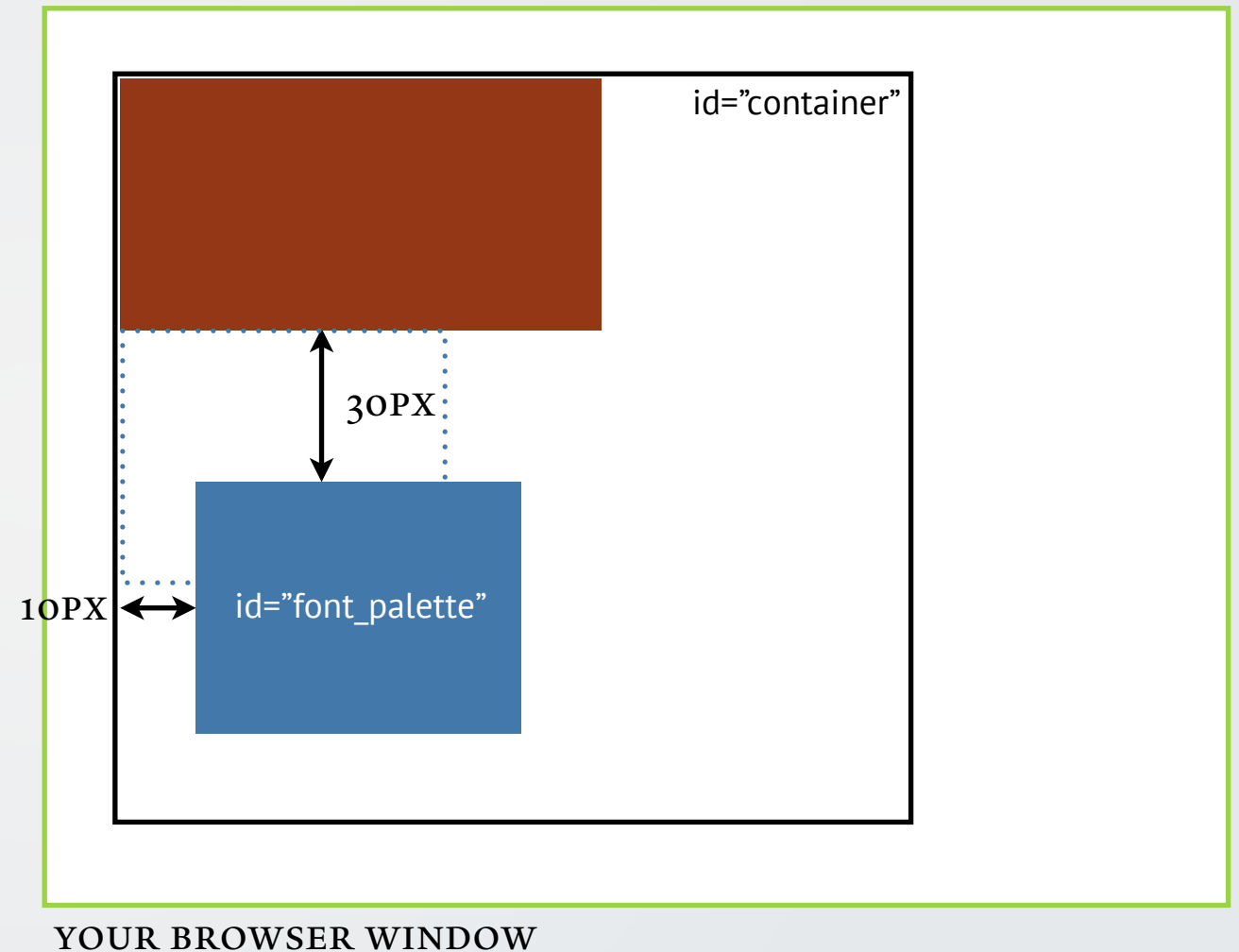
Tells the element how to position itself relative to neighboring sibling elements (i.e. elements with the same parent)

- top, left, bottom, and right properties tell the element where to position itself relative to where it would normally be
- similar to absolute positioning
- used to specify the exact location of an element

POSITION: RELATIVE

```
2 #font_palette {  
3   position: relative;  
4   top: 30px;  
5   left: 10px;  
6 }
```

Take the element with `id="font_palette"` and render it 30px from the top and 10px from the left of where it would normally be





The float property

`float: left;`



THE FLOAT PROPERTY


Causes element to ‘float’ toward one side of its parent

- Four values: left, right, none, and inherit
- IMPORTANT: the width of the element collapses to only the needed width of the content (think *inline*!)
- Multiple floated siblings will line up side by side

FLOAT: LEFT;

```
.floatleft {  
  display: block;  
  float: left;  
  width: 75px;  
  height: 75px;  
  background: turquoise;  
}
```

```
<p>  
  <span class="floatleft"></span>  
  Lorem ipsum dolor sit amet,  
  consectetur adipiscing elit ...  
</p>
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras at mauris sed dui mollis pellentesque non quis velit. Cras non sapien vel metus consequat volutpat vel a augue. Etiam ultrices ultricies ligula eu fringilla. Nulla quis augue velit, et blandit lacus. Nulla vitae lacus quis dolor rutrum auctor nec ac ipsum. Integer consectetur vestibulum metus, id vehicula purus fermentum non. Praesent ac ante porttitor tellus dictum sagittis in quis dolor.

FLOAT: LEFT;

```
.floatleft {  
  display: block;  
  float: left;  
  width: 75px;  
  height: 75px;  
  background: turquoise;  
}
```

```
<p>  
  <span class="floatleft"></span>  
  <span class="floatleft"></span>  
  Lorem ipsum dolor sit amet,  
  consectetur adipiscing elit ...  
</p>
```



Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Cras at
mauris sed dui mollis pellentesque non
quis velit. Cras non sapien vel metus
consequat volutpat vel a augue. Etiam

ultrices ultricies ligula eu fringilla. Nulla quis augue velit, et
blandit lacus. Nulla vitae lacus quis dolor rutrum auctor nec ac
ipsum. Integer consectetur vestibulum metus, id vehicula purus
fermentum non. Praesent ac ante porttitor tellus dictum sagittis
in quis dolor.


Note how the two boxes “float” next to each other, with no spacing between them.

FLOAT: RIGHT;

```
.floatright {  
  display: block;  
  float: right;  
  width: 75px;  
  height: 75px;  
  background: turquoise;  
}
```

```
<p>  
  <span class="floatright"></span>  
  Lorem ipsum dolor sit amet,  
  consectetur adipiscing elit ...  
</p>
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras at mauris sed dui mollis pellentesque non quis velit. Cras non sapien vel metus consequat volutpat vel a augue. Etiam ultrices ultricies ligula eu fringilla. Nulla quis augue velit, et blandit lacus. Nulla vitae lacus quis dolor rutrum auctor nec ac ipsum. Integer consectetur vestibulum metus, id vehicula purus fermentum non. Praesent ac ante porttitor tellus dictum sagittis in quis dolor.



THE CLEAR PROPERTY

Causes an element to drop below any floated objects that match its value

- 5 values: left, right, both, none, inherit
- e.g. `clear: left;` will drop below elements with `float: left;` (but not those with `float: right;`), while `clear: both;` will drop below all floated elements

CLEAR: BOTH;

```
.floatleft {  
  display: block;  
  float: left;  
  width: 75px;  
  height: 75px;  
  background: turquoise;  
}  
p { clear: both; }
```

```
<div>  
  <span class="floatleft"></span>  
  <p>Lorem ipsum dolor sit amet,  
  consectetur adipiscing  
  elit ...</p>  
</div>
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras at mauris sed dui mollis pellentesque non quis velit. Cras non sapien vel metus consequat volutpat vel a augue. Etiam ultrices ultricies ligula eu fringilla. Nulla quis augue velit, et blandit lacus. Nulla vitae lacus quis dolor rutrum auctor nec ac ipsum. Integer consectetur vestibulum metus, id vehicula purus fermentum non. Praesent ac ante porttitor tellus dictum sagittis in quis dolor.

CODING

LET'S COPY JENNIFER DEWALT

Our goal: Recreate this website.

- http://jenniferdewalt.com/secret_message.html
- Let's do it!

HOMEWORK

HOMework

Time to play developer! You have now learned enough HTML and CSS to construct many websites on the Internet. This week, you have received your first professional assignment: Your designer has created a wireframe of the blog site (or site of your choice!) they want you to build. Take their outline, keeping in mind dimensions and functionality specified, and put your own spin on it. Select your own colors, fonts, add additional functionality – the choice is yours! Just make sure to deliver on all of the basics that have been specified.

This assignment will take some time, so start early. The goal is to grapple with the challenges as best you can, working to complete as much of the design as possible. When you get stuck, we encourage you to collaborate with each other, come to office hours, and remember that we are here to help with any questions you might have!



HOMEWORK



http://hackyale.com/pdfs/homework_2_wireframe.pdf

HACK YALE

< FRONT END / >

WWW.HACKYALE.COM